

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Aural pattern recognition experiments and the subregular hierarchy

Citation for published version:
Rogers, J & Pullum, GK 2011, 'Aural pattern recognition experiments and the subregular hierarchy', *Journal of Logic, Language and Information*, vol. 20, no. 3, pp. 329-342. https://doi.org/10.1007/s10849-011-9140-2

Digital Object Identifier (DOI):
10.1007/s10849-011-9140-2

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Journal of Logic, Language and Information

Publisher Rights Statement:
Rogers, J., & Pullum, G. K. (2011). Aural Pattern Recognition Experiments and the Subregular Hierarchy. Journal of Logic, Language and Information, 20(3), 329-342, doi: 10.1007/s10849-011-9140-2 The final publication is available at link.springer.com

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 28. Mar. 2024

Aural Pattern Recognition Experiments and the Subregular Hierarchy

James Rogers[1] and Geoffrey K. Pullum[2]

[1] Dept. of Computer Science, Earlham College
[2] Linguistics & English Language, University of Edinburgh

Abstract. We explore the formal foundations of recent studies comparining aural pattern recognition capabilities of populations of human and non-human animals. To date, these experiments have focused on the boundary between the Regular and Context-Free stringsets. We argue that experiments directed at distinguishing capabilities with respect to the Subregular Hierarchy, which subdivides the class of Regular stringsets, are likely to provide better evidence about the distinctions between the cognitive mechanisms of humans and those of other species. Moreover, the classes of the Subregular Hierarchy have the advantage of fully abstract descriptive (model-theoretic) characterizations in addition to characterizations in more familiar grammar- and automata-theoretic terms. Because the descriptive characterizations make no assumptions about implementation, they provide a sound basis for drawing conclusions about potential cognitive mechanisms from the experimental results.
We review the Subregular Hierarchy and provide a concrete set of principles for the design and interpretation of these experiments.

1 Introduction

The last several years have seen a number of intriguing experiments aimed at differentiating the aural pattern recognition capabilities of various species [1–4]. Aside from the evidence these provide about potentially unique characteristics of human languages, when they contrast human capabilities with those of our evolutionary cousins they provide indirect evidence about those cognitive faculties of our common ancestors which may have provided the foundation for the evolution of human language.

The connection between the experimental data and the hypothesized cognitive mechanisms is provided by formal language theory (FLT). Characterizations of the structure of language classes (such a Nerode-style characterizations and pumping lemmas) have motivated the choice of stimulus patterns the experiments employ, and characterizations of those classes in terms of abstract computational mechanisms (such as grammars and automata) have provided the basis for inferences about the properties of the cognitive mechanisms in play.

Nearly all of the FLT employed in this work is based in the Chomsky Hierarchy (CH), but there are a number of reasons for questioning whether this is

really the right place to be looking. To begin with, viewed as an instrument in the context of these experiments, the CH seems to lack resolution. What is typically taken as emblematic of Context-Free Languages (CFLs) is far too hard for humans (processing $\{\mathbf{people}^n\ \mathbf{left}^n \mid n \geq 1\}$ rapidly outstrips any human being's aural pattern recognition abilities), while what is taken as representative of regular languages is far too easy (being able to recognize $\{(\mathbf{ding\ dong})^n \mid n \geq 1\}$ should hardly count as evidence of ability to handle arbitrary regular languages). Beyond that, grammar and automata theoretic classes seem to presuppose specific types of structure or specific classes of recognition mechanisms, raising questions about whether these are necessarily relevant to the cognitive mechanisms under study.

There is, in fact, a rich hierarchy of classes of stringsets which subdivides the class of Regular (Finite-State) stringsets. In addition to providing finer resolving power in the range of capabilities that seem to be relevant to these studies, this subregular hierarchy includes some of the most well-developed examples of **descriptive** characterizations of language-theoretic complexity classes [5–9].

Descriptive (model-theoretic) characterizations focus on the nature of the information about the properties of a string (or structure) that is needed in order to distinguish those which exhibit a pattern from those which do not. Consequently, they are independent of specific mechanisms for specifying or recognizing the patterns. Any mechanism which can handle only patterns which can be described within the means of a particular descriptive class will have no need to be sensitive to anything other than the kind of information that characterizes that class. Conversely, any mechanism which is is sensitive to that kind of information will be capable, in principle, of handling the range of patterns which fall within the descriptive class.

The descriptive characterizations, however, do not stand alone. These classes are also characterized by a range of abstract mechanisms including grammars, automata and, in some cases, artificial neural networks. These characterizations in more traditional FLT terms provide a means of reasoning about the structural properties both of the individual stringsets that satisfy a pattern and about the class of stringsets that can be described within the means of the class.

Note that the grammar and automata theoretic characterizations do not determine the actual form of a cognitive mechanism for recognizing patterns within the class. The fact that there are many equivalent characterizations of each class implies that no conclusions can be drawn about the specific details of the mechanism. What one can conclude, on the other hand, is that whatever the actual mechanism is it must be sensitive to the kind of information that characterizes the descriptive class and that the class of patterns that it can recognize will exhibit those properties that are characteristic of the class.

Thus classes that can be characterized both descriptively and automata or grammar theoretically, while not presupposing any concrete details of the mechanism, serve both ends of these aural pattern recognition studies. By providing characterizations of the structure of the stringsets that satisfy particular classes of patterns, they provide a means of designing experimental protocols that can

resolve contrasts between the cognitive capabilities of the subjects. By providing highly abstract characterizations of the mechanisms which can process particular classes of patterns they provide a means of drawing conclusions about the differences in the capabilities of cognitive mechanisms that can account for observed differences in recognition abilities.

In this paper we revisit the subregular hierarchy paying particular attention three aspects: the broad generality of the descriptive characterizations of the classes, the consequences their structural characteristics have for the design of pattern recognition experiments and the nature of the conclusions about the cognitive mechanisms involved that these experiments can support.

While most of the FLT results are either already well established or could probably be best attributed to folklore, we offer two substantiative methodological contributions. The first is specific to the current experiments in aural pattern recognition. There is good reason to suspect that distinctions between human capabilities in this realm and the capabilities of our evolutionary cousins may occur within this range of complexities. Heinz [10], for example, has shown that a wide range of stress patterns in human languages fall strictly within the Regular stringsets. We believe that experiments directed at distinguishing capabilities with respect to the subregular hierarchy may provide a great deal of evidence about the distinctions between the cognitive mechanisms of humans and those of other species. We provide a concrete set of principles for the design and interpretation of these experiments.

Our second methodological contribution is more general. We believe that our allocation of roles between the descriptive and mechanism-oriented characterizations, in which conclusions about potential cognitive mechanisms are based only on the more abstract descriptive characterizations and the mechanism-oriented characterizations are reserved for providing structural information about the definable sets, provides a clear foundation on which to interpret experimental results that is applicable to a wide range of pattern-based experiments. As the papers we cite above demonstrate, there has often been considerable variation in the conclusions that have been drawn from similar evidence and this appears to spring from variation in assumptions about the necessary properties of mechanisms capable of distinguishing a given class of stringsets. As we will show, many of these assumptions are unwarranted.

2 The Subregular Hierarchy

We refer to the 'subregular hierarchy' for brevity, but each of the classes in the hierarchy is actually the closure of an infinite hierarchy of subclasses, some of which can be further refined into sub-hierarchies. We concentrate on the four main classes.

2.1 Strictly Local Stringsets

The base of the hierarchy is the class of **Strictly Local** (SL) stringsets, those that can be distinguished simply on the basis of which symbols occur adjacently.

A Strictly k-Local description is a set of **k-factors** (or k-grams), length k sequences of symbols drawn from the alphabet augmented with start and end symbols. The stringset such a description defines is the set of strings which include only k-factors from the set. Formally, given a string w and a length k, the set of k-factors of w is:

Definition 1 (k-factors).

$$F_k(w) \stackrel{def}{=} \begin{cases} \{y \mid w = x \cdot y \cdot z, \quad x, y, z \in \Sigma^*, |y| = k\} & \text{if } |w| > k, \\ \{w\} & \text{otherwise.} \end{cases}$$

The set of k-factors of a stringset L is the set of k-factors of the strings that it contains:

$$F_k(L) \stackrel{def}{=} \bigcup_{w \in L} [F_k(w)].$$

A **strictly k-local description** is an arbitrary set of k-factors drawn from the alphabet Σ, possibly starting with $\rtimes$ and/or ending with $\ltimes$ where $\rtimes$ and $\ltimes$ are endmarkers not occurring in Σ:

Definition 2 (Strictly k-Local Descriptions). *$\mathcal{G}$ is a **strictly k-local description** iff*

$$\mathcal{G} \subseteq F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$$

A string w satisfies such a description iff the augmented string $\rtimes \cdot w \cdot \ltimes$ includes only k-factors given in the description. $L(\mathcal{G})$ is the stringset defined by $\mathcal{G}$, the set of finite strings which satisfy it:

$$L(\mathcal{G}) \stackrel{def}{=} \{w \mid F_k(\rtimes \cdot w \cdot \ltimes) \subseteq \mathcal{G},\ w \text{ finite}\}.$$

A stringset is Strictly k-Local (SL_k) iff it can be defined by a Strictly k-Local description. It is Strictly Local (SL) iff it is SL_k for some k.

The set $\{(\textbf{ding dong})^n \mid n \geq 1\}$ is SL_2, as witnessed by the set of 2-factors:

$$\{\rtimes \textbf{ding},\ \textbf{ding dong},\ \textbf{dong ding},\ \textbf{dong} \ltimes\}. \tag{1}$$

Computationally, Strictly k-Local stringsets are those that can be recognized by **scanners**, automata that simply scan a k-symbol window across the input failing if, at any point, the window contains a factor that is not in the permitted set. A grammar-theoretic characterization can be obtained by taking the k-factors to be productions permitting the extension of a string with the k^{th} symbol of the factor if the string currently ends with the first $k-1$ symbols.

The Strictly k-Local stringsets form a proper hierarchy in k, with the class of Strictly Local stringsets, in general, (SL) being the union of the Strictly k-Local stringsets for all finite k. Every finite stringset is SL_k for k no greater than the length of the longest string plus one. Hence, the finite stringsets are a (proper) sub-class of SL. Conversely, there is no k such that SL_k includes all the finite languages.

From a cognitive perspective, this is the class of stringsets that can recognized while only remembering the previously encountered block of symbols for blocks of some fixed finitely bounded size or, more generally, that can be distinguished solely on the basis of whether particular finitely bounded blocks of symbols occur independently of the content of the rest of the string. Any cognitive mechanism that is sensitive *only* to the individual length k blocks of events, in isolation, in the presentation of a string will be able to recognize *only* SL_k stringsets. For any fixed k, all the SL_k stringsets are learnable in the sense of being identifiable in the limit (Gold [11]), but the closure class SL is not.

The characteristic property of SL stringsets is that if there are two strings in the set in which the same sequence of $k-1$ symbols occurs, then the result of substituting the suffix starting at that sequence in one of them for the suffix starting at that sequence in the other must also be in the stringset. We refer to this property as **Suffix Substitution Closure**.

Theorem 1 (Suffix Substitution Closure). *A stringset L is* SL *iff there is some k such that whenever there is a string x of length $k-1$ and strings u_1, v_1, u_2, and v_2, such that $u_1xv_1 \in L$ and $u_2xv_2 \in L$ then $u_1xv_2 \in L$ as well.*

In fact, if $L \in \mathrm{SL}_k$ then $\mathcal{G}_L \stackrel{\text{def}}{=} \cup_{w \in L}[F_k(\rtimes w \ltimes)]$ is the minimal SL_k description of L. Note that this is a *characterization*: a stringset can be recognized while remembering only the preceding $k-1$ symbols if and only if it is closed under this type of substitution of suffixes.

One of the consequences of this characterization is that SL specifications cannot require some particular sequence of symbols to occur in every string unless the k-factors that may precede it are distinct from those that may follow it. For example, the set of strings of As and Bs in which at least one B occurs is not SL (for any k). We will refer to this set as "Some-B":

$$\text{Some-B} \stackrel{\text{def}}{=} \{w \in \{\mathsf{A}, \mathsf{B}\}^* \mid |w|_{\mathsf{B}} \geq 1\}. \tag{2}$$

That this is non-SL follows straightforwardly from suffix substitution closure. If Some-B were SL then it would be SL_k for some specific k. But then the string $\mathsf{A}^k\mathsf{B}\mathsf{A}^k$ witnesses the failure of k-suffix substitution closure:

$$\begin{aligned} \varepsilon \cdot \mathsf{A}^{k-1} \cdot \mathsf{B}\mathsf{A}^{k-1} &\in \text{Some-B}, \\ \mathsf{A}^{k-1}\mathsf{B} \cdot \mathsf{A}^{k-1} \cdot \varepsilon &\in \text{Some-B, but} \\ \varepsilon \cdot \mathsf{A}^{k-1} \cdot \varepsilon &\notin \text{Some-B}. \end{aligned} \tag{3}$$

An animal that used a strategy to recognize strings that was based only on remembering the most recently encountered fixed-length sequence of symbols could potentially recognize strings of the form $(\mathsf{A}\mathsf{B})^*$, distinguishing them from those, for instance, of the same form in which one or more of the Bs was replaced with an A or *vice versa*. But having learned to recognize strings of the form $\mathsf{A}^*\mathsf{B}\mathsf{A}^*$, an animal using a Strictly Local strategy would not be able to distinguish these from those consisting of only A^*.

The failure of an animal to recognize a particular SL stringset, however, does not imply that they do not employ this strategy. Their ability to apply the strategy may be limited in ways that exclude that set. Similarly, while success in recognizing an SL stringset establishes that the subject employs a strategy that is at least as powerful as scanning k-factors, it does not imply that they are capable of recognizing every SL stringset. So we can establish lower bounds experimentally (individuals of this species can recognize at least some SL stringsets) but experiments of this sort will provide firm evidence only of possible upper bounds (these individuals fail to recognize at least some SL stringsets). The strength of the evidence depends on the preponderance of negative results and the breadth of variation in the experimental setting.

2.2 Locally Testable Stringsets

While SL descriptions cannot require some factor to occur, they can forbid it. So Some-B *is* in co-SL. Since the class of Strictly Local stringsets is closed under intersection, if we close SL_k under complement as well we get closure under all Boolean operations. Descriptions of stringsets in this class can be taken to be formulae in a propositional language in which the propositional variables are k-factors which are taken to be true for a string iff they occur in that string.

Definition 3 (k-Expressions over Strings). *The language of k-expressions (over strings) is the smallest set including:*

- *Atomic formulae: $f \in F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$ is a k-expression.*
- *Disjunction: If φ_1 and φ_2 are k-expressions then $(\varphi_1 \vee \varphi_2)$ is a k-expression*
- *Negation: If φ_1 is a k-expression then $(\neg\varphi_1)$ is a k-expression.*

If w is a string and φ a k-expression, then

$$w \models \varphi \overset{def}{\iff} \begin{cases} \varphi = f \in F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\}) \text{ and } f \in F_k(\rtimes \cdot w \cdot \ltimes), \\ \varphi = (\varphi_1 \vee \varphi_2) \text{ and } w \models \varphi_1 \text{ or } w \models \varphi_2, \\ \varphi = (\neg\varphi_1) \text{ (i.e., otherwise) and } w \not\models \varphi. \end{cases}$$

Stringsets definable in this way are called **Locally k-Testable** (LT_k).

Any pattern that can be described in terms of Boolean combinations of permissible and non-permissible k-factors for some k defines a Locally Testable stringset. Some-B, for example, is LT_2:

$$\text{Some-B} = \{w \in \{\mathsf{A}, \mathsf{B}\}^* \mid w \models \rtimes \mathsf{B} \ \vee \ \mathsf{AB}\} \tag{4}$$

Again, these classes form a proper hierarchy in k with the class LT being the closure of the hierarchy under union. For each k, the class SL_k is a proper subset of LT_k, but SL_{k+1} is *not* a subset of LT_k nor is LT_k a subset of SL_{k+1}. In fact, LT_2 includes stringsets that are not SL for any k (as witnessed by Some-B).

Automata for recognizing LT_k stringsets can be obtained by extending scanners to keep track of which k-factors occur, feeding the result into an arbitrary

Boolean network. Since there are but finitely many k-factors over a fixed alphabet this is a finite-state recognition strategy.

As with SL, the LT_k stringsets are learnable in the limit if k is fixed but the closure class LT is not learnable in the limit.

The characteristic of Locally k-Testable sets is that strings that include exactly the same set of k-factors cannot be distinguished—they either both must be included in the set or they must both be excluded.

Theorem 2 (Local Test Invariance). *A stringset L is Locally Testable iff there is some k such that, for all strings x and y: if $F_k(\rtimes \cdot x \cdot \ltimes) = F_k(\rtimes \cdot y \cdot \ltimes)$ then $x \in L \Leftrightarrow y \in L$.*

Again, this is a characterization. If we say that two strings are k-equivalent if and only if they have the same set of k-factors, then a stringset is LT_k iff it is the union of some subset of the (finitely many) equivalence classes of Σ^* with respect to k-equivalence.

One of the consequences of this is that, while Some-B is Locally Testable, the set of strings of As and Bs in which *exactly* one B occurs:

$$\text{One-B} \stackrel{\text{def}}{=} \{w \in \{\mathsf{A}, \mathsf{B}\}^* \mid |w|_{\mathsf{B}} = 1\}, \tag{5}$$

is not Locally Testable (for any k). To see this, note that the strings $\mathsf{A}^k\mathsf{B}\mathsf{A}^k$ and $\mathsf{A}^k\mathsf{B}\mathsf{A}^k\mathsf{B}\mathsf{A}^k$ include exactly the same sets of k-factors, but the first is in One-B while the second is not.

So an animal that used a strategy based on keeping track of exactly which subset of the set of all k-factors over the relevant alphabet occurs in a string or which was otherwise sensitive to only this information could potentially distinguish strings of As and Bs in which some B occurs from those in which no B occurs, but would not be able to distinguish strings in which exactly one B occurs from those in which more than one occurs.

2.3 Locally Threshold Testable

A stronger strategy would be to keep track not just of which k-factors occur but *how many* times each occurs. If one limits this to counting up to a finite threshold (not distinguishing different numbers of occurrences if they both exceed the threshold) then this will still be finite-state. This would allow recognition of "n-B" for any value of n less than the threshold. Stringsets that can be distinguished in this way are called **Locally Threshold Testable** (LTT) and there are proper hierarchies in both k and t, the size of the threshold.

Strikingly, this turns out to be exactly the class of stringsets one can define using First-Order logic to reason about positions in strings using a monadic predicate for each symbol (picking out the set of positions in which it occurs) and a binary predicate relating positions to their immediate successor.

Definition 4 (FO(+1)). *An $\triangleleft$-model of a string w is a structure*

$$\langle \mathcal{D}, \triangleleft, P_\sigma \rangle_{\sigma \in \Sigma}$$

where the domain $\mathcal{D} \stackrel{def}{=} \{i \in \mathbb{N} \mid 0 \leq i < |w|\}$ *is the set of positions in* w, $\lhd$ *is the successor relation on these positions* ($x \lhd y \stackrel{def}{\iff} y = x + 1$) *and, for each* $\sigma \in \Sigma$, *the predicate* P_σ *picks out the set of positions at which* σ *occurs in* w.

FO($+1$*) is the class of stringsets definable in First-Order logic as* $\lhd$*-models.*

Any pattern that can be defined with a finite set of FO formulae over this signature is an LTT stringset.

Theorem 3 (Thomas [6]). *A set of strings is First-order definable relative to the class of finite* $\langle \mathcal{D}, \lhd, P_\sigma \rangle_{\sigma \in \Sigma}$ *models (in FO(*$\lhd$*)) iff it is Locally Threshold Testable.*

One-B, for example, is the set of strings over $\{\mathsf{A}, \mathsf{B}\}$ which satisfy the description:

$$(\exists x)[\mathsf{B}(x) \wedge (\forall y)[\mathsf{B}(y) \rightarrow x \approx y]\ ]. \tag{6}$$

$\mathrm{LTT}_{k,t}$ is, again, learnable in the limit if both k and t are fixed, but is not learnable otherwise. Consequently, the sets of strings definable in FO with successor is not learnable in the limit.

The characteristic of LTT stringsets is analogous to that of LT stringsets. We say that two strings are (k,t)-equivalent whenever, for each k-factor, they include either the same number of occurrences of that k-factor or they both include at least t occurrences. A stringset is LTT if and only if there is some k and t for which it does not distinguish (k,t)-equivalent strings: every pair of equivalent strings are either both included in the set or both excluded. Hence, a stringset is $\mathrm{LTT}_{k,t}$ iff it is the union of a subset of the finitely many equivalence classes of Σ^* with respect to (k,t)-equivalence.

One could probe the limits of LTT by exploring a range of values of k and t, but it is doubtful that a sufficiently large range could be covered in a practical set of experiments. A better approach would be to explore the ability of animals to recognize the set we will call "B-before-C".

$$\text{B-before-C} \stackrel{\text{def}}{=} \{w \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}^* \mid \text{at least one } \mathsf{B} \text{ precedes any } \mathsf{C}\}. \tag{7}$$

To see that this is not LTT for any k or t, it suffices to note that the strings $\mathsf{A}^k\mathsf{B}\mathsf{A}^k\mathsf{C}\mathsf{A}^k$ and $\mathsf{A}^k\mathsf{C}\mathsf{A}^k\mathsf{B}\mathsf{A}^k$ have exactly the same number of occurrences of every k-factor and therefore are (k,t)-equivalent for all t.

An animal that used a strategy of counting k-factors up to some threshold or which was otherwise sensitive to only this information, then, could potentially distinguish strings in which exactly one B occurs from those in which more than one B occur but would be unable to distinguish strings of the form $\mathsf{A}^*\mathsf{B}(\mathsf{A}|\mathsf{B}|\mathsf{C})^*$ (where — denotes alternation) from those of the form $\mathsf{A}^*\mathsf{C}(\mathsf{A}|\mathsf{B}|\mathsf{C})^*$.

2.4 Star-Free Stringsets

The next step is to extend the FO signature with a predicate for the transitive closure of the successor relation ("precedes" or "less-than"). The class of

stringsets FO-definable over this signature is called FO(<). McNaughton and Papert [5] showed that this class coincides with the **Star-Free** sets (SF), the stringsets definable by regular expressions which may employ complement but not Kleene-closure. Any pattern that can be described by a finite set of FO formulae over this signature defines a Star-Free language. B-before-C, for example, is the set of strings over $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ which satisfy:

$$(\forall x)[\mathsf{C}(x) \rightarrow (\exists y)[\mathsf{B}(y) \wedge y < x]\,]. \tag{8}$$

In abstract language-theoretic terms, SF turns out to be the closure of LT under concatenation and Boolean operations. This class is called **Locally Testable with Order** (LTO). Like SL, LT and LTT, it is the closure of an infinite hierarchy of language classes, LTO_k, each of which is closure of LT_k under concatenation and Boolean operations.

The cognitive analog of the SF class would be employing a fixed sequence of threshold-counting strategies—or equivalently, allowing the threshold counters to be reset up to a fixed, finite number of times—and keeping track of which succeed and which fail. LTO, since it extends LT, is not learnable in the limit.

Automata-theoretically, SF is the class of stringsets recognized by **Counter-Free** automata, automata for which the syntactic monoid is aperiodic. This is the case iff there is some $n > 0$ such that, for all strings u, v, w over Σ, if $uv^n w$ occurs in L then $uv^{n+i}w$, for all $i \geq 1$, occurs in L as well.

Theorem 4 (McNaughton and Papert [5]). *A stringset L is Star-Free iff it is **non-counting**, that is, iff there exists some $n > 0$ such that, for all strings u, v, w over Σ, if $uv^n w$ occurs in L then $uv^{n+i}w$, for all $i \geq 1$, occurs in L as well.*

Corollary 1. *A set of strings is First-Order definable relative to the class of finite $\langle \mathcal{D}, \lhd, <, P_\sigma \rangle_{\sigma \in \Sigma}$ models (in FO(<)) iff it is non-counting.*

One key consequence of this is that the automata cannot do modular counting. Membership in SF stringsets cannot depend on the number of times some factor occurs modulo some fixed value. (In other words, the threshold counters cannot be reset arbitrarily many times.) One simple stringset that requires modular counting is the set of strings of As and Bs in which the number of Bs is even:

$$\text{Even-B} \stackrel{\text{def}}{=} \{w \in \{\mathsf{A}, \mathsf{B}\}^* \mid |w|_{\mathsf{B}} \bmod 2 = 0\}. \tag{9}$$

So while an animal using a sequence of LT strategies could potentially recognize B-before-C, it would be unable to distinguish strings of the form $(\mathsf{A}^*\mathsf{B}\mathsf{A}^*)^{2n}$ from those of the form $(\mathsf{A}^*\mathsf{B}\mathsf{A}^*)^{2n+1}$.

2.5 Regular Stringsets

Finally, if we move to a **Monadic Second-Order** language, allowing quantification over subsets of positions rather than just individual positions (quantification over finite subsets, wMSO, actually suffices), using either signature (since

transitive closure is MSO definable), we obtain the Regular stringsets. This characterization of Regular stringsets dates back to the late 1950s, due to Büchi [11], Elgot [12] and Medvedev [13]. Automata theoretically, the Regular stringsets are the ones accepted by Finite-State automata. In cognitive terms, any mechanism for which there is a finite bound on the amount of information that is retained while processing a string recognizes, at most, a regular stringset.

It should be emphasized that the fact that there is a finite bound on the cognitive resources available to an animal (as there presumably is for all of us) does not imply that they are limited to mechanisms that implement a recognition strategy which requires only finitely bounded resources. They may well employ a strategy for which, in principle, there is no fixed finite bound on the amount of information retained while processing a string which simply fails on strings beyond a particular length or degree of complexity (or perhaps, *would* fail if it ever encountered such strings.)

2.6 Beyond Regular Stringsets

What CFLs introduce is a need, in principle, to retain an amount of information that depends on the length of the string and is, hence, not limited by a fixed finite bound. This can be captured in a descriptive setting by reasoning about additional structure beyond the linear ordering of symbols in the string. The grammar-theoretic characterization that gives the class its name generates strings, in effect, by tracing out one sort of additional structure; the class of derivation trees of CFLs is almost exactly the class that is definable by strictly 2-local definitions over trees.[3] This is nearly immediate if one takes the set of productions of a CFG to be a set of **local trees**, trees of height 1.

This is not the only way to organize the additional structure, though. The CFLs can also be characterized by **texts**, strings augmented with an additional partial ordering, in which the additional ordering is a **matching** [14]. This corresponds, in strong way, to the characterization of CFLs as the homomorphic image of the intersection of a Dyck Language and a Regular stringset [15].

The additional structure can be captured in automata-theoretic terms, of course, by augmenting finite-state automata with a stack, or some other store with a size that is not finitely bounded. For many CFLs, including $\mathsf{A}^n\mathsf{B}^n$, a proper stack is not necessary. These can be recognized by **Counter Machines**, FSA augmented with a counter which can be incremented, decremented and tested for zero. This corresponds to a **Pushdown Automaton** (PDA) with just a single stack symbol other than the bottom-of-stack marker. Note that the single bracket-type Dyck language D_1 (strings of well-nested parentheses) also requires only a single counter of this type. To get beyond counter machines to

[3] The differences come about because local definitions are not restricted to having a single symbol labeling the root of the trees (in other words, they admit multiple start symbols) and they are not required to label interior nodes and leaves with distinct symbol sets (in other words, they may permit symbols which may be terminal to also possibly be expanded).

PDAs with more than a single stack symbol one needs to move to D_2 in which there are two types of brackets that are well-nested both individually and with respect to each other.[4]

3 Pattern Recognition Experiments

Both familiarization/discrimination experiments (as in Fitch and Hauser [2]) and training experiments (as in Gentner, et al., [3]) require the subjects to generalize from a set of positive examples, testing the nature of the generalization by testing their ability to discriminate strings within the intended stringset from those not within it. The idea is that no subject will (or even can) generalize to an intended set that is formally more complex than can be recognized by the cognitive faculties they can bring to bear, and that subjects will not consistently fail to generalize to intended sets which are within the recognizing power of those faculties.

Any stringset that is a superset of the training (or familiarization) set is a consistent generalization. Consequently, it is not sufficient to simply test the subject on strings from a pair of stringsets that span a complexity boundary. Rather, one has to consider all possible sets that the subject may generalize to and test them on pairs of strings which can distinguish these. These consistent sets will always include both proper supersets of the intended set (e.g., the set of all strings over the relevant alphabet) as well as proper subsets (e.g., the finite training set, itself).

The set $\mathsf{A}^n\mathsf{B}^n$, which has often been used as the exemplar of CFLs, has a variety of features that might be generalized by a subject: all of the As precede all of the Bs ($\mathsf{A}^*\mathsf{B}^*$), the strings are all of even length ($\{\mathsf{A}^i\mathsf{B}^j \mid i+j \bmod 2 = 0\}$), the number of As is equal to the number of Bs ($|w|_\mathsf{A} = |w|_\mathsf{B}$), etc. In addition, since the training set is finite it is not inconsistent to infer a finite bound on the number of As and Bs. In order to resolve these, one must test the subject on pairs of strings that fall in the symmetric difference between these sets and the intended set.

Consider the situation in which the subject has been exposed to just the string AAABBB. Testing this against AABBBB, for example, can distinguish whether the subject has generalized to $\mathsf{A}^n\mathsf{B}^n$ or $\{\mathsf{A}^i\mathsf{B}^j \mid i+j \bmod 2 = 0\}$, which would provide evidence about whether or not they can recognize non-Finite-State stringsets. Pairing it with AABBB, in contrast, can separate subjects that have generalized to $\mathsf{A}^*\mathsf{B}^*$ (which will not find it novel) but will not provide any evidence about the Finite-State boundary: it will be novel both to subjects that have generalized to $\mathsf{A}^n\mathsf{B}^n$ and to $\{\mathsf{A}^i\mathsf{B}^j \mid i+j \bmod 2 = 0\}$. As $\mathsf{A}^*\mathsf{B}^*$ is SL_2 this pair actually spans the entire Subregular Hierarchy.

While it will never be possible to completely rule out the possibility that a subject has generalized to some finite superset of the training set which is also a subset of the intended set, one can be reasonably sure that this is not

[4] These can be recognized by FSA with two counters, but two counters actually suffice to recognize all **r.e.** stringsets.

the case by including strings that are longer than any string in the training set and which will, therefore, be unlikely to be included in any finite generalization. Thus pairing AAABBB with AAAABBBB can provide evidence about whether the subject has generalized to a finite set.

Finally, strings such as ABABAB will appear novel to subjects that have generalized to $\mathsf{A}^*\mathsf{B}^*$, requiring only the ability to recognize SL_2 stringsets, but would not appear novel to a subject who had overgeneralized to $|w|_{\mathsf{A}} = |w|_{\mathsf{B}}$, even though this is a CF stringset. Thus, the pair of stringsets that has been most widely used in these experiments actually fails to provide any evidence about the Finite-State boundary whatsoever.

4 Interpreting the Empirical Evidence

In addition to the difficulty of obtaining empirical evidence which resolves the boundaries of complexity classes, there have been substantial differences in interpretation of what such evidence indicates about the nature of the cognitive mechanisms involved. Fitch and Hauser [2] associate the ability to recognize CF stringsets such as $\mathsf{A}^n\mathsf{B}^n$ with the need for an "open-ended memory", but they identify this class with the ability to process Phrase-Structure Grammars and, in particular, the ability to embed strings within other strings (of the same constituent type). The ability to recognize long-distance dependencies is taken to be a consequence of this ability.

In Hauser, Chomsky and Fitch [1] the role of this self-embedding is taken a step further, underlying a hypothesis that a "computational mechanism of recursion" accounts for the difference between human and non-human capabilities in this realm. Gentner, et al., [3] echo this, identifying their experiments as establishing "recursive syntactic pattern learning" while Perruchet and Rey [4] focus their experiments on "center-embedding".

As we have seen, though, neither $\mathsf{A}^n\mathsf{B}^n$ nor D_1 (sets of properly nested parentheses) require center-embedding or even Phrase-Structure analysis at all. Moreover, long-distance dependencies actually show up at the LT_2 level. LT_2 includes, for example, the set of strings involving interaction of *wh*-extraction and number agreement of which Bresnan [16] wrongly claimed that they exhibit a "distant type of agreement" that "cannot be adequately described even by context-sensitive phrase structure rules", namely:

> *Which problem did your professor say (she thought)* was unsolvable?*
> *Which problems did your professor say (she thought)* were unsolvable?*

The problem for experimenters is that distinctions between mechanisms for recognizing non-Finite-State stringsets depend on the way in which the additional structure, beyond the string itself, is organized; these are issues that show up in the analysis of the string, not in its form as a sequence of events. While the mechanism-internal structure is occasionally explicitly marked in the string—see Huybregts [17] and Shieber [18], for example—it will usually depend on distinctions based on the meaning of the strings. Consequently, it is it is likely to be

very difficult to separate human capabilities from those of other animals within the non-Finite-State realm with pattern recognition experiments.

5 Conclusion

Formal language theory provides useful tools for exploring pattern recognition capabilities in general and, in particular, for designing and interpreting experiments for distinguishing differential capabilities of this sort across species. Paradoxically, the machine-oriented aspects—the theory of grammars and automata—have little to say about the cognitive mechanisms involved.

Descriptive or model-theoretic characterizations such as the ones we have considered here, in contrast, because they focus on the kind of information that distinguishes structures in a class, can provide clues about the cognitive mechanisms under study that are independent of *a priori* assumptions about the details of the physical implementation of those mechanisms.

What the characterizations in terms of grammars and automata do provide is highly specific information about the nature of the stringsets that can or cannot be recognized by these means, providing clear criteria on which to choose sets of stimuli to test the boundaries of the class. Together, these provide a sound foundation for studying classes of stringsets that might be relevant to the experimental study of prerequisites to language.

The subregular hierarchy encompasses a range of extremely general ways of defining classes of patterns which are well understood from both descriptive and machine-oriented perspectives. The prevailing focus on the boundary between the finite-state and the context-free seems to have been taken over from the work of Chomsky in 1956. But he was concerned with the capacities of human beings, who we already knew spoke complex languages. With tamarins and starlings we are fairly sure that no developed language capability is in place. What we seek is a glimpse of any cognitive capacity for syntactic pattern recognition they might have that could serve as an evolutionary building block for linguistic capacities.

As yet we do not even know whether an animal could recognize a pattern that is LT but not SL. The range of complexity classes that make up the subregular hierarchy spans a great deal of the territory that is likely to be relevant to distinguishing human aural pattern recognition capabilities from those of other species and may, in fact, span the full range of complexity classes which *can* be probed with experiments of this type. We hope that this paper will serve as a foundation for the design and analysis of further experiments in this realm.

References

1. Hauser, M.D., Chomsky, N., Fitch, W.T.: The faculty of language: What is it, who has it, and how did it evolve. Science **298** (2002) 1569–1579
2. Fitch, W.T., Hauser, M.D.: Computational constraints on syntactic processing in nonhuman primates. Science **303** (2004) 377–380

3. Gentner, T.Q., Fenn, K.M., Margoliash, D., Nusbaum, H.C.: Recursive syntactic pattern learning by songbirds. Nature **440** (2006) 1204–1207
4. Perruchet, P., Rey, A.: Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates? Psychonomic Bulletin and Review **12** (2005) 307–313
5. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge, MA (1971)
6. Thomas, W.: Classifying regular events in symbolic logic. Journal of Computer and Systems Sciences **25** (1982) 360–376
7. García, P., Ruiz, J.: Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **9** (1990) 920–925
8. Benedikt, M., Segoufin, L.: Regular tree languages definable in FO. In Diekert, V., Durand, B., eds.: 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2005). Volume 3404 of Lecture Notes in Computer Science. (2005) 327–339
9. Straubing, H.: Finite Automata, Formal Logic and Circuit Complexity. Birkhäuser (1994)
10. Heinz, J.: Inductive Learning of Phonotactic Patterns. PhD thesis, Dept. of Linguistics, UCLA (2007)
11. Büchi, J.R.: Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **6** (1960) 66–92
12. Elgot, C.C.: Decision problems of finite automata and related arithmetics. Transactions of the American Mathematical Society **98** (1961) 21–51
13. Medvedev, Y.T.: On the class of events representable in a finite automaton. In Moore, E.F., ed.: Sequential Machines—Selected Papers. Addison-Wesley (1964) 215–227 Originally in Russian in Avtomaty (1956), pp. 385–401.
14. Lautemann, C., Schwentick, T., Théien, D.: Logics for context-free languages. In Pacholski, L., Tiuryn, J., eds.: Computer Science Logic, Kazimierz, Poland (1994) 203–216
15. Chomsky, N.: Context-free grammars and pushdown storage. Quarterly Progress Report 65, MIT Res. Lab. Elect. (1962)
16. Bresnan, J.W.: Evidence for a theory of unbounded transformations. Linguistic Analysis **2** (1978) 353–393
17. Huybregts, R.: The weak inadequacy of context-free phrase structure grammars. In de Haan, G.J., Trommelen, M., Zonneveld, W., eds.: Van Periferie Naar Kern. Foris Publications, Dordrecht (1984) 81–99
18. Shieber, S.: Evidence against the context-freeness of human language. Linguistics and Philosophy **8** (1985) 333–343