Cognitive and Sub-Regular Complexity

James Rogers[1], Jeffrey Heinz[2], Margaret Fero[1], Jeremy Hurst[1], Dakotah Lambert[1], and Sean Wibel[1]

[1] Earlham College, Richmond IN 47374, USA,
[2] University of Delaware, Newark DE 19716, USA

Abstract. We present a measure of cognitive complexity for subclasses of the regular languages that is based on model-theoretic complexity rather than on description length of particular classes of grammars or automata. Unlike description length approaches, this complexity measure is independent of the implementation details of the cognitive mechanism. Hence, it provides a basis for making inferences about cognitive mechanisms that are valid regardless of how those mechanisms are actually realized.

1 Introduction

Identifying the nature of the cognitive mechanisms employed by various species, and the evidence which helps determine this nature, are fundamental goals of cognitive science. The question of the relative degree of difficulty of distinguishing (proto-) linguistic patterns has received a considerable amount of attention in recent Artificial Grammar Learning (AGL) research [1, 2], as well as in current research in phonology [3, 4]. In the AGL research, as in the phonological research, the complexity of the learning task has been central. This in no small part depends on the complexity of the patterns being learned.

This paper studies the pattern complexity of subclasses of the class of regular stringsets[3]from a model-theoretic perspective, which has its roots in the seminal work of McNaughton and Papert [5] (and, ultimately, Büchi [6] and Elgot [7]). An important aspect of this analysis is that it is *independent* of any particular representation. We argue that descriptive complexity of this model-theoretic sort provides a more consistent measure of complexity than typical complexity measures based on minimum description length. More importantly, we show how this notion of cognitive complexity can provide concrete evidence about the capabilities of the recognition mechanism that is valid for all mechanisms, regardless of their implementation.

This complexity analysis is exemplified with stress patterns in the world's languages. Stress patterns are rules that govern which syllables are emphasized,

[3] To minimize confusion between natural and formal languages will generally use the term "stringset" to denote a set of strings rather than the more traditional "language", except that we will use the original terminology when referring by name to concepts defined elsewhere in the literature.

or stressed, in words. The reason we use stress patterns to illustrate the complexity hierarchy is because the cross-linguistic typology of stress patterns has been well-studied [8, 9] and because finite-state representations of these patterns already exist [10, 11].

In the next section, we explain why approaches based on minimum description length fail to provide an adequate notion of cognitive complexity in these domains. We then (Section 3) develop a model-theoretic foundation for building hierarchies of descriptive complexity that do provide a consistent and useful notion of cognitive complexity. In Section 4 we develop such a hierarchy based on adjacency. This is the Local hierarchy of McNaughton and Papert, although our presentation is more abstract and provides a basis for the generalizations that follow. Section 4.1 treats the Strictly-Local sets. We do this in greater detail than we do in the subsequent sections, providing the general framework and allowing us to focus on specific variations at the higher levels of the hierarchy. Sections 4.2, 4.3 and 4.4 treat the Locally Testable, Locally Threshold Testable and Star-Free sets, respectively.

In Section 5 we repeat this structure for a hierarchy based on precedence rather than adjacency. The Piecewise Testable level (Section 5.2) of this hierarchy is well known, but the Strictly Piecewise level (Section 5.1) has only been studied relatively recently. The two hierarchies converge at the level of the Star-Free sets.

We conclude with a brief description of our current work applying these results to the phonology of stress patterns.

While the most of the language-theoretic details we present are not new here, we present them within a more general framework that provides better insight into the common characteristics of and parameters of variation between the classes. Our main contribution, however, is the use of these descriptive hierarchies as the basis of a measure of cognitive complexity capable of providing clear and reliable insights about obscure cognitive mechanisms.

2 Cognitive Complexity of Simple Patterns

The formal foundation for comparisons of the complexity of patterns has primarily been the information theoretic notion of minimum description length. This compares the total number of bits needed to encode a model of computation—for our purposes, a general recognition algorithm—plus the number of bits required to specify the pattern with respect to that model.

Our focus, here, is on patterns that can be described as regular stringsets. While there are many computational models that we might choose, we will focus on a few standard ones: Regular Grammars, Deterministic Finite State Automata (DFAs) and Regular Expressions [12]. All of these computational models are equivalent in their formal power and there is no significant difference in the size of the encodings of the computational models themselves[4] (the recognition

[4] Note that what is in question here is the encoding of the model, a representation of, say, a Turing machine program to process the descriptions, *not* the descriptions themselves. The encodings of the models vary in size by at most a constant.

algorithms), so there is no *a priori* reason to prefer one over another. The question is how well the relative size of the descriptions of patterns with respect to a given computational model reflects pre-theoretic notions of the relative complexity of processing the patterns. So we will concentrate on comparing complexity within a given computational model. This allows us to ignore the encoding of the computational model itself.

One does not have to look far to find examples of pairs of stringsets in which these three computational models disagree with each other about the relative complexity of the stringsets. Moreover each get the apparent relative complexity wrong on one or another of these examples. Figure 1 compares minimal descriptions, with respect to each of these computational models, of the set of strings of 'A's and 'B's that end with 'B', which we will refer to as EndB, and minimal descriptions of the set of strings of 'A's and 'B's in which there is an odd number of occurrences of 'B', which we will refer to as OddB.[5] Thinking simply in terms of what a mechanism has to distinguish about a string to determine whether it meets the pattern or not, what properties of strings distinguish those that fit the pattern from those that do not, EndB is clearly less complex than OddB. In the first case, the mechanism can ignore everything about the string except the last (or most recent) symbol; the pattern is 1-Definite, i.e., fully determined by the last symbol in the string. In the second, it needs to make its decision based on the number of occurrences of a particular symbol modulo two; it is properly regular in the sense that it is regular but not star-free (see Section 4.4).

The Regular Grammars get this intuition right, as do the regular expressions. The DFAs, on the other hand, differ only in the label of two transitions. There is no obvious attribute of the DFAs, themselves, that distinguishes the two.

Figure 2 compares minimal descriptions of the set of strings of 'A's and 'B's in which there is at least one occurrence of 'B's (SomeB) with minimal descriptions of strings in which there is exactly one occurrence of 'B' (OneB). Here, there can be no question of the relative complexity of the two stringsets: in order to recognize that exactly one 'B' occurs, one must be able to recognize that at least one 'B' occurs; in order to generate a string in which exactly on 'B' occurs, one must be able to generate a string with at least one 'B'. But for both the Regular Grammars and the Regular Expressions the size of the description of SomeB is greater than that of OneB. If we insist that DFAs be total, in the sense of having a total transition function—an out edge from each state for each symbol of the alphabet—then the minimal DFA for OneB is larger than that for SomeB. But if we trim the DFAs, deleting states that cannot fall on paths from the start state to an accepting states, the DFAs are identical except that OneB actually requires one fewer transition.

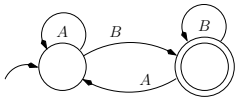
The point of these comparisons is that, even with just these four extremely simple patterns, all of these computational models disagree with each other about relative complexity and each of them get some of the relative complexities wrong. Relative information theoretic complexity, at this level, depends on the

[5] In the case of the DFAs, the minimality is easy to verify. For the other computational models minimality could be verified by enumeration, although this seems excessive.

Sequences of 'A's and 'B's which end in 'B' (EndB)

Regular Grammar: $S_0 \longrightarrow AS_0,\ S_0 \longrightarrow BS_0,\ S_0 \longrightarrow B$

DFA:

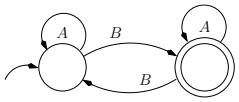


Regular Expression: $(A + B)^*B$

Sequences of 'A's and 'B's which contain an odd number of 'B's (OddB)

Regular Grammar: $S_0 \longrightarrow AS_0,\ S_0 \longrightarrow BS_1,$
$S_1 \longrightarrow AS_1,\ S_1 \longrightarrow BS_0,\ S_1 \longrightarrow \varepsilon$

DFA:



Regular Expression: $(A^*BA^*BA^*)^*A^*BA^*$

Fig. 1. Minimal descriptions: strings which end in 'B' vs. strings with an odd number of 'B's.

choice of computational model and none of these computational models consistently reflects the actual pre-theoretic relative complexity of distinguishing the patterns.

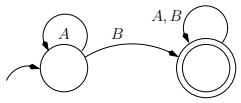
There are many ways to describe regular stringsets beyond the ones considered here [13] so the above is not a deductive proof that no such computational model exists. While searching for an appropriate computational model is one line of research, this program faces a fundamental limitation. Encoding complexity with respect to a particular computational model severely limits the validity of conclusions we might draw about actual cognitive mechanisms from relative complexity results. In the domain of language, the structure of the cognitive mechanisms that an organism uses to recognize a pattern is hotly debated. If a complexity measure is going to provide useful insights into the characteristics of the cognitive mechanisms that can distinguish a pattern, it is an advantage if it is agnostic about the operational details of the mechanisms themselves.

The alternative to searching for a computational model is to develop an abstract measure of complexity. This measure should be invariant across all possible cognitive mechanisms and depend only on properties that are necessarily common to all computational models that can distinguish a pattern. Such a measure has to be based on intrinsic properties of the (generally infinite) set of stimuli that match a pattern. We will take as the basis of the measure the properties of the stimuli that distinguish those that satisfy a pattern from those

Sequences of 'A's and 'B's which contain at least one 'B' (SomeB)

Regular Grammar: $S_0 \longrightarrow AS_0,\ S_0 \longrightarrow BS_1,$
$S_1 \longrightarrow AS_1,\ S_1 \longrightarrow BS_1,\ S_1 \longrightarrow \varepsilon$

DFA:

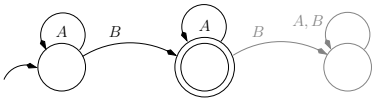


Regular Expression $A^*B(A+B)^*$

Sequences of 'A's and 'B's which contain exactly one 'B' (OneB)

Regular Grammar: $S_0 \longrightarrow AS_0,\ S_0 \longrightarrow BS_1,$
$S_1 \longrightarrow AS_1,\ S_1 \longrightarrow \varepsilon$

DFA:



Regular Expression: A^*BA^*

Fig. 2. Minimal descriptions: strings that contain at least one 'B' vs. strings that contain exactly one 'B'

that do not. These are the things to which a cognitive mechanism needs to be sensitive—the properties of strings it must be able to detect—to in order to correctly classify a stimulus with respect to a pattern.

3 Cognitive Complexity from First Principles

At the most fundamental level, we need to decide what kind of objects (entities, things) we are reasoning about and what relationships between them are we reasoning with. Since we are focusing on linguistic-like behaviors, we will assume that the cognitive mechanisms of interest perceive (process, generate) linear sequences of events.[6]

These we can model as strings, linear sequences of abstract symbols, which we will take to consist of a finite discrete linear order (isomorphic to an initial segment of the natural numbers) that is labeled with an alphabet of events. The labeling partitions the domain of the linear order into subsets, each the set of positions at which some event occurs. Representing these as ordinary relational structures [14], we get word models of Figure 3, in which we use the symbol '$\lhd$' to denote successor (adjacency) and '$\lhd^+$' to denote less-than (precedence). Concatenation with respect to these models is just the ordered sum of the linear orders.[7] We take these models simply to *be* strings; we use no other formalization.

We will distinguish three classes of models: (+1)—models which include only successor (restricted to be successor with respect to some linear order), (<)—models which include only less-than, and models which include both (word models in general).

4 Adjacency—Substrings

The first hierarchy of complexity classes we will consider is based on reasoning about adjacency, in general about substrings, i.e., blocks of consecutive symbols within a string. This gives us a well known sequence of stringset classes, based on generalizations of the *Strictly Local Languages* [5]. We formalize these classes here in a way that will support generalization to stringset classes that are based on other ways of reasoning about strings.

[6] Historically, the term "event" has referred to the entire sequence. But, in general the overall pattern may be hierarchically structured, i.e., sequences of subsequences each of which would, themselves, be an event. So the distinction, here, seems to be spurious and we will refer to the elements of any sequence as an event.

[7] That is to say, the concatenation of two word models is the disjoint unions of their domains and of their interpretations of the relation symbols extended so that the minimum point of the domain of the right word is the successor of the maximum point of the domain of the left. Note that the empty string is represented by a model with an empty domain, which is usually avoided, but this presents no problems for our applications.

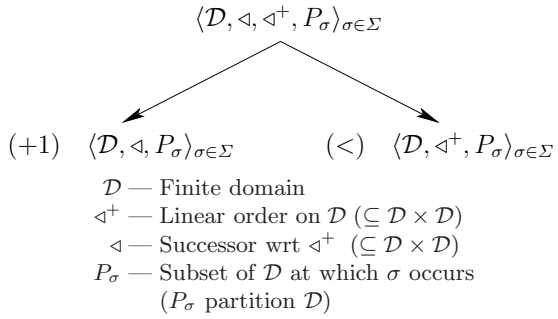


Fig. 3. Word models. (+1) models are the restriction of the general word models to $\lhd$; (<) models are the restriction to $\leq_2$.

All stringsets within these classes are defined in terms of their constituent substrings with the variation between the classes corresponding to how descriptions can be built from those substrings. Traditionally, the substrings that occur within a string are referred to as its *factors* (with respect to concatenation).

Definition 1 (k-Factor).
v is a factor *of w if $w = uvx$ for some $u, v \in \Sigma^*$.*
*v is a k-*factor *of w if it is a factor of w and $|v| = k$.*
The set of k-factors of a string w is:

$$F_k(w) \stackrel{def}{=} \begin{cases} \{v \in \Sigma^k \mid (\exists u, x \in \Sigma^*)[w = uvx]\} & \text{if } |w| \geq k, \\ \{w\} & \text{otherwise.} \end{cases}$$

This lifts to sets in the standard way $F_k(L) \stackrel{def}{=} \{F_k(w) \mid w \in L\}$.

k-factors are essentially k-grams without probabilities. Note that the set of k-factors of a word w which is shorter than k includes just w, itself. The set of all k-factors over an alphabet Σ is $F_k(\Sigma^*) = \{w \in \Sigma^* \mid |w| \leq k\}$, where $|w|$ denotes the length of w. $F_k(\Sigma^*)$ properly includes $F_{k-i}(\Sigma^*)$ for all $i < k$.

4.1 Strictly Local Sets

Definition 2 (Strictly Local Sets). *A strictly k-Local definition $\mathcal{G}$, over some alphabet Σ, is a set of k-factors over $\Sigma \cup \{\rtimes, \ltimes\}$, where '$\rtimes$' and '$\ltimes$' are new symbols: initial and final markers, respectively.*

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

A string w satisfies *$\mathcal{G}$ ($w \models \mathcal{G}$) iff the set of k-factors of $\rtimes \cdot w \cdot \ltimes$ is a subset of $\mathcal{G}$.*

$$w \models \mathcal{G} \stackrel{def}{\iff} F_k(\rtimes \cdot w \cdot \ltimes) \subseteq \mathcal{G}$$

The stringset licensed by a description $\mathcal{G}$ is the set of words that satisfy it.

$$L(\mathcal{G}) \stackrel{def}{=} \{w \mid w \models \mathcal{G}\}$$

A set of strings is Strictly k-local *(*SL_k*) iff it is $L(\mathcal{G})$ for some strictly k-local definition $\mathcal{G}$. It is* Strictly Local *(*SL*) iff it is* SL_k *for some k.*

The expression $\rtimes \cdot w \cdot \ltimes$ denotes w augmented with explicit initial and final markers. A strictly k-local description is the set of k-factors that are licensed to occur in the augmented string. Again, $F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$ contains factors of length less than k, but in this case they all begin with '$\rtimes$' and end with '$\ltimes$'. Hence, they license only words of length less than $k-1$.

The characteristic property of Strictly k-local sets is that they are closed under substitution of suffixes that start with the same $(k-1)$-factor.

Theorem 1 (Suffix Substitution Closure). *A stringset L is strictly k-local iff whenever there is a string x of length $k-1$ and strings w, y, v, and z, such that*

$$w \cdot \overbrace{x}^{k-1} \cdot y \in L \text{ and } v \cdot \overbrace{x}^{k-1} \cdot z \in L \Rightarrow w \cdot \overbrace{x}^{k-1} \cdot z \in L$$

(Sketch of proof:) Closure of SL_k stringsets under substitution of suffixes in this way is nearly immediate. If $w \cdot x \cdot y$ and $v \cdot x \cdot z \in L(\mathcal{G})$ for some SL_k definition $\mathcal{G}$, and $|x| = k-1$ then

$$F_k(\rtimes \cdot w \cdot x \cdot z \cdot \ltimes) \subseteq F_k(\rtimes \cdot w \cdot x \cdot y \cdot \ltimes) \cup F_k(\rtimes \cdot w \cdot x \cdot y \cdot \ltimes) \subseteq \mathcal{G}$$

For the other direction, suppose that a stringset L is closed under substitution of suffixes that start with the same $(k-1)$-factor. Let $\mathcal{G}_L = F_k(\{\rtimes\} \cdot L \cdot \{\ltimes\})$. That $L \subseteq L(\mathcal{G}_L)$ is immediate by construction. One can show that $L(\mathcal{G}_L) \subseteq L$ by constructing an arbitrary $w \in L(\mathcal{G}_L)$ in stages from strings in L that share successively long prefixes of w, extending the prefix, at each stage, by substitution of suffixes.

Note that this is a *characterization*, every SL_k stringset is closed under substitution of suffixes in this way and every stringset that is SSC-closed for some k can be defined by a SL_k definition.

SL_k and SL, as a whole, are closed under intersection but not union, complement, concatenation or Kleene-$*$ [5].

Example 1. Stress in the language Alawa is governed by two (actually three) phonological rules [11]:

- In words of all sizes, primary stress falls on the penultimate syllable.
- In words of all sizes, there is no secondary stress.

The third rule is implicit in all stress patterns

- Every word has exactly one syllable that receives primary stress.

This pattern is not SL_2 as witnessed by:

$$\rtimes\sigma\,\acute{\sigma}\,\sigma\ltimes \in L_{\text{Alawa}},\ \rtimes\,\acute{\sigma}\,\ltimes \in L_{\text{Alawa}},\ \text{but}\ \rtimes\sigma\,\acute{\sigma}\,\ltimes \notin L_{\text{Alawa}}.$$

On the other hand, we can capture L_{Alawa} with the constraints:

1. Don't permit 3-factors with multiple primary stress.
2. Don't permit unstressed penultimate syllables.
3. Don't permit primary stress to be followed by more than one syllable.
4. Don't permit unstressed monosyllables.
5. Don't permit empty words.

$$\begin{aligned} L_{\text{Alawa}} = L(\ & F_3(\rtimes \cdot \Sigma^+ \cdot \ltimes) \\ & - \{\ \rtimes\acute{\sigma}\acute{\sigma},\ \acute{\sigma}\acute{\sigma}\ltimes,\ \sigma\acute{\sigma}\acute{\sigma},\ \acute{\sigma}\sigma\acute{\sigma},\ \acute{\sigma}\acute{\sigma}\sigma,\ \acute{\sigma}\acute{\sigma}\acute{\sigma}, && (1) \\ & \quad \sigma\acute{\sigma}\ltimes,\ \sigma\sigma\ltimes, && (2) \\ & \quad \acute{\sigma}\sigma\sigma, && (3) \\ & \quad \rtimes\sigma\ltimes, && (4) \\ & \quad \rtimes\ltimes\}\quad) && (5) \\ = L(\ & \{\rtimes\sigma\sigma,\ \rtimes\sigma\acute{\sigma},\ \rtimes\acute{\sigma}\sigma,\ \sigma\sigma\sigma,\ \sigma\sigma\acute{\sigma},\ \sigma\acute{\sigma}\sigma, \acute{\sigma}\sigma\ltimes,\ \rtimes\acute{\sigma}\ltimes\}\ ) \end{aligned}$$

Hence $L_{\text{Alawa}} \in \mathrm{SL}_3 - \mathrm{SL}_2$.

The strictly local classes form a proper hierarchy in k.

Theorem 2 (SL-Hierarchy).

$$\mathrm{SL}_1 \subsetneq \mathrm{SL}_2 \subsetneq \mathrm{SL}_3 \subsetneq \cdots \subsetneq \mathrm{SL}_i \subsetneq \mathrm{SL}_{i+1} \subsetneq \cdots \subsetneq \mathrm{SL}$$

(Sketch of proof) The inclusions follow nearly immediately from the definition of an SL_k definition. The separations are easy to obtain using generalizations of the proof that the Alawa stress pattern is not SL_2.

The proper inclusions reflect the intuition that distinguishing a pattern that requires attending to a larger block of symbols is likely to be cognitively more difficult than distinguishing one that can be recognized by attending to smaller blocks.

While every finite stringset is SL_k for some k, there is no k for which SL_k includes all Finite stringsets. Note, also, that given fixed k and Σ, there are only finitely many SL_k stringsets. SL_k is learnable in the limit from positive data in the sense of Gold [15]; SL as a whole is not [16].

Example 2. Edlefsen, et al. [17] have categorized the 109 patterns in Heinz's Stress Pattern Database [18]:

9 are SL_2	Abun West, Afrikans, ... Cambodian, ... Maranungku
44 are SL_3	Alawa, Arabic (Bani-Hassan), ...
24 are SL_4	Arabic (Cairene),[8]...
3 are SL_5	Asheninca, Bhojpuri, Hindi (Fairbanks)
1 is SL_6	Icua Tupi
28 are not SL	Amele, Bhojpuri (Shukla Tiwari), Arabic Classical, Hindi (Keldar), Yidin,...

[8] The formalization of Arabic (Cairene) is controversial. Thomas Graf formalizes this in a way that is properly regular [19].

72% are SL, all $k \leq 6$. 49% are SL_3.

This suggests that the majority of stress patterns in natural languages are cognitively very simple and, perhaps even learnable in the limit.

Cognitive interpretation of SL It is important to note that the definition of the class SL and its characterization by suffix substition closure make no reference to any computational model of any sort. They are stated purely in terms of the structure of the stringset itself. Members of an SL_k stringset are distinguished from non-members purely on the basis of their k-factors. This assumes nothing about *how* those distinctions might be made by a particular computational mechanism. Any mechanism that can distinguish members of an SL_k stringset from non-members *must* be able, at least, to distinguish strings in this way. Any capabilities they may have beyond that are, in a sense, wasted, at least with respect to that stringset.

This gives us a general characterization of cognitive mechanisms that are capable of recognizing SL_k stringsets.

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) SL_k stringset must be sensitive, at least, to the length k blocks of consecutive events that occur in the presentation of the string.
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to the immediately prior sequence of $k-1$ events.

Note, again, that the cognitive mechanism is not required to analyze strings in terms of blocks of consecutive events, even if they are presented as sequences of events in time. It just needs to be able to make judgements, at each point in the presentation of the string, that depend on the sequences of $k-1$ events which occur prior to that point. Every regular stringset can be generated by a context-free grammar that is not a regular grammar, that does not analyze the string in terms of contiguous blocks of symbols. Nevertheless, if the stringset is strictly local, it will still need to get the judgements right; the set of all strings that it generates will be closed under substitution of suffixes; it will differ from the set it does not generate only in the blocks of consecutive symbols that occur in the strings.

4.2 Locally Testable Languages

The standard phonological assumption that in every word there is some syllable that receives primary stress [20] is problematic for SL. Letting $\Sigma = \{\sigma, \acute{\sigma}, \grave{\sigma}\}$ (representing unstressed syllables, those with primary stress and those with secondary stress, respectively), this assumption can be described with the following regular expression: $\Sigma^* \acute{\sigma} \Sigma^*$. Note that we are not (yet) ruling out the possiblity that more than one syllable receives primary stress. To see that this is not SL,

suppose, for contradiction, that it was. Then it would necessarily be SL_k for some particular k. But then

$$\rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \ltimes,\ \rtimes \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \in L_{\mathrm{Some}\acute{\sigma}} \quad \text{but } \rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \notin L_{\mathrm{Some}\acute{\sigma}}$$

SL patterns cannot, in general, require a factor to occur; they can, at most, forbid factors from occurring. Hence, they cannot enforce the requirement that primary stress occurs unless either the stress is required to fall within a fixed radius of one end of the word (as in the case of Alawa) or the factors preceding the stress can be distinguished in some other way from those following it.

The next level of the Local Hierarchy, the class of *Locally Testable* (LT) languages is the closure of SL under Boolean operations. Since this includes complement, it allows one to require the occurrence of specific factors. Rather than taking LT descriptions to be Boolean combinations of SL descriptions, we use a simple propositional calculus to describe LT sets. This provides a foundation for extending the descriptions to First Order descriptions.

Definition 3 (Local k-expressions). *The logic of Local k-expressions is based on the smallest set including the following forms, with the intended semantics as indicated.*

$$\begin{array}{cc} f \in F_k(\rtimes \cdot \Sigma^* \cdot \ltimes) & w \models f \overset{def}{\iff} f \in F_k(\rtimes \cdot w \cdot \ltimes) \\ \varphi \wedge \psi & w \models \varphi \wedge \psi \overset{def}{\iff} w \models \varphi \textit{ and } w \models \psi \\ \neg\varphi & w \models \neg\varphi \overset{def}{\iff} w \not\models \varphi \end{array}$$

The k-factors serve as our atomic propositions. While these are not devoid of internal structure, they are either a factor of a string or not. Hence, strings can be seen as valuations of the factors in the ordinary propositional sense. The calculus of k-expressions is just a idiosyncratic propositional calculus.

Definition 4 (Locally Testable Sets). *A stringset L over Σ is* k-Locally Testable *iff (by definition) there is some local k-expression φ over Σ (for some k) such that L is the set of all strings that satisfy φ:*

$$L = L(\varphi) \overset{def}{=} \{w \in \Sigma^* \mid w \models \varphi\}$$

A stringset is LT *iff it is* LT_k *for some k.*

Note that SL_k descriptions can be interpreted by local k-expressions:

$$L(\mathcal{G}) = L(\bigwedge_{f_i \notin \mathcal{G}} [\neg f_i])$$

Thus $\mathrm{SL}_k \subsetneq \mathrm{LT}_k$. In particular, SL stringsets are exactly those LT stringsets that can be expressed as conjunctions of negative constraints.

Since strings are, in effect, propositional valuations, the Locally Testable stringsets can be characterized by the fact that they must be the union of finitely many classes of strings that are equivalent with respect to the k-factors they comprise.

Theorem 3 (Local Test Invariance). *A stringset L is Locally Testable iff there is some k such that, for all strings x and y, if $\rtimes \cdot x \cdot \ltimes$ and $\rtimes \cdot y \cdot \ltimes$ have exactly the same set of k-factors then either both x and y are members of L or neither is.*

In other words, if

$$w \equiv_k^L v \overset{\text{def}}{\iff} F_k(\rtimes w \ltimes) = F_k(\rtimes v \ltimes).$$

the LT_k stringsets cannot break the equivalence classes of Σ^* with respect to $\equiv_k^L$. (The superscript L here refers to the fact that this is an equivalence with respect to Local criteria.)

LT_k and LT as a whole are closed under Boolean operations, by definition, but are not closed under concatenation or Kleene-$*$ [5].

Since there are finitely many $\equiv_k^L$ equivalence classes, for fixed k, there are finitely many LT_k stringsets over a given alphabet. LT_k is learnable in the limit, although LT is not [21].

Example 3. Stress in the Mongolic language Buriat is governed by two explicit constraints:

- Primary stress falls on the right-most non-final heavy syllable, else on the final syllable if it is heavy, else on the initial syllable.
- Secondary stress falls on the initial syllable and on heavy syllables.

The second constraint forbids any unstressed H. One of the consequences of these constraints is that if a word ends with $\acute{H}$ then there is no non-final $\grave{H}$. This is LT_2 as witnessed by the 2-expression:

$$\neg(\acute{H}\ltimes \wedge \grave{H}\sigma)$$

It is not, on the other hand, SL since

$$\rtimes \grave{H} \overbrace{L \cdots L}^{k-1} \acute{H} L \ltimes,\ \rtimes \grave{L} \overbrace{L \cdots L}^{k-1} \acute{H} \ltimes \in L_{\text{Buriat}} \quad \text{but} \quad \rtimes \grave{H} \overbrace{L \cdots L}^{k-1} \acute{H} \ltimes \notin L_{\text{Buriat}}$$

Furthermore, it is not LT_1, either:

$$\rtimes \grave{H}\acute{H}\grave{H} \ltimes \equiv_1^L \rtimes \grave{H}\grave{H}\acute{H} \ltimes$$

Theorem 4 (LT-Hierarchy).

$$\mathrm{LT}_1 \subsetneq \mathrm{LT}_2 \subsetneq \mathrm{LT}_3 \subsetneq \cdots \subsetneq \mathrm{LT}_i \subsetneq \mathrm{LT}_{i+1} \subsetneq \cdots \subsetneq \mathrm{LT}$$

Again, the hierarchy reflects the intuition that attention to larger blocks is likely to be cognitively more difficult than attention to smaller blocks (because, for one thing, it requires more memory).

Local Test Invariance implies that, in order to distinguish strings that satisfy an LT_k pattern from those that do not, a mechanism has to be sensitive to the *set* of k-factors that occur in a string, not just whether specific k-factors occur or not.

Cognitive interpretation of LT

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) LT_k stringset must be sensitive, at least, to the *set* of length k contiguous blocks of events that occur in the presentation of the string—both those that do occur and those that do not.
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to the set of length k blocks of events that occurred at any prior point.

Here again, this interpretation is fully independent of way that the mechanism actually parses the strings. However it may do that, it must be able to distinguish strings purely on the basis of their sets of k-factors.

4.3 FO(+1)—Locally Threshold Testable Languages

LT constraints can require primary stress, but they cannot rule out multiple occurrences of primary stress. To see this, let $L_{\mathrm{One}\acute{\sigma}}$ be the set of strings over $\{\sigma, \acute{\sigma}, \grave{\sigma}\}$ in which *exactly* one $\acute{\sigma}$ occurs. Suppose, for contradiction, that it is LT, hence LT_k for some k. Then

$$\rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \equiv_k^L \rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes$$

but the former is in $L_{\mathrm{One}\acute{\sigma}}$ while the latter is not. The problem is that LT automata can't count. Or, more precisely, they can count only to 1.

In order to distinguish strings in which some $\acute{\sigma}$ occurs from those in which more than one occurs, we will need to be able to distinguish one instance of $\acute{\sigma}$ from another. We need to state our constraints in terms of specific positions within a string. We do this with a standard First Order language for our (+1) word models, strings with successor but not less-than.

Definition 5. FO(+1) *is the standard First Order logical system over the models* $\langle \mathcal{D}, \lhd, P_\sigma \rangle_{\sigma \in \Sigma}$ *with equality on the domain:*

$$\begin{array}{rc}
x \lhd y & w, [x \mapsto i, y \mapsto j] \models x \lhd y \overset{def}{\iff} j = i + 1 \\
x \approx y & w, [x \mapsto i, y \mapsto j] \models x \approx y \overset{def}{\iff} j = i \\
P_\sigma(x) & w, [x \mapsto i] \models P_\sigma(x) \overset{def}{\iff} i \in P_\sigma \\
\varphi \wedge \psi & \vdots \\
\neg\varphi & \vdots \\
(\exists x)[\varphi(x)] & w, s \models (\exists x)[\varphi(x)] \overset{def}{\iff} w, s[x \mapsto i] \models \varphi(x) \\
 & \qquad \textit{for some } i \in \mathcal{D}
\end{array}$$

where $w, s[x \mapsto i] \models \varphi(x)$ *says that the string* w *satisfies the formula* $\varphi(x)$*, in which the variable* x *possibly occurs, with the position* i *taken to be the value of* x*. (So* i *witnesses that there is some position which satisfies the formula.)*

We take $FO(+1)$ to denote the class of $FO(+1)$-definable stringsets, as well. A stringset L is in the class $FO(+1)$ iff it is $FO(+1)$-definable:

$$L = L(\varphi) \stackrel{def}{=} \{w \mid w \models \varphi\}.$$

Note that local k-expressions can be captured in FO(+1) by Boolean combinations of existential formulae with k variables. (The same k variables can be reused in each existential subformula.) Hence LT $\subsetneq$ FO(+1).

Example 4. $L_{\text{Some}\acute{\sigma}}$ is FO(+1) as witnessed by the formula $(\exists x)[\acute{\sigma}(x)]$.
$L_{\text{At-Most-One}\acute{\sigma}}$ is FO(+1) as witnessed by $(\forall x, y)[\neg(\acute{\sigma}(x) \wedge \acute{\sigma}(y) \wedge x \not\approx y)]$
Consequently, $L_{\text{One}\acute{\sigma}}$ is also FO(+1).

Thomas [22] characterizes FO(+1) in terms of *Local Threshold Testability*, equivalence in terms of the multiplicity of k-factors up to some fixed finite threshold t.

Definition 6 (Locally Threshold Testable). *A set L is* Locally Threshold Testable *(LTT) iff there is some k and t such that, for all $w, v \in \Sigma^*$:*

if for all $f \in F_k(\rtimes \cdot w \cdot \ltimes) \cup F_k(\rtimes \cdot v \cdot \ltimes)$
either $|w|_f = |v|_f$ or both $|w|_f \geq t$ and $|v|_f \geq t$,
then $w \in L \iff v \in L$.

So a stringset is $\text{LTT}_{k,t}$ iff it does not distinguish between strings that, for any k-factor w, either have the same number of occurrences of w or have at least t occurrences; a stringset is LTT iff it is $\text{LTT}_{k,t}$ for some k and t.

Theorem 5 ([22]). *A set of strings is First-order definable over $\langle \mathcal{D}, \lhd, P_\sigma \rangle_{\sigma \in \Sigma}$ iff it is* Locally Threshold Testable.

Once again, there are only finitely many stringsets over a given alphabet if k and t are fixed. So $\text{LTT}_{k,t}$ is learnable in the limit, although LTT, and FO(+1), is not.

Cognitive interpretation of FO(+1)

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) FO(+1) stringset must be sensitive, at least, to the multiplicity of the length k blocks of events, for some fixed k, that occur in the presentation of the string, distinguishing multiplicities only up to some fixed threshold t.
- If the strings are presented as sequences of events in time, then this corresponds to being able count up to some fixed threshold.

4.4 FO(<)—Star Free Languages

While FO(+1) formulae can distinguish strings on the multiplicity of their k-factors, they cannot distinguish the order in which those factors occur.

Example 5. A second primitive constraint on stress in Buriat is that no syllable with primary stress can properly precede a non-final heavy syllable (which, necessarily would have secondary stress). But this is not a constraint that is FO(+1) definable since

$$\rtimes \grave{L} \overbrace{L \cdots L}^{k-1} \grave{H} \overbrace{L \cdots L}^{k-1} \acute{H} \overbrace{L \cdots L}^{k-1} \ltimes$$

and

$$\rtimes \grave{L} \underbrace{L \cdots L}_{k-1} \acute{H} \underbrace{L \cdots L}_{k-1} \grave{H} \underbrace{L \cdots L}_{k-1} \ltimes$$

have the same number of each k-factor.

This is a constraint that can be enforced in terms of less-than:

$$\neg(\exists x, y)[\acute{\sigma}(x) \wedge \grave{H}(y) \wedge x < y]$$

Note that less-than is not FO definable from successor (as witnessed by the example) although successor is FO definable from less-than. Hence FO(+1) $\subsetneq$ FO(<).

The characterization of FO(<) is the primary result of McNaughton and Papert [5].

Definition 7 (Local Testability with Order). *The class of stringsets that are Locally Testable with Order (*LTO*) is the closure of* LT *under concatenation and Boolean operations.*

Note that threshold testability is not required, since it can be reduced to concatenation. Any fixed number of occurrences of a factor can be captured as the concatenation of a fixed number of single occurrences [5].

Definition 8 (Star-Free Languages). *The class of star-free stringsets is the closure of the class of finite stringsets under union, concatenation and complement with respect to Σ^*.*

This is the class of stringsets that are the denotation of regular expressions extended with a complement operator but without Kleene star.

Theorem 6 ([5]). *For any stringset L, the following are equivalent*

- *L is First-order definable over $\langle \mathcal{D}, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$*
- *L is* LTO
- *L is* Star-Free.

This class of languages is not learnable in the limit because it contains every finite language and at least one infinite language [15].

Cognitive interpretation of SF **(FO(<))**

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) SF stringset must be sensitive, at least, to both the order and the multiplicity of the length k blocks of events, for some fixed k, that occur in the presentation of the string, distinguishing multiplicities only up to some fixed threshold t.
- If the strings are presented as sequences of events in time, then this corresponds to being able not only to count events up to some threshold but also to track the sequence in which those events occur.

5 Precedence—Subsequences

The Buriat constraint of Example 5 is a simple negative constraint on the order of syllables. If we specify our constraints in terms of precedence rather than adjacency this can be captured at the level corresponding to SL. We can do this simply by interpreting our atomic formulae as subsequences rather than factors. Let:

$$v \sqsubseteq w \overset{\text{def}}{\iff} v = \sigma_1 \cdots \sigma_n \text{ and } w \in \Sigma^* \cdot \sigma_1 \cdot \Sigma^* \cdots \Sigma^* \cdot \sigma_n \cdot \Sigma^*$$

So $v \sqsubseteq w$ iff v is a subsequence of w.

The logic of *Piecewise k-expressions* is identical in form and meaning to that of local k-expressions except that the atomic formulae are now strings of length less than or equal to k over Σ (with no end markers) which are interpreted as subsequences.

$$s \in \Sigma^{\leq k} \qquad w \models s \overset{\text{def}}{\iff} s \sqsubseteq w$$

To emphasize that we are working with subsequences rather than substrings, we will generally write the subsequence $\sigma_1\sigma_1$ as $\sigma_1 \,..\, \sigma_2$.

5.1 Strictly Piecewise Testable Sets

Strictly k-Piecewise Testable sets are those sets that are definable as conjunctions of negative atomic piecewise k-expressions. As with SL these are closed under intersection but not union, complement, concatenation or Kleene-$*$. And they form a proper hierarchy in k.

The class of SP constraints was characterized by Rogers, et al. [23]. Strikingly these are exactly the sets of strings that are closed under subsequence. The parameter k is the length of the longest string that is not included although all of its subsequences are. One immediate consequence of this is that no set of SP constraints will suffice to define the stress pattern of a human language since SP constraints cannot require some primary stress to occur.

Example 6. While SP constraints cannot require primary stress to occur, they can prohibit more than one primary stress, as witnessed by the piecewise 2-expression $\neg\acute{\sigma} \,..\, \acute{\sigma}$.

The Buriat constraint of Example 5 is SP_3, as witnessed by $\neg\acute{\sigma}\,..\,\grave{H}\,..\,\sigma$.

Since the sequence $\acute{\sigma}\grave{H}\sigma$ must be excluded but none of its subsequences may be (on the basis of this constraint) it is not an SP_2 definable constraint.

Note that while SP constraints can neither require strings to start or end with a particular symbol nor require them to not start or end with a particular symbol, they can forbid particular symbols from occurring anywhere except at the beginning or end of a string.

As with SL_k, SP_k is learnable in the limit if k is fixed. SP in general is not.

Cognitive interpretation of SP

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) SP_k stringset must be sensitive, at least, to the length k (not necessarily consecutive) sequences of events that occur in the presentation of the string.
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to up to $k-1$ events distributed arbitrarily among the prior events.

5.2 Piecewise Testable Sets

Continuing to follow the pattern of the local hierarchy, the *Piecewise Testable* sets are those which are definable by arbitrary piecewise k-expressions. These are well studied, having been introduced in Simon [24]. As with the LT sets, they can be characterized as the union of finitely many equivalence classes, but with strings being equivalent if they share the same set of subsequences, rather than the same set of factors.

Theorem 7 (k-Subsequence Invariance). *A stringset L is Piecewise Testable iff there is some k such that, for all strings x and y, if x and y have exactly the same set of subsequences of length less than or equal to k then either both x and y are members of L or neither is.*

Example 7. PT constraints can require primary stress to occur on exactly one syllable: $\acute{\sigma} \wedge \neg\acute{\sigma}\,..\,\acute{\sigma}$.

In general, they cannot require syllables to occur initially or finally unless that syllable cannot occur more than once. Hence the Buriat constraint of Example 3 can be captured in PT_2 with the expression $(\acute{H} \wedge \neg\acute{H}\,..\,\sigma) \rightarrow \neg\grave{H}$. This asserts that if there is some $\acute{H}$ but no non-final $\acute{H}$ (thus there is a final $\acute{H}$) then there are no $\grave{H}$.

In parallel with LT_k, PT_k for fixed k is learnable in the limit, but PT in general is not [21].

Cognitive interpretation of PT

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) PT_k stringset must be sensitive, at least, to the set of length k subsequences of events that occur in the presentation of the string—both those that do occur and those that do not.
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to the set of all length k subsequences of the sequence of prior events.

5.3 First Order

Still following the pattern of the local hierarchy, the next step is to move to a First Order language over $(<)$ models. But successor is FO definable from less-than, so at the FO level $(<)$ models are equivalent to models with both successor and less-than. It is at this point, the Star-Free sets, that the local and piecewise hierarchies meet.

6 Further Work

From a practical point of view, one of the most important characteristics of these hierarchies is that all of the classes are closed under intersection. This means that complicated patterns can be factored into the co-occurrence of primitive patterns of one type or the other (local or piecewise), as we have done here with Alawa and (partially) Buriat, with the overall complexity being the supremum of the complexities of the primitive constraints.

We are currently factoring all of the stress patterns in Heinz's database into primitive constraints for which we have determined the complexities with respect to the local and piecewise hierarchy. The results, though preliminary, are exciting. With the possible (controversial) exception of Cairene Arabic, all of the patterns are at worst Star-Free. Moreover, while there are patterns that are properly Star-Free from either the local or piecewise perspective (Buriat is an example), all of the patterns we have factored (nearly all of the patterns in the database) are co-occurrences of either LT and SP constraints or SL and PT constraints. This suggests that stress in every human language can be factored into co-occurrence of simple constraints, all of which are potentially learnable in the limit at least in principle if an upper bound on the length of the (sub)sequence is established.

These preliminary results are made possible by the abstract complexity measures introduced here. It is unclear how an approach based on the minimal description length of a particular model could have obtained this result. Furthermore, this complexity analysis, while introduced with examples from phonology, is much more far-reaching than that. They can be, and are being, applied to syntactic structures [25]. One reason this is possible is because these measures are sufficiently and appropriately abstract. They are agnostic about the operational details of models themselves, and therefore they provide a basis for making

inferences about cognitive mechanisms that are valid, regardless of how those mechanisms are actually realized.

References

1. Folia, V., Uddén, J., de Vries, M., Forkstam, C., Petersson, K.M.: Artificial language learning in adults and children. Language Learning **60** (2010) 188–220 Supplement 2.
2. 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**(5598) (2002) 1569
3. Heinz, J.: Learning long-distance phonotactics. Linguistic Inquiry **41**(4) (2010) 623–661
4. Heinz, J., Idsardi, W.: Sentence and word complexity. Science **333**(6040) (July 2011) 295–297
5. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press (1971)
6. Büchi, J.R.: Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **6** (1960) 66–92
7. Elgot, C.C.: Decision problems of finite automata and related arithmetics. Transactions of the American Mathematical Society **98** (1961) 21–51
8. Hayes, B.: Metrical Stress Theory. Chicago University Press (1995)
9. van der Hulst, H., Goedemans, R., van Zanten, E., eds.: A survey of word accentual patterns in the languages of the world. Mouton de Gruyter, Berlin (2010)
10. Heinz, J.: The Inductive Learning of Phonotactic Patterns. PhD thesis, University of California, Los Angeles (2007)
11. Heinz, J.: On the role of locality in learning stress patterns. Phonology **26**(2) (2009) 303–351
12. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (2001)
13. Kracht, M.: The Mathematics of Language. Mouton de Gruyter (2003)
14. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press (1972)
15. Gold, E.: Language identification in the limit. Information and Control **10** (1967) 447–474
16. Garcia, P., Vidal, E., Oncina, J.: Learning locally testable languages in the strict sense. In: Proceedings of the Workshop on Algorithmic Learning Theory. (1990) 325–338
17. Edlefsen, M., Leeman, D., Myers, N., Smith, N., Visscher, M., Wellcome, D.: Deciding strictly local (SL) languages. In Breitenbucher, J., ed.: Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics. (2008) 66–73
18. Heinz, J.: UD phonology lab stress pattern database. `http://phonology.cogsci.udel.edu/dbs/stress/` (March 2012)
19. Graf, T.: Comparing incomparable frameworks: A model theoretic approach to phonology. University of Pennsylvania Working Papers in Linguistics **16**(2) (2010) Article 10 Available at: `http://repository.upenn.edu/pwpl/vol16/iss1/10`.
20. Hyman, L.M.: How (not) to do phonological typology: the case of pitch-accent. Language Sciences **31**(2-3) (2009) 213 – 238 Data and Theory: Papers in Phonology in Celebration of Charles W. Kisseberth.
21. García, P., Ruiz, J.: Learning k-testable and k-piecewise testable languages from positive data. Grammars **7** (2004) 125–140

22. Thomas, W.: Classifying regular events in symbolic logic. Journal of Computer and Systems Sciences **25** (1982) 360–376
23. Rogers, J., Heinz, J., Bailey, G., Edlefsen, M., Visscher, M., Wellcome, D., Wibel, S.: On languages piecewise testable in the strict sense. In Ebert, C., Jäger, G., Michaelis, J., eds.: The Mathematics of Language: Revised Selected Papers from the 10th and 11th Biennial Conference on the Mathematics of Language. Volume 6149 of LNCS/LNAI. FoLLI/Springer (2010) 255–265
24. Simon, I.: Piecewise testable events. In: Automata Theory and Formal Languages: 2nd Grammatical Inference conference, Berlin, Springer-Verlag (1975) 214–222
25. Graf, T.: Locality and the complexity of minimalist derivation tree languages. In: Proceedings of the 16^{th} Conference on Formal Grammar. (2011) to appear.