JMLR: Workshop and Conference Proceedings 57:66–78, 2016

Using model theory for grammatical inference: a case study from phonology

Kristina Strother-Garcia KMSG@UDEL.EDU
Jeffrey Heinz HEINZ@UDEL.EDU
Hyun Jin Hwangbo HWANGBO@UDEL.EDU
Dept. of Linguistics & Cognitive Science, University of Delaware
125 E. Main St, Newark, DE 19716

Abstract

This paper examines the characterization and learning of grammars defined by conjunctions of negative and positive literals (CNPL) where the literals correspond to structures in an enriched model theory of strings. CNPL logic represents an intermediate between conjunctions of negative literals (CNL) and a propositional-style logic, both of which have been well-studied in terms of the language classes they describe. Model-theoretic approaches to formal language theory have traditionally assumed that each position in a string belongs to exactly one unary relation. Using enriched models (which do no satisfy this assumption) presents a new avenue for investigation with potential applications in several fields including linguistics, planning and control, and molecular biology. We demonstrate the value of such structures and CNPL logic with a particular learning problem in phonology.

Keywords: subregular languages, model theory, phonology

1. Introduction

In this paper we explore model-theoretic approaches to formal languages, and some of the consequences for grammatical inference. The paper is exploratory in nature; no hard proofs or new theorems are provided. However, we show that a carefully chosen model theory for strings can make the learning problem easier by lowering the complexity class of the target languages and thereby lowering the complexity of the algorithms necessary for learning them. Though the particular example presented in this paper draws from phonology—the study of systematic aspects of pronunciation in natural languages—the potential applications are far broader including other areas of linguistics, planning and control in cyber-physical systems, and modeling sequential data in biological systems.

Traditional model-theoretic approaches (such as Büchi (1960)) assume that each position in a string belongs to exactly one unary relation. Conceptualizing unary relations as position labels, this means that no position in a string may go unlabeled or have multiple labels. Abandoning this assumption allows similarities among different alphabetic symbols to be fully expressed. It also allows *underspecified* structures to be defined, which simplifies the logical expressions used to describe a formal language.

More broadly, we argue that the model-theoretic approach helps create a bridge between grammatical inference, formal language theory, and other branches of machine learning such as concept and relational learning (Flach, 2012; De Raedt, 2008). In addition to using enriched strings, we investigate an under-studied language class: namely, languages describable by the *conjunction of negative and positive literals* (CNPL). CNPL logic is more expressive than conjunction of negative literal (CNL) logic by definition, but lacks disjunction and is therefore less expressive than full propositional logic.

This paper is organized as follows. We first introduce notation and concepts important to model theory and formal language theory. Then we introduce the concept of substructure and logics in the context of the Subregular Hierachy (McNaughton and Papert, 1971; Rogers and Pullum, 2011), highlighting some of the insights the model-theoretic approach brings. We then further motivate our exploration by bringing out two advantages of model-theoretic approaches: grammars expressed as logical statements and enriched model theories of strings. Then we turn to the phonological example where we present an enriched model and show how it simplifies the grammar. The penultimate section shows how a simple twist on string extension learning (Heinz, 2010) can be used to learn a class of formal languages containing the target grammars. And then we conclude.

2. Model Theory, Logic, and Formal Language Theory

Let Σ denote a finite set of symbols, the alphabet, and Σ^* the set of elements of the free monoid of Σ under concatenation. We refer to these elements both as *strings* and as *words*. The ith symbol in word w is denoted w_i. We sometimes make use of left and right word boundary markers ($\rtimes$ and $\ltimes$, respectively), which are symbols not in Σ.

If u and v are strings, let uv denote the concatenation of u and v. For all $u, v, w, x \in \Sigma^*$, if $x = uwv$ then then w is a *substring* of x. If $x \in \Sigma^* w_1 \Sigma^* w_2 \Sigma^* \dots w_n \Sigma^*$ then w is a *subsequence* of x. A substring (subsequence) of length k is called a k-factor (k-subsequence). Let $\texttt{factor}_k(w)$ denote the set of substrings of w of length k. Let $\texttt{subseq}_k(w)$ denote the set of subsequences of w up to length k. The domains of these functions are extended to languages in the normal way.

A formal language is a subset of Σ^*. We assume some familiarity with regular and subregular language classes shown in Figure 1 (McNaughton and Papert, 1971; Rogers et al., 2010; Rogers and Pullum, 2011; Rogers et al., 2013). Here are definitions of the lower four classes of languages.

Locally Testable. Language L is Locally k-Testable (LT_k) iff there is some k such that, for all strings x and y: if $\texttt{factor}_k(\rtimes x \ltimes) = \texttt{factor}_k(\rtimes y \ltimes)$ then $x \in L \leftrightarrow y \in L$. L is Locally Testable (LT) if there is some k so $L \in \mathrm{LT}_k$ (Rogers and Pullum, 2011).

Piecewise Testable. A language L is Piecewise k-Testable (PT_k) iff there is some k such that, for all strings x and y: if $\texttt{subseq}_k(x) = \texttt{subseq}_k(y)$ then $x \in L \leftrightarrow y \in L$. L is Piecewise Testable (PT) if there is some k such that $L \in \mathrm{PT}_k$.

Strictly Local. A stringset L is Strictly k-Local (SL_k) iff whenever there is a string x of length $k-1$ and strings u_1, v_1, u_2, and v_2, such that $u_1 x v_1, u_2 x v_2 \in L$ then $u_1 x v_2 \in L$. (We say L is *closed under suffix substitution*). L is Strictly Local (SL) if $L \in \mathrm{SL}_k$ for some k (Rogers and Pullum, 2011).

Strictly Piecewise. A language L is Strictly k-Piecewise (SP_k) iff $\texttt{subseq}_k(w) \subseteq \texttt{subseq}_k(L)$ implies $w \in L$. L is Strictly Piecewise (SP) if there is a k such that it belong to SP_k; equivalently, L belongs to SP iff L is closed under subsequence (Rogers et al., 2010).

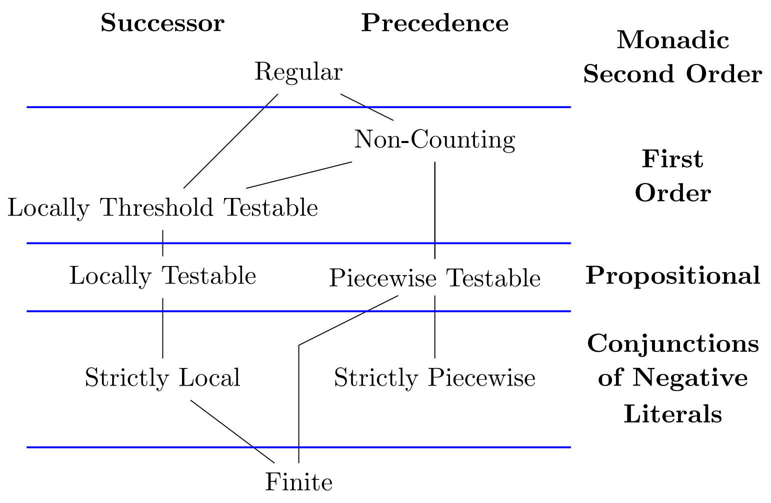

Figure 1: Subregular Hierarchies from a model-theoretic perspective.

We assume familiarity with Monadic Second Order (MSO) logic and First Order (FO) logic. Definitions of these logics are involved and omitted for space, but readers unfamiliar with them are encouraged to read Enderton (2001). We explain informally below how formal languages can be defined with logical sentences and word models.

Model theory studies objects in terms of mathematical logic (Enderton, 2001). A model of an object is a structure containing information about the object. The type of information present in a model theory $\mathfrak{M}$ of a set of objects Ω is given by the *model signature* and a set of mathematical statements about how structures are *interpreted*.

A model signature contains a domain (a set of elements), a set of constants, a set of relations, and a set of functions. These denote elements, relationships among elements, and maps from (tuples of) elements to other elements, respectively. We only consider model signatures with finite domains and whose signatures contain neither constants nor functions. In other words, we apply *finite* model theory to *relational structures*.

Formally, a model theory $\mathfrak{M}$ of a set of objects Ω has the signature $\langle \mathfrak{D}, \mathfrak{R} \rangle$ where there exists $n \in \mathbb{N}$ such that $\mathfrak{R}$ is a set of n relations, and for each $1 \leq i \leq n$, there exists a_i such that $R_i \in \mathfrak{R}$ is an a_i-ary relation. A *structure* in $\mathfrak{M}$ is thus any $\mathcal{S} = \langle D, \{R_1, R_2, \ldots, R_n\} \rangle$ where D is finite and each R_i is an a_i-ary relation over D. Since D is finite, its elements are standardly given as elements of $\mathbb{N}$: $D = \{1, \ldots k\}$ for some $k \in \mathbb{N}$. The size of $\mathcal{S}$ is $|\mathcal{S}| = k$.

Not all structures under $\mathfrak{M}$ are interpretable as objects of Ω. (An example is given below.) For $\mathfrak{M}$ to model Ω, each object in Ω must have some interpretable structure and distinct objects must have distinct interpretable structures. A structure that is interpretable as an object $o \in \Omega$ is a *model* of o (denoted $\mathcal{M}_o$). We present two distinct model theories for words to illustrate. For concreteness, let $\Sigma = \{a, b, c\}$ and $\Omega = \Sigma^*$.

Following Büchi (1960), Rogers and Pullum (2011), and Rogers et al. (2013), a common model for words in $\{a, b, c\}^*$ is the Successor Word Model ($\mathfrak{M}^{\triangleleft}$), which is given by the signature $\langle \mathfrak{D}, \{\triangleleft, R_a, R_b, R_c\} \rangle$ where $\triangleleft$ is the binary ordering relation *successor* and for each $\sigma \in \{a, b, c\}$, R_σ is a unary relation denoting which elements are labeled σ. This model will be contrasted with the Precedence Word Model ($\mathfrak{M}^{<}$), which has the signature

$\langle \mathfrak{D}, \{<, R_a, R_b, R_c\}\rangle$ where $<$ is the binary ordering relation *precedence* and the unary relations R_σ are the same as in $\mathfrak{M}^{\triangleleft}$. In both model theories, each string $w \in \Sigma$ of length k has a unique interpretable structure (model). The domain of D of $\mathcal{M}_w$ is $\{1, 2 \ldots k\}$. For each $\sigma \in \Sigma$, the unary relation $R_\sigma = \{i \in D \mid w_i = \sigma\}$. The only difference is the ordering relation. Under $\mathfrak{M}^{\triangleleft}$, the ordering relation $\triangleleft = \{(i, i+1) \in D \times D\}$ but under $\mathfrak{M}^{<}$, $<= \{(i,j) \in D \times D \mid i < j\}$.

Figure 2 illustrates these different word models with the word *cabb*, along with graphical representations of the models. In these graphs, nodes represent domain elements; binary relations are shown with directed labeled edges; and unary relations are shown as labels above the nodes. While significant aspects of the models illustrated in Figure 2 are the same, information about the order of the elements is represented differently.

$$\mathcal{M}^{\triangleleft}_{cabb} = \langle \{1,2,3,4\}, \{\langle 1,2\rangle, \langle 2,3\rangle, \langle 3,4\rangle\}, \{2\}, \{3,4\}, \{1\}\rangle$$

$$\mathcal{M}^{<}_{cabb} = \langle \{1,2,3,4\}, \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 2,3\rangle, \langle 2,4\rangle, \langle 3,4\rangle\}, \{2\}, \{3,4\}, \{1\}\rangle$$

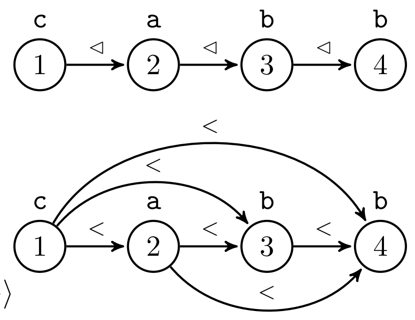

Figure 2: Successor and precedence models for word *cabb* with graphical representations.

It follows that certain conditions must be met for structures to be interpretable as strings. In both theories, for a structure $\mathcal{S}$ with domain D to be interpretable as a word, each element in D must have at least one label $((\forall i \in D)(\exists \sigma \in \Sigma)[i \in R_\sigma])$ and at most one label $((\forall \sigma, \sigma' \in \Sigma)[R_\sigma \cap R_{\sigma'} = \emptyset])$. Furthermore, in both theories every element must be ordered. For $\mathfrak{M}^{\triangleleft}$, this means $\triangleleft = \{(i, i+1) \in D \times D\}$. For $\mathfrak{M}^{<}$, it means $<= \{(i,j) \in D \times D \mid i < j\}$.

The structure $\mathcal{S} = \langle \{1,2\}, \{\emptyset, \{1\}, \{2\}, \emptyset\}\rangle$ is an example of a structure in both $\mathfrak{M}^{\triangleleft}$ and $\mathfrak{M}^{<}$ which is not interpretable as a string. $\mathcal{S}$ specifies two elements, one of which is labeled a and one is labeled b. But the order of these elements remains unspecified.

For any two relational structures $\mathcal{S}_1$ and $\mathcal{S}_2$ of the same theory, we say $\mathcal{S}_1$ is a *substructure* of $\mathcal{S}_2$ (written $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$) iff there exists a function h which maps every element in D_1, the domain of $\mathcal{S}_1$, to elements in D_2, the domain of $\mathcal{S}_2$, such that for all n-ary relations R and all n-tuples of elements of D_1, $R_1(x_1, \ldots x_n)$ iff $R_2(h(x_1), \ldots h(x_n))$. Observe that h is an injective homomorphism.

For example under $\mathfrak{M}^{\triangleleft}$, $\mathcal{M}_{ab}$ is a sub-structure of $\mathcal{M}_{cabb}$. (Let h map 1 to 2 and 2 to 3.) Under $\mathfrak{M}^{<}$, $\mathcal{M}_{cb}$ is a sub-structure of $\mathcal{M}_{cabb}$. (Let h map 1 to 1 and 2 to 3 (or 4).)

The lemma below is not difficult to prove.

Lemma 1 *For all $u, v \in \Sigma^*$, word u is a substring of v iff $\mathcal{M}^{\triangleleft}_u \sqsubseteq \mathcal{M}^{\triangleleft}_v$. Likewise, u is a subsequence of v iff $\mathcal{M}^{<}_u \sqsubseteq \mathcal{M}^{<}_v$.*

These facts help make clear similarities between substrings and subsequences observed in earlier works (García and Ruiz, 2004; Lothaire, 1997, 2005). They are both sub-structures.

For any structure $\mathcal{S}$ in model theory $\mathfrak{M}$ for Ω, let $\texttt{substrucs}(\mathcal{S}) \overset{\text{def}}{=} \{\mathcal{S}' \mid \mathcal{S}' \sqsubseteq \mathcal{S}\}$ and $\texttt{substrucs}_k(\mathcal{S}) \overset{\text{def}}{=} \{\mathcal{S}' \mid \mathcal{S}' \sqsubseteq \mathcal{S} \text{ and } |\mathcal{S}'| \leq k\}$. Note that $\texttt{substrucs}_k(\mathcal{S})$ includes sub-structures of $\mathcal{S}$ of length k or less.

It is well-known that logical expressions together with a model theory for words can define classes of formal languages. Given a logical language and a logical expression ϕ in this language, together with a model theory $\mathfrak{M}$ for words, then the formal language L defined by ϕ is simply those words whose models satisfy ϕ. Satisfaction (denoted $\models$) is defined as part of the semantics of the logic (see Enderton (2001)). Formally, $L(\phi) \overset{\text{def}}{=} \{w \in \Sigma^* \mid \mathcal{M}_w \models \phi\}$.

Here is an example. In the terminology of FO logic, if R is a symbol denoting a n-ary relation in $\mathfrak{R}$ then $R(x_1, \dots, x_n)$ is an *atomic formula* in the logical language (each x_i is a variable ranging over the domain). If $\phi \overset{\text{def}}{=} (\exists x)[R_a(x)]$ then $L(\phi) = \Sigma^* a \Sigma^*$ since every word w containing a satisfies ϕ. This is because there is an element i in the domain of $\mathcal{M}_w$ to which variable x can be assigned which makes $R_a(x)$ true since $i \in R_a$.

More generally, each structure $\mathcal{S} = \langle D, \{R_1 \dots R_n\}\rangle$ in signature $\langle \mathfrak{D}, \mathfrak{R}\rangle$ with each R_i relation of arity a_i and with $|D| = k$ defines the FO sentence $\phi_{\mathcal{S}} \overset{\text{def}}{=} (\exists x_1, \dots x_k)\left[\bigwedge_{1\leq i \leq n, \boldsymbol{x} \in R_i} R_i(\boldsymbol{x})\right]$ where $\boldsymbol{x}$ is a tuple of variables of size a_i. When $\mathcal{S}$ is a model of a word w, we write ϕ_w instead of $\phi_{\mathcal{M}_w}$. The lemma below is not hard to prove.[1]

Lemma 2 *For each structure $\mathcal{S}$ and $w \in \Sigma^*$, $\mathcal{S} \sqsubseteq \mathcal{M}_w$ iff $\mathcal{M}_w \models \phi_{\mathcal{S}}$.*

For example, under $\mathfrak{M}^{\lhd}$, $\phi^{\lhd}_w = (\exists x_1, \dots x_k)\left[\bigwedge_{1 \leq i \leq k-1} x_i \lhd x_{i+1} \ \bigwedge_{1\leq i \leq k} R_{w_i}(x_i)\right]$. Similarly, under $\mathfrak{M}^{<}$, $\phi^{<}_w = (\exists x_1, \dots x_k)\left[\bigwedge_{1 \leq i < j \leq k} x_i < x_j \ \bigwedge_{1\leq i \leq k} R_{w_i}(x_i)\right]$. It follows from Lemmas 1 and 2 that word u is a substring of v iff $\mathcal{M}^{\lhd}_v \models \phi^{\lhd}_u$ and that u is a subsequence of v iff $\mathcal{M}^{<}_v \models \phi^{<}_u$. We take advantage of these lemmas in the case study in sections 4 and 5.

The aforementioned concepts (logic and model theory) provide a unified simple way to understand the classes of formal languages shown in Figure 1. The class of languages definable with sentences of Monadic Second-Order (MSO) logic under $\mathfrak{M}^{\lhd}$ are exactly the regular languages (Büchi, 1960). (MSO logic with $\mathfrak{M}^{<}$ is also exactly the regular languages since both precedence and successor are MSO-definable from the other.) Restricting the logical language to First Order (FO) logic but keeping $\mathfrak{M}^{\lhd}$ yields the Locally Threshold Testable (LTT) class of formal languages (Thomas, 1997). On the other hand, languages of the sentences of FO logic under $\mathfrak{M}^{<}$ yields exactly the Non-counting languages (McNaughton and Papert, 1971). Note the Non-counting class properly contains LTT because successor ($\lhd$) is FO-definable from precedence ($<$) but not vice versa (see Figure 1).

In the same way that sentences of FO logic are a subset of sentences of MSO logic, restricting sentences of FO logic in particular ways yields the Locally Testable (LT), Strictly Local (SL), Piecewise Testable (PT), and Strictly Piecewise (SP) classes of languages. Call a FO sentence ϕ *propositional* if there is a finite set of words W such that ϕ is a Boolean combination of the formulae ϕ_w (with $w \in W$).[2] Then the class of LT languages is exactly the set of languages definable with propositional sentences under $\mathfrak{M}^{\lhd}$ of $\{\rtimes\}\Sigma^*\{\ltimes\}$ and

1. There is a technical assumption here that the domain elements occur in some relation in $\mathfrak{R}$. If not, we have to include 'equals' ($=$) into the logic and add statements like $x_i \neq x_j$ in the FO formula.
2. This means that ϕ is a formula which combines formulae ϕ_w ($w \in S$) only with logical connectives such as negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$), and the biconditional ($\leftrightarrow$).

the class of PT languages is exactly the set of languages definable with propositional sentences under $\mathfrak{M}^{<}$ of Σ^* (Thomas, 1997, Theorem 4.5).[3] The 'atomic' expressions in this 'propositional logic' are the formulae ϕ_w. We use the word *literal* to mean such an atomic expression or its negation.

The SL and SP classes are obtained by restricting the set of propositional sentences further. A propositional sentence ϕ is a *conjunction of negative literals* (CNL) if it has the form $\bigwedge_{w \in W} \neg\phi_w$. The SL languages are those obtained from CNL sentences under $\mathfrak{M}^{\triangleleft}$, and the SP languages are those obtained from CNL sentences under $\mathfrak{M}^{<}$ (Rogers et al., 2010, 2013). Thus SL (SP) languages essentially forbid a finite set of k-factors (k-subsequences).

Finally, a propositional sentence is a *k-sentence* if the longest word in W is of length k. Then the class of LT_k (PT_k) languages is the set of languages which are LT (PT) and definable with propositional k-sentences. Similarly, SL_k (SP_k) is the set of languages which are SL (SP) and definable with CNL k-sentences.

3. Motivation for the Present Work

Why model theoretic treatments of formal languages? There are two reasons. The first is that it provides a bridge to a range of techniques that have been used to learn concepts expressed as logical sentences. The second is that there are many more kinds of models of words we can imagine and explore than the successor and precedence models. We argue both of these are important for grammatical inference. But first we explain in more detail these two reasons. Our case study in the later sections illustrates both of these points.

Take learning. It is known that for a given k, the LT_k and PT_k languages are identifiable in the limit from positive data with different kinds of finite-state techniques (García and Ruiz, 2004; Heinz and Rogers, 2013), though neither of these papers addresses the data complexity in the sense of de la Higuera (1997). The logical characterizations provided in Section 2, however, allow different learning strategies to be applied, such as concept learning and related techniques (Flach, 2012; De Raedt, 2008). For example, Valiant (1984) presents an algorithm that PAC-learns bounded CNF (Conjunctive Normal Form) expressions. A CNF expression is the conjunction of clauses where each clause is a disjunction of atomic expressions or their negation (i.e., literals). A m-CNF expression ϕ is one where the number of atomic expressions in any clause of ϕ does not exceed m. It follows that this algorithm can be used to PAC-learn the Locally k-Testable and Piecewise k-Testable languages.

To see why, we have to show that for every k there is an m such that for every k-sentence ϕ there exists a m-CNF formula ψ such that ψ is equivalent to ϕ. First, there is the fact that every sentence ϕ of propositional logic can be expressed as an equivalent expression ψ which is in CNF (Enderton, 2001). Since the size of Σ and k bound the number of literals to $|\Sigma|^{k+1}$ we can set m accordingly.[4] Hence no clause can exceed this many literals. So m is a (possibly large) constant factor and these classes of languages are PAC-learned by Valiant's algorithm. This provides a window into the data complexity required to learn these concepts, at least in the approximately correct setting (which we acknowledge is different from de la Higuera's 1997 study of exact identification).

3. A reviewer adds that word boundaries can also be handled with theories with constants `min` and `max`.
4. Using word boundaries would alter this value but not in a way that undermines the main point.

The second reason to be interested in model theory is that much richer models of words can be considered (and exploited) in scientific fields where richer models are warranted. In both the successor and precedence models of words, it is the case that $R_\sigma \cap R_{\sigma'} = \emptyset$. No domain element can satisfy more than one unary relation; no position in a string can be more than one letter. Alphabetic symbols are taken to be independent and wholly distinct.

In many domains, however, there are reasons to believe this is not the case. In biology, the sequences of symbols $\{A, C, G, T\}$ are used to represent sequences of nucleotides in DNA structures. However, these symbols are not wholly independent; A and T are considered complementary in a way that A and G are not. This could be encoded in the representation in part by including a unary relation which is true of both the A and T symbols. Another example comes from robotic control and planning where sequences of symbols often denote series of actions. The symbols here are also not necessarily independent; distinct actions may involve common joints, postures, motors, and articulators. (See Coste (2016) and (Fu et al., 2015) for grammatical inference approaches to these fields, respectively.)

Another domain is phonology, the study of systematic aspects of the pronunciation of words and phrases in natural languages (Odden, 2014). It is well known that distinct speech sounds are organized along certain dimensions. For example, the speech sounds [f] and [v] are pronounced identically except the [v] involves additional vocal fold vibration called *voicing*. This is also the only difference between [s] and [z]. Under a model-theoretic approach, a theory can be constructed in which [v] and [z] both satisfy the unary relation `voicing` but [f] and [s] do not.[5] While a comprehensive study of model-theoretic approaches to these dimensions remains to be done,[6] we believe the model-theoretic approach provides a way to bridge linguistic theory with formal language theory and learning theory.

In the next two sections, we explore this kind of model-theoretic approach to the problem of representing and learning unbounded stress systems in phonology.

4. Case Study: The Nature of Unbounded Stress Patterns

Phonological stress refers to the auditory or perceptual prominence of a certain syllable in a word or phrase (Hayes, 1995). For example, the second syllable in *America* is stressed in English. In many languages the position of stress in words is predictable (van der Hulst et al., 2010). The problem we study is how language-specific stress patterns can be learned from positive data (Dresher and Kaye, 1990; Tesar and Smolensky, 1998; Heinz, 2009).

As explained in Hayes (1995), some stress patterns are *quantity-sensitive*, meaning they are related to syllable weight, another type of prominence. Languages differ in how they assign syllable weight, but this is not relevant to our analysis. Usually, there are two weights: light (L) and heavy (H). We use the acute accent to denote stress (e.g., Ĺ represents a stressed light syllable). Hence we let $\Sigma = \{$L, H, Ĺ, H́$\}$. Our interest is in those subsets of Σ^* which represent predictable stress patterns in natural languages.

If stress is always positioned predictably within a window of size k from the beginning or end of the word, the pattern is said to be *bounded*. As such, these patterns are SL$_k$. Unbounded stress patterns are those that are not bounded. There are four simple types of

5. Similarly, there are 52 symbols in the English alphabet, but these could be modeled with 27 unary relations: one for each of the 26 letters and another unary relation `uppercase`.
6. Though see Graf (2010) for the first thorough model-theoretic treatment of phonological theories.

unbounded stress patterns (Hayes, 1995) and at least a dozen more complex types (Heinz, 2007). Kwakiutl exhibits one simple unbounded pattern, where stress falls on the leftmost heavy syllable in words with heavy syllables and on the rightmost syllable in words without them (Bach, 1975). This is called LHOR (Leftmost Heavy, Otherwise Right) (Hayes, 1995). The three other simple unbounded patterns are variations on this theme. Table 1 shows some possible word types in an LHOR language.

Heinz (2014) shows that these simple unbounded patterns are neither SP nor SL. Recall that SP languages are closed under subsequence. Thus, LHOR is not SP since the word LLĹ belongs to LHOR but LL does not. To see why LHOR is not Strictly k-Local for any k, consider that both H́L^{k}L and L^{k}Ĺ belong to LHOR but H́L^{k}Ĺ does not. Thus LHOR is not closed under suffix substitution for any k.

H́	Ĺ	LĹ	H́L
LLH́	H́H	LLĹ	H́HH
…			

Table 1: Possible word types

Permissible				Forbidden			
LL	HH	LĹ	HL	HH́	ĹL	HĹ	ĹĹ
LH	H́H	LH́	H́L	H́H́	ĹH	H́Ĺ	ĹH́

Table 2: 2-subsequences (LHOR)

Heinz (2014) goes on to show that simple unbounded stress patterns are SP$_2$ languages intersected with the language $L = \Sigma^*\acute{\mathrm{L}}\Sigma^* \cup \Sigma^*\acute{\mathrm{H}}\Sigma^*$, which is the language where each word has at least one stress. Table 2 shows the permissible and forbidden 2-subsequences of LHOR. It is not hard to show that L belongs to neither SL nor SP, but that it is LT$_1$ and PT$_1$ (since for all $w \in L$, $\texttt{factor}_k(w)$ and $\texttt{subseq}_k(w)$ must contain Ĺ or H́). Rather than move up the hierarchy, Heinz suggests L is innate and does not need to be learned.

However, an obvious alternative given the discussion in section 3 is to apply a learning algorithm for PT$_2$ (since this contains all PT$_1$ and SP$_2$ languages and their intersections (Rogers et al., 2010)). Letting W = {HH́, H́H́, ĹL, ĹH, HĹ, H́Ĺ, HĹ, ĹĹ, ĹH́} (the forbidden 2-subsequences of LHOR), then the sentence ϕ_{PT} below exactly describes the LHOR language under $\mathfrak{M}^{<}$. Observe it is both a propositional 2-sentence (so belongs to PT$_2$) and a 2-CNF expression since the final clause has two literals.

$$\phi_{\mathrm{PT}} = \bigwedge_{w \in W} \neg\phi_w \wedge (\phi_{\acute{\mathrm{L}}} \vee \phi_{\acute{\mathrm{H}}}) \tag{1}$$

We set aside here the linguistic reason against this proposal (that a PT$_2$ theory of unbounded stress may significantly overgenerate the attested typology). We are more interested in reducing the time and/or data complexity of learning the simple unbounded stress patterns.

Our proposal comes in two parts. First, we consider a class of languages obtained by *conjunctions of positive and negative literals* (CNPL). CNPL sentences are just 1-CNF expressions since each clause contains exactly one literal, positive or negative. Given a model theory $\mathfrak{M}$ of a set of objects S, let $\mathbb{CNPL}(\mathfrak{M}, S)$ denote the set of CNPL sentences obtained under $\mathfrak{M}$ and let $\mathbb{CNPL}_k(\mathfrak{M}, S)$ denote the set of CNPL k-sentences obtained under $\mathfrak{M}$. It remains an open question what the exact nature of $\mathbb{CNPL}(\mathfrak{M}^{<}, \Sigma^*)$ and $\mathbb{CNPL}(\mathfrak{M}^{\triangleleft}, \{\rtimes\}\Sigma^*\{\ltimes\})$ are, other than they are sandwiched in between PT and SP, and LT and SL, respectively. Restricting the hypothesis space in this way can make learning easier. Specifically, Valiant's (1984) bounded CNF learning algorithm learns 1-CNF formulae with less time and data than 2-CNF formulae.

Secondly, we consider a non-standard word model. There is no 1-CNF formula for LHOR under the $\mathfrak{M}^<$ theory of Σ^* (note the final clause in (1) has two literals). As will be shown, our alternative model has a 1-CNF formula which describes exactly LHOR.

The other word model for Σ^* we consider is $\mathfrak{M}$ where $\mathfrak{R} = \{<, \texttt{light}, \texttt{heavy}, \texttt{stress}\}$. As before $<$ is the binary precedence relation. The other relations are unary, with the following interpretations: $\texttt{light} = \{i \in D \mid w_i \in \left\{\text{L}, \acute{\text{L}}\right\}\}$; $\texttt{heavy} = \{i \in D \mid w_i \in \{\text{H}, \acute{\text{H}}\}\}$; and $\texttt{stress} = \{i \in D \mid w_i \in \{\acute{\text{L}}, \acute{\text{H}}\}\}$.

It is important to note that, unlike $\mathfrak{M}^\triangleleft$ or $\mathfrak{M}^<$, domain elements of a structure in $\mathfrak{M}$ can belong to more than one (or no) unary relation. For example in the model of the word LLḢL, $3 \in \texttt{heavy}$ and $3 \in \texttt{stress}$. This model faithfully captures the linguistic truism that stress is an *additional* aspect of syllables. Visually, each node may have multiple (or no) labels, a key difference between this model of string structure and traditional models.

We use an in-text shorthand for structures. Figure 3 summarizes structures with the symbol we use in-text shown below each one. Ḣ and Ĺ are fully-specified structures of size one. H and L represent heavy and light syllables that are unspecified for stress. Similarly, $\acute{\sigma}$ represents a stressed syllable unspecified for weight, and σ a completely unspecified syllable.

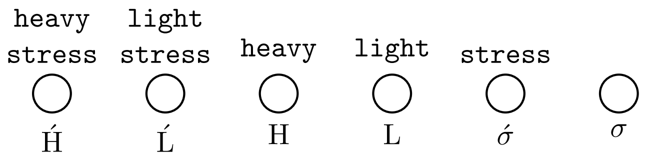

Figure 3: Structures of size 1 and their in-text shorthand.

We concatenate these symbols to represent longer structures. For example we write LLĹ to indicate the structure in Figure 4. Four of its sub-structures are shown in Figure 5.

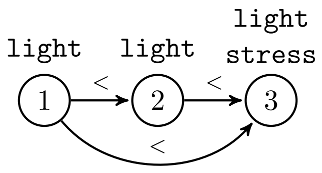

Figure 4: The structure LLĹ.

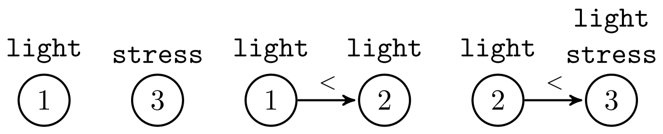

Figure 5: Four sub-structures of LLĹ: L, $\acute{\sigma}$, LL, LĹ.

We now construct a new sentence under $\mathfrak{M}$ that also describes LHOR exactly. Every word must have at least one syllable (regardless of quality) and one stressed syllable. Under $\mathfrak{M}$, these positive literals correspond to structures σ and $\acute{\sigma}$. Crucially, these structures are *underspecified*; they are not models themselves, but are sub-structures of $\mathcal{M}_{\acute{H}}$ and $\mathcal{M}_{\acute{L}}$.

The banned structures can also be simplified under $\mathfrak{M}$. A stressed light is only permissible if it is the final syllable. Thus one of the banned structures in the LHOR pattern is a stressed light followed by any other syllable (Ĺσ). Again, this structure is underspecified. It models no string in Σ^* but is a sub-structure of models of many strings. In particular, it is a sub-structure of the models of four forbidden 2-subsequences in Table 2: ĹH, ĹḢ, ĹL, and ĹĹ. The LHOR pattern also mandates that a word with one or more heavy syllables have stress on the leftmost heavy. Consequently, a heavy syllable may not be followed by any stressed syllable (H$\acute{\sigma}$). Forbidding this structure effectively bans the remaining four forbidden 2-subsequences from Table 2: HḢ, ḢḢ, HĹ, and ḢĹ. Thus, LHOR can be described with a 1-CNF formula with four literals under $\mathfrak{M}$, as shown in (2), which contrasts with the 2-CNF formula ϕ_{PT} under $\mathfrak{M}^<$ in (1).

$$\phi_{\text{LHOR}} = \phi_\sigma \wedge \phi_{\acute{\sigma}} \wedge \neg\phi_{\acute{\text{L}}\sigma} \wedge \neg\phi_{\text{H}\acute{\sigma}} \tag{2}$$

The 2-sentence ϕ_{LHOR} refers to structures of size 2 or less, which are analogous to 2- and 1-subsequences. Forbidding structures (as opposed to models) allows us to capture the core generalizations of LHOR without referring to a seemingly arbitrary list of subsequences.

It is natural to wonder whether a model with fewer unary relations could make these same distinctions. Is a `light` syllable just the absence of a `heavy` syllable in the same way an unstressed syllable is the absence of `stress`? We don't believe it would suffice to remove `light` from the theory. Without `light`, the structure $\acute{\sigma}\sigma$ is a sub-structure of the model of the ungrammatical word ĹL, but also the grammatical H́L. We only want to ban a stressed syllable followed by another syllable *if the first is light*, so the `light` label appears necessary to differentiate forbidden structures from permissible ones.

5. Learning k-CNPL

The learning algorithm identifies the target CNPL sentence by simultaneously building two sets: one for *permissible* structures (G_{per}) and another for *required* structures (G_{req}). Because conjunction is commutative, CNPL sentences can be rearranged into two groups of literals: negative literals and positive literals. For instance, $p \wedge \neg q \wedge r \wedge \neg s$ is equivalent to $(\neg q \wedge \neg s) \wedge (p \wedge r)$. As suggested by Lemmas 1 and 2, it is straightforward to construct a CNPL sentence once these structures are identified. Setting $G = (G_{per}, G_{req})$, let $\phi_G = \bigwedge_{\mathcal{S}\in G_{req}} \phi_{\mathcal{S}} \bigwedge_{\mathcal{S}\in\overline{G_{per}}} \neg\phi_{\mathcal{S}}$ where $\overline{G_{per}}$ is the complement of G_{per} with respect to a finite set of structures of bounded size (so k is understood).

The idea of learning forbidden structures from permissible ones goes back to García et al. (1990) and was later generalized by Heinz (2010) and Heinz et al. (2012). The required sub-structures are exactly the ones represented by positive literals in a CNPL sentence. Learning the required structures is akin to cross-situational learning (Siskind, 1996, inter alia), where learners seek what is common among different events.

Let $L(G_{per}) = \{w \in \Sigma^* \mid \texttt{substrucs}_k(\mathcal{M}_w) \subseteq G_{per}\}$. Let $L(G_{req}) = \{w \in \Sigma^* \mid G_{req} \subseteq \texttt{substrucs}_k(\mathcal{M}_w)\}$. Then the language of $G = (G_{per}, G_{req})$ is the intersection of $L(G_{per})$ and $L(G_{req})$. Clearly for any $\phi \in \mathbb{CNPL}_k(\mathfrak{M}, \Sigma^*)$, $L(\phi) = L(\phi_G) = L(G)$. $L(G)$ is exactly those words whose models contain all required structures and no forbidden ones.

The learning algorithm is defined in Equation 3. It sequentially draws from a sequence of words $t(1), t(2), \ldots$, each of which belongs to the target k-CNPL language $L \subseteq S$. For each $t(n)$, the algorithm returns a grammar $G(n) = (G_{per}(n), G_{req}(n))$ with $G_{per}(1) = G_{req}(1) = \texttt{substrucs}_k(t(1))$. As n increases, the learner identifies k-sub-structures of $t(n)$ and adds them to G_{per}. Thus learning G_{per} is a string extension learning (Heinz, 2010). In contrast, to learn G_{req}, the algorithm takes the *intersection* of the sets k-sub-structures of each datum, leaving only the k-sub-structures that are common to all observed words.

$$\begin{aligned}
(G_{per}(1), G_{req}(1)) &= (\texttt{substrucs}_k(t(1)), \texttt{substrucs}_k(t(1)). \\
G_{per}(n) &= G_{per}(n-1) \cup \texttt{substrucs}_k(t(n)) \text{ for } n > 1. \\
G_{req}(n) &= G_{req}(n-1) \cap \texttt{substrucs}_k(t(n)) \text{ for } n > 1.
\end{aligned} \tag{3}$$

We illustrate the algorithm on a sample run with the language LHOR and $k = 2$, as shown in Table 3. Let $t(1) = \text{LL\'{L}}$. The relational structure of LL\'L is given in Figure 4, and some of its sub-structures are shown in Figure 5. By definition, $G_{per}(1) = G_{req}(1) = \texttt{substrucs}_2(t(1)) = \{\text{LL}, \text{L\'{L}}, \text{L}, \text{\'{L}}, \sigma, \acute{\sigma}, \ldots\}$.

Let $t(2) = \text{\'{H}HL}$. The learner builds $G_{per}(2)$ by taking the union of $\texttt{substrucs}_2(t(2))$ and $G_{per}(1)$, adding the 2-sub-structures of $t(2)$ to its previously hypothesized grammar. In contrast, $G_{req}(2)$ takes the intersection of $\texttt{substrucs}_2(t(2))$ and $G_{req}(1)$, leaving only the common sub-structures: $\{\sigma, \acute{\sigma}, \text{L}\}$.

Let $t(3) = \text{\'{H}}$. As before, $\texttt{substrucs}_2(t(3))$ are added to $G_{per}(2)$. The only common structures of $G_{req}(2)$ and $\texttt{substrucs}_2(t(3))$ are σ and $\acute{\sigma}$. Since this is correct for LHOR, no more structures will be added to G_{req}; nor will more structures be removed from G_{req} because every word in LHOR contains both structures.

Let $t(4) = \text{L\'{H}}$. This includes two crucial 2-subsequences not present in $t(1)$ through $t(3)$: LH, and L\'H. The learner has now observed all permissible 2-subsequences in Table 2. Because their models contain all permissible 2-structures, it follows that *all* permissible 2-structures have been observed. In other words, $G_{per}(4)$ contains exactly those 2-structures that satisfy $\neg\phi_{\text{\'{L}}\sigma} \wedge \neg\phi_{\text{H}\acute{\sigma}}$. Thus $L(G_{per}(4)) = L(\neg\phi_{\text{\'{L}}\sigma} \wedge \neg\phi_{\text{H}\acute{\sigma}})$. Also, it is clear that $L(G_{req}(4)) = L(\phi_\sigma \wedge \phi_{\acute{\sigma}})$. By definition, $L(G(4)) = L(\phi_\sigma \wedge \phi_{\acute{\sigma}}) \cap L(\neg\phi_{\text{\'{L}}\sigma} \wedge \neg\phi_{\text{H}\acute{\sigma}})$. It follows that $L(G(4)) = L(\phi_\sigma \wedge \phi_{\acute{\sigma}} \wedge \neg\phi_{\text{\'{L}}\sigma} \wedge \neg\phi_{\text{H}\acute{\sigma}}) = L(\phi_{LHOR})$. The learner has therefore converged on the correct grammar for the LHOR stress pattern as described in (2).

i	$t(i)$	$G_{per}(i)$	$G_{req}(i)$
1	LL\'L	$\texttt{substrucs}_2(t(1))$ $= \{\text{LL}, \text{L\'{L}}, \text{L}, \text{\'{L}}, \sigma, \acute{\sigma}, \ldots\}$	$\texttt{substrucs}_2(t(1))$ $= \{\text{LL}, \text{L\'{L}}, \text{L}, \text{\'{L}}, \sigma, \acute{\sigma}, \ldots\}$
2	\'HHL	$G_{per}(1) \cup \texttt{substrucs}_2(t(2))$ $= \{\text{LL}, \text{L\'{L}}, \text{HH}, \text{\'{H}H}, \text{HL}, \text{\'{H}L}, \ldots\}$	$G_{req}(1) \cap \texttt{substrucs}_2(t(2))$ $= \{\sigma, \acute{\sigma}, \text{L}\}$
3	\'H	$G_{per}(2) \cup \texttt{substrucs}_2(t(3))$ $= \{\text{LL}, \text{L\'{L}}, \text{HH}, \text{\'{H}H}, \text{HL}, \text{\'{H}L}, \ldots\}$	$G_{req}(2) \cap \texttt{substrucs}_2(t(3))$ $= \{\sigma, \acute{\sigma}\}$
4	L\'H	$G_{per}(3) \cup \texttt{substrucs}_2(t(4))$ $= \{\text{LL}, \text{L\'{L}}, \text{HH}, \text{\'{H}H}, \text{HL}, \text{\'{H}L}, \text{LH}, \text{L\'{H}}, \ldots\}$	$G_{req}(3) \cap \texttt{substrucs}_2(t(4))$ $= \{\sigma, \acute{\sigma}\}$

Table 3: $G(i)$, $1 \leq i \leq 4$

Of course, it would be interesting to apply this algorithm to the full range of unbounded stress patterns discussed in earlier research.

6. Conclusion

This paper presents a model-theoretic approach to formal languages and grammatical inference. First, we confronted the assumption that unary relations on string positions be mutually exclusive and considered some advantages of dispelling it. The string representations made possible by model theories allow sub-structures to capture similarities between distinct alphabetic symbols, which seems appropriate in certain scientific fields.

Second, we pointed out that a type of logic, CNPL, or equivalently 1-CNF, corresponds to a class of subregular languages not yet formally described. We presented an example that showed how a CNPL sentence of enriched string sub-structure literals can be used reduce the learning complexity of simple unbounded stress patterns (from 2-CNF to 1-CNF). We also described a simple learning algorithm for CNPL sentences in section 5.

The goal of this paper was to show that model theory offers new ways to think about grammatical inference. It raises, of course, more questions than it answers. What is the exact nature of the class of languages defined by CNPL? What exactly are the time and data complexity savings with certain model-theoretic representations of strings? And must the model theory $\mathfrak{M}$ be given a priori or can it too be learned? We believe answering these will lead to new developments in grammatical inference.

Acknowledgments

We thank Rémi Eyraud, Greg Kobele, and Jim Rogers for helpful discussion, and three anonymous reviewers for valuable feedback. This work is supported by NIH#R01HD087133.

References

Emmon W Bach. Long vowels and stress in Kwakiutl. In *Texas Linguistic Forum*, volume 2, pages 9–19, 1975.

J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

François Coste. Learning the language of biological sequences. In Jeffrey Heinz and José Sempere, editors, *Topics in Grammatical Inference*, chapter 8, pages 215–247. Springer-Verlag Berlin Heidelberg, 2016.

Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.

Luc De Raedt. *Logical and Relational Learning*. Springer-Verlag Berlin Heidelberg, 2008.

Elan Dresher and Jonathan Kaye. A computational learning model for metrical phonology. *Cognition*, 34:137–195, 1990.

Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition, 2001.

Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

Jie Fu, Herbert G. Tanner, Jeffrey Heinz, Konstantinos Karydis, Jane Chandlee, and Cesar Koirala. Symbolic planning and control using game theory and grammatical inference. *Engineering Applications of Artificial Intelligence*, 37:378–391, January 2015.

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

Pedro García, Enrique Vidal, and José Oncina. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338, 1990.

Thomas Graf. Logics of phonological reasoning. Master's thesis, University of California, Los Angeles, 2010.

Bruce Hayes. *Metrical stress theory: Principles and case studies.* University of Chicago Press, 1995.

Jeffrey Heinz. Learning unbounded stress systems via local inference. In *Proceedings of the 37th Meeting of the Northeast Linguistics Society*, 2007. University of Illionois, Urbana-Champaign.

Jeffrey Heinz. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351, 2009.

Jeffrey Heinz. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden, July 2010.

Jeffrey Heinz. Culminativity times harmony equals unbounded stress. In Harry van der Hulst, editor, *Word Stress: Theoretical and Typological Issues*, chapter 8. Cambridge University Press, Cambridge, UK, 2014.

Jeffrey Heinz and James Rogers. Learning subregular classes of languages with factored deterministic automata. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 64–71, Sofia, Bulgaria, August 2013.

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127, October 2012.

M. Lothaire, editor. *Combinatorics on Words.* Cambridge University Press, Cambridge, UK, New York, 1997.

M. Lothaire, editor. *Applied Combinatorics on Words.* Cambridge University Press, 2nd edition, 2005.

Robert McNaughton and Seymour Papert. *Counter-Free Automata.* MIT Press, 1971.

David Odden. *Introducing Phonology.* Cambridge University Press, 2nd edition, 2014.

James Rogers and Geoffrey Pullum. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342, 2011.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. On languages piecewise testable in the strict sense. In *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artifical Intelligence*, pages 255–265. Springer, 2010.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. Cognitive and sub-regular complexity. In *Formal Grammar*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

Jeffrey Mark Siskind. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):39 – 91, 1996.

Bruce Tesar and Paul Smolensky. Learnability in optimality theory. *Linguistic Inquiry*, (29):229–268, 1998.

Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, chapter 7. Springer, 1997.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

H.G. van der Hulst, R. Goedemans, and E. van Zanten, editors. *A survey of word accentual patterns in the languages of the world.* Mouton de Gruyter, Berlin, 2010.