October 1994

Kolmogorov Complexity and the Information Content of Parameters

Robin Clark
University of Pennsylvania, rclark@babel.ling.upenn.edu

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-17.

Kolmogorov Complexity and the Information Content of Parameters

Abstract

A key goal of linguistic theory is to account for the logical problem of language acquisition. In particular, linguistic constraints can be taken as constraining the learner's hypothesis space and, so, reducing its computational burden. In this paper, I will motivate an information theoretic approach to explaining some linguistic constraints. In particular, the theory attempts to relate ease of acquisition with the simplicity of linguistic representations and their frequency in the learner's input text. To this end, the paper reviews some results in information theory and Kolmogorov complexity and relates them to a theory of parameters.

Comments

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-17.

The Institute For
Research In Cognitive
Science

Kolmogorov Complexity and the Information Content of Parameters

by

Robin Clark

**University of Pennsylvania
3401 Walnut Street, Suite 400C
Philadelphia, PA 19104-6228**

October 1994

Site of the NSF Science and Technology Center for
Research in Cognitive Science

University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 94-17

Kolmogorov Complexity and the Information Content of Parameters

Robin Clark[*]
Department of Linguistics
University of Pennsylvania
Philadelphia, PA 19104
e-mail: rclark@babel.ling.upenn.edu

Abstract

A key goal of linguistic theory is to account for the logical problem of language acquisition. In particular, linguistic constraints can be taken as constraining the learner's hypothesis space and, so, reducing its computational burden. In this paper, I will motivate an information theoretic approach to explaining some linguistic constraints. In particular, the theory attempts to relate ease of acquisition with the simplicity of linguistic representations and their frequency in the learner's input text. To this end, the paper reviews some results in information theory and Kolmogorov complexity and relates them to a theory of parameters.

1 Introduction

A classic problem in linguistic theory is the relationship between linguistic principles and learning. Since at least Chomsky (1965), the logical problem of language acquisition has provided a foundation for linguistic theorizing. One might hope that a variety of linguistic principles and relations could be grounded in the learning theory; thus, c-command, subjacency

[*] I received generous support for this work from the Research Foundation of the University of Pennsylvania. Audiences at CUNY, the State University of New York at Stony Brook and USC have patiently listened to various parts of this paper. I would also like to thank David Embick, Shyam Kapur, Mark Johnson and the regular participants of the Information Theory Reading Group at the University of Pennsylvania: Srinivas Bangalore, Jason Eisner, Mitch Marcus, Dan Melamed, Lance Ramshaw and Jeff Reynar. Naturally, this doesn't imply that they endorse anything contained herein.

and government might all have a grounding in the learning theory. In this paper, I will argue that learning theory can ground many linguistically significant relations. I will pay particular attention to developing a theory of the information content of parameters. This theory will, in turn, be used in the characterization of the input data available to the learner. A formal characterization of the complexity of the input data may well be crucial in constraining the class of possible learners and, thus, be of some methodological importance to linguistics (see, in particular, Osherson & Weinstein, 1992).

To take one example, the size of the government domain might be a direct result of properties of the learner; a larger government domain might require computational capacities that surpass those of a conventional learner, resulting in a class of languages that are grammatically possible but unlearnable. It may well be the case that there are possible but unlearnable languages in the space made availabe by Universal Grammar, although one can only speculate as to how we could study the space empirically were this to be true. Indeed, I would prefer to be driven to the position that there are possible but unlearnable languages rather than adopting such a position from the outset of the study. I will, therefore, adopt the following hypothesis:

(1) The form of parameters is a direct consequence of the fact that they must be set by a learning algorithm on the basis of evidence from an input text.

The rest of this paper can be read as a formalizaion of the hypothesis in (1). I will propose a formal theory of the bound on the amount of information that can be contained in any parameter. In particular, I will propose a limit based on formal properties of the evidence available to language learners; I will argue that there must be a strict relationship between the complexity of linguistic representations and the complexity of the input text. This relationship places a bound on what can in principle be learned from the input text and, thus, limits the amount of cross-linguistic variation that is possible. Finally, I will consider a number of applications of this theory to the study of parameters and a characterization of the input evidence available to the learner.

In section 2 I will discuss certain background assumptions on the relationship between formal learning theory and the study of locality in linguistic theory. Section 2.1 discusses degree 2 learnability (Wexler & Culicover, 1980) and the account of one locality principle, subjacency, which that theory was able to give. In section 2.2 I will turn to the theory of parameter setting and argue that the current theory must be supported by a theory of locality analogous to the degree 2 theory. A notable gap in the current theory is the lack of a suitable subtheory of what constitutes a "linguistically possible parameter" (if anything); such a theory, it should be noted, would place substantive empirical constraints on typological variation.

The following sections attempt to make up for this gap. Section 3 is devoted to the formal underpinnings of a theory of linguistic complexity. I will first turn to some mathematical background necessary for understanding the theory of Kolmogorov complexity in section 3.1. Kolmogorov complexity is a theory of the inherent information content of an object. In many ways, it seems analogous to the evaluation metric of Chomsky (1965), although the interpretation given here is rather different from the traditional interpretation of the evaluation metric. Section 3.1 is, in many ways, quite demanding and the reader may wish to consult Cover & Thomas (1991) or Li & Vitányi (1993) for more leisurely and, no doubt, more comprehensible introductions to the field. Kolmogorov complexity is of deep significance to linguistic theory and is a topic of interest in its own right, as a perusal of the papers collected in Zurek (1990) will quickly show.

Section 3.2 is an example of how one can encode phrase markers as strings of binary numbers. The example is convenient, since it will allow us to compare the complexity of parameters and texts in a straightforward fashion. However, nothing crucial rests on the discussion in this section and the reader is invited to skip the section and move on to the results in section 4. It is here that a number of the basic relations between the complexity of the input text and the complexity of parameters are established. In particular, I will argue that the former can provide an upper bound on the latter.[1] Finally, I will turn to some of the results of Osherson & Weinstein (1992) and speculate that the theory of complexity developed here can give us some insight into constraining the class of learners.

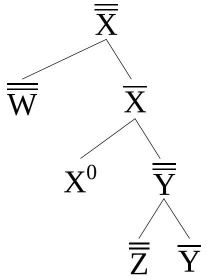
2 Learnability and the explanation of locality

An important contribution that the theory of learnability can make to linguistic theory is to explain some aspects of locality. In syntax, locality is manifest almost everywhere. Notions like government, bounding, subcategorization, predication and even c-command and m-command basically serve to place constraints on the space within which important linguistic relations can play. Consider, for example, the definition of *government* in (2) taken from Chomsky (1986):

(2) α *governs* β if and only if α m-commands β and every barrier for β dominates α.

The above definition basically serves to define the following abstract tree fragment:

(3) The abstract domain of government:

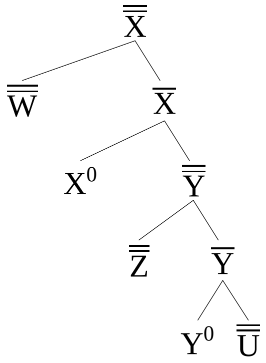


[1] These results are very much in the spirit of PAC (“probably approximately correct”) learning where establishing sample size is of prime importance. For general introductions to PAC learning see Natarajan (1991) and Anthony & Biggs (1992). Niyogi & Berwick (1993) develop an application to principles and parameters theories in their analysis of Gibson & Wexler (1994).

In the above tree, certain linguistic processes (case-marking and θ-role assignment, for example) may take place between X^0 and $\overline{\overline{Y}}$, X^0 and $\overline{\overline{W}}$ (Spec-head agreement) or X^0 and $\overline{\overline{Z}}$ but not between X^0 and material dominated by $\overline{\overline{Z}}$, the latter being too large a domain.

While one can cogently argue that notions like government and command are epiphenomenal, it appears, nevertheless, that the domains selected by these relations have some linguistic significance. The question of *why* these relations are significant principles of organization in natural languages is a legitimate object of linguistic theorizing. One could easily imagine that Universal Grammar would have selected a broader domain, "extended government", over which linguistic relations could take place:

(4) The abstract domain of extended government:


$\overline{\overline{X}}$
$\overline{\overline{W}}$
$\overline{X}$
X^0
$\overline{\overline{Y}}$
$\overline{\overline{Z}}$
$\overline{Y}$
Y^0
$\overline{\overline{U}}$

The domain of the relation shown in (4) is somewhat larger than the domain of government and it predicts that linguistically significant relations could take place between X^0 and $\overline{\overline{W}}$. Even a relation, like antecedent government, between $\overline{\overline{W}}$ and $\overline{\overline{U}}$ might best be broken up into a pair of relations: one between $\overline{\overline{W}}$ and $\overline{\overline{Z}}$ and the other between $\overline{\overline{Z}}$ and $\overline{\overline{U}}$. Note that these two relations collapse down to the domain of the government relation shown in (3).

In the absence of such relations, we can conclude that extended government is not a significant relation and that UG does not single out X^0 and $\overline{\overline{W}}$ for special treatment. The contrast between government and extended government poses an interesting challenge for linguistic theory: Why has UG selected a domain like the one selected by government rather than the one selected by extended government? A coherent approach to this challenge has been to attempt to reduce linguistically relevant domains to domains relevant for language learnability. The core intuition behind this approach is that linguistically significant relations must be expressed on a highly restricted syntactic domain if they are to be expressed at all. Since the linguistically significant relations are reflected within a domain of low complexity, the probability that learner will detect the effects of these relations increases and convergence to the target (i.e., successful acquisition) becomes more likely.

2.1 Degree 2 Learnability

The best known learnability proof, that of Wexler & Culicover (1980), relied heavily on locality in order to guarantee convergence to the target. Their system learned a transformational component of the type familiar from the standard theory (Chomsky, 1965) and thus was not concerned with parameter setting in the current sense. Nevertheless, it is worth briefly considering their system. The system was presented with a pair consisting of a base tree (output of the phrase structure component, corresponding to the semantic structure of the sentence[2]) and the surface string. The learner would then run its transformational component on the base form, b, to check if the output corresponded to the surface string, s.

Underlying the proof of convergence is the notion of *detectable error*:

(5) *Error and Detectable Error*
If a transformational component C [the learner's transformational component — RC] maps a base phrase-marker P onto a surface structure that is different from the surface structure obtained when A [the target (adult) transformational component — RC] is applied to P, we say that C *makes an error* on P. If C makes an error on P, and if the surface *sentence* when C does the mapping is different from the surface sentence when A does, we say that C *makes a detectable error* on P. [Emphasis in the original text — RC]

The learner has made an error if the output of the learner's grammar differs from the output of the target adult system on some datum. Notice that the two systems might differ without an observer being able to detect the difference. For example, the two grammars might output the same string with different hierarchical structures. The definition of *detectable error* singles out errors on which the string (the surface sentence) generated by the learner's grammar does not correspond to the string generated by the target grammar.

Detectable errors were a crucial ingredient in the Wexler & Culicover proof. The learner only changes its hypothesis when evidence from the external world forces it to do so. If the learner's hypothesis is incorrect, then, it will only change its hypothesis if it makes an error on some input datum generated by the target grammar. The learner would never change its hypothesis without the motor of detectable errors to drive it; if its current hypothesis is successfully able to account for the input, the learner will not change its hypothesis to test some other alternative.

Notice that the learner's grammar and the target grammar might agree up to strings of a high-level of complexity. For example, the learner's grammar could agree with the target up to structures with 20 levels of embedding and then differ from it on structures with 21 levels of embedding or more. Since the learner would only change its hypothesis due to a detectable error, it will change its hypothesis only when it has made an error on one of these highly complex

[2] Notice that this base form was taken as linguistically invariant under the *Universal Base Hypothesis* (UBH); this factored out the problem of learning the base component. See Kayne (1993) for a recent proposal that revives the UBH. Notice that, if this is correct, then the locus for cross-linguistic variation and, hence, learning, resides somewhere other than in base structures.

structures. Since these structures are very rare in realistic input texts, the learner is unlikely to encounter such an example. As the probability of encountering the crucial input decreases, the amount of time required for the learner to converge increases. In the worst case, the learner would never encounter the relevant examples, thus making the error that the learner has made effectively undetectable and the amount of time required to converge approaches infinity. In other words, the learner is effectively placed in the situation of being unable to converge since the time required is so high.

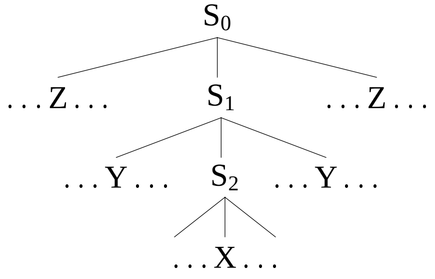
For the above reason, it is crucial that the complexity of the structures on which the learner makes a detectable error be strictly limited. That is, it must be guaranteed that if the learner makes an error, then it will make an error on a sufficiently simple example. This is the content of the *Boundedness of Minimal Degree of Error* (from Wexler & Culicover, 1980):

(6) *Boundedness of Minimal Degree of Error* (BDE)
For any base grammar B there exists a finite integer U, such that for any possible adult transformational component A and learner (child) component C, if A and C disagree on any phrase-marker b generated by B, then they disagree on some phrase-marker B' generated by B, with b' of degree at most U.

Here *degree* refers to the number of S nodes embedded in the representation. Since simple examples dominate the input, the BDE increases the probability that the learner will make a detectable error if its hypothesis is incorrect. The increased probability of making an error decreases the amount of time that the learner must spend searching the hypothesis space before it converges. Thus, the learner is not only more likely to converge (the probability of convergence approaches 1 in the limit, as Wexler & Culicover demonstrate), but it is more likely to converge in less time. Since real language acquisition is an automatic process which takes place over a relatively small time period, this property is a crucial one for a psychologically plausible theory of learning.

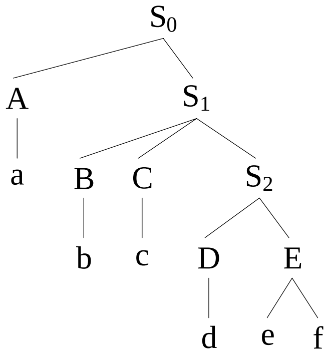
The smaller we can make the constant U in (6) the more the learner's task will be facilitated. Wexler & Culicover propose that U corresponds to phrase-markers of degree 2, as shown in (7):

(7) A degree 2 phrase-marker:



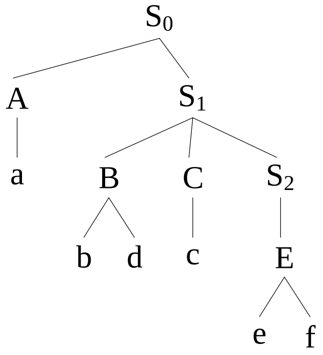
To understand how the proof worked, let us consider a concrete example. First, we will assume that rules apply in a strict cycle. That is, where S is a cyclic node, the most deeply embedded clause is the first domain of rule application, the next most deeply embedded S is the next domain and so forth. Let us take the case where the learner's transformational component raises the element d in the following tree from S$_2$ to S$_1$ in the following base structure:

(8) Base structure:



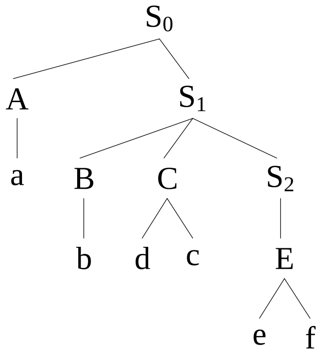
Suppose further that in the adult transformational component, d is adjoined as a right-daughter of B, as follows:

(9) Adult intermediate structure:



while in the child grammar, d is mistakenly adjoined as a left-daughter of C:

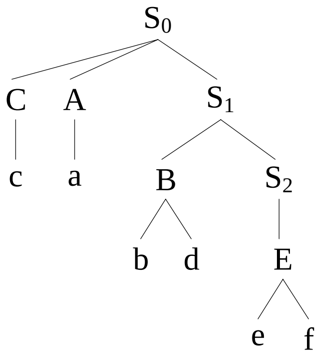
(10) Child intermediate structure:



Notice that both grammars generate the string *abdcef* although they assign different representations to the string. Thus, the error that the learner has made is not yet detectable.

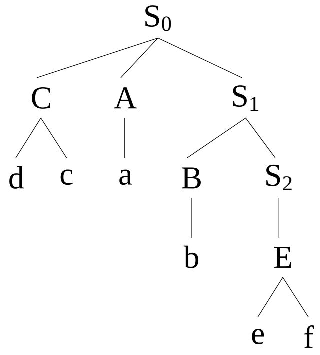
The BDE requires that the error reveal itself on a phrase-marker of the appropriate complexity; that is, the error must be made manifest on a degree 2 structure. Clearly, the error in (10) can be revealed on a degree 2 structure. Suppose that both the child and the adult transformational components contain a rule which raises the constituent C and makes it a left-daughter of S_0. The output of the adult transformational component will be as in (11):

(11) Adult output structure:



The output of the child's transformational component will be as in (12):

(12) Child output structure:



Notice that the adult's grammar has generated the string *cabdef* while the child's grammar has generated the string *dcabef*. Since the strings are not equal, the child's error has been revealed and the child must change its hypothesis.

Notice in particular the interplay between rule application and detectable errors in the above example. The movement rule applied within a restricted syntactic domain, the subtree dominated by S_1, in both the child grammar and the adult grammar. The child's error is revealed within the superordinate domain of rule application, the tree dominated by S_0. Wexler & Culicover present a number of constraints which serve to limit the application of grammatical processes to domains of complexity less than degree 2, that is, the constant specified by the constant U in the

BDE. They argue that degree 2 trees are the smallest that can contain raising rules plus a domain of application which will reveal the learner's errors. Notice that degree 2 trees correspond, in an interesting way, to the domain of classical subjacency as defined in Chomsky (1973):

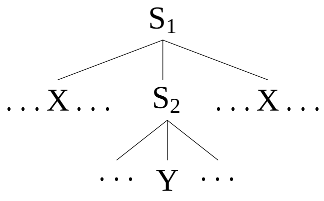
(13) *Subjacency*
No rule may relate X, Y in the structure:

$\ldots X \ldots [_{\alpha} \ldots [_{\beta} \ldots Y \ldots _{\beta}] \ldots _{\alpha}] \ldots X \ldots$

where α, $\beta \in \{$S,NP$\}$.

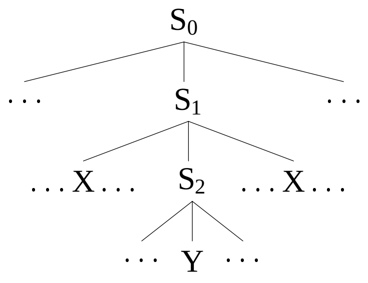
The definition in (13) restricts rule application to relating positions X and Y in trees of the following type, for example:

(14)



If we add one more cyclic domain to the above tree, we have a classic degree 2 structure of the same type as in (7):

(15)



Thus, a degree 2 tree was the least tree to contain a cyclic node dominating a domain for subjacency. If Wexler & Culicover were on the right track, then, notions like *cyclicity* and *subjacency* which were useful for syntactic analysis would have their ultimate grounding in a theory of learnability. In other words, an appropriately constrained and elaborated learning theory could potentially provide an explanation of why certain domains were relevant to syntactic operations.

2.2 Parameter Setting

Principles and parameters (*P&P*) theories seem to change the nature of the learning problem substantially. In brief, these theories make the claim that the set of (core) languages is finite. Core languages are generated by a fixed set of universal principles whose behavior is determined by a small set of parameters which can be set to a finite number of values. Parameters themselves are points of cross-linguistic variation and can perhaps most fruitfully be thought of as propositions which are true or false of a given language. The following, for example, are possible parameters:

(16) a. Verbs assign a θ-role to their right. {true, false}

b. Verbs assign Case to their right. {true, false}

c. The verb may assign accusative Case to the subject of an infinitive to which it does not assign a θ-role. {true, false}

d. Prepositions assign oblique Case. {true, false}

The propositions themselves should be made up of “linguistically natural” predicates like *X governs Y* or *X assigns accusative Case to Y* and so on. Following Clark (1990;1992), I will assume that grammars can be indexed according to the truth-values associated with the parameters. That is, let us associate ‘0’ with *false* and ‘1’ with *true* and establish a fixed order for the parameters. Taking the cases in (16), suppose we fix the order as $\langle$(16a), (16b), (16c), (16d)$\rangle$, the sequence “0001” would denote that language in which verbs assign Case and a θ-role to their left, cannot assign Case to a non-thematically dependent subject of an infinitive and in which prepositions assign an oblique Case.

The method outlined above provides an enumeration of the set of possible natural languages, once the set of parameters has been defined. Notice that the enumeration may contain gaps, since certain combinations of parameters setings may be ruled out either by Universal Grammar or for independent reasons. This simply means that the set of possible natural languages is smaller than our already finite enumeration. Assuming that there are *n* binary parameters, there will be 2^{n} possible core grammars. Since finite collections of recursively enumerable language are learnable, a learning function, φ, must exist which will identify this collection of languages (Osherson, Stob & Weinstein, 1986). Given that finite sets are learnable, some might be tempted that formal learning theory has little to contribute either to the question of natural language acquisition or to theoretical problems like constraints on cross-linguistic variation. On this view, one would better work directly on developmental psycholinguistics or on descriptive problems associated with comparative grammar than waste time on formal learning theory.

I will argue, contrary to the above view, that formal investigations into the computational problem of language learnability are more relevant than ever. As noted above, formal learning theory can contribute to the study of linguistic complexity and to grounding local relations like government. In particular, no constraint has been placed on the content of the parameters. Nothing in the syntactic theory rules out parameters like the following, for example:

(17) A special form of agreement is used in the main clause when the main verb governs an embedded wh-question, the specifier of which $\overline{\text{A}}$-binds a trace contained in the complement of a raising verb which occurs in a clause which is a complement to an infinitive non-factive verb.

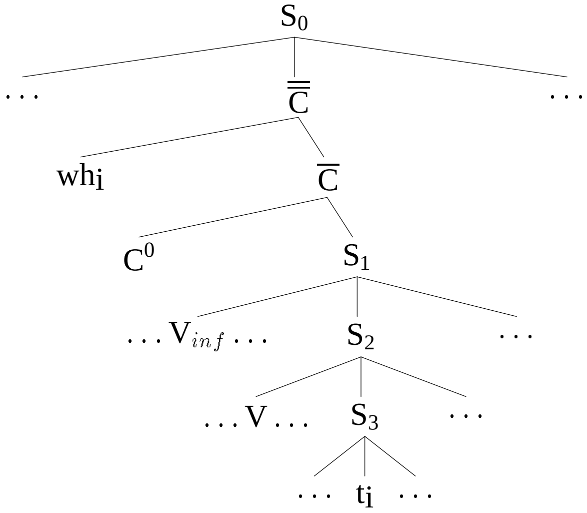
We now define the notion of parameter expression (from Clark, 1992):

(18) *Parameter Expression*
A sentence σ expresses a parameter p_i just in case a grammar must have p_i set to some definite value in order to assign a well-formed representation to σ.

Parameter expression entails that there is some empirical stake in the learner's setting a parameter p_i to some particular value v_j; there are sentences which the learner cannot represent so long as p_i is not set to v_j. Nothing in the definition requires that a sentence express only one parameter nor does it require that a sentence express a parameter unambiguously. For example, the order "Verb Object" might be the expression of whatever parameters govern the ordering of verbal complements relative to the head or it might express a parameter that allows rightward shifting over verbal complements across the head. In the long term, the learner will have to distinguish between these possibilities, although any one example may be highly ambiguous.

Notice, now, that the alleged parameter in (17) cannot be expressed in a degree 2 phrase-marker. Rather, it can only be expressed in a degree 3 phrase-marker as shown in (19):

(19)



The example in (17) is deliberately artificial. It would be extremely surprising to find a language with the property of having special verbal morphology in precisely the context described there. Notice, however, that the parameter is a boolean combination of otherwise linguistically natural predicates. Furthermore, it is at least imaginable that Universal Grammar could contain many parameters which could only be expressed in structures of at least the complexity shown in (19).

While linguists may have the intuition that languages do not vary in the way described in (17), the theory itself is silent on the existence of such complex parameters.

The problem here is one of complexity. Intuitively, the kind of data required to set (17) involves a degree of syntactic complexity that the learner is unlikely to encounter. In particular, setting the parameter (17) to the value that (19) expresses will require a great deal of time, since there is a low probability that the learner will encounter data of the type shown in (19). Thus, the amount of time required to converge on the target sequence of parameter settings will increase. We would expect, then, that there is some non-arbitrary relation between the syntactic complexity encoded by a parameter, the frequency with which the parameter is "expressed" in the input text and the amount of time required for the learner to fix the parameter to the correct setting. Parameters of low complexity can be "expressed" in structures of low complexity; these structures have a higher probability of occuring in the input text and, as a result, the learner encounters more structures which "express" the target parameter value. Hence, the learner will be able to fix the parameter to the correct value relatively quickly and the correct parameter value should be acquired fairly early in the developmental sequence.

In order to formalize these intuitions, we will assume for the present that at each step of time the learner outputs a hypothesis about the target. This hypothesis can be an integer which will serve as the index of the target grammar. Recall that we have introduced a binary representation for parameters which worked to enumerate the possible grammatical systems. The learner can "unpack" the index of the hypothesized system and the result is a sequence, $\langle x_1, x_2, \ldots, x_n \rangle$, of n parameter values. Following Clark (1992) we will assume the following definition of *convergence*, where L is a set of possible learning systems:[3]

(20) *Convergence*
A learning system, $\mathscr{L} \in L$, converges to a target p_a (a sequence of parameter values)just in case:

$$\lim_{T \to \infty} \overline{\mu}(\mathscr{L}, T) \;=\; 1$$

where $\overline{\mu}(\mathscr{L}, T)$ is a measure of average system performance over time.

We can define $\overline{\mu}$ as:

$$\overline{\mu}(\mathscr{L}, T) \;=\; \frac{1}{T} \sum_{t=1}^{T} \mu(\mathscr{L}(t))$$

The idea is that the function μ is an evaluation metric which can measure the distance between the learner's current best hypothesis and the target. A score of 1 implies that the learner's current best hypothesis is the same as the target. For example, μ could act like a multiple choice test, in that it simply sums the number of correctly set parameters and divides that result by the number of parameters. The function $\overline{\mu}$ is just the average results over time.

[3] I will have occasion to revise this definition below. For the moment, however, it will serve as a baseline.

Clark (1992) argued that, in order to converge in the above sense, a number of properties would have to hold of the parametric system, the learner and the input text. First, the parameter would have to be *expressable* in the sense defined in (21):

(21) *Parameter Expressability*
For all parameters $\mathrm{x_i}$ in a system of parameters $\mathscr{P}$ and for each possible value $\mathrm{v_j}$ of $\mathrm{x_i}$, there must exist a datum $\mathrm{d_k}$ in the input text such that a syntactic analysis τ of $\mathrm{d_k}$ express $\mathrm{v_j}$.

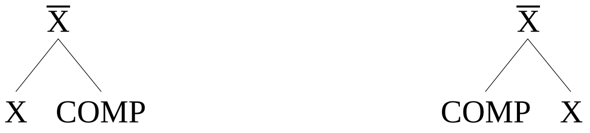
That is, the learner must be able to detect the effects a parameter setting has on the output of the grammar somewhere in the input text. If this were not so, if no input sentence showed the effects of a particular parameter set to some particular value, then the learner would have no stake in setting the parameter to that value. Its output would be, for all practical purposes, indistinguishable in behavior from the target sequence of parameter settings. A parameter value which is inexpressable should be unsettable.

Parameter expressability alone is not, however, a strong enough constraint on the system. A parameter which is only expressed once in the input stream should be indistinguishable from noise as far as the learner is concerned. For example, if a parameter were expressed in the input text with less frequency then a particular type of speech error, then the learner should find it difficult to distinguish between that parameter setting and a speech error; since we don't want parameter setting to take place on the basis of a speech error, this parameter would be unsettable. This suggests that, given a particular parameter $\mathrm{p_m}$ which is to be set to a particular value $\mathrm{v_n}$, there is some basic threshold frequency $\phi_{(n,m)}$ that must be met in order to set $\mathrm{p_m}$ to $\mathrm{v_n}$. Letting $f_{(n,m)}(\mathrm{s_i})$ be the actual frequency perceived in the input text, we can formalize this intuition by:

(22) *Frequency of Parameter Expression*
Given an input text $\mathrm{s_i}$, a target parameter sequence $\mathrm{p_a}$, and a learning system $\mathscr{L}$, $\lim_{T\to\infty} \overline{\mu}(\mathscr{L}, T) = 1$ if, for all parameter values v in positions m in the target $\mathrm{p_a}$, $f_{(v,m)}(\mathrm{s_i}) \geq \phi_{(v,m)}(\mathrm{p_a})$.

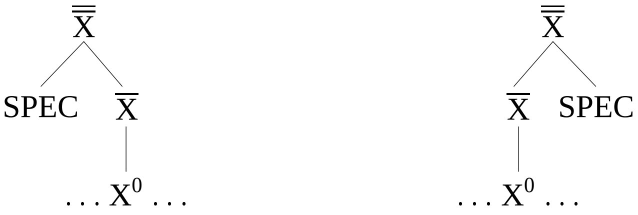
Here, we intend the threshold represented by $\phi_{(v,m)}(\mathrm{p_a})$ to be the number of times the learner must encounter a construction which expresses the value v of parameter m in the input text in order to correctly set the parameter. Notice that parameters that are expressed in "simple" structures are likely to be expressed with fairly high frequency. Consider, for example, those parameters which express the relative order of a head and its complements. The minimal tree on which these parameters can be expressed is quite simple:

(23)



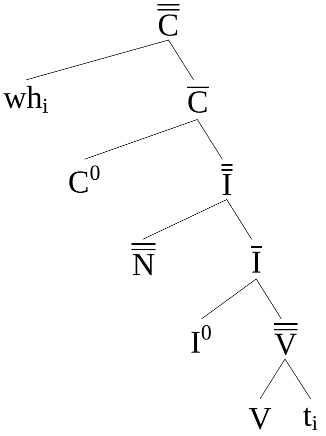
The subtrees in (23) are frequent in parsing the input text, so that the learner should quickly master the relevant parameters; these parameters will pass threshold relatively early. The minimal tree upon which specifier-head relations can be expressed is slightly more complex, as shown in (24):

(24)



The minimal tree which would exhibit non-string vacuous wh-movement is still more complex:

(25)



The intuition underlying both (21) and (22) is that parameters which are expressed by small, "simple" structures, like head-complement and specifier-head order in (23) and (24), will be expressed with high frequency, since these structures are likely to be embedded in larger structures or simply occur on their own. Thus, the learner is likely to set these parameters rapidly since it will have been exposed to the effects of the target parameter setting at a level which exceeds threshold fairly early on. Parameters which are expressed in more complex structures, like non-vacuous syntactic application of "Move α" as in (25) will be expressed less frequently since these are not as likely to be embedded within larger structures or occur on their own.[4] More complex parameters will achieve threshold frequency later than simple ones.

The constraints in (21) and (22), however, do not capture the intuition that ease of acquisition is related to complexity. Clark (1992) attempts to capture this by stipulating the following:

[4] I assume here, as is standard in the acquisition literature, that the learner's input is made up, for the most part, of simple grammatical sentences.

(26) *Boundedness of Parameter Expression*
For all parameter values v_{i} in a system of parameters $\mathscr{P}$, there exists a syntactic construction τ_{j} that express $\mathrm{v_i}$ where the complexity $\mathrm{C}(\tau_{\mathrm{j}})$ is less than or equal to some constant U.

Notice that (26) is related to the BDE (see (6) on page 6) in that both attempt to place an upper bound on the complexity of the input stream required for convergence. The idea is to select a constant U that will guarantee that there is a relatively simple structure which expresses any parameter in the system $\mathscr{P}$. If U is sufficiently small, then there is a good chance that the frequency of expression for each parameter in the input will exceed threshold. Thus, we might define the notion of a *minimal text*:

(27) Let σ_{min} be a set of sentences drawn from the language $\mathrm{L_i}$ such that, when parsed according to the grammar for $\mathrm{L_i}$, all grammatically admissible trees τ_{j} of complexity $\mathrm{C}(\tau_{\mathrm{j}}) \leq U$ are exemplified once in σ_{min}; σ_{min} is a minimal text for $\mathrm{L_i}$.

The idea is that a minimal text contains one example of each type of grammatical construction of complexity less than the constant U given by (26). Notice that a minimal text is finite, since arbitrary embeddings are ruled out by the complexity bound U. Given a minimal text, we can define a *fair text* in the following way:

(28) Let r be the maximum threshold frequency, $\phi_{(v,m)}(\mathrm{p_a})$, for all parameters, m, and values, v, in some system of parameters $\mathrm{p_a}$ and let σ_{i} be a minimal text for a language $\mathrm{L_j}$. The text that results from concatenating σ_{i} to itself r times is a fair text for $\mathrm{L_j}$.

In other words, a fair text can be constructed from a minimal text in such a way that each construction is repeated enough times to guarantee that the thresholds for parameter setting is exceeded. Thus, no information is withheld from the learner in a fair text. We can define the learnability property as follows (note that this restates (22) in terms of a fair text):

(29) A system of parameters $\mathrm{p_a}$ is learnable if and only if there exists a learner φ such that for every language $\mathrm{L_i}$ determined by $\mathrm{p_a}$ and every fair text σ_{j} for $\mathrm{L_i}$, $\varphi(\sigma_{\mathrm{j}})$ converges to $\mathrm{L_i}$.

So a system has the learnability property just in case there is some learner that learns the languages determined by that system from any arbitrarily selected fair text.

The complexity bound U established for the constraint in (26) should serve to limit the complexity of the input text; in particular, given U we can establish an upper bound on both the sample size and the time required by the learner. This is so since U established a limit on the size of the minimal texts. As U grows, the minimal texts for each language will also grow. But the size of a fair text is just $|\sigma_{\mathrm{min}}|^{r+1}$, so the fair texts will also grow. Assuming, as seems reasonable, that the time to converge is a function of the size of the text φ learns on, then the time-complexity of learning is also a function of U. Recall that U is a bound on parameter expression; no parameter can contain more information than can be expressed by a phrase marker of complexity at most U. Thus, U also limits the information that can be encoded by any one parameter. Finally, since cross-linguistic variation is determined by the different parameter values, U also limits the amount of variation that is possible across languages.

The constant U is of some linguistic interest. The problem now is to determine the value for U since doing so would greatly constrain the linguistic theory. Clark (1992) does not attempt to assign a specific value to U. Recent proposals by Morgan (1986), Lightfoot (1989;1991) and Rizzi (1989) can be taken as specific empirical proposals for the value of this constant. For any proposed value for U, we can ask whether some smaller value might not be adequate. The best theory is one which establishes a firm lower bound on the value of U such that a minimal text generated from a smaller value of the constant would result in a fair text that cannot correspond to an empirically viable input text; such a text could not, in principle, allow the learner to fix linguistically plausible parameters since it would be too simple. In the sections that follow we will develop such a theory.

3 Formal Preliminaries

In this section, I will discuss some background material for the mathematical analysis of parameters. The approach can be summarized as follows: In order to place an upper bound on the information content of parameters in general, I will consider the behavior of the system that is the output of parameter setting. This follows from parameter expressability (see (21) in section 2.2); in particular, the content of parameters bears a systematic relationship to the structural descriptions that the grammar determined by the parameters admits. I will attempt to characterize the complexity of these objects by developing a standardized description language for phrase markers (section 3.2). Since this project is rather dry, I will first turn to the formal theory of theory of descriptive complexity and algorithmic information theory (section 3.1) in order to give a general sense of the complexity theory and motivate the work in section 3.2. Finally, in section 4 I will turn to particular applications of this formal theory to the theory of parameters and learnability.

3.1 Descriptive complexity and algorithmic information theory

We are interested in the inherent descriptive complexity of an object. Is there some general method for calculating the amount of information associated with an object, whether the object is a phrase-marker, a linguistic derivation, a strand of DNA or a lump of coal? Suppose, for example, that we wish to transmit a description of an object to some receiver; the complexity of the object should correspond (roughly) to the effort we must go through in order to encode and transmit the description. The best measure of effort available is just the length of the description since it is likely to take less effort to transmit a short description than a long description.

Consider, for example, the following three strings:

(30) a. 011011011011011011011011011011011011011011011

b. 0110101000001001111001100110011111110011101111001100100100001-000

c. 100000101100111011100110010111110000010010100

The string in (30a) appears to have a good deal of structure. Indeed, our description might be the program “Print the sequence *011* fifteen times”, which would allow the receiver to completely reconstruct the string. Assuming that the print instruction can be encoded in two bits, the repeat instruction in two bits then the length of the description would be $2 + 2 + 4 + 3 = 11$ bits (that is, the bit length of the two instruction plus 4 for encoding 15 plus the length of “011”) which is less than 45 bits (the length of the string).

Compare the string in (30a) with the one in (30b). The string in (30b) appears to have much less structure than the string in (30a); indeed, the string passes many of the tests for randomness (Cover & Thomas, 1991). In fact, the string in (30b) is the binary expansion of $\sqrt{2} - 1$. Thus, the transmitter could encode and transmit a set of instructions specifying the receiver to compute $\sqrt{2} - 1$ and again, transmit a message that is less complex than the original string. Thus, although the sequence appears complex at first glance, there is structure present that the transmitter can exploit. The example raises the interesting problem of how to decide when a given string is random; in particular, effective tests for randomness (proportion of sequences like “00”, “10”, “01” and “11” in the string, and so forth) are not guaranteed to give the correct answer. Thus, the randomness of a string may not be decidable (see Li & Vitanyi, 1993 for some discussion) which brings up the interesting relationship between Kolmogorov complexity and Gödel’s incompleteness theorem.

Consider, finally, the string in (30c). This string has little to no structure, having been generated by a series of coin tosses. There would seem to be no description of (30c) that is shorter than (30c) itself. Thus, the transmitter has little choice but to transmit all 45 bits of (30c). Notice the connection, made informally here, between the complexity of the description of an object, computation, and *randomness*. This is an important intuition underlying descriptive complexity and we will rely heavily on this intuition throughout. Objects with structure should have short descriptions because the description can rely on the structure of the object to tell the receiver how to compute the description. A random object has no discernible structure for the transmitter to exploit so the transmitter has no choice but to transmit the entire description. Thus, if an object is genuinely random, its description should be uncompressible. Languages have a great deal of structure, so we would expect them to have a relatively low descriptive complexity; the question of their actual complexity is one of some theoretical interest.

The intuition underlying the above discussion is that, given a description language D, the complexity of an object should correspond to the length of the shortest description in D. In accord with the discussion above, the description language should be powerful enough to describe computations. In other words, D can be thought of as a programming language. In particular, we will take as given a universal Turing machine $\mathscr{U}$ and that D is a programming language for $\mathscr{U}$.[5] Suppose that x is a program written in the language D. We will let $\mathscr{U}(x)$ stand for the process of running the universal Turing machine $\mathscr{U}$ on x. For present purposes, we will conflate the description of an object with the object itself; thus, if x is a description of the object y in D then we will write $y = \mathscr{U}(x)$, even though the output of $\mathscr{U}(x)$ is a description of y and not necessarily y itself. Given this formalism, we give the following definition for *Kolmogorov complexity*:

(31) The *Kolmogorov complexity* $K_{\mathscr{U}}(x)$ of a string x with respect to a universal computer $\mathscr{U}$ is defined as:

$$K_{\mathscr{U}}(x) \;=\; \min_{p \,:\, \mathscr{U}(p)=x} l(p)$$

where $l(p)$ denotes the length of the program p.

In other words, the Kolmogorov complexity of an object x is the length of the shortest program, p, for $\mathscr{U}$ that allows $\mathscr{U}$ to compute a description of x. It should be emphasized that x itself can be anything we can describe. For example, we might estimate the complexity of Marcel Duchamp's "Nude Descending a Staircase" by scanning the picture and performing our calculations on the resulting binary file.

It might seem as though the above definition of complexity is of only limited interest, since it is defined relative to a particular universal Turing machine, $\mathscr{U}$. In fact, Kolmogorov complexity is machine independent as shown by the following theorem (see Cover & Thomas, 1991, for a complete proof):

(32) *Universality of Kolmogorov Complexity*
If $\mathscr{U}$ is a universal computer, then for any other computer $\mathscr{A}$,

$$K_{\mathscr{U}}(x) \;\leq\; K_{\mathscr{A}}(x) + c_{\mathscr{A}}$$

for all strings $x \in \{0,1\}^*$, where the constant $c_{\mathscr{A}}$ does not depend on x.

Briefly, suppose that $\mathscr{A}$ is a Turing machine and that $K_{\mathscr{A}}(x)$ is the complexity of x relative to $\mathscr{A}$. Since $\mathscr{U}$ is a universal Turing machine, it can simulate any other Turing machine. In particular, it can simulate $\mathscr{A}$. Let $c_{\mathscr{A}}$ be the Kolmogorov complexity of the program, y that $\mathscr{U}$ uses to simulate $\mathscr{A}$. We can compute a description of x on machine $\mathscr{U}$ using the program we used to compute x on machine $\mathscr{A}$ plus y, the simulation program. Thus, the Kolmgorov complexity of

[5] A universal Turing machine is one which can simulate the behavior of any other Turing machine given a program and an input. The reader is invited to consult Papadimitriou (1994) for an excellent introduction to Turing machines and complexity.

x relative to $\mathscr{U}$ is bounded from above by the Kolmogorov complexity of x relative to machine $\mathscr{A}$ plus the Kolmogorov complexity of y. The absolute Kolmogorov complexity of x relative to $\mathscr{U}$ may well be less than this amount, but it can never exceed $K_{\mathscr{A}}(x) + c_{\mathscr{A}}$.

In other words, our complexity calculations are independent of the architecture of the universal computer $\mathscr{U}$ we have chosen; any other choice would lead to a variation in the complexity bounded by a constant term and, thus, well within the same order of magnitude of our estimate of complexity. Given the result in (32), we can drop reference to the particular machine we use to run the programs on.

Having seen that Kolmogorov complexity is invariant up to a constant across computing machines, let us turn, briefly, to some general results that bound the complexity of descriptions. Let us first define *conditional Kolmogorov complexity* as in (33):

(33) *Conditional Kolmogorov Complexity*
If $\mathscr{U}$ is a universal computer then the conditional Kolmogorov complexity of a string of known length x is:

$$K_{\mathscr{U}}(x|l(x)) = \min_{p\,:\,\mathscr{U}(p,l(x))=x} l(p)$$

The definition in (33) is the shortest description length if $\mathscr{U}$ has the length of x made available to it. From the above definition, it is a fairly routine matter to prove the following:

(34) *Bound on conditional Kolmogorov complexity*
$K(x|l(x)) \;\leq\; l(x) + c$

In this case, the length of the string x is known before hand. A trivial program for describing x would, therefore, be merely “Print the following $l(x)$ bits: $x_1 x_2 \ldots x_{l(x)}$”. That is we simply transmit the description along with a print instruction. The length of the above program is therefore $l(x)$ plus the print instruction, c. Hence, $K(x|l(x))$ is bounded from above by $l(x) + c$. This means that the conditional complexity of x is less than the length of the sequence x. Notice that the conditional complexity of x could be far less than $l(x)$; we have guaranteed that the complexity of an object will never exceed its own length.

What happens if we don’t know the length of x? In this case, the end of the description of x will have to be signalled or computed somehow. This will add to complexity of the description, but by a bounded amount. Thus, the following is a theorem (see Cover & Thomas, 1991 for a formal proof):

(35) *Upper bound on Kolmogorov complexity*
$K(x) \;\leq\; K(x|l(x)) + 2\log l(x) + c$

The addition term, $2\log l(x)$, comes from the punctuation scheme that signals the end of x.

We have seen so far that we can estimate the inherent descriptional complexity of an object by the expedient of using programs which compute a description of the object and that this metric is universal. Once a program that computes a description of the object has been discovered, we can use it as an upper bound on the actual Kolmogorov complexity of that object. Can we ever discover the actual Kolmogorov complexity of the object? It is perhaps surprising to realize that we can't. Recall that we are measuring complexity relative to programs for a universal Turing machine, $\mathscr{U}$. Suppose that we were to enumerate the possible programs in lexicographic order (starting from the shortest program and proceeding in alphabetical order). We could then run each program on $\mathscr{U}$. Suppose that $\mathscr{U}(p_i) = y$ (that is, $\mathscr{U}$ halts on p_i, yielding a description of y); we can enter $l(p_i)$ as an estimate of $K(y)$. But there may be programs shorter than p_i such that $\mathscr{U}$ has yet to halt on these programs. In particular, suppose that there is a program p_j such that $l(p_j) < l(p_i)$ and $\mathscr{U}(p_j)$ has not yet halted. It could be that $\mathscr{U}(p_j)$ will eventually halt with $\mathscr{U}(p_j) = y$. If so, then $l(p_j)$ is a better estimate of $K(y)$ than $l(p_i)$. If we could know that $\mathscr{U}(p_j) = y$ then we could find the actual Kolmogorov complexity of y. But this entails that we know that $\mathscr{U}(p_j)$ halts, which in turn entails that we have a solution to the Halting Problem. Since the Halting Problem is unsolvable, we cannot guarantee that we have arrived at the true Kolmogorov complexity of an object once we have a program which computes its description. In other words:

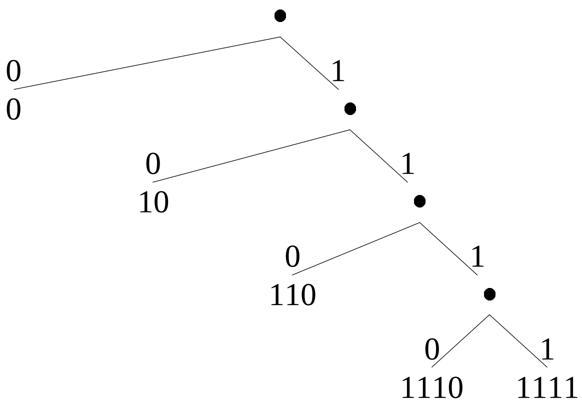
(36) An upper bound on the Kolmogorov complexity of an object can be found, but a lower bound cannot.

So far, we have allowed any program for the universal Turing machine to count as a possible description. A number of interesting results hold if we require that the programs be prefix-free. We can take the programs for $\mathscr{U}$ to be codes which map from a description of the object onto a binary encoding of the description. A *prefix code* can be defined as follows:

(37) A code is a prefix or instantaneous code if no codeword is a prefix of any other codeword.

In other words, a prefix code can easily be decoded without reference to possible continuations of the codeword precisely because the end of the codeword can be immediately detected; it is a "self-punctuating" code. Let us consider an example of a prefix code to illustrate the principle. Suppose we need to transmit the names and order of entry of a horse race. The five horses are named Rimbaud, Oliver, Bill, Indigo, and Newton. We can generate the following five codewords to create a prefix code for the five horses:

(38)



Notice how the above tree is constructed. Only the leaf nodes are labeled; each leaf is labeled by a codeword. Left-branches are associated with a “0” while right-branches are associated with a “1”. Taking a left-branch results outputs a codeword whose end is signalled by “0”. Only one codeword, “1111”, lacks this property. Its end is signaled by its length. Thus, the code is self-punctuating. Let us associate the horse with codewords via the encoding function E : HORSES $\longrightarrow$ CODEWORDS:

(39) $E(\text{Rimbaud}) = 0$
$E(\text{Oliver}) = 10$
$E(\text{Bill}) = 110$
$E(\text{Indigo}) = 1110$
$E(\text{Newton}) = 1111$

Suppose that the sequence “11111100101110” is transmitted over the channel. This sequence can be unambiguously decomposed into the codewords “1111” followed by “110” followed by “0” followed by “10” followed by “1110”. Adopting the convention that order in the sequence corresponds to order across the finish line, then we can interpret the string as indicating that Newton was first, followed by Bill in second place, Rimbaud in third, Oliver in fourth and Indigo in last place. A little experimentation should show that any sequence of the codewords in (39) can be unambiguously segmented.

Notice that the code in (39) is not necessarily optimal. In our toy example, this doesn’t matter since we had to report on all the horses in the race. Suppose, however, that we needed to report only the winner of the race. In order to optimize our resources, we would want to assign the shortest codeword to the most likely winner, and so on. Notice the association between shortness and probability and recall the discussion above concerning randomness and description length; the association between description length and probability apparent here. Suppose that we have the following probabilities of winning:

(40) Pr(X = Rimbaud) $= \frac{1}{2}$
Pr(X = Oliver) $= \frac{1}{4}$
Pr(X = Bill) $= \frac{1}{8}$
Pr(X = Indigo) $= \frac{1}{16}$
Pr(X = Newton) $= \frac{1}{16}$

Now the code given in (39) is optimal. The most likely winner, Rimbaud, is associated with the shortest code word since E(Rimbaud) $=$ 0 which is of length 1. Analogously, the least likely winners, Indigo and Newton, are both associated with codewords of length 4.

In order to firm up the relationship between codes and probability, let us first note the existence of the so-called *Kraft inequality* (see Cover & Thomas, 1991, for proof):

(41) *Kraft Inequality*
For any prefix code over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality:

$$\sum_i D^{-l_i} \leq 1$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

Applying the Kraft inequality to our toy code in (39) we see that each codeword length associated with a horse is exactly the probability that the horse will win according to the distribution in (40). Thus, there is an interesting relationship between probabilities and codeword lengths in an optimal prefix code. This relationship can be best understood by considering the *entropy* of the random variable ranging over the things we wish to encode. Entropy is a measure of the degree of uncertainty of a random variable. Let X be a random variable ranging over an alphabet $\mathscr{X}$ with probability mass function $p(x) =$ Pr$\{X = x\},\ x \in \mathscr{X}$. Then:

(42) The *entropy* $H(X)$ of a discrete random variable X is defined by:

$$H(X) = -\sum_{x \in \mathscr{X}} p(x) \log p(x).$$

In the discrete case, entropy measures the expected number of bits required to report what value the random variable X has taken on in an experiment. Note that in our horse race example in (40), the entropy is: $-(\frac{1}{2}\log\frac{1}{2} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{8}\log\frac{1}{8} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{16}\log\frac{1}{16}) = 1.875$ bits.

Naturally, there is a tight relationship between entropy and optimum codes. Intuitively, the best code is one which is just long enough to transmit a message and no longer. If a code is too short (below the number of bits required by entropy), then information is lost. If it is too long, then there are redundancies (and, hence, wasted effort) in the system. In fact, the following is a theorem (see Cover & Thomas, chapter 5, for a proof and discussion):[6]

(43) Let $l_1, l_2, \ldots l_m$ be the optimal codeword lengths for a source distribution **p** and a D-ary alphabet and let L be the associated expected length of the optimal code ($L = \sum p_i l_i$). Then:

$$H_D(X) \leq L < H_D(X) + 1.$$

The theorem in (43) just says that entropy of the source provides a bound on the length of the optimum codewords for encoding that source. Indeed, many data compression schemes rely on the relationship between entropy, probability and prefix codes to approach optimum compression. Returning to the code in (39) and the probability distribution in (40) we see that $L = \sum p_i l_i = ((\frac{1}{2} \times 1) + (\frac{1}{4} \times 2) + (\frac{1}{8} \times 3) + (\frac{1}{16} \times 4) + (\frac{1}{16} \times 4)) = 1.875$ which is the same as the entropy of the distribution; thus, the code given in (39) is optimal relative to the probability distribution in (40).

Let us return, now, to Kolmogorov complexity proper. Given that Kolmogorov complexity is concerned with optimum description length, it should come as no surprise that there is an intimate relationship between the theory of optimum codes (that is, data compression) and Kolmogorov complexity. Presumably, the shortest description of an object is already in its most compressed form (otherwise, it wouldn't be the shortest description). Let us assume that we encoded the programs for our universal Turing machine $\mathscr{U}$ using a prefix code. The following theorem can be seen as the complexity analog of the Kraft inequality in (41):

(44) For any computer $\mathscr{U}$:

$$\sum_{p\,:\,\mathscr{U}(p)\text{ halts}} 2^{-l(p)} \;\leq\; 1.$$

In fact, from (44) we see that the halting programs for our machine $\mathscr{U}$ must form a prefix code.

Given the relationship between optimal codeword lengths and entropy in (43) and the fact, from (44), that the halting programs form a prefix code, we would expect that entropy of a random variable X ranging over an alphabet $\mathscr{X}$ should provide a useful bound on the Kolmogorov complexity of objects described by $\mathscr{X}$. This is indeed the case, as the following rather imposing looking theorem states:

[6] Note that $H_D(X)$ is the entropy of X calculated with a base D log.

(45) *The relationship between Kolmogorov complexity and entropy*
Let the stochastic process $\{X_i\}$ be drawn in an independent identically distributed fashion according to the probability mass function $f(x)$, $x \in \mathscr{X}$, where $\mathscr{X}$ is a finite alphabet. Let $f(x^n) = \prod_{i=1}^{n} f(x_i)$. Then there exists a constant c such that

$$H(X) \leq \frac{1}{n} \sum_{x^n} f(x^n) K(x^n|n) \leq H(X) \ + \ \frac{|\mathscr{X}| \log n}{n} + \frac{c}{n}$$

for all n. Thus

$$E\frac{1}{n}K(X^n|n) \longrightarrow H(X).$$

That is, the average expected Kolmogorov complexity of length n descriptions should approach entropy as sample size grows.

We have so far noted a relationship between Kolmogorov complexity, prefix codes and entropy. The relationship is both surprising and deeply suggestive. Recall that Kolmogorov complexity is defined relative to symbolic objects, namely Turing machine programs; we can, in fact, think of these programs as programs for a physical computer if we like. Entropy is a statistical notion, a measure of the amount of uncertainty in a system. Nevertheless, as (45) shows, there is a systematic relationship between entropy and Kolmogorov complexity.

In order to firm up this intuition, consider the following thought experiment. Suppose we started feeding a computer randomly generated programs. Sticking to the binary programming language we have been using for Turing machines, we might generate these programs by tossing a coin and using “1” for *heads* and “0” for *tails*. In general, these programs will crash (halt with no output), but every once in a while one of them will halt with a sensible output. Thus, the following quantity is well-defined:

(46) The *universal probability* of a string x is

$$P_{\mathscr{U}}(x) \ = \sum_{p\,:\,\mathscr{U}(p)=x} 2^{-l(p)} \ = \ \mathrm{Pr}(\mathscr{U}(p) = x),$$

which is the probability that a program randomly drawn as a sequence of fair coin tosses $p_1, p_2, \ldots$ will print out the string x.

Notice the similarity between the definition in (46) and the Kraft inequality in (41). Given the relationship between optimal prefix codes and probability, we would expect that there should be a tight relationship between universal probability and Kolmogorov complexity. Indeed, the following is a theorem (see Cover & Thomas, 1991):

(47) $P_{\mathscr{U}}(x) \ \approx \ 2^{-K(x)}$

That is, we can approximate the universal probability of x by using its Kolmogorov complexity. Intuitively, this is because the high probability things are encoded by short strings, as we have seen. Thus, simple objects are much more likely than complex ones. Anticipating later discussion somewhat, suppose that for each parameter we take the Kolmogorov complexity of the smallest structure which expresses it. Those parameters associated with low complexity should be more likely to be expressed in the input text, since they will have relatively high universal probability by (47). By the argumentation in section 2.2, we would expect that parameters with low Kolmogorov complexity to be set relatively early. This follows from the interaction between the Frequency of Parameter Expression (see (22) on page 13) and Boundedness of Parameter Expression (see (26) on page 15). We initially formalized this via minimal texts and fair texts. The intuition was that the simpler the parameter was, the more frequently it would be expressed and, therefore, the more likely it was to be set correctly. Notice, though, that the properties of these texts follow directly from the complexity theory outlined here. In particular, if we take the constant U in the definition of Boundedness of Parameter Expression to be a function of the (average) Kolmogorov complexity of the parameters in the system, then the frequency of expression of the parameters will follow from the Kolmogorov universal statistic. Thus, the theory of Kolmogorov complexity, and its association with universal probability, formalizes the informal argument made in section 2.2.

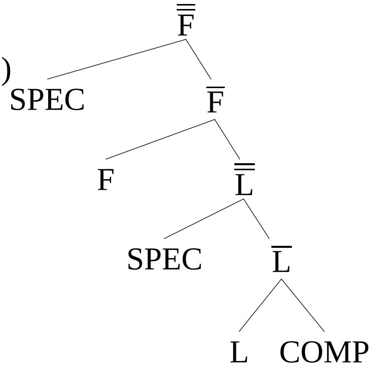
3.2 The binary encoding of phrase-markers

In this section, I will develop a simple method for encoding trees as string of binary numbers. The intent is to create a representation that is appropriate for the complexity theory that was discussed in section 3.1. We need an encoding system that can be used to represent grammatical representations, parameters and input texts. The general line of attack will be to represent all of the above data structures using phrase markers and to encode the phrase markers as bit strings. I will argue for this move in section 4 below. For the moment, I will simply develop a general encoding scheme.

The encoding method that I will discuss here is based on the method developed in Clark (1993) and can be thought of as a programming language for syntactic representations. It should be noted, however, that the method described here is not intended to be optimal, nor is it intended to be completely general. In particular, the scheme given here is not a prefix code for tree structures. Instead, it is intended to be a worked-out example of a binary encoding scheme for phrase-markers. Notice, however, that once the phrase-marker has been translated to binary form, the result can be compressed by an appropriate compression algorithm (eg., Huffman coding) and won't vary too much from the optimal representation.

In order to simplify the exposition, I will assume that phrase-markers are binary branching (see Kayne, 1984;1993) and that lexical categories tend to be associated with functional categories. Thus, the following is a minimal phrasal skeleton, where F is a functional category and L is a lexical category:

(48)



In the above tree, the functional category F corresponds to (inflectional) morphological properties associated with the lexical category L or it corresponds to a closed class element (complementizers, for example). For simplicity, I will assume that all lexical categories can (and must) be associated with a functional category; thus, the tree in (48) is minimal. I will put aside the case where one functional category occurs as a complement to another one, although nothing hinges on this assumption.

The categories that F and L can range over are:[7]

(49) N(oun), V(erb), Adj(ective), Adposition (P), Adv(erb), C(omplementizer), I(nflection), D(eterminer), Conj(unction), Int(ensifier)

The inventory in (49) is impoverished, but it will serve to illustrate the basic method. Extending the inventory to other categories (degree words, for example) will not substantially alter the size of the encodings and it is this that we are most interested here. For the purposes of our encoding, it is crucial that the number of distinct grammatical categories is finite, a reasonable assumption by anyone's account.

Our ultimate goal is to develop an effective procedure that, when given an arbitrary binary branching tree whose node labels are drawn from the inventory in (49), will produce a binary encoding of that tree. We do not require that the binary encoding encode only well-formed (grammatically correct) phrase-markers; it is the job of the grammar to distinguish well-formed from ill-formed structures. The binary encoding need only represent formally correct trees. That is, the tree must have a unique root and the branches are not permitted to cross. In addition, The binary encoding must preserve the following information from the original tree:

[7] For simplicity, I will ignore recent proposals to divide categories like I into tense, aspect, agreement and other categories.

(50) a. The node label (grammatical category) of each node must be represented.

b. Morphological information must be preserved in the binary encoding.

c. Dominance relations must be encoded in such a way as to be effectively recoverable. We will encode immediate dominance.

d. Linear precedence between nodes sharing the same mother must be represented. We will crucially appeal to the binary branching convention here. A more general representation scheme can be developed which does not presuppose binary branching, however.

e. Indexing of nodes should be preserved. Given that the indexation of nodes in a given a phrase-marker is correctly encoded, coindexation relations can be derived from the encoding.

Given the finite list of categories in (49) we can adopt the enumeration given in (51). Any enumeration with do for our purposes, so there is nothing of special theoretical interest in (51) apart from its existence:

(51)

N =	0000	C =	0101
V =	0001	I =	0110
Adj =	0010	D =	0111
P =	0011	Conj =	1000
Adv =	0100	Int =	1001

As (51) shows, we can easily establish a binary encoding for the grammatical categories.

Morphological features may seem somewhat more complex. Nevertheless, given a finite inventory of n binary features, we will have 2^n possible feature matrices. Thus, we can again sort to a brute force enumeration as we did with the grammatical categories. Let us suppose that each node may be marked with the following features:

(52) ±sing
±1pers
±2pers
±fem
±past
±acc

As above, the inventory in (52) is impoverished; naturally, the set could easily be augmented to include further features. Here, "sing" represents the singular/plural distinction, "1pers" and "2pers" are first and second person respectively; thus, third person is represent as [−1pers, −2pers]. The feature "fem" is for feminine gender; thus, masculine is represented as [−fem]. The feature "acc" encodes case; thus, a [−acc] element will be interpreted as nominative while [+acc] is accusative. A more complete system would encode other cases. Finally, the feature "past" encodes tense. Clearly, we would have to extend the above features to include participles and so forth. There are some redundancies and coocurrence restrictions in even this small system which we will put aside for the moment; the grammar itself will have to express these restrictions, not our simple encoding procedure.

Notice that the five features in (52) yield 64 possible feature matrices. We need a string of five binary digits to encode all the possibilities. Arranging the features in the order ⟨sing, 1pers, 2pers, fem, past, acc⟩, we can simply represent the "+" value as "1" and the "−" value as "0". Thus, the sequence "010100" would stand in for the matrix:

(53) $$\begin{bmatrix} -\text{sing} & +\text{1pers} \\ -\text{2pers} & +\text{fem} \\ -\text{past} & -\text{acc} \end{bmatrix}$$

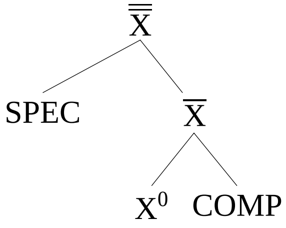
That is, the binary number decodes as the feature array for an element that is first-person singular and feminine. Thus, a string of 6 binary digits is sufficient to encode all the feature matrices possible in this system.

For the sake of generality in the encoding scheme, let us assume that the features in (52) can be associated with each node in the phrase-marker. As a result, nouns will be marked as [±past], although the grammar may well declare that nouns are undefined for past tense marking. Similarly, adverbs and adjectives will bear nominal morphological features. No real harm is done by this. Indeed, verbs and adjectives can agree with nominal elements, although the agreement relationship is usually seen as mediated by a functional head. The intent here is to keep a constant block length in the code associated with each node in the phrase-marker and thus simplify the procedure for translating between phrase-markers and binary encodings. This assumption can be dispensed with if the binary encoding for each node can encode information about its own length; in that case, the assumption about fixed block length could be dispensed with.

We now have a method for encoding the grammatical category as well as a method for encoding feature matrices; if we concatenate these two encoding schemes, we can encode the grammatical category and feature matrix of nodes in a phrase marker. Let g represent the grammatical category of a node x as enumerated in (51) and h represent the encoding of x's feature matrix; then $g(x)^\frown h(x)$ will be a string of nine binary digits the first 4 of which encode grammatical category and the next 5 of which encode the feature matrix. A node with the grammatical category "verb" and the feature matrix in (53) would be encoded as "0001010100" since "0001" encodes the category verb and "010100" encodes the matrix in (53).

The next bit of information we must encode is the bar-level of the node as dictated by $\overline{\text{X}}$-theory. Uncontroversially, I will assume that there are three distinct bar levels: X^0 for heads, $\overline{\overline{\text{X}}}$ for phrasal level projections, and $\overline{\text{X}}$ for intermediate projections:

(54)



Two bit positions are sufficient to encode the bar-level of a node:

(55)			
	X^0	=	00
	$\overline{X}$	=	01
	$\overline{\overline{X}}$	=	10

Letting j the encoding in (55), we can represent the encoding of the grammatical category, feature matrix and bar-level of a node x be: $g(x)^\frown h(x)^\frown j(x)$. Thus a V^0 with the feature matrix in (53) would be encoded as "000101010000" where the last two digits in the string encode bar-level.

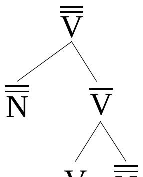
Our next problem is to encode hierarchical structure and linear precedence. One method would be to start from the root and then proceed left-wards down the tree in a depth first fashion. Recall that we are assuming that trees are binary so that, for each node, we need to encode whether it is the root, or, if not, whether it is a left-daughter, a right-daughter or the sole descendant of another node. Since there are four possibilities, we can use two bit positions to encode this information:

(56)			
	00	=	root
	01	=	right-daughter
	10	=	left-daughter
	11	=	sole daughter

Notice that the notation in (56) does not encode what the node is a daughter of (if anything). Precise information about a node's ancestry will be implicitly encoded by the position of the block of binary digits encoding the node.

It is helpful to consider an example. Consider the following tree fragment:

(57)


$\overline{\overline{V}}$
$\overline{\overline{N}}$
$\overline{V}$
V
$\overline{\overline{N}}$

For present purposes, I will ignore arrays of morphological features in the translation and encode only (1) grammatical category, (2) $\overline{X}$ level and (3) hierarchical structure and linear order. This will simplify the exposition and allow us to concentrate on the encoding of hierarchical structure in a string representation.

Let us establish the following conventions. Each node will be represented by a sequence of binary digits of fixed length. I will refer to this sequence as a *block*. Information about descendance will be encoded as the prefix of the block followed by information about bar-level and grammatical category. The blocks will be concatenated in such a way as to make information about hierarchical structure derivable from the string plus the information about descendance. The root of (57) is a $\overline{\overline{\mathrm{V}}}$. Since it is the root, its block begins with "00"; since it is a maximal projection, the block continues with "10"; since it is of category V, the block terminates with "0001". This the entire block encoding the subtree consisting only of the root $\overline{\overline{\mathrm{V}}}$ is:

(58) $\overline{\overline{\mathrm{V}}}$ = 00100001

The left-daughter of the root is a $\overline{\overline{\mathrm{N}}}$. This will be encoded as "10100000" since it consists of a left-daughter (the "10" which begins the block) which is a maximal projection (the next "10") and of category N (the trailing "0000"). Thus, the subtree consisting of the root $\overline{\overline{\mathrm{V}}}$ and its left-daughter, $\overline{\overline{\mathrm{N}}}$, is unambiguously represented by 0010000110100000.

Consider, next, the right-daughter of $\overline{\overline{\mathrm{V}}}$, the $\overline{\mathrm{V}}$. Since it is a right-daughter, it will be prefixed by "01". Notice that it cannot be taken as a right-daughter of $\overline{\overline{\mathrm{N}}}$ since the latter lacks a left-daughter. This illustrates an important point: in order for the encoding scheme to encode phrase markers unambiguously, the information about daughterhood must be taken such that a sole daughter is distinct from both a left-daughter and a right-daughter. If a sole daughter were taken as a degenerate case of either a right- or left-daughter, the scheme would become ambiguous. Continuing with the example at hand, the next two elements in the sequence will be "01" since the node is a single-bar projections. Finally, this block will terminate with the sequence "0001", which encodes the grammatical category V. Thus, the third block is "01010001".

The next block encodes the left-daughter of $\overline{\mathrm{V}}$, namely V^0. This block is "10000001" (left-daughter$^\frown$head$^\frown$V). Finally, the right-daughter of $\overline{\mathrm{V}}$, $\overline{\overline{\mathrm{N}}}$, is encoded as "01100000" (right-daughter$^\frown$phrase$^\frown$N). Thus, the entire tree fragment is unambiguously encoded by the string:

0010000110100000010100011000000101100000

The reader should convince himself that the only way to decode the above string is as the tree fragment in (57).

Let us consider an example of translating from a bit string to a phrase marker. Consider the following string:

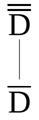
(59) 001001111101011110000111011000001010001001100000

Since block length is rigidly fixed, the string in (59) can be unambiguously broken down into the following six blocks:

1. 00100111
2. 11010111
3. 10000111
4. 01100000
5. 10100010
6. 01100000

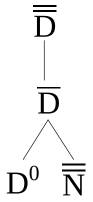
The first block decodes as a root node which is a maximal projection of category D. The second block begins with “11” indicating that it is a sole-daughter. It is a single-bar projection of category D. Hence, the following tree fragment is encoded by the first two blocks:

(60)



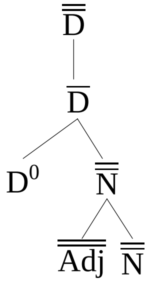
The next two blocks, “10000111” and “01100000” encode a left-daughter D^0 and a right-daughter $\overline{\overline{\mathrm{N}}}$, respectively. Thus, the first four blocks encode the following fragment:

(61)


$\overline{\overline{\mathrm{D}}}$
$\overline{\mathrm{D}}$
D^0
$\overline{\overline{\mathrm{N}}}$

The final two blocks “10100010” and “01100000” encode a left-daughter $\overline{\overline{\overline{\mathrm{Adj}}}}$ and a right-daughter $\overline{\overline{\mathrm{N}}}$. Hence, the string unambiguously encodes the following tree fragment:

(62)


$\overline{\overline{\mathrm{D}}}$
$\overline{\mathrm{D}}$
D^0
$\overline{\overline{\mathrm{N}}}$
$\overline{\overline{\overline{\mathrm{Adj}}}}$
$\overline{\overline{\mathrm{N}}}$

The current encoding scheme deals only with binary trees. It should be clear, however, that the scheme can be extended to cover ternary branching trees or, indeed n-ary branching trees.

Fixed block length can also be eliminated by dynamical encoding of the block length for each node. That is, suppose that ω_i is the encoding of a node Γ_i and that $l(\omega_i) = n$. We can prefix ω_i with a string of n 1s followed so that the encoding, E, of Γ_i would be $E(\Gamma_i) = 1^n 0\omega_i$. This trick will allow us to eliminate some of the redundancy in the encoding scheme.

4 Applications and Consequences

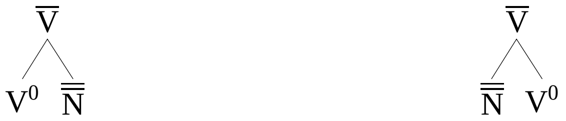
Let us summarize the argument so far. The central thesis rests on the hypothesis that learners can only set a parameter if they are exposed to data expressing that parameter with frequency sufficient to exceed some threshold value. Intuitively, the simpler the structures on which a parameter can be expressed the more frequent parameter expression should be. In section 3.1 I outlined the theory of Kolmogorov complexity which explicitly ties the simplicity of descriptions to high probability. In general, simple structures, those with low Kolmogorov complexity, are the most probable. This result is both extremely pleasing and rather surprising since it ties optimum descriptions to probabilities.

Notice that there are different ways that an object could have low Kolmogorov complexity. One way is that the object is simply small. To take a linguistic example, let us return to the expression of head/complement order. Let us suppose that language particular orderings are encoded by the parameter in (63):

(63) The head precedes its complement. $\{$yes,no$\}$

The parameter in (63) can be expressed in extremely compact tree fragments. Thus, the following tree fragments would seem to be minimal:

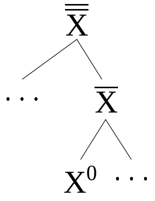
(64)



Since the parameter in (63) is associated with such a small structure, we would expect it to have a low Kolmogorov complexity and, hence, a high frequency of expression in the input text.

Another way to minimize Kolmogorov complexity is to be highly regular, as we have seen. An algorithmically random object is one that cannot be compressed since none of its structure is predictable; this implies that the object's description is as large as the object itself. Natural language is far from being algorithmically random. Consider, for example, the $\overline{\text{X}}$ skeleton shown in (65):

(65)



The $\overline{\text{X}}$ skeleton contains a good deal of predictable structure. The fact that each projection is headed, for example, is entirely predictable, although the placement of the head within the projection is subject to variation. Notice that the encoding of a phrase marker according the the procedure outlined in section 3.2 will contain many redundancies, in particular, the repetition of structure due to headedness or the predictable fact that bar-levels are largely predictable. The intuition here is these regularities in natural language (the $\overline{\text{X}}$ skeleton and so on) pave the way for data compression, allowing parameters to range over larger structures.

The first step in applying the theory must be to define the complexity of a parameter value. For each possible value of a parameter, we must have some method of calculating a good upper bound on its Kolmogorov complexity. The crucial part of the learning system, the part upon which parameter setting rests, is parameter expression (see (18) on page 11 for the definition). Let us extend the definition in the following way:

(66) *Generalized Parameter Expression*
A string ω with representation τ expresses the value v_i of a parameter p just in case p must be set to v_i in order for the grammar to represent ω with τ.

The definition in (66) generalizes the original one in (18) since it allows sentence fragments to express parameters. Thus, the examples in (67) can express parameters, although any one of them can be taken as a sentence fragment:

(67) a. on the table

b. the cat

c. walked the dog

The new definition in (66) relatives parameter expression to representations. This move is controversial since the learner has no access to linguistic representations in the input text. Nevertheless, parameter expression is only coherent when it is relativized to representations. A given string may be associated with a number of distinct representations each of which may express distinct, even conflicting, parameter values. Consider, in this light, the examples in (68):

(68) a. John wanted the ice cream to melt.

b. John believed Bill to annoy Mary.

c. John persuaded Bill to annoy Mary.

All the examples in (68) involve the sequence: V⌒NP⌒to⌒VP. The infinitive in (68a) can be intepreted either as a complement or as a purpose clause; that is, he wanted that the ice cream melt or he wanted the ice cream so he could melt it. A similar reading can be associated with (68b), although it is remote. Notice that (68b) can passivize comfortably, unlike (68a); thus, the two examples have different properties which must be learned:

(69) a. *the ice cream was wanted to melt

b. Bill was believed to annoy Mary.

Finally, the example in (68c) involves object control and, so has a very different representation from those of (68a) and (68b) both of which involve exceptional Case marking. Thus, although the strings of grammatical categories in (68) are the same, the parameters expressed in each example are different. The simplest account of this is to allow structures to express parameters and not simply strings. Thus, the basis for calculations of complexity should be the representations of grammatical strings and not just the strings that occur in the input text.

Finally, we should be careful to note that parameters need not be expressed in isolation nor are they expressed unambiguously. It is unlikely that any example drawn randomly from the input text would show the effect of only a single parameter value; rather, each example is the result of the interaction of a several different principles and parameters (see Clark, 1990;1992;1994 for discussion). Similarly, a given example could be ambiguously generated by a number of different parameter settings and, so, be taken as expressing these values ambiguously. For example, SVO in a sentence could be the base order or the result of V2. Nothing in the definition in either (18) or (66) requires that parameter expression be unambiguous. The computational problem of sorting through the interactions of the principles and parameters to arrive at the correct set of parameter settings for the target language is discussed in Clark (1990;1992) and Clark & Roberts (1993); I will leave that problem aside and focus on the proper metric for the information content of parameters.

Let us take the information content of a parameter value to be the Kolmogorov complexity of the least structure that expresses that parameter value:

(70) *Parameter Complexity*
The complexity of a parameter p_i set to a value v_j is defined by

$$K(p_i(v_j)) \;=\; \min_{\tau \text{ expresses } p_i(v_j)} K(\tau).$$

The reasoning here is that the best measure of the information content of a parameter value is in terms of the smallest structure on which that value has a demonstrable effect.

With the above definition of complexity in mind, let us return to the Boundedness of Parameter Expression, repeated here:

(71) *Boundedness of Parameter Expression*
For all parameter values v_i in a system of parameters $\mathscr{P}$, there exists a syntactic struction τ_j that express $\mathrm{v_i}$ where the complexity $\mathrm{C}(\tau_\mathrm{j})$ is less than or equal to some constant U.

A fruitful approach to the problem of boundedness would be to define the constant U in (71) in terms of the Kolmogorov complexity of the parameters in $\mathscr{P}$. Recall that the complexity of a parameter is defined as the complexity of the minimal structure which expresses that parameter. Each parameter in the system $\mathscr{P}$ might therefore have a different complexity. Let us take $\overline{K}(\mathscr{P})$ to be the average complexity of the parameters in $\mathscr{P}$ and $\mathrm{var}_{\mathscr{P}}$ to be the variance. We might allow U to range over $\overline{K}(\mathscr{P}) \pm \mathrm{var}_{\mathscr{P}}$. That is, U is a function of the average Kolmogorov complexity of the parameterized system. As I will argue, below, this in turn can be approximated by using the entropy of (fair) input texts.

With the above in mind, let us now return to the notions of minimal text and fair text (see (27) and (28) on pages 15 and 15 respectively). The essential idea was that a minimal text contained all trees whose complexity was less than U while a fair text was one where each tree was repeated enough times to exceed threshold for parameter setting. To convert the latter into a text for the learner, one would, of course, obliterate the trees leaving only strings of terminals. Given that we are defining U in terms of the average Kolmogorov complexity $\overline{K}(\mathscr{P})$ of the system $\mathscr{P}$ of parameters, it is apparent that there must be a systematic relationship between the complexity of $\mathscr{P}$ and the fair text. Indeed, if we draw a random example, s_i from the fair text as defined above, the complexity of the example, $K(s_i)$ should bound the complexity of the parameters expressed by s_i. If we let P stand for the parameters expressed by s_i, we would expect:

(72) $\forall_{p \in P} K(p) \;\leq\; K(s_i)$

Generalizing the above, there is a systematic relationship between what we can call “text complexity” and “grammar complexity”. In the perfect world, text complexity would bound grammar complexity from above. In other words, the learner could not induce more complexity than is resident in the input text. Let us take text complexity to be the average complexity of representations in the input text; letting σ_n denote the first n examples in the fair text, then we can define text complexity by:

(73) *Text Complexity*
The complexity of an initial sequence σ_i of an input text σ is given by:

$$\overline{K}(\sigma_i) \; = \; \frac{1}{n}\sum_{i}^{n} K(\sigma_i).$$

which is to say that (73) gives the average complexity of representations drawn from a fair text. Let us denote the quantity in (73) as $\overline{K}(\sigma)$ for a fair text, σ. We expect that:

(74) For any G, a grammar determined by the parameterized system $\mathscr{P}$, and σ, a fair text for G:

$$\overline{K}(\mathscr{P}) \; \leq \; \overline{K}(\sigma).$$

That is, text complexity provides an upper bound on grammar complexity. From (74) and the hypothesis that the bound U lies within the variance of the average complexity of the parameterized system, it follows that:

(75) $U \; \leq \; \overline{K}(\sigma) + \text{var}_\sigma$

That is, we have a principled method of estimating U in (71). In other words, all parameters must be expressed within the range set by the inequality in (75), which establishes on upper bound on the amount of information that can be packed into any one parameter; if a parameter could only be expressed on a structure which exceeded the bound established in (75), we predict that it will be unsettable on a fair input text and, therefore, a parameterized system which contained such a parameter would be unlearnable.

Notice that, once U is known, we can use it to infer a bound on the amount of typological variation possible. This is because U limits the size of trees on which parameters can be expressed; as such, it eliminates arbitrary embeddings. Thus, all linguistic variation must take place within the limited structural domain defined by U. Recall, though, that U is defined so as to contain most of the simple representations; we might speculate that the average complexity of the parameter system, $\overline{K}(\mathscr{P})$ is itself significant. We might speculate that the average complexity corresponds to linguistically significant relationships like government (recall, in this light, the discussion of learnability and locality in section 2). If this is on the right track, then average complexity could provide a key to the learning theoretic foundations of linguistically significant relations. I should note, however, that this hypothesis remains highly speculative.

Finally, we should recall the connection between Kolmogorov complexity and statistics, discussed in section 3.1 (see particularly the discussion of (46) on page 24). The lower the Kolmogorov complexity of an object is, the higher its probability, given that we are using a prefix code for the Turing machine programs that compute the objects. This entails that, given target parameter settings $p_i(v_i)$ and $p_j(v_j)$ such that $K(p_i(v_i)) \leq K(p_j(v_j))$, $p_i(v_i)$ will be expressed more frequently than $p_j(v_j)$. We would expect, then, that $p_i(v_i)$ will exceed threshold for parameter setting in the learning system before $p_j(v_j)$ will. Given, Frequency of Parameter Expression (see (22) on page 13), we would expect the learner to master $p_i(v_i)$ before $p_j(v_j)$. Thus, the Kolmogorov complexity of individual parameters should give us some insight into developmental sequencing.

A question that immediately arises is whether or not there is an effective method for estimating grammar complexity and, thereby, placing an upper bound on U. Recall that, in the limit, the expected Kolmogorov complexity of a random variable X goes to its entropy, $H(X)$ (see the discussion of (45) on page 24). This suggests that we can use entropy to estimate complexity. In this case, the random variable X should range over linguistic representations. This suggests that one technique for estimating U would be, first, to parse a significantly large text. Each tree can be translated into string form using some procedure analogous to the one discussed in section 3.2. Let X be taken as ranging over the nodes in the parse trees, that is blocks in the string encoding. $H(X)$ can then be estimated from the encoding of the parsed text. Thus, we have the interesting result that bounds on the size of linguistic representations can be statistically estimated from properties of texts.

The reader should compare the use of Kolmogorov complexity developed here with the discussion of Berwick (1985). Berwick correctly notes that grammar size is not the relevant metric since, given a nativist account of learning, a learner could acquire an arbitrarily large grammar. Notice that we have not used grammar size as the metric here but rather the complexity of parameter expression. Arguably, this latter is the correct metric for ease of acquisition since, as we have argued, in order to set a parameter to a particular value the learner must have evidence in the form of parameter expression. Thus, while the grammar can be arbitrarily large, its effects on the input text must be on a relatively small syntactic domain. Otherwise, by the Kolmogorov universal statistic, the learner will be unlikely to encounter the effects of the target parameter value and is, therefore, unlikely to converge to the correct setting.

Let us turn, now, to some interesting results due to Osherson & Weinstein (1992). Following much work in the Gold model (Gold, 1967; Osherson, Stob & Weinstein, 1986), they take learners (*Children* in their terms) to be functions from texts (SEQ, or sequences of examples drawn from the target language) to N, indices for languages. Thus the following definitions are standard:[8]

[8] The notation $t[n]$ denotes a text of length n.

(76) Let $C \in$ *Children* and text t be given.

(a) C ***converges*** on t to $i \in N$ just in case $C(t[n]) = i$ for all but finitely many $n \in N$.

(b) C ***identifies*** t just in case there is $i \in N$ such that

(i.) C converges on t to i, and

(ii.) *range*$(t) = W_i$.

(c) C ***identifies*** language L just in case C identifies every text for L.

(d) C ***identifies*** collection $\mathscr{L}$ of languages just in case C identifies every $L \in \mathscr{L}$. In this case, $\mathscr{L}$ is said to be identifiable.

The model in (76) is a standard formalization of the problem of language acquisition. The learner is presented, at each step, with a datum drawn from the target language and offers a hypothesis as to what the target language is. It identifies the target just in case it converges to the correct grammar and does not change its mind and, furthermore, it correctly converges on any text for the language. A collection of languages is identifiable if there is a learner that identifies every language in the collection.

Notice that the framework outlined in (76) is quite general. One could take the collection of languages to be scientific theories, for example, and the learners to be scientists. In this case, the scientist might be presented with a prediction of the theory. Interestingly, Osherson & Weinstein (1992) apply the Gold model in (76) to linguists. To do so, they define a *data sequence* for $C \in$ *Children* as the infinite sequence $(\sigma_0, C(\sigma_0)), (\sigma_1, C(\sigma_1)), \ldots$; each datum is a pair consisting of a sequence $\sigma_i \in$ SEQ (the set of all sequences) and the index of the language C returns on that sequence. The idea is that once the linguist has a working hypothesis about Universal Grammar and the learning function, he could run the theory on input texts and see what happens. The set of finite initial sequences drawn from any data sequence is denoted SEG.

We can take a linguist, then, to be a function from SEG to N where each $n \in N$ is a hypothesis about the nature of the learner. Given $l \in$ *Linguists*, $i \in N$, and a data sequence $d \in SEG$, then l *converges* on d to i just in case $l(d[n]) = i$ for all but finitely many $n \in N$. Furthermore, l *identifies* $C \in$ *Children* just in case l converges on d to an index for C. A collection $\mathscr{C} \subseteq$ *Children* is *identifiable* just in case l identifies each $C \in \mathscr{C}$. Notice the formal similarities between the characterization of linguists and the Gold model in (76).

Osherson & Weinstein (1992) then define the following constraints on possible children. Notice that each of the following constraints represent plausible hypothesis about development psycholinguistics and many, if not all, of them have been proposed in the literature:

(77) *Learning Properties*

a. *Consistent*: $C \in$ *Children* is *consistent* just in case for all $\sigma \in SEQ,$ *range*$(\sigma) \subseteq W_{C(\sigma)}$. That is, consistency requires that the current hypothesis generate at least the current data.

b. *Conservative*: $C \in$ *Children* is *conservative* just in case for all $\sigma \in SEQ$, if *range*$(\sigma) \subseteq W_{C(\sigma-)}$ then $C(\sigma) = C(\sigma-)$ where $\sigma-$ is the result of removing the last member of σ. That is, a conservative learner doesn't abandon a hypothesis that generates all the available data.

c. *Memory-limited*: Let $C \in$ *Children* be given. C is *memory-limited* just in case for all $\sigma, \tau \in SEQ$, if $C(\sigma-) = C(\tau-)$ and $\sigma_{\text{last}} = \tau_{\text{last}}$ then $C(\sigma) = C(\tau)$, where σ_{last} is the last element of σ. That is, the learner's current conjecture depends only on its previous conjecture and the current datum.

d. *Prudent*: Let $C \in$ *Children* be given. C is *prudent* just in case *scope*$[C] = \{W_i | \text{for some } \sigma \in SEQ, C(\sigma) = i\}$, where *scope*$[C]$ is the collection of languages that C identifies. That is, learners never hypothesize a language that is not in the collection that the learner identifies.

e. *Decisive*: Let $C \in$ *Children* be given. C is *decisive* just in case for all $\sigma \in SEQ$, if $W_{C(\sigma)} \neq W_{C(\sigma-)}$ then there is no $\tau \in SEQ$ that extends σ with $W_{C(\tau)} \neq W_{C(\sigma-)}$. In other words, once the learner has abandoned a hypothesis, it never returns to it.

f. *Boundedness*: Let $C \in$ *Children* and total recursive function $h : N \longrightarrow N$ be given. C is *h-bounded* just in case for all $\sigma \in SEQ, \varphi_{C(\sigma)}$ operates in h-time ($\varphi_n(x)$ operates in h-time if it is defined within $h(x)$ steps of computation, h a total recursive function from N to N). That is, the amount of time available to the learner is bounded.

Notice that each of the properties in (77) narrows the set of children that the linguist must identify.

Osherson & Weinstein (1992) prove the following theorem:

(78) *The nonexistence of reliable linguists*

There is a collection $\mathscr{C} \subseteq$ *Children* with the following properties:

(a) Each $C \in \mathscr{C}$ is consistent, conservative, memory-limitied, prudent and decisive.

(b) There is total recursive $h : N \longrightarrow N$ such that each $C \in \mathscr{C}$ is h-bounded.

(c) $\mathscr{C}$ is not weakly characterizable.

Weak characterization is a fairly liberal success criterion:

(79) Let $l \in$ *Linguists* and $C \in$ *Children* be given and let d be the data sequence for C. l *weakly characterizes* C just in case l converges on d to $i \in N$ such that

(a) *scope*$[C] \subseteq \{W_j | j \in W_i\}$

(b) $\{W_j | j \in W_i\}$ is identifiable.

That is, the linguist is able to characterize learners whose scope is a subset of the identifiable collection of languages.

The theorem in (78) is quite strong. It entails that the linguist will be unable to identify $\mathscr{C}$, the target collection of children, even when that class is limited to a narrow collection of learners. It might be thought that the problem is that we are still considering too broad a class of learners, even though we are limiting our attention to h-bounded, consistent, conservative, memory-limitied, prudent and decisive learners. An important property of principles and parameters systems is that they admit only a finite class of languages. It might be thought that this will help the linguist narrow her hypothesis sufficiently to allow for characterization of $\mathscr{C}$. Osherson & Weinstein (1992) prove the following:

(80) Let $\mathscr{C}$ be the collection of children that idenify less than three, nonempty languages. Then $\mathscr{C}$ is not weakly characterizable.

The theorem in (80) has the following corollary:

(81) The collection of children whose scope is finite is not identifiable.

Thus, finitude does not help the linguist become reliable.

The results in (78), (80) and (81) would seem to offer a bleak prospects for the success of linguistic theory. In fact, having formalized the problem, it is possible to try to extend the investigation to discover what sorts of children linguists could, in principle, discover. In particular, Osherson & Weinstein offer the following:

(82) h-*fast children*

a. Given $C \in$ *Children* and total recursive $h : N \longrightarrow N$, call C h-*fast* just in case $C(\sigma)$ is defined in $h($*size*$(\sigma))$ units of time, if at all.

b. PROPOSITION: Let total recursive $h : N \longrightarrow N$ be given and let $\mathscr{C} \subseteq$ *Children* be the collection of all h-fast children. Then $\mathscr{C}$ is identifiable

Notice that the function h which bounds the learner is defined in terms of the size of the input text. That is, the linguist need only consider those $\mathscr{C} \subseteq$ *Children* that converge in time bounded by $h(\textit{size}(\sigma))$. If this is correct, then complexity considerations of the sort addressed by Kolmogorov complexity are crucial for constraining the problem of language learnability. Crucially, h relies on a complexity metric on the input required for convergence. It seems sensible, then, to hypothesize that the same metric which connects the complexity of the input text to the information encoded in parameters, Kolmogorov complexity, could be used to bound the time needed for learners to converge. Recall that we started with the intuition that learning can be accomplished on a fair text which is, itself, constructed from a minimal text. As we have seen earlier in this section, minimal texts can be defined in terms of the Kolmogorov complexity of individual parameters. We might suppose, then, that h is a function from N to N which takes $\overline{K}(\sigma)$, the average complexity of the input text. Thus, the learner would be h-fast relative to the average complexity of the input data. Given the systematic relation between $\overline{K}(\mathscr{P})$ and $\overline{K}(\sigma)$, h-fast learners could, then, grounded in the complexity of the parametric system $\mathscr{P}$ itself, an attractive result.

References

Anthony, M. & N. Biggs (1992). *Computational Learning Theory.* Cambridge University Press, Cambridge.

Berwick, R. (1985). *The Acquisition of Syntactic Knowledge.* The MIT Press, Cambridge, MA.

Chaitin, G. (1975). “Randomness and mathematical proof”. *Scientific American*, 232, pp. 47-52.

Chaitin, G. (1987). *Algorithmic Information Theory.* Cambridge University Press, Cambridge.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* The MIT Press, Cambridge, MA.

Chomsky, N. (1973). “Conditions on transformations” in S.R. Anderson and P. Kiparsky (eds) *A Festschrift for Morris Halle.* Holt, Rinehart and Winston, New York.

Chomsky, N. (1986). *Barriers.* The MIT Press, Cambridge, MA.

Chomsky, N. (1992). “A Minimalist Approach to Linguistic Theory”. *MIT Working Papers in Linguistics*, Occasional Papers in Lingiustics No. 1.

Clark, R. (1990).*Papers on Learnability and Natural Selection*, Technical Report in Formal and Computational Linguistics, No. 1, Département de linguistique, Université de Genève.

Clark, R. (1992). “The Selection of Syntactic Knowledge” *Language Acquisition*, 2.2, pp. 83-149.

Clark, R. (1993). “From a Context-Free Initial State to a Mildly Context-Sensitve Grammar via Classifiers: Self-Organizing Grammars and Parameter Setting”. ms. University of Pennsylvania.

Clark, R. (1994). “Finitude, Boundedness and Complexity: Learnability and the Study of First Language Acquisition”. Barbara Lust, Magui Suñer & Gabriella Hermon (eds) *Syntactic Theory and First Language Acquisition: Crosslinguistic Perspectives (Vol. II)*. Lawrence Erlbaum, Inc., Princeton, NJ.

Clark, R. & I. Roberts (1993). “A Computational Model of Language Learnability and Language Change”, *Linguistic Inquiry*, 24.2, pp. 299-345.

Cover, T. & J. Thomas (1991). *Elements of Information Theory*. John Wiley & Sons, Inc., New York.

Gibson, E. & K. Wexler (1994). “Triggers”. *Linguistic Inquiry*, 24, to appear.

Gold, E. M. (1967). “Language identification in the limit”. *Information and Control*, 10, pp. 447-474.

Kayne, R. (1984). *Connectedness and Binary Branching*. Foris Publications, Dordrecht, the Netherlands.

Kayne, R. (1993). “The Antisymmetry of syntax”. ms. CUNY.

Kolmogorov, A. N. (1965). “Three approaches to the quantitative definition of information”. *Problems in Information Transmission*, 1, pp.1-7.

Li, M. & P. Vitányi (1993). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York.

Lightfoot, D. (1989). “The Child’s Trigger Experience: Degree-0 Learnability”. *Behavioral and Brain Sciences*, 12.2, pp. 321-375.

Lightfoot, D. (1991). *How to Set Parameters*. The MIT Press. Cambridge, MA.

Morgan, J. (1986). *From Simple Input to Complex Grammar*. The MIT Press. Cambridge, MA.

Natarajan, B. (1991). *Machine Learning: A Theoretical Approach.* Morgan Kaufmann Publishers, Inc. San Mateo, CA.

Niyogi, P. & R. Berwick (1993). “Formalizing Triggers: A Learning Model for Finite Spaces”. A.I. Memo No. 1449. MIT.

Osherson, D & S. Weinstein (1992). “On the study of first language acquisition”. ms. IDIAP and University of Pennsylvania.

Osherson, D., M. Stob & S. Weinstein (1986). *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* The MIT Press. Cambridge, MA.

Papadimitriou, C. H. (1994). *Computational complexity*, Addison-Wesley, Reading, MA.

Rizzi, L. (1989). “On the Format for Parameters”. *Behavioral and Brain Sciences*, 12.2, pp. 355-356.

Solomonoff, R. J. (1964). “A formal theory of inductive inference”. *Information and Control*, pp. 1-22, 224-254.

Wexler, K. & P. Culicover (1980). *Formal Principles of Language Acquisition.* The MIT Press, Cambridge, MA.

Zurek, W. H. (1990). *Complexity, Entropy and the Physics of Information.* Addison Wesley Publishing Co., Redwood City, CA.