A Generalized Algorithm for Learning Positive and Negative Grammars with Unconventional String Models

Sarah Payne
Stony Brook University
sarah.payne@stonybrook.edu

Abstract

This paper introduces an algorithm for learning positive and negative grammars with enriched representational models. In conventional model-theoretic treatments of strings, each position belongs to exactly one unary relation. Strother-Garcia et al. (2016) introduce unconventional string models, in which multiple positions can have shared properties, and demonstrate their utility for grammatical inference. Chandlee et al. (2019) develop this approach for learning negative grammars. Here, we show that by fixing k — the size of the elements in the grammar — Chandlee et al.'s approach can be further generalized to learn both positive and negative grammars over unconventional string models. We prove that this algorithm finds the most general grammars which cover the data.

1 Introduction

A great deal of work on learning formal languages has made use of **conventional string models**, in which each position in a string belongs to exactly one unary relation (Heinz, 2010b; Heinz et al., 2012, i.a.). In this paper, we focus on learning over **unconventional string models**, in which positions in a string can have multiple, shared properties (§2.3; Strother-Garcia et al., 2016; Vu et al., 2018). For phonological applications, we can think of conventional string models as operating exclusively over segments — atomic, undecomposable, units — while unconventional string models operate over phonological features.

We focus on the learning of formal languages that can be defined by a set of banned (under a **negative** grammar, Rogers et al. 2013) or allowed (under a **positive** grammar, Heinz et al. 2012) substructures. These include the Strictly k-Local and Strictly k-Piecewise classes (Rogers et al., 2010; Rogers and Pullum, 2011, i.a.); many phonological and phonotactic generalizations fall into these classes (e.g., Heinz, 2018). While Strother-Garcia et al. (2016), Chandlee et al. (2019), and Rawski (2021) develop algorithms for learning negative grammars with unconventional string models, how to learn positive grammars with these models remains an open question. Recent work on language acquisition suggests that the child may construct positive phonological (Belth, 2023) and phonotactic (Payne, 2023) grammars, in line with evidence for positive syntactic and morphological grammars (e.g., Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023). While arguments have also been made for negative phonologicial grammars (e.g., Prince and Smolensky, 1993; Hayes and Wilson, 2008), these findings demonstrate that the learning of positive grammars from unconventional string models warrants further exploration.

When learning over conventional string models, positive and negative grammars are straightforwardly interdefinable (Heinz, 2010b); we may learn a positive grammar simply by learning a negative one and applying a post-hoc conversion. However, such a conversion is exponentially more expensive for unconventional string models (§4.3). What's more, the polarity of the grammar to be learned has implications for the learning trajectory: while the language of the grammar continuously *shrinks* as a negative grammar grows, it continuously *expands* as a positive grammar grows (§8). Hence, there exist independent psycholinguistic and computational motivations for learning positive grammars directly from unconventional string models.

In this paper, we adapt the learning algorithm of Chandlee et al. (2019) to learn both positive and negative grammars over unconventional string models. Specifically, Chandlee et al. exploit the partially-ordered hypothesis space given by unconventional string models to learn the most general *negative* grammars. We demonstrate that if the size of substructures in the grammar is fixed to be exactly k, then we can immediately adapt this algorithm to learn both the most general positive and

Proceedings of the Society for Computation in Linguistics (SCiL) 2024, pages 75-85.
Irvine, California, June 27-29, 2024

negative grammars. What's more, for any negative grammar learnable by the Chandlee et al. algorithm, our algorithm learns an equivalent negative grammar (§4.2). This paper is organized as follows: §2 provides preliminaries of model theory, §3 introduces subfactors and maxfactors, and §4 defines positive and negative grammars and their languages in terms of these structures. §5 defines the learning criteria, adapted from Chandlee et al., and §6 introduces a generalized learning algorithm that provably satisfies these criteria. The algorithm is applied to the example of Samala sibilant harmony in §7 and implications are discussed in §8.

2 Preliminaries

This section and the next follow closely from §2-3 of Chandlee et al. (2019), since the current work builds closely on their algorithm.

2.1 Formal Language Theory

Formal language theory allows us to study languages as mathematical objects which exist independently of the specific grammar (Heinz, 2016) The set of all possible finite strings generated from a finite alphabet Σ is denoted Σ^*, and the set of all strings of length k is given by Σ^k. In formal language theory, languages are defined as subsets of Σ^*. The length of a string w is denoted $|w|$.

2.2 Finite Model Theory

Finite model theory provides a unified vocabulary for representing many kinds of objects as relational structures, allowing for algorithms that are largely agnostic to the choice of linguistic representation (Enderton, 2001; Libkin, 2004; Chandlee et al., 2019; Lambert et al., 2021; Rawski, 2021). We consider finite relational models of strings in Σ^*.

Definition 1 (Models)**.** A **model signature** is a set of relations $R = \{R_1, R_2, ..., R_n\}$ where each R_i is an m_i-ary relation. An R-**structure** is a tuple of elements $S = \langle D; R_1, R_2, ..., R_n\rangle$, where D is a finite set of elements, the domain, and each R_i is a subset of D^{m_i}. The **size** $|S|$ of an R-structure S corresponds to the cardinality of its domain. A **model** for the set of objects Ω is a total, one-to-one function from Ω to R-structures.

Consider the **precedence** model for strings in Σ^*, defined as $M^{<}(w) := \langle D^w; <, [R^w_\sigma]_{\sigma\in\Sigma}\rangle$ where $D^w = \{1, ..., |w|\}$ is the domain of positions in w and $<:= \{(i, j) \in D^w \times D^w \mid i < j\}$ is the general precedence relation (Büchi, 1960; McNaughton and Papert, 1971; Rogers et al., 2013). With this model and $\Sigma = \{a, b, c\}$, we have:

$$\begin{aligned} M^{<}(abba) = \langle D = \{1, 2, 3, 4\}; \\ <= \{(1, 2), (1, 3), (1, 4), \\ (2, 3), (2, 4), (3, 4)\}, \\ R_a = \{1, 4\}, R_b = \{2, 3\}, R_c = \emptyset\rangle \end{aligned} \tag{1}$$

The **successor** model differs from the precedence model only in the ordering relation, given by $\lhd := \{(i, i + 1) \in D^w \times D^w\}$. The precedence and successor models of $abba$ are shown in Figure 1.

Since R-structures may be *any* mathematical structure conforming to a model signature, not all possible R-structures are valid models of strings: the R-structure in Figure 2 is not a model of any $w \in \Sigma^*$. To limit the R-structures we consider, we introduce the notion of **connectedness**.

Definition 2 (Connected R-Structure)**.** An R-structure $S = \langle D; R_1, R_2, ..., R_n\rangle$ is connected iff $(\forall x, y \in D)[(x, y) \in C^*]$, where C^* is defined as the symmetric transitive closure of:

$$\begin{aligned} C = \{(x, y) \in D \times D \mid \\ \exists i \in \{1...n\}, \exists (x_1...x_m) \in R_i \\ \exists s, t \in \{1...m\}, x = x_s, y = x_t\} \end{aligned} \tag{2}$$

Intuitively, domain elements x and y of S belong to C if they belong to some non-unary relation R_i in S. It is easy to see that $S_{ab,ba}$ (Figure 2) is not connected: neither (2, 3) nor (3, 2) is contained in

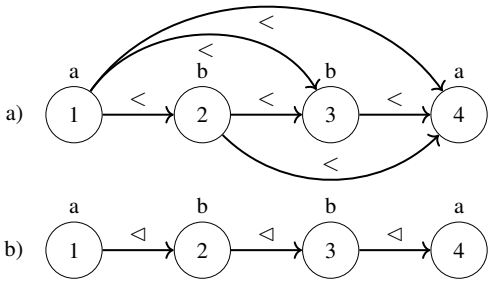

Figure 1: The precedence ($M^{<}$, subfigure a) and successor ($M^{\lhd}$, subfigure b) models of the string *abba*.

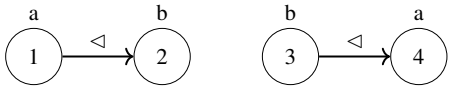

Figure 2: A visualization of the R-structure given by $S_{ab,ba} = \langle D = \{1, 2, 3, 4\}; \lhd = \{(1, 2), (3, 4)\}, R_a = \{1, 4\}, R_b = \{2, 3\}, R_c = \emptyset\rangle$.

C and thus none of (1,3), (1,4), (2,3), (2,4), etc. are contained in C^*. In contrast, both $M^{<}(abba)$ and $M^{\lhd}(abba)$ in Figure 1 are connected R-structures.

2.3 Unconventional String Models

The models shown in Figure 1 are **conventional string models**: besides the ordering relation, they include only mutually-exclusive unary relations (e.g., R_a) which label each domain element with a single property of being some $\sigma \in \Sigma$. In contrast, **unconventional string models** recognize that distinct alphabetic symbols may share properties and expand the model signature by including these properties as non-exclusive unary relations (Strother-Garcia et al., 2016; Vu et al., 2018).

Unconventional string models allow for more generalized representations, and thus have a number of useful linguistic applications. Consider the example of sibilant harmony in Samala: subsequences such as [s...s] that agree in ±ANTERIOR are allowed but subsequences such as [s...ʃ] which disagree are banned, so [hasxintilawas] is licit but [hasxintilawaʃ] is not (Hansson, 2010). Under a conventional string model, we must separately represent that [s...ʃ], [z...ʃ], [s...ʒ], etc. are banned, or equivalently that [s...s], [z...z], [ʃ...ʒ], etc. are allowed. Under an unconventional string model, however, we can simply represent that [+STR, +ANT][+STR, -ANT] subsequences are banned, or that [+STR, +ANT][+STR, +ANT] and [+STR, -ANT][+STR, -ANT] subsequences are allowed.

3 Subfactors and Maxfactors

We define a partial order over R-structures by establishing the notions of **restrictions, subfactors**, and **maxfactors**, building on Chandlee et al. (2019).

Definition 3 (Restriction)**.** An R-structure A is a restriction of an R-structure B if $D^A \subseteq D^B$ and for each m-ary relation R_i in the model signature, $R_i^A = \{(x_1, ..., x_m) \in R_i^B \mid x_1, ..., x_m \in D^A\}$.

A restriction is made by identifying a subset D^A of the domain of B and retaining only those relations in B whose elements are wholly within D^A. For example, Figure 3 shows the restriction of $M^{<}(abba)$ as defined in Equation 1 to $D' = \{1, 2, 3\}$. This restriction is given by: $M^{<}(abb) = \langle D' = \{1, 2, 3\}; <= \{(1, 2), (1, 3), (2, 3)\}, R_a = \{1\}, R_b = \{2, 3\}, R_c = \emptyset\rangle$.

Definition 4 (Subfactor)**.** A connected R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if there exists a restriction B' of B and a bijection h such that for all $R_i \in R$, if $R_i(x_1, ..., x_m)$ holds in A, then $R_i(h(x_1), ..., h(x_m))$ holds in B'. If $A \sqsubseteq B$, B is a **superfactor** of A.

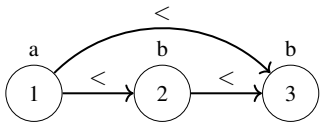

Figure 3: A restriction of $M^{<}(abba)$ shown in Fig. 1a.

Definition 5 (Maxfactor)**.** A connected R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) iff $A \sqsubseteq B$ and for each m-ary relation R_i, whenever $R_i(x_1, ..., x_m)$ holds in B, $R_i(h^{-1}(x_1), ..., h^{-1}(x_m))$ holds in A. Equivalently, $A \leq B$ if $A \sqsubseteq B$ and there is no R-structure A' non-isomorphic to A and B such that $|A| = |A'|$ and $A \sqsubseteq A' \sqsubseteq B$. [1]

Intuitively, A is a subfactor of B if there is a mapping between D^A and some subset of D^B and all relations that hold in A also hold over the corresponding elements in B. Note that this requirement is *unidirectional*. By contrast, maxfactors additionally require that all relations that hold in B also hold over the corresponding elements in A. We can thus think of maxfactors as the *maximally specified* subfactors of an R-structure. We use **factor** when the distinction between subfactor and maxfactor is irrelevant. This is true for conventional string models: since there is no underspecification in these models, any subfactor must also be a maxfactor.

If $A \sqsubseteq B$ and $|A| = k$, then A is a **k-subfactor** of B, and if $A \leq B$ and $|A| = k$, then A is **k-maxfactor** of B. Let the set of k-subfactors of an R-structure B be given by:

$$\text{SFAC}_k(B) \coloneqq \{A \mid A \sqsubseteq B, |A| = k\} \quad (3)$$

and the set of k-maxfactors of B be given by:

$$\text{MFAC}_k(B) \coloneqq \{A \mid A \leq B, |A| = k\} \quad (4)$$

For all $w \in \Sigma^*$ and any model M of Σ^*, the k-subfactors and k-maxfactors of w are given by $\text{SFAC}_k(M(w))$ and $\text{MFAC}_k(M(w))$, respectively; we also write $\text{SFAC}_k(M, w)$ and $\text{MFAC}_k(M, w)$ for readability. Finally, we define:

$$\text{SFAC}_k(M, \Sigma^*) = \bigcup_{w \in \Sigma^*} \text{SFAC}_k(M, w) \quad (5)$$

[1] In model-theoretic terms, Definition 5 simply means that A is a connected substructure of B (Libkin, 2004).

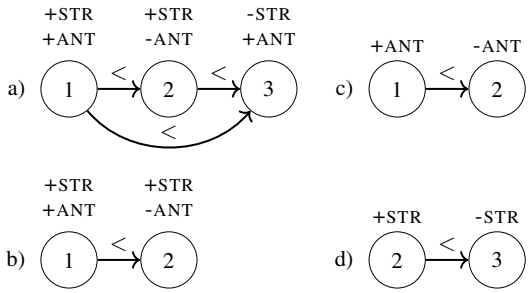

Figure 4: A visualization of $M^{<}(\int st)$ (a) and three 2-subfactors; only subfactor (b) is also a 2-maxfactor.

and likewise for $\textsc{Mfac}_k(M, \Sigma^*)$. From Definitions (4) and (5), we have:

$$\textsc{Mfac}_k(M, w) \subseteq \textsc{Sfac}_k(M, w) \tag{6}$$

Note that our definition of $\textsc{Sfac}_k()$ differs from the that of Chandlee et al. (2019) in that Chandlee et al. require the size of the subfactors to be *bounded* by k rather than *equal* to k. This difference is due to the constant size needed to define a positive grammar, discussed in §4.1. To differentiate between our definition of $\textsc{Sfac}_k()$ and that of Chandlee et al. (2019), we denote the latter as:

$$\textsc{Sfac}_{\leq k}(B) \coloneqq \{A \mid A \sqsubseteq B, |A| \leq k\} \tag{7}$$

Returning to Samala sibilant harmony (§2.3), consider the precedence model of [sʃt] given by:

$$\begin{aligned} M^{<}(\text{sʃt}) = \langle D = \{1,2,3\}; \\ <= \{(1,2),(1,3),(2,3)\}, \\ R_{\text{STR}} = \{1,2\}, R_{\text{ANT}} = \{1,3\}\rangle \end{aligned} \tag{8}$$

This model, along with three of its 2-subfactors, is shown in Figure 4. The R-structures (b), (c), and (d) are subfactors of $M^{<}(\text{sʃt})$, since they are all connected and the relations that hold within them also hold in $M^{<}(\text{sʃt})$. However, *only* (b) is a 2-maxfactor: it is the only subfactor for which all relations that hold in $M^{<}(\text{sʃt})$ also hold within it.

We now introduce two lemmas that will be used to define grammars and their languages in §4.

Lemma 1 (Maxfactor-Subfactor Containment)**.** *Let k be some positive integer and let M be some model of Σ^*. For any $w \in \Sigma^*$ and for any $F \in \textsc{Sfac}_k(M, w)$, we have that:*

$$[\exists G \in \textsc{Mfac}_k(M, w)](F \sqsubseteq G) \tag{9}$$

Proof. Let G be the restriction of $M(w)$ to $h(D^F)$, where $h : F \to M(w)$ is given by Definition 4. Clearly, $G \sqsubseteq M(w)$, and by Definition 3, $R_i^G = \{(x_1, ..., x_m) \in R_i^{M(w)} \mid x_1, ..., x_m \in h(D^F)\}$ for all $R_i \in R$. Thus, $G \sqsubseteq M(w)$ and whenever $R_i(x_1, ..., x_m)$ holds in $M(w)$, $R_i(x_1, ..., x_m)$ holds in G. By Definition 5, $G \leq M(w)$.

By Definition 4, for all $R_i \in R$, if $R_i(x_1, ..., x_m)$ holds in F, then $R_i(h(x_1), ..., h(x_m))$ holds in some restriction M' of $M(w)$, and thus in $M(w)$ by Definition 3. But since G is defined over $h(D^F)$ and contains all relations in $M(w)$ defined over D^G, it must also be the case that if $R_i(x_1, ..., x_m)$ holds in F, then $R_i(h(x_1), ..., h(x_m))$ holds in G. By Definition 4, this means that $F \sqsubseteq G$. □

Lemma 2 (Union of Subfactors of Maxfactors)**.** *Let k be some positive integer and let M be some model of Σ^*. For any $w \in \Sigma^*$, we have that:*

$$\bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S) = \textsc{Sfac}_k(M, w) \tag{10}$$

Proof. ($\subseteq$) Consider some $f \in \textsc{Sfac}_k(S)$, where $S \in \textsc{Mfac}_k(M, w) \subseteq \textsc{Sfac}_k(M, w)$ and thus $f \sqsubseteq S \in \textsc{Sfac}_k(M, w)$. By Equation 3, this means that $f \sqsubseteq S \sqsubseteq M(w)$, and thus that $f \sqsubseteq M(w)$. Since $f \sqsubseteq M(w)$ and $|f| = k$, by Equation 4, $f \in \textsc{Sfac}_k(M, w)$.

($\supseteq$) Consider some $g \in \textsc{Sfac}_k(M, w)$. By Lemma 1, $[\exists g' \in \textsc{Mfac}_k(M, w)](g \sqsubseteq g')$, and since $g \sqsubseteq g'$ and $|g| = k$, Equation 3 tells us that $g \in \textsc{Sfac}_k(g')$. Since $g \in \textsc{Sfac}_k(g')$ and $g' \in \textsc{Mfac}_k(M, w)$, it must be the case that $g \in \bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S)$. □

4 Grammars and Their Languages

4.1 Positive vs. Negative Interpretations

We define a grammar G as a finite set of subfactors; the language that G defines differs based on whether we interpret it as a positive or negative grammar. We first discuss these interpretations informally, then formalize them in Definitions 6-7.

Under a **negative** interpretation (notated G^-), the elements of G^- are forbidden, and strings in $L(G^-)$ contain no forbidden subfactors. This approach has parallels to logical expressions which are "conjunctions of negative literals" (Rogers et al., 2013; Chandlee et al., 2019): the forbidden subfactors are simply interpreted as the negative literals. Returning to Samala sibilant harmony (Figure 4), if (c) is in the grammar (i.e., [+ANT][-ANT] $\in G^-$), this is sufficient to determine that sʃt $\notin L(G^-)$, since [+ANT][-ANT] $\sqsubseteq M^{<}(\text{sʃt})$ (Figure 4a).

Under a **positive** interpretation (notated G^+), the elements of G^+ are permissible, and strings in $L(G^+)$ are those which are *covered* by these elements; we can think of the subfactors in G^+ as *tiling* the strings in $L(G^+)$ (Rogers and Heinz, 2014). Returning to Figure 4, if both (c) and (d) are in G^+ (i.e., [+STR][-STR] $\in G^+$, [+ANT][-ANT] $\in G^+$), then s∫t $\in L(G^+)$, since (c) covers [s∫] and (d) covers [∫t]. However, if (d) but not (c) is in G^+, then s∫t $\notin L(G^+)$, because there is *no subfactor* in G^+ that covers the first two indices of $M^<$(s∫t). The notion of tiling is greatly simplified when the subfactors used to tile are of equal size, as in Rogers and Heinz (2014). As such, our definitions of negative and positive grammars and their languages below operate over fixed values of k.

Definition 6 (Negative Grammar)**.** Let k be some positive integer and M be a model of Σ^*. A **negative grammar** G^- is a subset of $\textsc{Sfac}_k(M, \Sigma^*)$, and the language $L(G^-)$ of G^- is given by:

$$L(G^-) = \{w \in \Sigma^* \mid (\forall S \in \textsc{Mfac}_k(M, w)) \; [\textsc{Sfac}_k(S) \cap G^- = \emptyset]\} \tag{11}$$

or equivalently by:

$$L(G^-) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{Mfac}_k(M, w)) \; [\textsc{Sfac}_k(S) \cap G^- \neq \emptyset]\} \tag{12}$$

The class of such languages is defined as:

$$\mathscr{L}^-(M, k) = \{L \mid (\exists G^- \subseteq \textsc{Sfac}_k(M, \Sigma^*)) \; [L(G^-) = L]\} \tag{13}$$

Definition 7 (Positive Grammar)**.** Let k be some positive integer and M be a model of Σ^*. A **positive grammar** G^+ is a subset of $\textsc{Sfac}_k(M, \Sigma^*)$, and the language $L(G^+)$ of G^+ is given by:

$$L(G^+) = \{w \in \Sigma^* \mid (\forall S \in \textsc{Mfac}_k(M, w)) \; [\textsc{Sfac}_k(S) \cap G^+ \neq \emptyset]\} \tag{14}$$

or equivalently by:

$$L(G^+) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{Mfac}_k(M, w)) \; [\textsc{Sfac}_k(S) \cap G^+ = \emptyset]\} \tag{15}$$

The class of such languages is defined as:

$$\mathscr{L}^+(M, k) = \{L \mid (\exists G^+ \subseteq \textsc{Sfac}_k(M, \Sigma^*)) \; [L(G^+) = L]\} \tag{16}$$

Defining the languages of positive and negative grammars in terms of quantification over $\textsc{Mfac}_k(M, w)$ allows us to tile a word w with k-subfactors. We can think of Equations 11 through 15 as realizing two primary distinctions: **quantification** ($\forall$ vs. $\exists$) and **membership** in G. For universal quantification ($\forall$), if all k-maxfactors of w *are* superfactors of some k-subfactor in G^+, then $w \in L(G^+)$ (Equation 14), and if all k-maxfactors of w are *not* superfactors of some k-subfactor in G^-, then $w \in L(G^-)$ (Equation 11). For existential quantification ($\exists$), if there exists a k-maxfactor of w that *is* a superfactor of some k-subfactor in G^-, then $w \notin L(G^-)$ (Equation 12). If there exists a k-maxfactor of w that is *not* a superfactor of some k-subfactor in G^+, then $w \notin L(G^+)$ (Equation 15). Figure 5 illustrates how these distinctions define the languages of positive and negative grammars.

	$\in G$	$\notin G$
$\forall$	**Positive Grammar** (Equation 14)	**Negative Grammar** (Equation 11)
$\exists$	**Negative Grammar** (Equation 12)	**Positive Grammar** (Equation 15)

Figure 5: Positive and negative grammars and their languages, organized by quantification and attestation.

To further illustrate the differences between negative and positive grammars, consider a grammar G such that $L(G) = \Sigma^*$, and let $k = 1$. If G is positive, then it must contain subfactors of *all possible* 1-maxfactors of any $w \in \Sigma^*$. The empty 1-subfactor [ ] satisfies this, so we have $G^+ = \{[\,]\}$ and $L(G^+) = \Sigma^*$. Conversely, if we have $G^- = \{[\,]\}$, we will have $L(G^-) = \emptyset$, since there is no $w \in \Sigma^*$ whose 1-maxfactors are *not* superfactors of [ ]. To define a negative grammar accepting Σ^*, we must instead ensure that *no* possible 1-maxfactor of any $w \in \Sigma^*$ is a superfactor of an element in G^-; this is easily achieved with $G^- = \emptyset$. At the same time, if we have $G^+ = \emptyset$, then no word $w \in \Sigma^*$ will have its 1-maxfactors contained by elements in G^+, and thus $L(G^+) = \emptyset$.

4.2 Equivalence to Chandlee et al. (2019)

In contrast to the current work, Chandlee et al. (2019) focus only on the learning of negative grammars, defined as follows:

Definition 8 (Chandlee et al. Negative Grammar)**.** Let k be some positive integer and M be a model of Σ^*. A negative grammar G^- is a subset of $\textsc{Sfac}_{\leq k}(M, \Sigma^*)$, and the language $L(G^-)$ is given by:

$$L(G^-) = \{w \in \Sigma^* \mid \textsc{Sfac}_{\leq k}(M, w) \cap G^- = \emptyset\} \tag{17}$$

Since Chandlee et al. consider negative grammars, they need only to set an *upper bound* on k. To learn both positive and negative grammars, however, we fix k. We thus wish to demonstrate an equivalence between the grammars and languages in Definition 8 and in Definition 6, namely:

Theorem 1. *Let L_{17} refer to $L(G^-)$ as in Equation (17) and L_{11} refer to $L(G^-)$ as in Equation (11). For any $G_1^- \subseteq \text{SFAC}_{\leq k}(M, \Sigma^*)$, $\exists G_2^- \subseteq \text{SFAC}_k(M, \Sigma^*)$ such that $L_{17}(G_1^-) = L_{11}(G_2^-)$.*

A full proof is provided in Appendix A.

4.3 The Cost of Interdefinability

When defined over conventional string models, negative and positive grammars are straightforwardly interdefinable: $G^+ = \Sigma^k \setminus G^-$ and $G^- = \Sigma^k \setminus G^+$ (Heinz, 2010b). However, there are two complications for unconventional string models that make the interdefinability significantly more costly.

Firstly, the number of potential k-subfactors is significantly larger for unconventional string models: consider a model with n binary features, defining $s \leq 2^n$ segments. Under a conventional string model, the number of k-factors is no more than $(s)^k \leq (2^n)^k$, since one segment is chosen at each position (Heinz, 2010b). Under an unconventional string model, however, a feature can be either positive, negative, *or underspecified* at each position, yielding $(3^n)^k$ possible k-subfactors, exponentially more than for the conventional string model.

Secondly, the conversion itself is less straightforward for unconventional string models. To illustrate this, we again consider Samala sibilant harmony (§2.3). For conventional string models, we must simply check whether some k-factor $f \in \Sigma^k$ is in G^- to determine if it is added to G^+: if [s...ʃ] $\in G^-$, for example, then [s...ʃ] $\notin G^+$. For unconventional string models, however, this is not sufficient. Returning to Figure 4, if subfactor (c) is in G^-, then we should not include (b) in the corresponding G^+, even though $b \notin G^-$, since $c \sqsubseteq b$. Likewise, if $b \in G^-$, then $c \notin G^+$. Thus, to determine whether some k-subfactor f should be added to G^+ for an unconventional string model, we must check not only if $f \in G^-$, but also if $(\exists g \in G^-)[f \sqsubseteq g \vee g \sqsubseteq f]$. Hence, both the number of possible k-subfactors and the method of conversion indicate that interdefinability is prohibitively costly for unconventional string models, further motivating the learning of positive grammars directly from these models.

5 The Learning Problem

Heinz (2010b) and Heinz et al. (2012) demonstrate that positive grammars like those in Definition 7 are learnable in the limit from positive evidence in the sense of Gold (1967), as well as PAC-learnable (Valiant, 2013) in some cases. In this work, G^+ is defined as the collection of all k-factors in the data sample, and a word w is in $L(G^+)$ if and only if all of its k-factors are in G^+. Because Heinz (2010b) and Heinz et al. (2012) are not working in the model-theoretic framework, it is sufficient to check that the k-factors are in the grammar, rather than superfactors of elements in the grammar.

Chandlee et al. (2019) learn negative grammars from positive evidence with unconventional string models. Rather than convergence to a *correct* grammar in the limit, Chandlee et al. define the learning problem in terms of returning an *adequate* grammar given a finite positive sample, in the sense of De Raedt (2008). We adapt Chandlee et al.'s definition of the learning problem to apply to both negative and positive grammars as follows:

Definition 9 (The Learning Problem)**.** Fix Σ, model M, positive integer k, and polarity p. For any language $L \in \mathscr{L}^p(M, k)$ and for any finite sample $D \subseteq L$, return a grammar G^p such that:

1. G^p is **consistent**, that is, $D \subseteq L(G^p)$.
2. $L(G^p)$ is a **smallest language** in $\mathscr{L}^p(M, k)$ which covers D, so that for all $L \in \mathscr{L}^p(M, k)$ where $D \subseteq L$, we have $L(G^p) \subseteq L$.
3. G^p includes R-structures S that are restrictions of R-structures S' in other grammars G' that also satisfy (1) and (2). That is, for all G' satisfying (1) and (2) and for all $S' \in G'$, there exists some $S \in G^p$ such that $S \sqsubseteq S'$.

The first criterion is self-explanatory: we want to at least cover the training data. Following Chandlee et al., the second criterion is motivated by Angluin (1980)'s analysis of identification in the limit. The third criterion requires us to learn the most *general* subfactors: for positive grammars, this means the subfactors that most generally encompass the allowed maxfactors, while for negative grammars, it means the most general constraints. In the case of Samala sibilant harmony (§2.3), for example, if we see [ʃ...ʃ] and [ʒ...ʒ], but not [s...ʃ], [z...ʒ], [s...ʒ], or [z...ʃ], then we would add to our positive grammar [-ANT, +STR][-ANT, +STR], rather than the

Algorithm 1 A generalized bottom-up learning algorithm for positive and negative grammars

Require: positive sample D, size k, and polarity p
1: $Q \leftarrow \{s_0\}$
2: $G \leftarrow \emptyset$
3: $V \leftarrow \emptyset$
4: **while** $|Q| > 0$ **do**
5: $\quad s \leftarrow Q$.dequeue()
6: $\quad V \leftarrow V \cup \{s\}$
7: $\quad$ **if** $((p = -) \wedge (\exists s' \in \text{EXT}_k(s), \exists x \in D)[s' \leq x]) \vee$
8: $(p = +) \wedge (\exists s' \in \text{EXT}_k(s))[\forall x \in D : s' \not\leq x]$ **then**
9: $\quad\quad S \leftarrow$ NextSupFact(s)
10: $\quad\quad S' \leftarrow \{s \in S \mid (\nexists g \in G)[g \sqsubseteq s] \wedge s \notin V\}$
11: $\quad\quad Q$.enqueue(S')
12: $\quad$ **else**
13: $\quad\quad G \leftarrow G \cup \{s\}$
14: $\quad$ **end if**
15: **end while**
16: **return** G

two more specified factors. From the same data, we would add to our negative grammar [+ANT, +STR][-ANT, +STR], rather than the four more specified factors. The third criterion is specific to unconventional string models: as discussed in §3, maxfactors and subfactors are equivalent for conventional string models, so there is no parallel notion of generality when learning over them.

6 A Generalized Learning Algorithm

The learning algorithm we present is based closely on that of Chandlee et al. (2019). Since our goal is to add the most *general* subfactors to our grammar, the algorithm is **bottom-up** in the sense of De Raedt (2008): we begin with the most general subfactors and traverse upwards in the partial order during learning. Indeed, once a subfactor has been identified as an element of the grammar, none of its superfactors need to be considered: all of them will be banned in the case of a negative grammar or allowed in the case of a positive grammar.

A bottom-up learner that learns both positive and negative grammars is given in Algorithm 1. As input, this algorithm takes a positive data sample D, an integer k corresponding to the size of the subfactors, and a polarity p indicating whether to learn a positive or negative grammar. As in Chandlee et al. (2019), Algorithm 1 makes use of a queue Q, which initially contains just the empty structure of length k, s_0 (Line 1). The algorithm also initializes two empty sets: G (Line 2), the grammar to be returned, and V (Line 3), the set of subfactors that have already been visited. Algorithm 1 considers the subfactors in Q one at a time in first-in-first-out order, and as each subfactor s is considered, it is added to the set of visited subfactors V (Line 6). Depending on the polarity p of the grammar to be learned, we condition as follows for a given s (Line 7): For a negative grammar, we check whether *any* of the possible extensions of s is a k-maxfactor of some element in D. For a positive grammar, we check whether *any* of the possible extensions of s is *not* a k-maxfactor of any element in D. The extensions of s are defined as follows:

$$\begin{aligned}\text{EXT}_k(s) = &\{A \in \text{SFAC}_k(M, \Sigma^*) \mid \\ &s \sqsubseteq A \wedge (\nexists A')[|A'| = k \wedge A \sqsubseteq A']\}\end{aligned} \tag{18}$$

In other words, the extensions of s are all k-maxfactors that are superfactors of s. For example, if we have s = [+ANT] and the only two features available are $\pm$ANT and $\pm$STR, then we have $\text{EXT}_k(s) = \{$[+ANT, -STR], [+ANT, +STR]$\}$.

Given Definitions 6 and 7, if the conditions on Line 7 are not satisfied, we add s to G (Line 13). If either of the conditions are satisfied, however, we must consider more specified subfactors than s. For a negative grammar, this is because the potential constraint s is violated by some $w \in D$ and thus cannot be added to G. For a positive grammar, this is because at least one k-maxfactor licensed by s is unattested in D, and thus s cannot be added to G. We extract the more specific superfactors of s by calling NextSupFact(s) (Line 9) where NextSupFact() is defined as follows:

$$\begin{aligned}\text{NextSupFact}(s) = &\{A \in \text{SFAC}_k(M, \Sigma^*) \mid \\ &s \sqsubseteq A \wedge (\nexists A')[s \sqsubseteq A' \sqsubseteq A]\}\end{aligned} \tag{19}$$

Intuitively, NextSupFact() returns the *least* superfactors for s. The set S of superfactors is then filtered (Line 10) to contain only those that have not been previously visited and contain no element of G as a subfactor. This is because if there is some $g \in G$ such that $g \sqsubseteq s$, then for any word w for which $s \sqsubseteq w$, we have $g \sqsubseteq s \sqsubseteq w$ and thus $g \sqsubseteq w$, and by Definitions 6 and 7, s will not add any new information to the grammar. The structures that pass this filter are then added to Q.

Note that Algorithm 1 is nearly identical to the algorithm of Chandlee et al. except that Line 7 conditions for both positive and negative grammars, and we consider subfactors of *exactly* size k rather than bounded in size by k. As discussed in §4.1, it is the latter modification that allows us to learn both positive and negative grammars in the same way. We now demonstrate that Algorithm 1 meets the criteria outlined in Definition 9.

Theorem 2. *For any $p \in \{+,-\}$, positive integer k, any $L \in \mathscr{L}^p(M,k)$ and any finite set $D \subseteq L$ provided to Algorithm 1, it returns a grammar G^p satisfying Definition 9.*

Proof. **(Condition 1)** Assume towards contradiction that there exists some $w \in D, w \notin L(G^p)$.

If $p = +$, then by Definition 7, there is some $x \in \textsc{Mfac}_k(M,w)$ such that $(\forall y \in \textsc{Sfac}_k(x))[y \notin G^p]$. By Algorithm 1, this means that for all $y \in \textsc{Sfac}_k(x)$ there is some $z \in \textsc{Ext}_k(y)$ such that $(\forall w' \in D)[z \not\leq w']$. However, $x \in \textsc{Sfac}_k(x)$ by Definition 4, and since $x \in \textsc{Mfac}_k(M,w)$, we have $\textsc{Ext}_k(x) = \{x\}$. Thus, if $x \notin G^+$, then $(\forall w' \in D)[x \not\leq w']$, so $w \notin D$, a contradiction.

If $p = -$, then by Definition 6, there is some $x \in \textsc{Mfac}_k(M,w)$ such that $(\exists y \in \textsc{Sfac}_k(x))[y \in G^p]$. Since $y \in \textsc{Sfac}_k(x)$ and $x \in \textsc{Mfac}_k(M,w)$, we have $x \in \textsc{Ext}_k(y)$. By Algorithm 1, $y \in G^-$ means that $(\forall y' \in \textsc{Ext}_k(y), \forall w' \in D)[y' \not\leq w']$, but since $x \in \textsc{Ext}_k(y)$, this means that $(\forall w' \in D)[x \not\leq w']$, and $w \notin D$, a contradiction.

(Condition 2) Consider any $L' \in \mathscr{L}^p(M,k)$ with $D \subseteq L'$ and any $w \in L(G^p)$. Since $w \in L(G^p)$, we have $\textsc{Sfac}_k(M,w) \subseteq \textsc{Sfac}_k(M,D)$ and since $D \subseteq L'$, we have $\textsc{Sfac}_k(M,D) \subseteq \textsc{Sfac}_k(M,L')$. Thus, $\textsc{Sfac}_k(M,w) \subseteq \textsc{Sfac}_k(M,L')$, and $w \in L'$. As such, $(\forall w \in L(G^p))[w \in L']$, and $L(G^p) \subseteq L'$.

(Condition 3) Assume towards contradiction that there is some G^p learned by Algorithm 1 such that for some $s \in G^p$, $\exists s' \sqsubseteq s$ that should be included in G^p: either $p = +$ and $(\forall x \in \textsc{Ext}_k(s'))[\exists w \in D, x \leq w]$, or $p = -$ and $(\forall x \in \textsc{Ext}_k(s'))[\nexists w \in D, x \leq w]$. Since $s' \sqsubseteq s$, s' will be added to Q before s' is generated by `NextSupFact()` under Algorithm 1, and since Q is a first-in-first-out queue, s' will be removed from Q for consideration before s is generated. Since we have either $p = +$ and $(\forall x \in \textsc{Ext}_k(s'))[\exists w \in D, x \leq w]$, or $p = -$ and $(\forall x \in \textsc{Ext}_k(s'))[\nexists w \in D, x \leq w]$, s' will be added to G by Line 7. Then, when s is generated by `NextSupFact()`, it will not pass the filter in Line 10, since $s' \sqsubseteq s$ and $s' \in G^p$. As such, s is never added to G^p, and $s \notin G^p$, a contradiction. □

7 Example: Samala Sibilant Harmony

We illustrate our learning algorithm by applying it to a toy example based on Samala sibilant harmony (§2.3; Hansson, 2010). For simplicity, we use only two features: ±ANT and ±VOI; the former is necessary to define the phonotactic restriction and the latter is not. We define the size of the subsequences to be $k = 2$, and assume that all licit subsequences are attested in D (c.f. Heinz, 2010a). The partially-ordered structure of the hypothesis space is shown in Figure 6, with lines indicating subfactor-superfactor relations (see Chandlee et al. 2019 for further discussion).

Following Line 1, we initialize Q to contain only the empty 2-subfactor []⋊[], shown at the bottom of Figure 6. Learning begins by dequeuing and considering []⋊[] (Lines 5-6). If we are learning a negative grammar, we check whether there is any element in $\textsc{Ext}_k(\square\square)$ which *is* a 2-maxfactor of some $x \in D$. By definition, $\textsc{Ext}_k(\square\square)$ will contain all fully-specified 2-factors that are superfactors of []⋊[]. This means, for example, that [+VOI, +ANT][+VOI, +ANT] $\in \textsc{Ext}_k(\square\square)$, and this corresponds to the licit subsequence [z...z] which is attested in D. If we are learning a positive grammar, we check whether any element in $\textsc{Ext}_k(\square\square)$ is *not* a 2-maxfactor of any $x \in D$. We have, for example, [+VOI, +ANT][+VOI, -ANT] $\in \textsc{Ext}_k(\square\square)$, and this corresponds to the illicit subsequence [z...ʒ] which is not attested in D. As such, the condition on Line 7 is satisfied for either polarity.

Following Line 9, we then extract the *least* superfactors of []⋊[]; these are shown in the level above []⋊[] in Figure 6. Since none of these subfactors have been seen and G is empty, they are all added to Q (Lines 10-11). However, as each is dequeued and considered, it still satisfies the criteria in Line 7: any subfactor with only ±ANT specified in a single position will have both licit and illicit maxfactors in its extension (e.g., [+ANT]⋊[]⋉ [+ANT][+ANT] means that Line 7 will be satisfied for negative grammars, but [+ANT]⋊[]⋉ [+ANT][-ANT] means that it will also be satisfied for positive grammars). Similarly, any subfactor with only ±VOI specified will have both licit and illicit maxfactors in its extension. As such, specification of the subfactors under consideration will be increased once more, corresponding to the third level in Figure 6.

It is here that we are able to add subfactors to G. When [+ANT]⋊[+ANT] is dequeued for the positive grammar, every factor in $\textsc{Ext}_k$([+ANT]⋊[+ANT]) (namely [s...s], [z...z], [s...z], and [z...s]) is attested, so Line 7 is not satisfied, and [+ANT]⋊[+ANT] is added to G (Line 13). Similarly, when [+ANT]⋊[-ANT] is dequeued for the negative grammar, no factor in $\textsc{Ext}_k$([+ANT]⋊[-ANT]) (i.e., none of [s...ʃ],

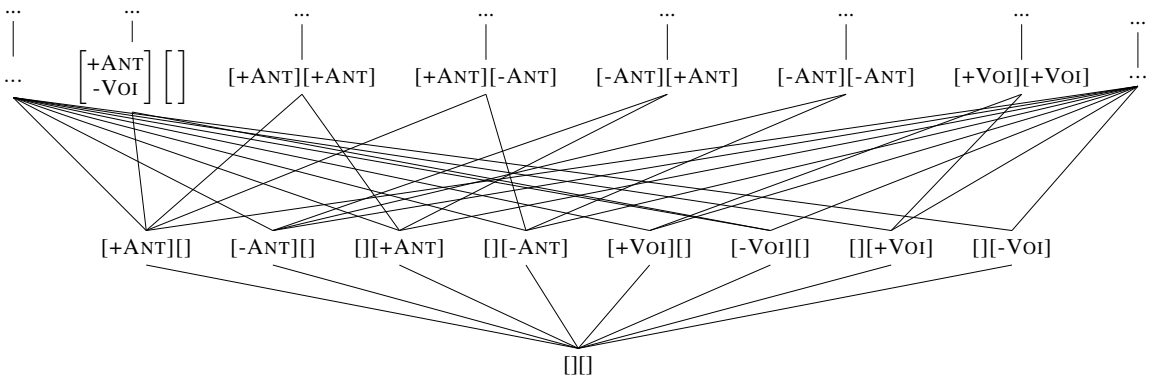

Figure 6: A partial illustration of the hypothesis space for learning Samala sibilant harmony with $k = 2$.

[s...ʒ], [z...ʃ] or [s...ʒ]) is attested, so [+ANT][-ANT] is added to G (Line 13). We may later dequeue [+ANT][+ANT, +VOI] in the positive case, but will not consider it (Line 10): [+ANT][+ANT] being allowed entails [+ANT][+ANT, +VOI] being allowed, so there is no reason to consider [+ANT][+ANT, +VOI] separately. Similarly, [+ANT][-ANT, +VOI] will not be considered in the negative case since [+ANT][-ANT] being banned entails [+ANT][-ANT, +VOI] being banned.

8 Discussion

The polarity of the grammar has several implications that warrant future exploration. In implementation, it is often necessary to terminate the search defined in Algorithm 1 before reaching the most specific k-maxfactors, but the implications of this termination differ based on the polarity of the grammar. Consider some positive data sample D, and let G^p be the grammar that will be learned by Algorithm 1 from D if the search space is traversed in its entirety. Let $G^p(t)$ be the intermediate grammar at some time t. It is easy to see that $G^p(t) \subseteq G^p$, since at any time t, elements in Q — as well as their superfactors — have not yet been considered.

However, the implications of $G^p(t) \subseteq G^p$ differ depending on the value of p. Specifically, $L(G^+(t)) \subseteq L(G^+)$ but $L(G^-) \subseteq L(G^-(t))$, since the additional elements in $G^p \setminus G^p(t)$ will either be interpreted as additional constraints (for negative p) or additional permitted elements (for positive p). Recall from §4.1 that $L^+(\emptyset) = \emptyset$ and $L^-(\emptyset) = \Sigma^*$, and from Algorithm 1 that G^p is initialized to $\emptyset$. This, in conjunction with the subset relations above, entails that Algorithm 1 consistently expands $L(G^+)$ during learning of a positive grammar by adding more allowed subfactors to G^+, while it consistently shrinks $L(G^-)$ during learning of a negative grammar by adding more banned subfactors to G^-. Future work should investigate how these differing predictions map onto developmental findings. While some findings have suggested an initial stage of conservatism in child productions (Fikkert, 1994; Levelt et al., 2000; Rose, 2000, i.a.), there is also evidence for early generalization in perception (Cristia and Peperkamp, 2012; Hallé and Cristia, 2012; Bernard and Onishi, 2023, i.a.), particularly based on phonological features and syllable position. Do children begin by positing that *anything* is allowed and later backtrack, or do they begin by positing that *nothing* is allowed, and only add items to their grammar once they have been observed in the input?

9 Conclusion

In this paper, we showed that if we fix the size k of subfactors in the grammar, then the algorithm of Chandlee et al. (2019) can be straightforwardly extended to learn both positive and negative grammars over unconventional string models in a unified way. The enriched representations provided by unconventional string models allow us to provably find the most general subfactors that are allowed or banned in a given language by conducting a bottom-up search of the partial ordering of k-subfactors.

Acknowledgements

I am grateful to Jeff Heinz, Thomas Graf, Jon Rawski, Logan Swanson, and the SCiL reviewers for their feedback. This work was supported by the Institute for Advanced Computational Science Graduate Research Fellowship and the National Science Foundation Graduate Research Fellowship Program under NSF Grant No. 2234683.

References

Dana Angluin. 1980. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135.

Caleb Belth. 2023. *Towards an Algorithmic Account of Phonological Rules and Representations*. Ph.D. thesis, University of Michigan.

Caleb Belth, Sarah Payne, Deniz Beser, Jordan Kodner, and Charles Yang. 2021. The greedy and recursive search for morphological productivity. *Proceedings of the 43rd annual meeting of the Cognitive Science Society*, 43:2869–2875.

Amélie Bernard and Kristine H Onishi. 2023. Novel phonotactic learning by children and infants: Generalizing syllable-position but not co-occurrence regularities. *Journal of Experimental Child Psychology*, 225:105493.

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

Jane Chandlee, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski. 2019. Learning with partially ordered representations. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 91–101, Toronto, Canada. Association for Computational Linguistics.

Alejandrina Cristia and Sharon Peperkamp. 2012. Generalizing without encoding specifics: Infants infer phonotactic patterns on sound classes. In *Proceedings of the 36th Annual Boston University Conference on Language Development (BUCLD 36)*, pages 126–138.

Luc De Raedt. 2008. *Logical and relational learning*. Springer Science & Business Media.

Herbert B Enderton. 2001. *A mathematical introduction to logic*. Elsevier.

Paula Fikkert. 1994. *On the acquisition of prosodic structure*. ICG Printing.

E Mark Gold. 1967. Language identification in the limit. *Information and Control*, 10(5):447–474.

Pierre Hallé and Alejandrina Cristia. 2012. Global and detailed speech representations in early language acquisition. In *Speech planning and dynamics*, pages 11–38. Peter Lang.

Gunnar Ólafur Hansson. 2010. *Consonant harmony: Long-distance interactions in phonology*, volume 145. University of California Press.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, 39(3):379–440.

Jeffrey Heinz. 2010a. Learning long-distance phonotactics. *Linguistic Inquiry*, 41(4):623–661.

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

Jeffrey Heinz. 2016. Computational theories of learning and developmental psycholinguistics. In Jeffrey Lidz, William Synder, and Joe Pater, editors, *The Oxford Handbook of Developmental Linguistics*, pages 633–663. Oxford University Press, Oxford, UK.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. *Phonological typology, phonetics and phonology*, pages 126–195.

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

Dakotah Lambert, Jonathan Rawski, and Jeffrey Heinz. 2021. Typology emerges from simplicity in representations and learning. *Journal of Language Modelling*, 9.

Clara C Levelt, Niels O Schiller, and Willem J Levelt. 2000. The acquisition of syllable types. *Language acquisition*, 8(3):237–264.

Daoxin Li and Kathryn D Schuler. 2023. Acquiring recursive structures through distributional learning. *Language Acquisition*, pages 1–14.

Leonid Libkin. 2004. *Elements of finite model theory*, volume 41. Springer.

Gary F Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T John Rosen, Fei Xu, and Harald Clahsen. 1992. Overregularization in language acquisition. *Monographs of the society for research in child development*, pages i–178.

Robert McNaughton and Seymour A Papert. 1971. *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press.

Sarah Payne. 2023. Marginal sequences are licit but unproductive. Poster presented at the 2023 Annual Meeting of Phonology.

Alan Prince and Paul Smolensky. 1993. *Optimality Theory: Constraint interaction in generative grammar*. John Wiley & Sons.

Jonathan Rawski. 2021. *Structure and Learning in Natural Language*. Ph.D. thesis, State University of New York at Stony Brook.

James Rogers and Jeffrey Heinz. 2014. Model theoretic phonology. In *Workshop slides in the 26th European Summer School in Logic, Language and Information*.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The Mathematics of Language: 10th and*

11th Biennial Conference, Revised Selected Papers, pages 255–265. Springer.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar: 17th and 18th International Conferences, Revised Selected Papers*, pages 90–108. Springer.

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

Yvan Rose. 2000. *Headedness and prosodic licensing in the L1 acquisition of phonology*. Ph.D. thesis, McGill University.

Kristina Strother-Garcia, Jeffrey Heinz, and Hyun Jin Hwangbo. 2016. Using model theory for grammatical inference: a case study from phonology. In *Proceedings of The 13th International Conference on Grammatical Inference*, pages 66–78.

Leslie Valiant. 2013. *Probably approximately correct: nature's algorithms for learning and prospering in a complex world*. Basic Books.

Mai H Vu, Ashkan Zehfroosh, Kristina Strother-Garcia, Michael Sebok, Jeffrey Heinz, and Herbert G Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics and AI*, 5:76.

Charles Yang. 2016. *The price of linguistic productivity: How children learn to break the rules of language*. MIT press.

A Proof of Theorem 1

Proof. To construct G_2^- from G_1^-, for each $g \in G_1^-$, if $|g| = k$, then add g directly to G_2^-. If $|g| < k$, then add to G_2^- all superfactors of g of length k given by $\{f \mid g \sqsubseteq f, |f| = k\}$. Assume towards contradiction that $L_{17}(G_1^-) \neq L_{11}(G_2^-)$. This means that either $(\exists w)[w \in L_{17}(G_1^-), w \notin L_{11}(G_2^-)]$ or $(\exists w)[w \in L_{11}(G_2^-), w \notin L_{17}(G_1^-)]$.

(Case 1) $(\exists w)[w \in L_{17}(G_1^-), w \notin L_{11}(G_2^-)]$**:** By Equation (11), $w \notin L_{11}(G_2^-)$ means that:

$$(\exists S \in \textsc{Mfac}_k(M, w))[\textsc{Sfac}_k(S) \cap G_2^- \neq \emptyset]$$

or equivalently that:

$$\left(\exists f \in \bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S)\right)[f \in G_2^-]$$

By Lemma 2 this means that:

$$[\exists f \in \textsc{Sfac}_k(M, w)](f \in G_2^-)$$

Given our construction of G_2^-, it is either the case that $f \in G_1^-$ or that $[\exists g \in G_1^-](g \sqsubseteq f)$. In the first case, $f \in \textsc{Sfac}_k(M, w) \subseteq \textsc{Sfac}_{\leq k}(M, w) \Rightarrow f \in \textsc{Sfac}_{\leq k}(M, w)$, but if $f \in \textsc{Sfac}_{\leq k}(M, w)$ and $f \in G_1^-$, then $\textsc{Sfac}_{\leq k}(M, w) \cap G_1^- \neq \emptyset$ and $w \notin L_{17}(G_1^-)$, a contradiction. In the second case, $g \sqsubseteq f \in \textsc{Sfac}_k(M, w) \Rightarrow g \in \textsc{Sfac}_{\leq k}(M, w)$, but if $g \in \textsc{Sfac}_{\leq k}(M, w)$ and $g \in G_1^-$, then $\textsc{Sfac}_{\leq k}(M, w) \cap G_1^- \neq \emptyset$ and $w \notin L_{17}(G_1^-)$, a contradiction.

(Case 2) $(\exists w)[w \in L_{11}(G_2^-), w \notin L_{17}(G_1^-)]$**:** By Equation 17, $w \notin L_{17}(G_1^-)$ means that

$$[\exists g \in \textsc{Sfac}_{\leq k}(M, w)](g \in G_1^-)$$

Given our construction of G_2^-, it is either the case that $|g| = k$ and $g \in G_2^-$ or that $[\forall f \mid g \sqsubseteq f, |f| = k](f \in G_2^-)$. In the first case, since $|g| = k$, $g \in \textsc{Sfac}_k(M, w)$ and thus:

$$g \in \bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S)$$

by Lemma 2. But this, in conjunction with $g \in G_2^-$, means that:

$$\left(\bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S)\right) \cap G_2^- \neq \emptyset$$

and $w \notin L_{11}(G_2^-)$, a contradiction. In the second case, $g \in \textsc{Sfac}_{\leq k}(M, w) \Rightarrow [\exists g' \in \textsc{Sfac}_k(M, w)](g \sqsubseteq g')$, but since $\{f \mid g \sqsubseteq f, |f| = k\} \subseteq G_2^-$, it must be the case that $g' \in G_2^-$. Since $g' \in \textsc{Sfac}_k(M, w)$ and $g' \in G_2^-$, by Lemma 2:

$$\left(\bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S)\right) \cap G_2^- \neq \emptyset$$

and $w \notin L_{11}(G_2^-)$, a contradiction. □