Animal Pattern-Learning Experiments: Some Mathematical Background*

Geoffrey K. Pullum and James Rogers
Radcliffe Institute for Advanced Study, Harvard University
Putnam House, 10 Garden Street, Cambridge MA 02138

1 Introduction

A number of recent studies on the learning of invented patterns in symbol sequences by animals such as cotton-top tamarins (Hauser, Chomsky & Fitch 2002:1578; Fitch & Hauser 2004; Hauser 2005; O'Donnell, Hauser & Fitch 2005), songbirds (Gentner 2005), and undergraduates (Fitch & Hauser 2004; Perruchet & Rey, in press) have shown interest in the contrast between two classes of sets of strings: the finite-state and the context-free. The claim of Fitch & Hauser (2004) is that tamarins can learn characteristic finite-state string patterns but not those that are context-free, while humans can learn both. Perruchet & Rey dispute the latter claim about humans; Gentner suggests that starlings show signs of being able to learn distinctively context-free patterns of recursion (self-embedding); and so on.

Showing that a mechanism can learn some stringset in a certain class does not, of course, show that it can learn every stringset in that class (unless that stringset is *complete* for that class in the technical sense of computational complexity theory). But it *is* possible to provide evidence that the learnability boundary for some learning mechanism (the line between the stringsets it can and cannot learn) *may* fall between two hierarchically related stringset classes. One could do this by utilizing a pair of stringsets: one in the stronger (larger) class which cannot be learned (showing that not all stringsets in the stronger class are learnable) and one in the weaker (smaller) class which can be learned (showing that at least one stringset in the weaker class is learnable).

But one would want to find a pair of stringsets that were relatively close to each other with respect to the hierarchy. One would not, for instance, choose a finite stringset and a strictly context-sensitive stringset if the aim was to find evidence that some mechanism could learn context-free but not context-sensitive stringsets.

In at least some of these studies, the boundary between finite-state and context-free stringsets is explored using stringsets that are not good exemplars of the former class. This appears to be a result of the mistaken assumption that the finite-state stringsets are at the lowest level of the relevant hierarchy, hence the smallest class of stringsets that needs to be considered. But the finite-state stringsets are by no means the smallest relevant stringset class. There are infinite hierarchies of interesting classes with general mathematical characterizations and operational characteristics that are potentially interesting from a cognitive perspective, each containing infinitely many infinite stringsets, and each a proper subset of the finite-state class. Some of these classes are learnable in the specific formal sense defined by Gold (1967).

Moreover, in some of the cited studies the experiments are apparently designed to distinguish the ability to learn strictly local relationships from the ability to learn unbounded or 'long-distance' dependencies. But this distinction does not correspond to the distinction between the finite-state and context-free stringsets. First, strict locality characterizes only a very weak subclass of the finite-state stringsets. Second, finite-state stringsets can exhibit clearly unbounded dependencies.

That these points are not better known may be due to the fact that so few mathematical studies were done on proper subclasses of the finite-state stringsets in the early years of formal language theory. The situation changed radically during the 1960s, though (as noted by McNaughton & Papert 1971:xiii). The stringset classes that are proper subclasses of the finite-state are now fairly well understood. An understanding of their properties would appear to be highly relevant to the work on animal learning of regularities, and thus perhaps to broader studies of animal precursors of the human ability to recognize grammaticality over indefinitely large sets of expressions.

Our goal in this paper is to provide an introduction to some interesting proper subclasses of the finite-state class, with particular attention to their possible relevance to the problem of characterizing the capabilities of language-learning mechanisms. We survey a sequence of these classes, strictly increasing in language-theoretic complexity, discuss the characteristics of the classes both in terms of their formal properties and in terms of the kinds of cognitive capabilities they correspond to, and suggest contrasting patterns which could serve to distinguish the adjacent classes in language learning experiments.

*This is the draft of January 25, 2006. It is being circulated to a few interested scholars in relevant fields, but has not at present submitted for publication. Please contact the authors — `gpullum@radcliffe.edu` and `jrogers@radcliffe.edu` — before citing or quoting this paper. Comments will be welcomed. Our work has been partially supported by the Radcliffe Institute for Advanced Study at Harvard University. We are grateful to Marc Hauser, Gary Marcus, Andrew Nevins, and Tim O'Donnell for relevant conversations and correspondence. Special thanks to Jean Davidson, Gerald Gazdar, Andràs Kornai, Barbara Scholz, Stuart Shieber, Dmitri Tymoczko, and Tom Wasow, who read earlier drafts, made substantive contributions, spotted errors, and supplied critical comments. They deserve credit for many improvements but bear no responsibility for any shortcomings.

We also note that the sense in which 'recursion' may be thought of as distinguishing finite-state from context-free stringsets is much more subtle than seems to be normally assumed.

2 Preliminary overview

We begin with a brief and somewhat psychology-oriented overview, in the hope that the reader will find it useful to have a non-mathematical, intuitive introduction to the classes of stringsets we are concerned with, stating up front what we take to be the key features that might have psychological relevance. The main classes of stringsets we deal with are these (we proceed from weaker to stronger classes):

— The **strictly local** (SL) stringsets are described solely in terms of adjacency relations — which fixed substrings are allowed to be immediately next to which other fixed substrings. (For each number k, there is a class of strictly local stringsets in which the maximum length of the mentioned substrings is k symbols.) In terms of psychology, these stringsets have a clear connection with the capacity to associate immediately adjacent stimuli — not necessarily to recall whether a certain stimulus was present at all, but to recognize a pair of stimuli when they are presented together in sequence.

— The **locally testable** (LT) stringsets include all the strictly local ones together with those that can be defined out of them through the set-theoretic operations of **union** (the union of two sets contains every string that is either in the first or in the second), **intersection** (the intersection of two sets contains every string that is both in the first and in the second), and **complement** (the complement of a set over some vocabulary contains every string over the same vocabulary that is *not* in that set). In terms of psychology, this turns out to correspond to the capacity to verify that a certain stimulus contains some perceptible element at some point. (This is not nearly so easy to see, but it will become clearer as we proceed.)

— The **star-free** (SF) stringsets include all the locally testable stringsets together with all those that can be defined from them by means of four operations: union, intersection, complement, and concatenation. (The operation of **concatenation** on two stringsets L_A and L_B forms the set L_AL_B of all strings that have a first part from L_A and a second part from L_B.) The corresponding psychological ability is the capacity to recognize that a certain stimulus occurred at least n times for some fixed number n — counting only up to some constant limit and thereafter losing track.

— The **finite-state** (FS) stringsets include all the star-free stringsets plus those that can be defined out of them through the operation of repeating substrings arbitrarily many times (the operation that forms from a set A the set of all strings consisting of zero or more concatenated elements of A is referred to here as asteration). The psychological ability that is added when we move from the star-free to the finite-state is essentially the ability to do modular arithmetic: to recognize that a certain stimulus occurred exactly n times in excess of some multiple of a modulus m (as in the way clocks count hours) — that is, to be able to count up to a set threshold and then re-set the counter and start again.

— The **context-free** (CF) stringsets are a well-known proper superset of the finite-state stringsets that allow for what can be computationally modeled by operations on an unbounded-depth pushdown stack memory. (Unlike finite-state automata, pushdown automata do not have a fixed finite bound on the amount of working memory needed for computations.) Context-free stringsets have sufficient expressive power to cover such operations as integer addition in unary notation (for instance, the set of all strings like '`1+1=11`', '`1+11=111`', '`11+1=111`', '`11+11=1111`', and all others where the third block of `1` symbols is as long as the first two joined together, can be described with a context-free grammar.)

Fleshing out these intuitive characterizations so that their potential cognitive implications can be seen more clearly is the purpose of the following sections, in which we offer a catalogue of formal language theory results pertaining to the stringset classes that fall below the finite-state in the hierarchy. We attempt to minimize notational technicalities, and we keep the overall structure modular enough that the reader will be able to skip technical details of one subsection without rendering the next unintelligible. But we do make use of a number of standard notations from elementary logic and formal language theory. We write:

— 'iff' for 'if and only if';
— $\{w \mid \phi(w)\}$ for the set of all strings w such that the condition $\phi(w)$ holds;
— $\#_a(w)$ for the number of as in the string w;
— ε for the unique 'empty' string of zero length;
— ab for the string consisting of a followed by b;
— a^n for a string of n successive as;
— $(ab)^n$ for n successive ab sequences;
— $\{a, b\}^*$ for the **asteration** of $\{a, b\}$ — the set of all strings consisting of those symbols (any length, ≥ 0), and
— $\{a, b\}^+$ for the **positive asteration** of $\{a, b\}$ (all strings with length ≥ 1).

For the asteration of a singleton set we often omit set brackets; so we may write a^* for $\{a\}^*$ or $(ab)^+$ for $\{ab\}^+$.

Other notations will be explained as we come to use them. Notice that although we maintain a terminological distinction between stringsets, classes of stringsets, and hierarchies of classes, this is purely for expository purposes. Mathematically, all of our classes and hierarchies are just sets (hierarchies being ordered by proper inclusion).

3 Strictly Local

We begin with a highly limited class of stringsets, one that really is limited to purely local dependencies in strings (as the finite-state stringsets are not): the **strictly local** (SL) stringsets. These are defined by reference to dependencies that are purely local in the sense that the only thing mentioned in describing them is the composition of their strings in terms of fixed-length substrings. They provide a formal reconstruction of a notion that will be familiar to anyone who knows the history of 20th-century psychology: the strict associationism that takes learning to be simply a matter of becoming sensitized to which sensorily perceptible phenomena occur in temporal proximity to which others — a process of coming to associate two stimuli if they regularly occur adjacently. An animal that could learn to recognize patterns in symbol strings only if they were definable in SL terms could indeed be said to be limited to strictly local dependencies, in a sense that can be made quite precise (we return to the implications of this point in section 8).

Hauser, Chomsky & Fitch (2002:1577) may be alluding to the SL class when they say:

> At the lowest level of the hierarchy are rule systems that are limited to local dependencies, a subcategory of so-called "finite-state grammars." Despite their attractive simplicity, such rule systems are inadequate to capture any human language.

The SL stringsets are, indeed, a very limited proper subset of the finite-state stringsets.

3.1 Definition

A strictly local stringset over a vocabulary Σ is a subset of Σ^* that is fully describable by means of a finite list Γ of strings not longer than some upper length limit k. In order to belong to the stringset that Γ describes (the stringset that we denote by $L(\Gamma)$), a string in Σ^* must consist wholly of substrings in Γ. The sequences appearing in Γ (and the substrings against which they are matched) are often called **factors** (since formal language theorists write $a \cdot b$ or ab to denote a followed by b, and a^3 for three as concatenated together, the same notation used in algebra for multiplication). By definition, every k-length substring of every string in $L(\Gamma)$ is a member of Γ.

Permitting reference to beginnings and ends of strings in Γ can be accomplished in various ways, but here we will add two pseudo-symbols (intuitively, end-markers) to the symbol set: $\rtimes$ for the beginning and $\ltimes$ for the end. Hence Γ will actually be a set of strings over $\Sigma \cup \{\rtimes, \ltimes\}$, with $\rtimes$ occurring only initially and $\ltimes$ occurring only finally. A string $\rtimes w$ means that a string in the stringset being defined may begin with w; and $z\ltimes$ means that a string may end with z. There is no other determinant of grammaticality for a stringset that is strictly k-local other than that every substring of length k must be on the list of factors.

We can distinguish more finely among the stringsets in SL. There are actually infinitely many distinct stringset classes making up SL, because for each limit k on factor length there is a different class of stringsets — the class of strictly k-local stringsets for that choice of k, denoted by SL_k. A description of an SL_k stringset is simply a finite set of factors each no more than k symbols in length. The class SL is the infinite union of all the SL_k classes.[1]

The strings in the description for an SL_k stringset divide into four types:

(1)

$\sigma_1 \cdots \sigma_k$	(a string of k symbols)
$\rtimes\sigma_1 \cdots \sigma_{k-1}$	(the left end-marker plus a string of $k-1$ symbols)
$\sigma_1 \cdots \sigma_{k-1}\ltimes$	(a string of $k-1$ symbols plus the right end-marker)
$\rtimes \underbrace{\sigma_1 \cdots \sigma_i}_{i \leq k-2} \ltimes$	(a string of not more than $k-2$ symbols flanked by the two end-markers)

Letting $F_k(w)$ denote the set of k-factors occurring in the string w, the stringset licensed by an SL_k description Γ is:

(2) $\{w \mid F_k(\rtimes w \ltimes) \subseteq \Gamma\}$

Setting $k = 1$ gives a degenerate case where we cannot define a grammaticality distinction at all. If both endmarkers are included in Γ, all strings over the symbols mentioned in Γ are defined as grammatical (so over that vocabulary everything is grammatical); if either endmarker is missing we get the empty stringset $\emptyset$ (i.e., nothing is grammatical); and there are no other possibilities.

The smallest non-degenerate case is where $k = 2$, giving the **strictly 2-local** stringsets, SL_2. These could be called the **bigram** stringsets, since their descriptions are simply sets of bigrams — two-symbol factors that are stipulated to be permissible subsequences of strings.

3.2 Abstract characterization

The following theorem provides a characterization of the class SL in terms of a property of strings:

(3) **Definition**: Suffix Substitution Closure
A stringset L is strictly local (SL) iff there exists some $k \geq 1$ such that for all $x \in \Sigma^{k-1}$ and $v, w, y, z \in \Sigma^*$, the following holds:

$$w \cdot x \cdot y \in L \text{ and } v \cdot x \cdot z \in L \text{ implies } w \cdot x \cdot z \in L.$$

That is, the SL stringsets are exactly those for which, beyond a certain factor length, what can precede a factor is entirely independent of what can follow it.

That is, in SL stringsets there is a length beyond which the description is unable to impose conditions on both what precedes a factor x (of the necessary length, $k-1$ symbols long for a stringset in SL_k) and what follows it; so it allows *anything* that can precede v to co-occur with *anything* that can follow.

[1] Strictly speaking we didn't define infinite unions. But we don't really need to appeal to them, because we could just say $\mathrm{SL} = \{L \mid \exists k \geq 2[L \in \mathrm{SL}_k]\}$.

Thus all you have to do to prove a stringset is not SL_k is to find two strings wxy and vxz (x being $k-1$ symbols long) that are in the set, where wxz does not belong. The following facts suffice to show that the set E of strings of words that are grammatical in English cannot be SL_2:

(4) a. *I absented myself.*
b. *You absented yourself.*
c. **I absented yourself.*

Let w = *I*, x = *absented*, y = *myself*, v = *You*, z = *yourself*. Since (4a) and (4b) show that wxy and vxz are in E, wxz should be too if the set is SL_2; but (4c) indicates that it is not. Therefore E is not SL_2.

3.3 Canonical example of non-membership

The hallmark configuration that guarantees a stringset will not be SL is for there to be some required factor that must occur in strings arbitrarily far from the ends. For example, the set denoted $a^*(ba^*)^+$ is a very simple finite-state stringset containing all and only those strings over $\{a, b\}$ that contain at least one b, but it has no SL_k description for any k. The apparently simple notion 'contains a b' is not expressible.

Given that every English transitive clause contains a verb, and given that both subject and object noun phrases may be arbitrarily long, and the verb must lie between them, we now know that the set of English clauses, conceived as a stringset, is not SL_k for any k. To see this, take a transitive verb V and a noun phrase N, where N is at least $k-1$ words long and can function either as the subject of or as the direct object of V. Since N can start a clause with V as its verb, and it can end such a clause, an SL_k description will wrongly permit a transitive clause to consist of just N on its own. (In this case, $w = \varepsilon$, $x = N$, $y = V \cdot N$, $v = N \cdot V$ and $z = \varepsilon$.) But a noun phrase on its own is not a transitive clause. Hence the set of transitive clauses in English is not SL.

3.4 Hierarchies

There are infinitely many strictly k-local classes of stringsets: the bigram stringsets (SL_2), the trigram stringsets (SL_3), and so on. Each is properly included in the next one up: $\mathrm{SL}_2 \subsetneq \mathrm{SL}_3 \subsetneq \cdots \mathrm{SL}_k \subsetneq \mathrm{SL}_{k+1} \subsetneq \cdots$. The class SL is the infinite union of all of them: $\mathrm{SL} = \mathrm{SL}_2 \cup \mathrm{SL}_3 \cup \mathrm{SL}_4 \cdots$.

The relation between the SL stringsets and the finite ones is slightly subtle. The class Fin of finite stringsets is a proper subset of SL, but not of SL_k for any k. Nonetheless, for any given finite stringset L you can find some k such that L is strictly k-local. That is, both the following are true:

(5) a. There is no length limit k such that Fin is a subset of SL_k.
b. For every stringset L in Fin there is a length limit k such that L is in SL_k.

A corollary of this is that for every k we can find finite sets of strings with no strictly k-local description. For example, $\{abc, cba\}$ is a two-member finite set that has no bigram description and thus is not in SL_2.

4 Locally Testable

We turn now to a class of strictly more complex stringsets, the **locally testable** or LT stringsets. They are straightforwardly definable from the SL class.

4.1 Definition

The locally testable (LT) class of stringsets is the closure of the strictly local class under the boolean operations. That is, every SL stringset is in the class LT, and so is every stringset that can be formed by taking the union of two stringsets in LT, the intersection of two stringsets in LT, or the complement of a stringsets in LT. We can use this as our basic definition:

(6) **Definition**: Locally Testable
The class of **locally testable** (LT) stringsets is the smallest class that (i) contains every SL stringset, and (ii) is closed under the operations of union of two sets in the class, intersection of two sets in the class, and complement of a set in the class.

Again we have an infinite hierarchy of stringset classes: for each k, the class LT_k is the closure of SL_k under the boolean operations. The class LT is the union of all the LT_k (that is, $\mathrm{LT} = \bigcup_k [\mathrm{LT}_k]$).

4.2 Example

An example of an LT stringset that is not SL is $a^+(ba^+)^+$, a minor variation on the example of Section 3.3. This is the set of strings in $\{a, b\}^+$ in which in which at least one b occurs, and b always occurs alone, preceded and followed by at least one a. This set is LT (in fact LT_2), because it is the intersection of three SL_2 sets: (i) the set of strings in $\{a, b\}^+$ which both start and end with a; (ii) the set in which bb does not occur (an SL_2 description can simply permit all pairs of as and bs *except* for bb); and (iii) the set in which ab does occur (SL_2 in virtue of being the complement of the set in which ab does not occur). But it is not SL_k for any k, since $a \cdot a^{k-1} \cdot ba^k$ and $a^k b \cdot a^{k-1} \cdot a$ are both in the set but $a \cdot a^{k-1} \cdot a$ is not.

4.3 Abstract characterization

The following statement provides an abstract characterization of the LT class.

(7) **Definition**: Local Test Invariance
A stringset $L \subseteq \Sigma^*$ is LT iff there exists some $k > 0$ such that for all strings $w, v \in \Sigma^*$

$$(F_k(\rtimes w \ltimes) = F_k(\rtimes v \ltimes)) \Rightarrow (w \in L \Leftrightarrow v \in L).$$

This says that a stringset is LT iff there is a number k such that membership of a string in the set is determined solely by

which k-factors occur in the string. Any two strings that share the same set of k-factors will either both be members of the set or both be non-members. Notice that every LT stringset must be LT_k for some finite number k, and the definition asserts the existence of that k.

4.4 Canonical example of non-membership

It is impossible to ensure in an LT stringset that some factor occurs at least n times for some $n > 1$, while allowing each instance to occur arbitrarily far apart and arbitrarily far from the ends of the string. So this stringset (the set of all strings over $\{a, b\}$ in which a occurs at least twice) is not LT:

(8) $\{w \in \{a,b\}^* \mid \#_a(w) \geq 2\}$

Likewise, the variants with $\#_a(w) = 2$ or $\#_a(w) < 2$ are not LT.

It is also impossible to require two factors to occur in some particular order if the two may occur arbitrarily far apart and arbitrarily far from the ends of the string. So this stringset is not LT:

(9) $\{u\,ab\,v\,ba\,w \mid u, v, w \in \{a,b,c\}^*\}$

4.5 Hierarchies

Again we have an infinite hierarchy of stringset classes making up the entire class LT:

$$\mathrm{LT}_2 \subsetneq \mathrm{LT}_3 \subsetneq \cdots \mathrm{LT}_k \subsetneq \mathrm{LT}_{k+1} \subsetneq \cdots$$

And again, the relationship to the finite stringsets is not simple: it is true that $L \in \mathrm{Fin} \Rightarrow L \in \mathrm{LT}_k$ for some k, so $\mathrm{Fin} \subsetneq \mathrm{LT}$. However, there is no k such that $\mathrm{Fin} \subseteq \mathrm{LT}_k$. Thus for every integer $k \geq 2$, there are finite stringsets that are not LT_k.

The relations between the SL and LT hierarchies are as follows. For every k, SL_k is a proper subset of LT_k; but for no k is SL_{k+1} a subset of LT_k.[2] On the other hand, there is no k such that LT_k a subset of SL_{k+1}. In fact it is not even the case that $\mathrm{LT}_2 \subseteq \mathrm{SL}$, so for no k is LT_k a subclass of SL.[3]

5 Star-Free

We now define an important proper superset of all of the classes mentioned so far: the **star-free** (SF) stringsets.[4]

We first define the class SF inductively:

(10) **Definition**: Star-Free Stringsets (SF). The class of star-free stringsets over a vocabulary Σ is the smallest class of stringsets that (i) includes the empty set $\emptyset$, the singleton set $\{\varepsilon\}$ containing the empty string, and for each symbol a in Σ the singleton set $\{a\}$, and (ii) is closed under boolean operations and concatenation.

(The reader already familiar with finite-state stringsets will notice how similar this is to the definition of that class. However, it lacks the mention of closure under the '*' operation — hence the name 'star-free' — and it includes closure under complement in the definition.)

There is a straightforward relation between SF and LT. Closing LT under concatenation and the boolean operations yields a class of stringsets known under the name **k-locally testable with order** (LTO_k), and the infinite union of these for all k is the class LTO. So we have this definition:

(11) **Definition**: Locally Testable with Order (LTO). The class of LTO stringsets is the smallest class that contains all the LT stringsets and all those that can be formed from LTO stringsets by means of the union, intersection, complement, and concatenation operations.

McNaughton & Papert (1971) prove that this gives us exactly the same class as the previous definition, i.e., that LTO = SF.

5.1 Example

An example of an SF stringset that is not LT is a^+ba^+. This is the set of strings in $\{a,b\}^+$ in which *exactly one* b occurs. It is SF since it is the concatenation of the set of strings in $\{a,b\}^+$ in which the only b is the final symbol (an SL_2 set licensed by the pairs $\{\rtimes a,\ aa,\ ab,\ b\ltimes\}$) and the set of strings in which only a occurs (similarly an SL_2 set). It is not LT_k for any k since the strings a^kba^k and $a^kba^kba^k$ share the same set of k-factors but the first is a member of the stringset while the second is not.

By similar analyses, each of the canonical non-LT stringsets given in Section 4.4 can be shown to be SF.

[2] To see this, note that for any k there is an SL_{k+1} description for $\{a,b\}^* - (\{a,b\}^*\{a\}^{k+1}\{a,b\}^*)$ — the set of all strings over $\{a,b\}$ with no string of $k+1$ adjacent occurrences of a; but this stringset will not be in LT_k, because the k-length factors of $\rtimes a^kba^k\ltimes$ are identical with the k-length factors of $\rtimes a^{k+1}ba^{k+1}\ltimes$.

[3] This can be seen from the fact that $L_{ab} = \{a,b\}^*\{ab\}\{a,b\}^*$, the set of all strings over $\{a,b\}$ in which ab occurs, is LT_2 (it's the complement of the set of strings in which ab does *not* occur). But it cannot be SL, for this reason: for any choice of k, the string b^kabb^k cannot belong to an SL_k set unless b^k also belongs, so for all k we have a counterexample to the claim that L_{ab} is SL_k; and that means L_{ab} is not SL at all.

[4] At least two other infinite hierarchies of stringsets are left undiscussed here, though it would be appropriate to include them in a fuller survey of this material. One has been called the 'dot-depth hierarchy'. We start with a base class of stringsets like SL or the even more primitive 'definite event' stringsets (in which grammatically is determined entirely by the content of the last k symbols for some fixed k), and we form the class of stringsets obtainable by concatenating any two of them, and then we close under boolean operations. This gives us dot-depth 1. Then we repeat with the dot-depth 1 stringsets as the base, to obtain the dot-depth 2 class. An infinite strict hierarchy of stringset classes is obtained. Another way to go is to generalize the LT capability for recognizing that a substring x occurs at least once, and replace 1 by a constant k. That is, we allow for recognizing of a string w that a substring x occurs at least j times, for $j \leq k$. This yields a class of stringsets that are called 'locally threshold testable' in Straubing (1994:47) and 'generalized locally testable' in Thomas (1982:372). For each $k \geq 1$ there is a class of k-locally threshold testable (or generalized k-locally testable) stringsets. Again the hierarchy is strict, and the infinite union of its members for all k makes up the locally threshold testable (or generalized locally testable) class that we could call LTT. An interesting model-theoretic characterization is given by Straubing (1994:46–50) and Thomas (1982:372–373): LTT is exactly characterized by first-order logic, interpreted on string models, if there is only one relation symbol and it is interpreted as 'is immediately followed by'. All these classes are proper subclasses of the better-known star-free class to which we turn in this section.

5.2 Abstract characterization

It is possible to characterize the same class in a different and more abstract way, in terms of a direct condition on what strings they contain and exclude.

(12) **Definition**: Non-Counting Stringsets. A stringset L over a vocabulary Σ is **non-counting**[5] iff there exists some $n > 0$ such that for all strings $u, v, w \in \Sigma^*$ and for all $i \geq 1$, the presence of $uv^n w$ in L implies that $uv^{n+i}w$ is also in L.

In other words, in a stringset that is non-counting there is a threshold length beyond which repeated substrings cannot be counted: wherever there are n consecutive occurrences of any substring v, it would also be grammatical to have $n + 1$ consecutive occurrences of v.

Schützenberger (1965) proves that the non-counting stringsets are exactly the star-free stringsets. So that yields a third distinct characterization of the SF class.

5.3 Canonical example of non-membership

A representative example of a kind of stringset that is not in SF is that of the stringsets in which some factor must occur an even number of times. Thus the following stringset is not SF:

(13) $\{w \mid w \in \{a, b\}^* \mid \#_b(w) \text{ is even}\}$

Notice that there is no number so large that beyond that number of b tokens divisibility by 2 ceases to matter. However many occurrences of b may occur in a string in this set, the same string with one extra b will not be in the set. To define the set you have to be able to count modulo 2. But in SL stringsets you can't count modulo n for any $n > 1$.

5.4 Model-theoretic characterization

One other distinction of the star-free stringsets that is particularly important in our view, though we touch on it only very lightly here. There is a particularly simple way of characterizing these stringsets using the model-theoretic techniques used to provide semantics for logical languages.

Strings can be equated with relational structures of the sort familiar in model theory. A relational structure is a set of elements D known as the domain together with a set $\mathcal{R}$ of relations on D. Thus a graph is a relational structure: D is the set of nodes, and there is one relation in $\mathcal{R}$, the relation that holds between two nodes if they are immediately connected by a edge in the graph. A string of symbols can be seen as a finite graph in which the edge relation strictly orders the domain and a set of unary relations (i.e., properties, one for each distinct symbol) partition it. Call these **string structures**.

[5]The term used in McNaughton & Papert (1971) is 'counter-free'. We avoid this term simply because its initials are unfortunately the initials of the phrase 'context-free', which we use frequently below. The term 'aperiodic' may also be encountered in the literature.

Consider the string *abab*. It corresponds to a string structure with $D = \{n_1, n_2, n_3, n_4\}$ and $\mathcal{R} = \{\prec, P_a, P_b\}$, where $\prec$ (binary) is immediate precedence (we assume $n_1 \prec n_2$, $n_2 \prec n_3$, and $n_3 \prec n_4$), P_a picks out the *a*s (so it has the extension $\{n_1, n_3\}$), and P_b picks out the *b*s (its extension is $\{n_2, n_4\}$). Various first-order formulae will be true in this structure; for example, the closed formula $\forall x \exists y[P_a(x) \Rightarrow (x \prec y \wedge P_b(y))]$ says that every a immediately precedes an occurrence of b, and the structure just described satisfies this condition.

For any first-order sentence, there will be some set of finite string structures that satisfy it. The following remarkable result about first-order logic on string models was proved in McNaughton & Papert (1971) (for more modern approaches to proving the result, see Straubing 1994, Ebbinghaus & Flum 1999, and Libkin 2004):

(14) A stringset is star-free iff it corresponds to the set of all the finite string structures satisfying some closed formula of first-order logic.

Thus the star-free stringsets can be characterized exactly by first-order logic descriptions interpreted on string structures. And we can put that another way: having the ability to learn arbitrary star-free stringsets from presented examples would be tantamount to being able to learn *any arbitrary property of strings that is definable in the first-order predicate calculus*. This would be a truly remarkable ability to attribute to any animal.

Yet so far we are only talking about the star-free stringsets. The finite-state stringsets are a superset with a still richer structure. We now turn to them.

6 Finite-state

The asteration operation permits the definition of a proper superset of the star-free stringsets, and of all the other classes just discussed, the **finite-state** or **regular** stringsets, first defined by Kleene (1956) under the name 'regular events' (he was thinking of events in neural nets). These have a large array of distinct characterizations that have been of enormous importance in theoretical computer science and continue to be important in computational linguistics. They are much better known than the former stringset classes, so we will just very briefly summarize.

6.1 Definition

The class under discussion can be characterized directly by means of an inductive definition, using singleton stringsets as the basis:

(15) **Definition**: Regular stringsets
The class of regular stringsets over a vocabulary Σ is the smallest class of stringsets that (i) includes the empty set, the singleton set containing the empty string, and for each symbol in Σ the singleton set containing only that symbol, and (ii) is closed under union, concatenation, and asteration.

6.2 Automaton characterization

Now consider the (apparently) quite distinct notion of a **finite-state automaton** — the most powerful possible computing device that is finite in every respect including the total amount of working memory used in computations, and thus in a sense a model for any physical computing device we can build in a finite universe.

(16) **Definition**: Finite-State Automaton.
A finite-state automaton (FSA) is a system of five components:

— a finite set Q of states,

— a finite set Σ of symbols,

— a function T from $Q \times \Sigma$ to Q,

— a distinguished start state $q_0 \in Q$, and

— a distinguished set of final states $F \subseteq Q$.

The intuition here is that the automaton corresponds to a machine that starts in state q_0 reading a string of symbols from Σ on a tape or other ordered digital medium, changing state as dictated by the function T (if the value of $T(q_i, \sigma_k)$ is q_j, then reading σ_k while in state q_i makes the machine switch to state q_j). A sequence of states that the machine goes through while reading a string is called a *run*, and if it ends with a state that belongs to F, it is a *successful* run. The stringset recognized by an FSA $\mathcal{A}$ is $\mathcal{A} = \langle Q, \Sigma, T, q_0, F\rangle$ — that is, the set of all and only those strings $\sigma_1 \ldots \sigma_n$ over Σ on which $\mathcal{A}$ has a successful run.

We call a stringset **finite-state** (FS) iff there is an FSA which recognizes it. Kleene (1956) connected the notion of recognizability by FSA to the class of regular stringsets defined above:

(17) **Theorem** (Kleene)
The finite-state stringsets (those recognizable by finite-state automata) are exactly the regular stringsets.

6.3 Example

An example of a finite state stringset that is not SF is $(aa)^*$. This is the set of all even length strings containing just a. That is, it is simply the asteration of the result of concatenating two copies of the singleton stringset $\{a\}$. To see that it is not SF, note that, for any n, either the string aa^na or aa^naa is in the stringset but, in both cases, the result of iterating the a one additional time is not. Similarly, example 13 of Section 5.3 is regular.

6.4 Abstract characterization

There is a more abstract algebraic characterization of the same class. We start with the notion of Nerode Equivalence, which is an equivalence relation involving sharing of continuation strings:

(18) **Definition**: Nerode Equivalence
A string w is **Nerode equivalent** to u with respect to a stringset $L \subseteq \Sigma^*$, and we write $w \equiv_L u$, iff for all strings $v \in \Sigma^*$, $wv \in L \Leftrightarrow uv \in L$.

Intuitively, two strings are Nerode equivalent with respect to L iff they have exactly the same set of possible continuations in L — any continuation to one of them that would make a string in L would also work for the other, and any continuation string that would make one of them ungrammatical in L would do the same to the other. We now define the **Nerode equivalence class** of a string w with respect to a stringset L as the set of all strings u that are Nerode-equivalent to it (with respect to L):

(19) **Definition**: Nerode Equivalence Class
The *Nerode equivalence class* of w with respect to L is $[w]_L \stackrel{\text{def}}{=} \{u \in \Sigma^* \mid w \equiv_L u\}$.

So the Nerode equivalence class with respect to L for a string contains all and only the strings that are Nerode equivalent to it with respect to L.

Now we can state an additional characterization of the FS stringsets in terms of how many Nerode equivalence classes there are (this characterization derives from the results reported in Myhill 1957 and Nerode 1958):

(20) Myhill–Nerode Theorem
A stringset $L \subseteq \Sigma^*$ is finite-state iff $\equiv_L$ partitions Σ^* into finitely many Nerode equivalence classes.

This says that a stringset over Σ is FS iff its Nerode equivalence relation breaks up the entire universe of strings over Σ into just a finite number of Nerode equivalence classes. That will mean that you can make a finite list of the kinds of continuation string and say for each whether they will lead to grammaticality or ungrammaticality in L. For example, let $\Sigma = \{a, b\}$ and $L = a^*b^*$. Then there are just Nerode equivalence classes: (i) strings in a^* (these can be continued with with any string in a^*b^* to yield a string in L); (ii) strings in a^*b^+ (these can be continued with with any string in b^* to yield a string in L); and (iii) all other strings (any continuations of these always lead to strings not in L). Since the number of Nerode classes is finite, L is FS.

6.5 Grammar characterization

Another characterization of the FS stringsets can be given in terms of a rewriting system or generative grammar. A rewriting system consists of a finite set N of nonterminals (categories), a finite set Σ of terminals (symbols or words that appear in strings), a designated start symbol S that belongs to N, and a finite set of rules. The **regular grammars** are those rewriting systems in which the rules are all of the following two forms (where $X, Y \in N$, $\sigma \in \Sigma$, and '$\rightarrow$' means 'can be rewritten as'):

(21) a. $X \rightarrow \sigma Y$ b. $X \rightarrow \sigma$

When a grammar has rules only of the forms shown in (21), the set of all and only those sequences of symbols in Σ that can be obtained by starting with S and rewriting according to rules chosen at random will always be FS, and every stringset can be described by some grammar of this form. (In fact such a grammar can be constructed directly from an FSA in such a way that it will generate the stringset that the FSA recognizes. It is not too hard to see the similarity between rules and automaton instructions: a rule '$X \rightarrow \sigma Y$' means that reading σ while in state X makes the machine switch to state Y.)

Grammars of this sort can also be called **finite-state grammars**, and for convenience when quoting Hauser et al. below, we'll follow them in using this term, and also the abbreviation FSG. The following are two examples of rewriting rule sets for FSGs:

(22) a. $S \rightarrow aA,\ A \rightarrow aS,\ A \rightarrow a$

b. $S \rightarrow aA, S \rightarrow bS, S \rightarrow b, A \rightarrow aS, A \rightarrow bA, A \rightarrow a$

The grammar with the rules in (22a) generates all and only the even-length strings of repetitions of a. The rules in (22b) generate all and only the strings over $\{a, b\}$ with an even number of a occurrences. Those stringsets are thus shown to be FS.[6]

6.6 Canonical example of non-membership

The classic case of a configuration that identifies a stringset as not being FS is that of a dependency between a certain number n of symbol occurrences that need to be matched by exactly n occurrences of another symbol elsewhere in the string. The simplest such case is $\{a^n b^n \mid n \geq 0\}$, for which the syntactic requirement is simply that there be n occurrences of a followed by exactly the same number of bs. The Nerode equivalence relation for this stringset yields infinitely many equivalence classes (in fact every string of the form $a^i b^j$ is in its own unique equivalence class: if $a^i b^j w$ is in the set for some $i, j \geq 0$ and so is $a^i b^j u$, then $w = u$), so by the Myhill–Nerode theorem it cannot be FS. Other examples of non-FS stringsets include the set of palindromes over a given vocabulary or alphabet (the strings that read the same forwards as they do backwards), and the set of non-palindromes.

6.7 Model-theoretic characterization

As with the star-free stringsets, the FS stringsets have a logical characterization that we will state without detailed discussion. It makes reference to *weak monadic second-order logic* (wMSO), which is like first-order predicate calculus except that there are additional variables for quantifying over finite sets of elements of the domain. Büchi (1960) proved the following theorem:

(23) A stringset is finite-state iff it corresponds to the set of all the finite string structures satisfying some closed formula of weak monadic second-order logic.

Again, then, what this means is that being able to learn arbitrary FS stringsets from presented examples would be tantamount to being able to induce *any arbitrary property of strings that can be defined in weak monadic second-order logic*. This seems to us extraordinarily unlikely for any species of animal.

7 Identification in the limit

There is one obvious, very specific sense in which it is certainly impossible to have a general capacity to learn FS stringsets: the class FS is not 'identifiable in the limit from text' in the technical sense of Gold (1967).

Identifying a stringset in this context is trivial: to identify a stringset is simply to output a description for it. So every r.e. stringset can be identified instantly by an algorithm: if G is a grammar that generates L, an algorithm which simply outputs G is an algorithm that identifies L. Matters only become non-trivial when we talk about classes of stringsets, and the question posed is whether some procedure can detect for any stringset in the class which stringset it is, and identify it by (for example) naming a generative grammar for it.

A class $\mathcal{L}$ of stringsets is identifiable in the limit from text iff there exists a single mechanical procedure meeting this condition: When presented with an infinite sequence of strings from an arbitrary target stringset L in $\mathcal{L}$ — that is, a sequence that contains each string in L at some point (possibly with repetitions) — produces a sequence of 'guesses' at the correct description for L (one after each presented string), guaranteed to converge after a finite amount of time on a correct guess — a fully accurate description of L — and never diverge from it after that, no matter what further strings are presented.

Gold's proof that the FS class is not identifiable depends on an elementary fact about the class: it is **superfinite**, which means it contains every finite stringset over Σ plus some infinite ones as well. His theorem to the effect that no superfinite class is identifiable in the limit from text implies that the SF, LT, and SL classes aren't identifiable either. Notice, though, that his proof does not go through for the SL_k or LT_k class given some fixed k, because each such class excludes some finite stringsets.

In fact it is easy to see that for any k, there is an algorithm that can be guaranteed to eventually identify from text the correct description for any SL_k stringset. The procedure is just to keep recording k-length factors and adding them to the current guess at what the right description is. For example, if the SL_3 identification algorithm were presented with the string *abba*, it would add to its currently guessed description the factors $\rtimes ab$, *abb*, *bba*, and $ba \ltimes$, and guess that the result is correct. The set of factors will simply grow until all the right factors are in the description, and if the algorithm is

[6] We are now in a position to provide something that we did not provide in section (3) because we had not introduced grammars: we can give a simple grammar characterization for the SL_2 class. A regular grammar generates an SL_2 stringset iff the relation 'is followed by in some rule' on the symbol set is a function — that is, if for any given terminal there is at most one nonterminal that can follow it in the right hand side of a rule.

presented with a text in Gold's special sense then eventually the set of factors will be the correct one, and from then on it will not undergo any further changes (because it will already contain every 3-length factor of every string in the set). For more detail on this see García & Vidal (1990).

The LT_k class turns out also to be learnable from positive data for any fixed k, although the matter is more complex; see García and Ruiz (2004).

No animal is going to learn patterns by identifying stringsets in the manner suggested by Gold's paper, which involves a brute force procedure of working methodically through an enumeration of an entire (typically infinite) class of grammars, ruling them out one after another until the right one is found. His method was never intended to emulate a learning process; it is a tool for showing that in principle successful learning processes for the class do exist (or that they don't). But for any experimenter on pattern learning by animals it is worth keeping in mind what we know about this area so far. For every fixed k, the entire class of stringsets with SL_k descriptions is identifiable, so we know that in that sense we are in the realm of stringsets for which inductive learning is possible in principle. The same is true for LT_k. But for the classes SL, LT, SF, FS, and classes higher in the hierarchy, learning in the sense of identification in the limit from text — roughly, effective induction on the basis of a finite sequence of exemplars — is impossible in principle.

8 Iteration, recursion, and infinitude

There is a misconception found in the literature to the effect that the finite-state stringsets allow for only local relationships and cannot accommodate long-distance dependencies between symbols. In Fitch & Hauser (2004) we read:

> The weakest class in [the Chomsky] hierarchy are finite state grammars (FSGs), which can be fully specified by transition probabilities between a finite number of "states" (e.g., corresponding to words or calls). ... In addition to concatenating items like an FSG, a PSG can embed strings within other strings, thus creating complex hierarchical structures ("phrase structures"), and long-distance dependencies.

Set aside the fact that standard FSGs do not depend on or keep track of probabilities; that is just a matter of replacing probabilities (real values between 0 and 1) with possibilities (0 or 1). However, it is not true that the states in an FSA or the nonterminals in an FSG correspond to words or symbols in a string. An automaton may be able to move to any of a number of different states on reading a certain symbol, and may have many different symbols that it can read while in a given state.[7] It is also not really true that FSGs do not just "concatenate items"; crucially, they can iterate on sequences. In consequence, complex strings can be embedded within other strings, and complex unbounded dependencies may hold.

For example, Pullum & Gazdar (1982) point out that a finite-state grammar can easily describe an infinite subset of English in which fronted *wh*-phrases agree with verbs arbitrarily far away from them, as seen in the contrast between such pairs as these:

(24) *Which girls do they think that you think that he thinks (...) were responsible?*

Which girl do they think that you think that he thinks (...) was responsible?

The finite-state stringsets are in fact a very rich and diverse class. Imagine setting an animal (or indeed, a human being) the task of learning the pattern shared by the following strings:

(25) *lo me la*
lu me lu
lo ki ki la
lu me ki ki lu
lo me ki me ki la
lu me me ki ki lu
lo me ki me me ki me la
lu me me ki me ki me lu
lo ki me me ki me ki me ki me la
lu me ki ki me me ki me ki me lu
lo me ki me me me ki me ki me ki me la
lu me me ki me me me ki me me ki me ki me lu
...

Here is the intended solution: these strings are all of the form $lo \cdots la$ or $lu \cdots lu$ where '$\cdots$' is a sequence of arbitrary length (n.b., an unbounded dependency) composed of *ki* and/or *me* in which the count of (not necessarily adjacent) tokens of *ki* is exactly divisible by 2. It seems highly unlikely that any animal would be able to learn this pattern. But the infinite stringset it characterizes is finite-state: if we let

$$R = (\{me\}^* \cdot \{ki\} \cdot \{me\}^* \cdot \{ki\} \cdot \{me\}^*)^*$$

then the stringset in question is the union of $\{lo\} \cdot R \cdot \{la\}$ and $\{lu\} \cdot R \cdot \{lu\}$, which is easily seen to meet the definition of regular stringsets in (15).

The most important contrast studied in recent animal experimentation on stringset learning, particularly with respect to cotton-top tamarins, has been the contrast between stringsets isomorphic to $(ab)^+$ (containing strings like *abababab*) and stringsets isomorphic to $\{a^n b^n | n \geq 1\}$ (containing strings like *aaaabbbb*). The former has been taken to be typical of the FS stringsets, and the latter of those that are CF. But as we have seen, the finite-state stringsets are a nowhere near the bottom of the hierarchy of stringset classes. And $(ab)^+$ is in *all* of the classes we have discussed, all the way down to SL_2. (A bigram set that defines it is $\{\rtimes a, ab, ba, b\ltimes\}$.)

[7] If an FSG meets the condition mentioned in footnote 6, however, then in a sense the states do correspond to terminals — for a given terminal there is only one state that can be next. This is another indication that although Fitch & Hauser talk about FSGs, they may tacitly have SL stringsets in mind.

So it can hardly be thought a representative member of the class of finite-state stringsets. There is no reason to think, simply on the basis that an animal has been shown to be capable of recognizing a pattern like $(ab)^+$, that it is capable of learning anything beyond strictly 2-local sets, a very limited class of stringsets indeed. Far from being an established lower bound, the finite-state stringsets might be far and away beyond the animal's learning capacity.

Similarly, the CF stringsets, as a class, may be far and away beyond the learning capacity of human beings. The class of CF stringsets includes such stringsets over $\{a, b, c\}$ as the following (and of course infinitely many others, some much more complicated):

(26) $\{w_1 \cdots w_k \mid \forall i \leq k \, \exists n \geq 0[(w_i = a^n b^n) \vee (w_i = b^n c^n)]\}$
(all and only those strings made up of consecutive substrings each composed either of zero or more *a*s followed by an equal number of *b*s or of zero or more *b*s followed by an equal number of *c*s)

(27) $\{cw_1c \cdots cw_kc \mid \forall i \leq k[\#_a(w_i) = \#_b(w_i)]\}$
(all and only those strings composed of an arbitrary number of *c*-separated substrings over $\{a, b\}$ each having a number of *a* occurrences equal to the number of *b* occurrences)

(28) $\{xcy \mid x \in \{a, b\}^* \wedge y \in \{a, b\}^* \wedge x \neq y\}$
(all and only those strings in which nonidentical initial and final substrings over $\{a, b\}$ are separated by a single *c*)

Whether humans could be said to be capable of learning patterns of arbitrary CF types such as these has not been investigated.

Fitch & Hauser (2004) do report success in teaching college students to recognize the pattern $\{a^n b^n \mid n \geq 1\}$. Perruchet & Rey (in press) are skeptical, and claim to have shown that humans do not in fact learn the pattern involved (as opposed to spotting which of a very small finite set of strings are the ones the investigator is concerned with). This seems plausible to us, in light of the well-known fact that humans find processing such patterns is almost impossible for $n \geq 3$. Consider this string, of $a^n b^n$ form:

(29) *People people people left left left.*

Few English speakers see this as grammatical before the trick has been explained to them. The trick is to see that *left* can mean either "departed" or "'abandoned", and *people people left* can mean *people whom people abandoned*. The sentence in (29) is standardly regarded as grammatical, but mainly for theoretical reasons. For example, it corresponds to the perfectly acceptable passive *People* [*who were*] *left by people* [*who were*] *left by people left*, with the fully understandable meaning "People who were abandoned by people who were abandoned by people departed." The standard view is that expressions like (29) exemplify a structural regularity that the grammar of English has to permit but that the parsing capabilities of human beings cannot cope with. (A very interesting non-standard view is provided by Kornai 1985, who regards it as completely obvious that strings of the form $People^n \; left^n$ where $n \geq 4$ are not grammatical, and provides arguments that the stringset of English is SF but not LT.)

The stringsets on which Fitch and Hauser (2004) actually tested both their animals and their human subjects were finite (and very small) sets — sets with the homomorphic images $\{abab, ababab\}$ and $\{aabb, aaabbb\}$. There is no language-theoretic complexity difference here: an SL_7 description suffices in each case. It is only the infinite extensions to $(ab)^+$ and $\{a^n b^n | n \geq 1\}$, respectively, that separate them in complexity terms; and they turn out to be separated very widely indeed.

Hauser, Chomsky & Fitch (2002) appear to hold that the generalization to infinite extensions is the fundamental hallmark of the unique capacity for language differentiating humans from all other animals. The key notion is "recursion". They give nothing detailed about what this means, but on on p. 1571 they say this about the human faculty of language in the narrow sense ("FLN"):

> All approaches agree that a core property of FLN is recursion, attributed to narrow syntax in the conception just outlined. FLN takes a finite set of elements and yields a potentially infinite array of discrete expressions.

This suggests that recursion is nothing more than whatever permits the generation of infinite stringsets on finite vocabularies. They go on to say:

> Natural languages go beyond purely local structure by including a capacity for recursive embedding of phrases within phrases, which can lead to statistical regularities that are separated by an arbitrary number of words or phrases. Such long-distance, hierarchical relationships are found in all natural languages for which, at a minimum, a "phrase-structure grammar" is necessary. It is a foundational observation of modern generative linguistics that, to capture a natural language, a grammar must include such capabilities . . .

This suggests that recursion is whatever distinguishes the non-FS CF stringsets from the FS.

What seems to us to be the standard interpretation of the term 'recursion' in formal language theory is that it refers to the rewriting of a string $\phi_1 A \phi_2$ so that at a later (not necessarily immediately following) stage it looks like $\phi_1 \psi_1 A \psi_2 \phi_2$, where ψ_1 and ψ_2 are not both empty and A is a nonterminal that yields terminals in at least some derivations of this sort and so do at least some of the nonterminals in either ψ_1 or ψ_2. In other words, an A constituent is allowed to be properly contained in another, longer, A constituent.

The property of exhibiting recursion in this sense distinguishes grammars that generate infinite stringsets from those that generate only finite stringsets. As such, it cannot be characteristic of the distinction between mechanisms that can learn only FS stringsets and those that can learn CF stringsets. Both of the regular grammars in (22) are recursive

in this sense. Indeed, the same is true for every grammar for a non-finite SL stringset.

The notion of recursion that actually distinguishes regular from CF stringsets is the property called 'self-embedding' in Chomsky (1959). Self-embedding involves derivations that go from $\phi_1 A \phi_2$ (not necessarily directly) to $\phi_1\psi_1 A \psi_2\phi_2$, where *neither* ψ_1 *nor* ψ_2 is empty and A yields terminals in at least some derivations of this sort and so do at least some of the nonterminals in both ψ_1 and ψ_2. Self-embedding could be said to set up potentially unbounded dependencies: a grammar with the rules $S \rightarrow aSb$ and $S \rightarrow ab$ produces derivations like

$$S \rightsquigarrow aSb \rightsquigarrow aaSbb \rightsquigarrow aaaSbbb \rightsquigarrow aaaabbbb$$

in which for each $j \geq 1$ it could be said that the presence of a b at a position j symbols to the right of the middle of a string in the set depends on there being an a at a position j symbols to the left of the middle, with no upper bound on j, and thus no upper bound on the distance between the two.

But the presence of self-embedding in a grammar is not in itself a sufficient condition to yield a non-finite-state stringset. Consider a CF grammar containing (at least) these rules:

(30) a. $S \rightarrow ABC$
b. $A \rightarrow a$
c. $B \rightarrow bBb$
d. $B \rightarrow bB$
e. $B \rightarrow b$
f. $C \rightarrow c$

Such a grammar has the self-embedding property. It not only embeds phrases within phrases, it embeds phrases labeled B within larger phrases labeled B, with material both to the left and to the right (notice rule c), and non-trivially so, since all nonterminals yield terminal strings. Certainly this grammar can be said to allow for configurations that are "separated by an arbitrary number of words or phrases": every well-formed string must begin with a that is followed by a final c, with indefinitely many bs between (an unbounded dependency). Yet (as the reader can determine by a little experimentation with bigrams) the stringset generated by this grammar (the concatenation of $\{a\}$, $\{b\}^+$, and $\{c\}$) is not just finite-state but in fact SL_2. We do not offer this as an important or surprising fact; we are merely underlining the point that notions like 'recursion', 'embedding', and 'unbounded dependency' need to be much more carefully defined than they have been in some of the recent literature.

The condition on CF grammars that guarantees that a stringset will not be finite-state is that if *all* the CF grammars for some stringset have the self-embedding property, then the stringset is not FS. This turns out to be a particularly difficult condition to work with. It is true that often one can prove of a stringset either that it is FS (and hence does not require self-embedding) or that it is CF (and hence does require it); but there is no algorithm for determining whether arbitrary CF grammars generate FS stringsets or not. That is, whether a given CF stringset has a non-self-embedding grammar is not algorithmically decidable.

Merely pointing out that recursion "yields a potentially infinite array" or allows configurations to be "separated by an arbitrary number of words or phrases" is not enough to delineate a difference between animal pattern-learning capacities and human linguistic abilities. Many of the distinctions that these experiments appear to be designed to illuminate, turn out, in fact, to be characteristic of distinctions between classes at the very bottom end of this range. We hope that this brief survey may inspire some research exploring some of the territory between SL and the finite-state stringsets.

9 Some criterial contrasts

We close with a short list of crucial cases of stringsets that distinguish between classes mentioned above, together with an indication of the cognitive ability to which they intuitively correspond.

Strictly Local vs. Locally Testable

— While $(ab)^+$ is strictly local, $a^+(ba^+)^+$ is locally testable but not strictly local.

— Criterial test pair: an SL_k description cannot distinguish between all cases of $a^k b a^k$ and a^{2k+1}, but an LT_k description can. That is, acceptance cannot require the presence of a b unless it occurs within the first or last k stimuli.

— Psychological correlate: ability to recognize that every a is immediately followed by a b versus ability to detect that at least one b was present somewhere.

Locally Testable vs. Star-Free

— While $a^+(ba^+)^+$ is locally testable, a^+ba^+ is star-free but not locally testable.

— Criterial test pair: an LT_k description cannot distinguish between all cases of $a^k b a^{2k+1}$ and $a^k b a^k b a^k$ (it cannot guarantee that there will only one b), but an SF description can.

— Psychological correlate: ability to recognize whether a b is present versus ability to recognize whether b occurred just once, or at most k times for some $k > 1$.

Star-Free vs. Finite-State

— While the set $\{a, b\}^*$ of all strings consisting of a and b is star-free (despite the star we use for convenience in denoting it here!), the set

$$\{w \mid w \in \{a, b\}^* \wedge \#_b(w) \cong 0 \bmod 2\}$$

(which contains all and only the strings consisting of a and b that have an even number of b occurrences) is finite-state but not star-free.

— Criterial test pair: a star-free description cannot distinguish between all cases of $(a^k b)^{2j+1} a^k$ and $(a^k b)^{2j} a^k$; a finite-state description (e.g., a finite-state automaton) can.

— Psychological correlate: ability to recognize whether b occurred at least k times versus ability to recognize whether b occurred a number of times that is divisible by some number n (i.e., to count up to some threshold n and then reset the counter).

Finite-State vs. Context-Free

— While the set $\{a^i b^j \mid i + j \cong 0 \bmod 2\}$ of all strings consisting of an even total of a and b occurrences is finite-state, the set $\{a^i b^j \mid i = j\}$ of all strings consisting of an equal number of a and b occurrences is CF but not FS.

— Criterial test pair: a FS description cannot distinguish between all cases of $a^{i-k}b^{i+k}$ and $a^i b^i$ (for all i and for all k); but a CF description can.

— Psychological correlate: ability to do modulo arithmetic versus ability to match brackets, or to do arbitrary integer addition in unary.

Each of these contrasts indicates a point at which some relevant distinction between cognitive capacities might be located, and thus suggests the topic for a potentially rewarding experiment.

References

Büchi, J. Richard. 1960. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **6**: 66–92.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* **2**: 137–167.

Chomsky, Noam and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg (eds.), *Computer Programming and Formal Systems*, 118–161. Amsterdam: North-Holland.

Ebbinghaus, Heinz-Dieter and Jörg Flum. 1999. *Finite Model Theory*. Berlin: Springer.

Fitch, W. Tecumseh and Hauser, Marc D. 2004. Computational constraints on syntactic processing in a nonhuman primate. *Science* **303**, 5656 (16 January 2004), 377–380.

Gentner, Timothy Q. 2005. Recursive syntactic pattern learning in songbirds. Presented at the Society for Language Development's Annual Society Symposium on Prerequisites to Language in Animal Cognition, Boston University, 3 November 2005.

García, Pedro and Enrique Vidal. 1990. Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition. *IEEE Transactions on Patern Analysis and Machine Intelligence* **12** (9): 920–925.

García, Pedro and José Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars* **7**: 125–140

Gold, Mark. 1967. Language identification in the limit. *Information and Control* **10**: 447–474.

Hauser, Marc D. 2005. The evolution of the language faculty: semantics, syntax, and interfaces. Presented at the Society for Language Development's Annual Society Symposium on Prerequisites to Language in Animal Cognition, Boston University, 3 November 2005.

Hauser, Marc D., Noam Chomsky, and W. Tecumseh Fitch. 2002. The faculty of language: What is it, who has it, and how did it evolve? *Science* **298**, 5598 (22 November 2002): 1569–1579.

Kleene, Stephen C. 1956. Representation of events in in nerve nets and finite automata. In Claude E. Shannon and J. McCarthy (eds.), *Automata Studies*, 3–42. Princeton, NJ: Princeton University Press.

Kornai, Andràs. 1985. Natural languages and the Chomsky hierarchy. *Proceedings of the 2nd European Conference of the Association for Computational Linguistics*, ed. by Margaret King, 1–7. Facsimile reproduction at `http://acl.ldc.upenn.edu/E/E85/E85-1001.pdf`.

Kozen, Dexter. 1997. *Automata and Computability*. Berlin: Springer.

Libkin, Leonid. 2004. *Elements of Finite Model Theory*. Berlin: Springer.

McNaughton, Robert and Seymour Papert. 1971. *Counter-Free Automata*. Research Monograph No. 65. Cambridge, MA: MIT Press.

Myhill, John. 1957. Finite automata and the representation of events. WADD TR-60-165, 112–137. Wright Patterson Air Force Base, Dayton, OH.

Nerode, Anil. 1958. Linear automaton transformations. *Proceedings of the American Mathematical Society* **9**: 541–544.

O'Donnell, Timothy J., Marc D. Hauser and W. Tecumseh Fitch. 2005. Using mathematical models of language experimentally. *TRENDS in Cognitive Sciences* **9**.6: 284–289.

Perruchet, Pierre and Arnaud Rey. In press. Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates? To appear in *Psychonomic Bulletin and Review*.

Pullum, Geoffrey K. and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and Philosophy* **4**: 471–504.

Schützenberger, M.-P. 1965. On finite monoids having only trivial subgroups. *Information and Control* **8**: 190–194.

Straubing, Howard. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Boston: Birkhäuser.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences* **25**: 360–376.