Typology emerges from simplicity in representations and learning

Dakotah Lambert[1]*, Jonathan Rawski*[2]*, and Jeffrey Heinz*[1]
[1] Stony Brook University
[2] San José State University

ABSTRACT

We derive well-understood and well-studied subregular classes of formal languages purely from the computational perspective of algorithmic learning problems. We parameterise the learning problem along dimensions of representation and inference strategy. Of special interest are those classes of languages whose learning algorithms are necessarily not prohibitively expensive in space and time, since learners are often exposed to adverse conditions and sparse data. Learned natural language patterns are expected to be most like the patterns in these classes, an expectation supported by previous typological and linguistic research in phonology. A second result is that the learning algorithms presented here are completely agnostic to choice of linguistic representation. In the case of the subregular classes, the results fall out from traditional model-theoretic treatments of words and strings. The same learning algorithms, however, can be applied to model-theoretic treatments of other linguistic representations such as syntactic trees or autosegmental graphs, which opens a useful direction for future research.

Keywords: model theory, subregularity, grammatical inference, formal language theory, phonology, learning complexity

1 INTRODUCTION

This paper presents an analysis supporting the view that the computational simplicity of learning mechanisms has considerable impact on the types of patterns found in natural languages.

First, we derive well-understood and well-studied subregular classes of formal languages purely from the computational perspective of algorithmic learning problems. We present a family of four learning algorithms, generalizing the String Extension learners in Heinz (2010b). We show that these algorithms, over different data structures, naturally structure the subregular Hierarchy of language classes purely by difficulty of learning. We show that the simplest classes of languages in these hierarchies are precisely the ones whose learning algorithms use the least computational resources, in particular space complexity. In fact, these are the only ones that are not prohibitively expensive to learn. A reasonable prediction is that learned natural language patterns would be most similar to patterns in the simplest of these classes, and this expectation is supported by previous typological and linguistic research in the domain of phonology.

The second result is that we introduce *linear-time* learning algorithms for some subregular classes, a further restriction of the typology beyond space-efficiency. As we explain, these algorithms are helpful in certain cases and not so helpful in others, depending on the extent to which the target patterns interact with other constraints. At issue is that a set of data points which may be helpful in identifying one constraint do not occur because they also happen to violate another. A virtue of this analysis is that we can identify precisely the situations where the linear-time learning algorithms can be applied.

Our third result is that the learning algorithms presented here are completely agnostic to choice of linguistic representation. These learning algorithms essentially parameterise the learning problem in two ways: the structural knowledge salient to the learner (the representation), and the way the learner collects and combines this structural information to derive sets of acceptable and unacceptable linguistic structures. In the case of the subregular classes of formal languages, the results emerge from traditional model-theoretic treatments of words and strings on the representational side and how the combinatorics of the grammars relate to kinds of logical languages on the other side.

Since the algorithms are agnostic to the representations, the same learning algorithms can be applied to model-theoretic treatments of other linguistic representations such as syntactic tree structures or autosegmental graphs. Of course, the real-life learning problem is complicated by the fact that language learners do not have direct access to linguistic structures like trees. Nonetheless the generality of these learning algorithms means the real-life learning problems may be reduced to these algorithms coupled with appropriate parsing mechanisms.

Priors in language learning 1.1

Language acquisition succeeds despite sparse, underdetermined, Zipf-distributed input, compounded by a lack of invariance in the signal – the so-called poverty of stimulus (Yang 2013). This holds across all domains of language, from phonological to syntactic induction.

It is uncontroversial that *some* bias or innate component restricts a learner's hypothesis space regardless of its strategy to solve this induction problem, often referred to as Universal Grammar (Nowak *et al.* 2002). The question is its nature. How is it rich, and how is it poor?

Data-driven statistical learning does not change this basic calculus. One reason is that children often learn language in ways that defy adult distributions (Legate and Yang 2002). Another is that induction from a data distribution without a prior may only recapitulate the training data (Fodor and Pylyshyn 1988; Mitchell 1982, 2017), and cannot generalize. Without a lens in which linguistic experience is viewed, even the input distribution cannot be recovered, simply because distributions are based on the structure of their parameters (Lappin and Shieber 2007). Consequently, the nontrivial open question central to learnability research in linguistics instead concerns the characteristics of this additional prior knowledge or bias such that learners *generalize* from limited experience (Rawski and Heinz 2019). This point is not specific to language. Any cognitive theory requires carefully constructed computational restrictions on the hypothesis space in order to be tractable and analytically verifiable (van Rooij and Baggio 2021).

1.2 *Regular and subregular patterns*

Recent typological and experimental work highlights the Regular region of languages as a sufficient structural bound on computational expressivity for phonological and morphological grammars. This Regular characterization has been extended to syntactic distributions when the data structure characterizing the computational trace is formulated as a tree rather than strings, which enforce syntactic membership in the Mildly Context-Sensitive class of languages (Kobele 2011; Graf 2011). However, the Regular class is not learnable under various learning scenarios including identification in the limit from positive data, and the Probably Approximately Correct (PAC) framework (Gold 1967; Valiant 1984; de la Higuera 2010). Additionally, the range of distributions present in phonology and morphology that sit in the Regular region do not require the full complexity of Regular power (Heinz 2018; Chandlee 2017).

For these reasons, phonological constraints are hypothesised to inhabit structured subclasses of the Regular languages, lumped under the term subregular (Heinz 2010a, 2018). Various connections between logic, formal languages and automata defining these classes have been explored in great detail. These characterizations build on two classical results in formal language theory: Büchi's monadic second-order characterization of the Regular languages (1960), and the first-order characterization by McNaughton and Papert (1971) of the Star-Free languages, which are also characterized by aperiodic deterministic finite-state automata (Schützenberger 1965). Refinements of these results from logical, automata-theoretic, and algebraic viewpoints have defined the Local and Piecewise hierarchies (Rogers *et al.* 2012). Linguistically, these refinements have garnered interest since the morphological and phonological typology correlates with these refinements, favouring the weakest subclasses in the subregular hierarchy. Experimental work also favours this characterization (Finley 2008; Lai 2015; McMullin and Hansson 2019). Our learning algorithms can be applied to model-theoretic treatments of other linguistic representations such as syntactic trees or autosegmental graphs, which opens a useful direction for future research.

Outline 1.3

This paper proceeds as follows. Section 2 defines a general model-theoretic treatment of linguistic representations, and analyzes several types of linguistic structures based on different model signatures. Section 3 defines a typology of online learning algorithms and derives the subregular language classes, hierarchically organised by space complexity. Section 4 characterizes this space of algorithms according to time complexity, and picks out of the least space-intensive subregular classes those that can be learned in linear time. Section 5 characterizes interactions of constraints defined in and between these classes. Section 6 discusses model signatures for other linguistic representations. Section 7 describes related work. Section 8 concludes with future directions.

MODEL THEORIES 2

This section will introduce the structural representations that the learning algorithms will work over. We will first discuss a general notion of structural information, and use it to derive a notion of *substructures*. In contrast to previous approaches, this will allow us to describe several distinct representations of words in a uniform way. Structural information is defined relationally in terms of model theory. Finite model theory provides a unified ontology and a vocabulary for representing many kinds of objects, by considering them as *relational structures* (see Libkin 2004 for a thorough introduction). This allows flexible but precise definitions of the structural information in an object, by explicitly defining its parts and the relations between them. This makes model-theoretic representations a powerful tool for analyzing the information characterizing a certain structure. This application of model theory is nothing new. It has been applied to syntax by Johnson (1988), King (1989), and Rogers (1998), to phonology by Potts and Pullum (2002), Rogers *et al.* (2012), and Strother-Garcia (2019), and to tonal systems and autosegmental representations by Jardine (2017a), Jardine *et al.* (2021), and Oakden (2020).

The discussion of this section is organized around different notions of order: successor, precedence, and relativized successor. The successor and precedence orders give rise to the Local and Piecewise branches of the subregular hierarchy, and the relativized successor gives rise to the Tier-Based Local branch. We assume some familiarity with these classes. Because this presentation focuses on deriving these subregular classes from a model-theoretic and learning perspective, we postpone most references to these classes and related work to Section 7.

A relational structure in general is a set of domain elements, $\mathscr{D}$, which is augmented with a set of relations of arbitrary arity, $R_i \subseteq \mathscr{D}^{n_i}$. The relations provide information about the domain elements. The *model signature* $\mathscr{M} = \langle \mathscr{D}; R_i \rangle$ collects these parts and defines the nature of the structure in terms of the information in the model. Let w be a string over some alphabet Σ. Then a model for a word w is a structure:

$$\mathscr{M}_{\Sigma}^{R_i}(w) := \left\langle \mathscr{D}_w; R_i, \sigma_w \right\rangle_{\sigma \in \Sigma}$$

where $\mathscr{D}_w$ is isomorphic to an initial segment $\langle 1, \ldots, |w| \rangle$ of the nonzero natural numbers and represents the positions in w, and each σ_w is a unary relation that holds for all and only those positions at which σ occurs. Note that it is assumed that the set $\{\sigma_w\}_{\sigma \in \Sigma}$ is a partition of $\mathscr{D}_w$.[1] Without loss of generality, consider an alphabet $\Sigma = \{\text{s}, \text{ʃ}, \text{á}, \text{à}\}$, which represent two types of sibilants and a vowel with either low or high tone. Strings are combinations of these symbols at certain events, like the word 'sásàʃá'.

The remaining R_i are the other salient relations, which are used to define order in a particular structure. One model signature for strings, called the *precedence model*, is given as

$$\mathscr{M}^{<}(w) = \langle \mathscr{D}_w; <_w, \text{s}_w, \text{ʃ}_w, \text{á}_w, \text{à}_w \rangle.$$

This model says that for every symbol σ in alphabet Σ, there is a unary relation R_σ in $\mathscr{R}$ that can be thought of as a labelling relation for that symbol. For our set $\Sigma = \{\text{s}, \text{ʃ}, \text{á}, \text{à}\}$, $\mathscr{R}$ includes the unary relations R_s,

[1] One can convert a model in which multiple unary relations may apply to a given domain element into a partitioned normal form by simply replacing these unary relations with their powerset.

$R_{\text{ʃ}}$, $R_{\text{à}}$, and $R_{\text{á}}$. It also defines a binary relation $(x < y)$, the general precedence relation on the domain $\mathscr{D}$. A visual of the word model for ‘sásàʃá’ under this signature is given in Figure 1.

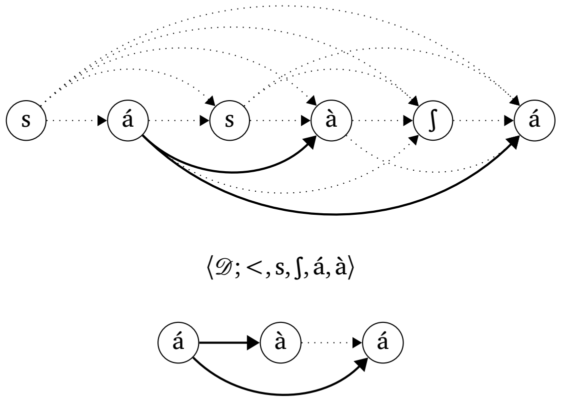


Figure 1:
The general precedence model of ‘sásàʃá’, along with the 3-factor ‘áàá’. Each edge defined by the relation is pictured, while the thick solid edges designate those that form the window from which this 3-factor is derived

The general precedence relation describes a notion of structural information purely in terms of whether a node precedes another one. While the information that, say, the last element in a string comes after the first is immediately accessible from the model, this distinction collapses the notions of immediate and general structural adjacency. Building on this precedence relation we can derive different types of relational structure. These refine the model of a word to describe immediate, relativized, or multiply-relativized adjacency.

Perhaps we would like to consider only immediately adjacent elements. Rather than a general precedence relation $<$, we may consider an immediate precedence, or successor, relation $\lhd$. The standard successor relation is the transitive reduction of the precedence relation and is first-order definable from the latter as follows:

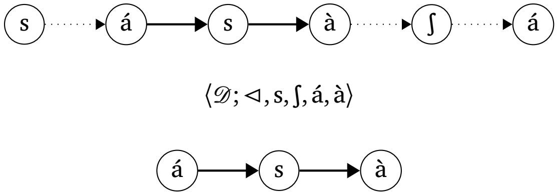
$$x \lhd y := x < y \land (\forall z)[x < z \Rightarrow y \leq z].$$

This relation gives a different word model, where elements are arranged according to immediate adjacency, commonly called the *successor model*. The signature for this model is given as

$$\mathscr{M}^{\lhd}(w) = \langle \mathscr{D}_w; \lhd_w, \mathrm{s}_w, \text{ʃ}_w, \text{á}_w, \text{à}_w \rangle.$$

A visual of the successor word model for the word ‘sásàʃá’ is given in Figure 2.

Figure 2:
The immediate successor model of 'sásàʃá', along with its 3-factor 'ása̍'



The general precedence relation can alternatively be refined to discuss a form of immediate adjacency relativized to certain unary relations in the signature. In particular, we can form relations between subsets of the alphabet, commonly called a *tier-alphabet*. For example, we may want to discuss the relations between only the sibilant elements present in a word, to the exclusion of all others. Similarly to how the successor relation is derived, we can restrict the precedence relation to the intended tier-alphabet τ and first-order define a similar tier-successor relation $\triangleleft^{\tau}$:

$$x \triangleleft^{\tau} y := \tau(x) \wedge \tau(y) \wedge x < y \wedge \big(\forall z\big)\big[\big(\tau(z) \wedge x < z\big) \Rightarrow y \leq z\big].$$

Figure 3 depicts the relationships among these ordering relations.

Figure 3:
Relationships between the general precedence relation and others first-order definable from it

$$\begin{tikzcd}
< \arrow[r, "\text{restrict}"] \arrow[d, "\text{reduce}"] & <^{\tau} \arrow[d, "\text{reduce}"] \\
\triangleleft \arrow[r, "\text{relativize}"] & \triangleleft^{\tau}
\end{tikzcd}$$

Adjusting the model signature appropriately, shown below, we get a tier-based notion of structure, shown visually in Figure 4.

$$\mathscr{M}^{\triangleleft^{\{\mathrm{s},\mathrm{ʃ}\}}}(w) = \langle \mathscr{D}_w; \triangleleft_w^{\{\mathrm{s},\mathrm{ʃ}\}}, \mathrm{s}_w, \mathrm{ʃ}_w, \mathrm{á}_w, \mathrm{à}_w \rangle.$$

Because the unary relations partition the domain elements, we can create a tier-adjacency relation for each element of the powerset of these relations. This merely amounts to adding tier-adjacency relations to the model signature to create a multi-tier signature. A model of the multi-tier relations is shown in Figure 5.

$$\mathscr{M}^{\triangleleft^{\{\mathrm{s},\mathrm{ʃ}\}}, \triangleleft^{\{\mathrm{à},\mathrm{á}\}}}(w) = \langle \mathscr{D}_w; \triangleleft_w^{\{\mathrm{s},\mathrm{ʃ}\}}, \triangleleft_w^{\{\mathrm{à},\mathrm{á}\}}, \mathrm{s}_w, \mathrm{ʃ}_w, \mathrm{á}_w, \mathrm{à}_w \rangle.$$

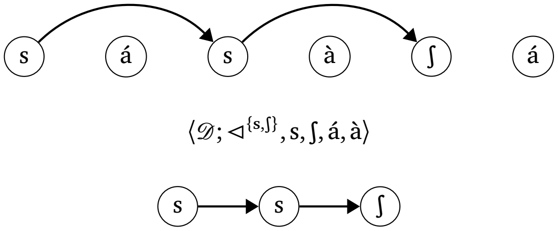


Figure 4:
The tier-successor model of ‘sásàʃá’ relativized over the set $\tau = \{\text{s}, \text{ʃ}\}$, along with its only 3-factor ‘sʃs’

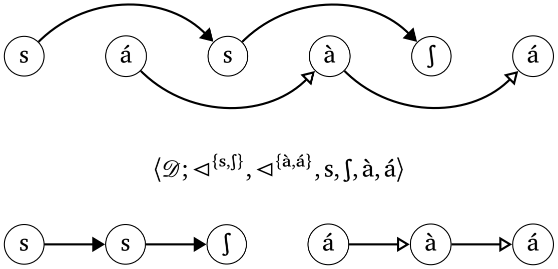


Figure 5:
The multi-tier-successor model of ‘sásàʃá’ relativized over the sets {s, ʃ} and {à, á}, along with its only two 3-factors, ‘sʃs’ and ‘áàá’

These four model signatures are by no means the only relational word models that may be considered. However, for the purposes of this paper we restrict ourselves to these signatures. Additionally, the definability of these signatures from other signatures leads to a general ability to define a notion of substructure, which we cover below.

Windows and factors 2.1

Now that we have a general model-theoretic notion of structure, we would like a way to define certain parts of each structure, each of which is a structure in itself defined by the signature. Here, we generalize the method of Lambert and Rogers (2020) in defining these restrictions on models.

In order to pick out the subparts of a word model, we first pick out sets of elements that will define the substructure. Given a homogeneous relation R of arity a, the set

$$\mathscr{W}_a^R(m) := \left\{ \left\{ \langle \overset{i}{x_i}, \overset{i+1}{x_{i+1}} \rangle : 1 \le i < a \right\} : \langle x_1, \ldots, x_a \rangle \in R_m \right\}$$

is the set of a-windows over R in the context of the model m. These are merely directed acyclic graphs (represented by their edge sets alone) constructed from the relations in R, such that each instance of a given domain element in the tuple is represented by a distinct node in the window, rather than merging all instances into a single node. Concretely, if 1 were a domain element and $\langle 1, 1\rangle$ an element of the relation, the corresponding 2-window would have two distinct nodes, both labelled by an index and the domain element 1: $\langle \overset{1}{1}, \overset{2}{1}\rangle$. The set of windows of length greater than a is defined inductively by

$$\begin{aligned}\mathscr{W}_{k+1}^{R}(m) := \Big\{ A \cup \langle \overset{j_{a-1}}{x_{a-1}}, \overset{k+1}{x_a}\rangle : A \in \mathscr{W}_{k}^{R}(m) &\text{ and } \langle x_1, \ldots, x_a\rangle \in R\\ &\text{and } \{j_1, \ldots, j_{a-1}\} \subseteq \{1, \ldots, k\}\\ &\text{and } \{\langle \overset{j_i}{x_i}, \overset{j_{i+1}}{x_{i+1}}\rangle : 1 \leq i < a-1\} \subseteq A\\ &\text{and } (\exists y, \ell)[\langle \overset{j_{a-1}}{x_{a-1}}, \overset{\ell}{y}\rangle \in A \text{ or } \langle \overset{\ell}{y}, \overset{j_{a-1}}{x_{a-1}}\rangle \in A]\\ &\text{and } \big(\forall j_a \in \{1, \ldots, k\}\big)\big[\langle \overset{j_{a-1}}{x_{a-1}}, \overset{j_a}{x_a}\rangle \notin A\big]\Big\}.\end{aligned}$$

This means that for each k-window, we find a linear subgraph (a path) that maps to the initial $a-1$ domain elements of one of the a-tuples that comprise R and add an edge from the final node of this path to a newly constructed node representing the final domain element from that tuple. The conditions are arranged in such a way that each iteration actually adds a new step to the path rather than simply repeating an older step, while still allowing cycles to be taken arbitrarily many times. Each of these larger windows can then be thought of as a graph of positions that are formed from a set of overlapping a-windows, which in turn are merely representations of tuples in the relation R. However, we may also wish to discuss a window which is of shorter length than the arity of the relation that defines it. To do so, we simply state that any connected subgraph of a window is itself a window.

For a given window x of a word model m, we define *the factor at* x (written $[\![x]\!]_m$) as the restriction of m to the domain elements that occur in x. This lets us define the set of all k-factors of m as follows:

$$\mathscr{F}_{k}^{R}(m) := \big\{[\![x]\!]_m : x \in \mathscr{W}_{k}^{R}(m)\big\}.$$

Note that a window is distinct from a factor in that the former is a graph of positions while the latter describes a word model whose domain consists of only a certain set of positions.

As example, consider the tier successor model of the word ‘sásàʃá’ as above. Consider a 3-window x which contains all and only the domain elements $\{1,3,5\}$. Here, the restriction of the word model that defines this 3-factor is

$$[\![x]\!]_m = m \restriction x = \big\langle \{1,3,5\}; \{\langle 1,3\rangle, \langle 3,5\rangle\}, \{1,3\}, \{5\}, \varnothing, \varnothing \big\rangle.$$

Similar examples can be seen above in Figures 1–5. Various parallels emerge. The precedence word model contains a strict superset of the factors of every other word model we have considered. The tier-based and multi-tier-based word models have ‘sʃs’ as a 3-factor, but the immediate successor model does not. On the other hand, ‘ásà’ is a 3-factor of only the precedence and immediate successor models. Only the precedence and multi-tier successor models have both ‘sʃs’ and ‘áàá’ (a sequence of High-Low-High tone vowels) as 3-factors.

Anchored word models 2.2

The word models considered up to this point do not encode domain boundaries explicitly. However, many prior treatments, including that of Lambert and Rogers (2020), explicitly assume such boundaries. One approach that has not been explicitly considered in this prior work is a model whose string yield is biinfinite. Here, left and right boundary symbols (labelled ⋊ and ⋉, respectively) exist in the model and both participate in and are self-related under any ordering relations. This approach naturally captures words shorter than k symbols in its concept of a k-factor, without having to consider a union of smaller factor widths. The successor model for ‘sásàʃá’ is shown along with each of its 3-factors in Figure 6.

The learning algorithms that we consider in this work are not bound to any particular model signature. Thus, we may consider the standard word models as shown in, for example, Figure 1, or we might consider these anchored word models.

This section has shown concretely how relational structures provide a uniform language for describing the structural information in representations of words. In this way, the differences between distinct subregular classes are isolated according to the relevant structural information. Also, the models considered in this section are just some

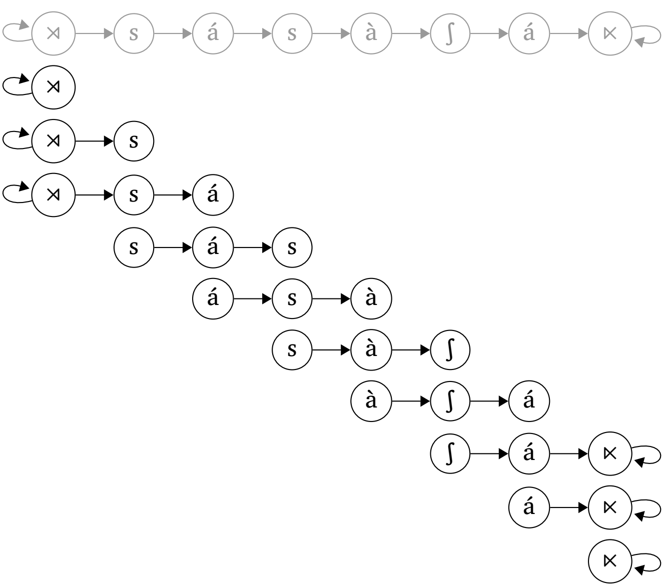


Figure 6: The anchored word model under successor for 'sásàʃá' along with each of its 3-factors. Note that every factor that includes a boundary symbol has an infinite yield. Those factors shorter than 3 symbols are formed from windows of length 3 that repeat the boundary symbols

of many models. The contents of each model signature clarify precisely what structural information the learner has immediate access to when making inferences during learning, described by the notion of a k-factor, and which it must computationally infer. For example, the non-local information that is immediately present in the precedence model requires more work in the successor model, its transitive reduct. These properties are encoded into the grammars being learned, and directly carve out the properties of classes of languages that result from a particular learning algorithm inferring such structures.

3 SPACE COMPLEXITY AND THE SUBREGULAR GRID

This section will introduce and examine four learning algorithms—algorithms I, II, III, and IV—where stringsets are learned in the limit from arbitrary positive data. Indeed, we will only be considering subclasses of a style of learning algorithm presented by Heinz (2010b, expanded upon by Heinz *et al.* 2012). We show that of these subclasses, some require substantially more space than others to properly account for the distinctions that must be made in the course of learning, and we argue that this alone would cause linguistic typology to tend toward the simpler, less space-intensive classes.

First we briefly discuss some background from learning theory. Generally, our presentation follows the style of Gold (1967). While issues with this theoretical framework have been pointed out (Johnson 2004; Clark and Lappin 2011), these criticisms stem from misunderstandings (see Heinz 2016, and references therein). Gold's framework is the basis for much influential work on learning formal languages (Jain *et al.* 1999; Nowak *et al.* 2002; Niyogi 2006; de la Higuera 2010; Clark and Lappin 2011).

More importantly, however, the algorithms we present here are largely independent of the particular learning framework that we use to evaluate their behaviour. They can be studied with respect to the various identifiability-in-the-limit paradigms of Gold, but they can also be studied with respect to other paradigms (Mohri *et al.* 2012). For example, all of the algorithms presented here are not only identifiable in the limit from positive data in polynomial time, they are also PAC-learnable.[2] While the assumptions of PAC learning, including the use of negative evidence and approximate identification, seem to make the learning problem easier, in fact the conclusions show the learning problem is harder. For example, the finite languages, learnable in the limit from positive data, are not PAC learnable. Interestingly, not all PAC-learning algorithms even require negative evidence. The standard textbook examples of rectangles (Kearns and Vazirani 1994) and rays (Anthony and Biggs 1992) only use positive data just like our algorithms here. Despite these differences, both frameworks focus the learning problem on generalization which has led some researchers to provide a unified analysis of these different frameworks (Niyogi 2006). Nonetheless, irrespective of the framework, we demostrate that the space complexity requirements are severe for algorithms III and IV, but not for algorithms I and II.

It is important to note that, while we present only four algorithms here that are sufficient to learn the well-understood subregular classes

[2] This is because when the parameters k (and t) are fixed, the defined class has a finite VC dimension (since the class has finite cardinality) (Vapnik 1995).

under consideration, these are not the only possible algorithms. Others do exist and may well meet the criteria for these learning frameworks. The complexity results here are general, applying to any algorithm that can learn the classes, simply because they are based on the kinds of distinctions that must be made.

First, we describe the general learning setup. Let L be a set of strings drawn from Σ^* and let $L_{\circledcirc}$ represent L with an adjoined element $\circledcirc$. An online learner is a function $\varphi\colon \mathscr{G} \times L_{\circledcirc} \to \mathscr{G}$, where $\mathscr{G}$ is some kind of grammar representation, a mechanism by which one can decide whether a given string is in L. In other words, an online learner begins with some guess as to what the grammar might be and updates this guess for each input word. Let $\mathscr{L}\colon \mathscr{G} \to \mathscr{P}(\Sigma^*)$ be the function that maps a grammar to its extensions, the set of strings it represents. Two grammars G_1 and G_2 are equivalent ($G_1 \equiv G_2$) iff they are extensionally equal, that is, $\mathscr{L}(G_1) = \mathscr{L}(G_2)$.

A text for L is a function $t\colon \mathbb{N} \to L_{\circledcirc}$, a sequence of strings drawn from L or pauses in which data does not appear. Following traditional mathematical notation for sequences, we use t_n to represent $t(n)$. If $\varnothing$ represents an initial guess at what the grammar might be, then the recursively-defined sequence

$$a_n(t) := \begin{cases} \varnothing & \textit{if } n = 0 \\ \varphi(a_{n-1}, t_n) & \textit{otherwise.} \end{cases}$$

represents the learning trajectory over a given text. Then given a text t for a language L, we say that a learning algorithm φ *converges* on t iff there is some $i \in \mathbb{N}$ such that for all $j > i$ it holds that $a_j(t) \equiv a_i(t)$. If for every possible text t over L it is the case that φ converges on t and $\mathscr{L}\big(\lim_{n\to\infty} a_n(t)\big) = L$, then we say φ converges on L. As a second lift, if for every stringset L in a class $\mathbb{L}$ it is the case that φ converges on L, then we say φ converges on $\mathbb{L}$.

3.1 *String extension learning*

Heinz (2010b, expanded by Heinz *et al.* 2012) defined string extension learning, a general notion of learning from gathered substructures. Originally treated only as a batch learner, the online definition is trivial to derive. Given a function $f : \mathscr{M} \to \mathscr{S}$ that extracts informational content from a word model, where $\mathscr{S}$ represents some notion of structural content, along with a combinator $\oplus\colon \mathscr{G} \times \mathscr{S} \to \mathscr{G}$ that somehow informs the grammar of these structures, we define

$$\varphi(G, w) := \begin{cases} G & \textit{if } w = \circledcirc \\ G \oplus f\big(\mathscr{M}(w)\big) & \textit{otherwise.} \end{cases}$$

In a simple case, $\mathscr{G}$ and $\mathscr{S}$ will be the same type, and $\oplus$ will simply be set union, but this is not a necessary requirement.

Although the present discussion has been contextualized in the presence of a complete text, the algorithms can only ever operate on a finite sample. No infinite complete presentation is ever needed or even available. The analysis with complete texts guarantees that no matter the order of the input there is always some finite point in time, some finite sample, at which point every piece of informational content that could occur has occured, and the algorithm will converge exactly to the target grammar (Heinz *et al.* 2012). For samples that do not meet this criterion, the smallest stringset in the target class that is consistent with the data will be learned instead of the target stringset itself (Heinz *et al.* 2012).

The space required by any string extension learning algorithm is bounded below by the output grammar size. This is dependent on the type of information that the grammar must retain. For the subsequent discussion, no additional space is necessary, so all that is relevant is the size of the grammar representation. Generally the worst case is when the target language is Σ^* and every factor, set, or multiset will need to be observed and stored.

Learning with factors 3.1.1

This simple case is exemplified by a learner that makes distinctions only between permitted and nonpermitted factors. This learner is parameterised by a factor width k. We have $\mathscr{G} = \mathscr{S} = \mathscr{P}(\Sigma^k)$ and $G \oplus S = G \cup S$. The information extraction function is

$$f(m) := \mathscr{F}_k(m).$$

Upon convergence, a word w is accepted iff all of its factors occur in G as shown in Figure 7.

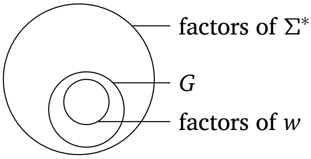


Figure 7:
Grammars returned by Algorithm II accept all and only those strings w whose factors are all in G

Since the grammar only needs to maintain a single merged set of attested factors, the space complexity for this class of learner is $\mathcal{O}(|\Sigma|^k)$. This will be referred to as Algorithm II. A variant, which will be Algorithm I, will be discussed in Section 3.1.4.

3.1.2 Learning with sets

The primary difference between learning with factors and learning with sets thereof is the grammar augmentation combinator. Rather than set union, $G \oplus S = G \cup S$, we have set insertion, $G \oplus S = G \cup \{S\}$. This of course means that $\mathscr{G}$ and $\mathscr{S}$ are no longer equal, with $\mathscr{G}$ being the powerset $\mathscr{P}(\mathscr{S})$, adding a layer of structure. Upon convergence, a word w is accepted iff its set of factors is an element of G as shown in Figure 8. Since a given grammar is in this case a set of sets of factors, with this larger grammar the space complexity is $\mathcal{O}(2^{|\Sigma|^k})$. These set-based classes can make more distinctions than the purely factor-based classes, but this power comes at a cost. This is Algorithm III.

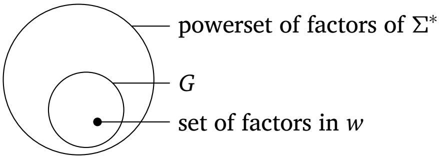


Figure 8:
Grammars returned by Algorithm III accept all and only those strings w whose set of factors is an element of G

3.1.3 Learning with multisets

A set is simply a structure that contains for each possible element a Boolean value describing whether or not that element is included. Given the natural isomorphism between the Booleans and the subset of $\mathbb{N}$ consisting of 0 and 1, one might consider a natural expansion of this structure which denotes number of occurrences saturating not at 1 but at some arbitrary value t. (In other words, t is the largest number one can count to.) We can learn classes in which well-formedness is characterized by the saturating multisets of factors in a word as follows. With

$\mathscr{S} = \mathscr{P}(\Sigma^k \times \mathbb{N}_t)$ and $\mathscr{G} = \mathscr{P}(\mathscr{S})$, we can maintain from the set-based learner the augmentation combinator where $G \oplus S = G \cup \{S\}$. However, the function that extracts informational content must be modified to include the t-counts associated with a given factor as follows

$$f(m) := \wr [\![x]\!]_m : x \in \mathscr{W}_k(m) \int_t .$$

The notation $\wr \ldots \int_t$ represents a multiset that saturates at a count of t. Note that this parallels the window-based definition of factors in a model, except that a saturating multiset is formed rather than merely a set. Upon convergence, a word w is accepted iff its saturating multiset of factors is an element of G as shown in Figure 9. The space complexity here is much like that of Algorithm III, except that the base of the exponent is changed to correspond with the number of values each factor may be associated with: $\mathscr{O}\big((t+1)^{|\Sigma|^k}\big)$. This is Algorithm IV. Using this algorithm with $t = 1$ is equivalent in every way to Algorithm III, so in fact there are only three algorithms under discussion. That said, we will retain this separation for the current discussion.

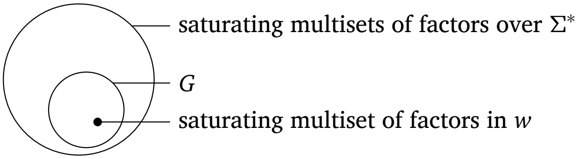

saturating multisets of factors over Σ^*
G
saturating multiset of factors in w

Figure 9:
Grammars returned by Algorithm IV accept all and only those strings w whose saturating multiset of factors is an element of G

Learning with factors, revisited 3.1.4

A variant of Algorithm II ignores all input words longer than k symbols. The only difference then is the information extraction function

$$f(m) := \begin{cases} m & \textit{if } |m| \le k \\ \varnothing & \textit{otherwise.} \end{cases}$$

Upon convergence, a word w is accepted iff it contains a factor that also occur in G as shown in Figure 10. This is Algorithm I. Notably, using the anchored word models with this algorithm produces only finite languages. In contrast to the other algorithms, translating such models into unanchored ones provides an increase in expressive power.

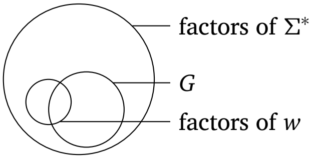


Figure 10:
Grammars returned by Algorithm I accept all and only those strings w whose sets of factors are not disjoint with G

3.1.5 Illustration

Considering a standard unanchored word model, with the algorithmic parameters k and t both set to 2, Table 1 represents the outputs of these four learning algorithms after seeing the single word 'aaaab'. Notably, this word is not short enough to inform Algorithm I of anything. Also, despite the fact that 'aa' occurs as a substring three distinct times, Algorithm IV saturates at a count of 2 under these assumed parameters.

Table 1:
Encountering the single word 'aaaab' with each learner

Algorithm	Resulting Grammar
I	$\varnothing$
II	$\{\mathrm{aa}, \mathrm{ab}\}$
III	$\big\{\{\mathrm{aa}, \mathrm{ab}\}\big\}$
IV	$\big\{\{\langle \mathrm{aa}, 2\rangle, \langle \mathrm{ab}, 1\rangle\}\big\}$

3.2 *The grid*

These learning algorithms are model-agnostic. As long as there exists some way to extract windows or factors (*i.e.*, substructures) from a model, the algorithms will work with that. When allowed to range over selected model signatures, the classes learned by each algorithm are shown in Figure 11. Each of the cells of the grid represent a particular class of languages. For example, the strictly local class contains languages for which no word may contain any of a finite set of local factors.

For clarity, we restrict our discussion of the fifteen classes present in the subregular Grid to the Appendix. There, we provide a brief description of the class, as well as a sample of attested linguistic patterns that it accounts for, as well as an interpretable implementation of the grammar for each of those patterns.

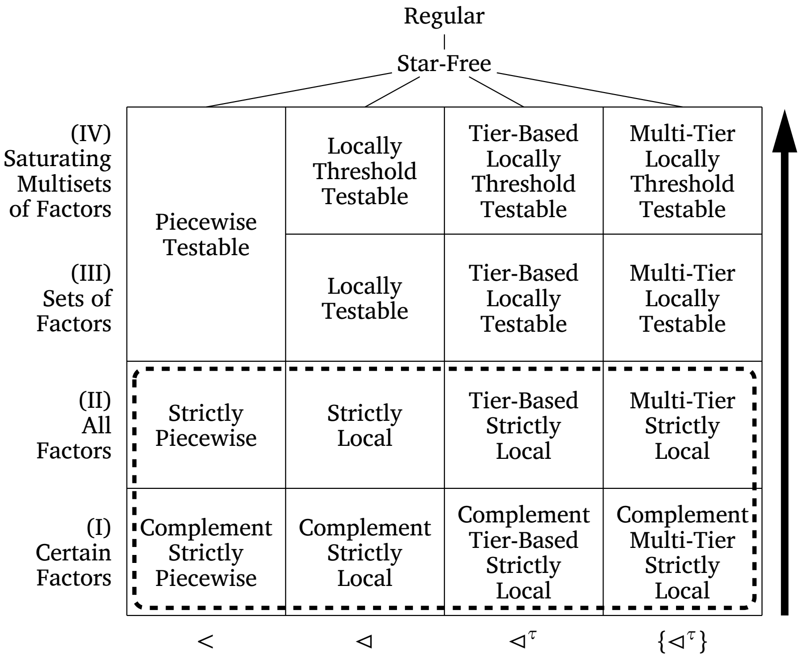

Regular
Star-Free
(IV) Saturating Multisets of Factors
(III) Sets of Factors
(II) All Factors
(I) Certain Factors
Piecewise Testable
Locally Threshold Testable
Tier-Based Locally Threshold Testable
Multi-Tier Locally Threshold Testable
Locally Testable
Tier-Based Locally Testable
Multi-Tier Locally Testable
Strictly Piecewise
Strictly Local
Tier-Based Strictly Local
Multi-Tier Strictly Local
Complement Strictly Piecewise
Complement Strictly Local
Complement Tier-Based Strictly Local
Complement Multi-Tier Strictly Local
$<$
$\lhd$
$\lhd^{\tau}$
$\{\lhd^{\tau}\}$

Figure 11: The subregular classes. Learning difficulty increases along the vertical axis. The horizontal axis is categorical, describing the type of substructure. The dotted line indicates the predicted region of phonological typology

Note that Algorithm I learns only strong classes (those in which domain boundaries, i.e. anchors, are unreferenced), while the others do not have this restriction. In the Piecewise case, where the model signature contains the general precedence relation ($<$), the strong classes are equivalent to the general classes and this distinction is irrelevant.

We note that the amount of space required to store the grammar is fairly large for any of these algorithms. But Algorithms III and IV require exponentially more space than I and II. These space requirements are shown in Figure 12, where it is apparent that even on a binary alphabet, the smallest possible nontrivial alphabet, the Locally 5-Testable class of languages, for example, requires more storage space than there are synapses in an average human being (Azevedo *et al.* 2009; Herculano-Houzel and Lent 2005). With the larger alphabet sizes commonly encountered in natural language the restrictions become even tighter. The interested reader could as an exercise consider how this graph would change if the size of the alphabet were around

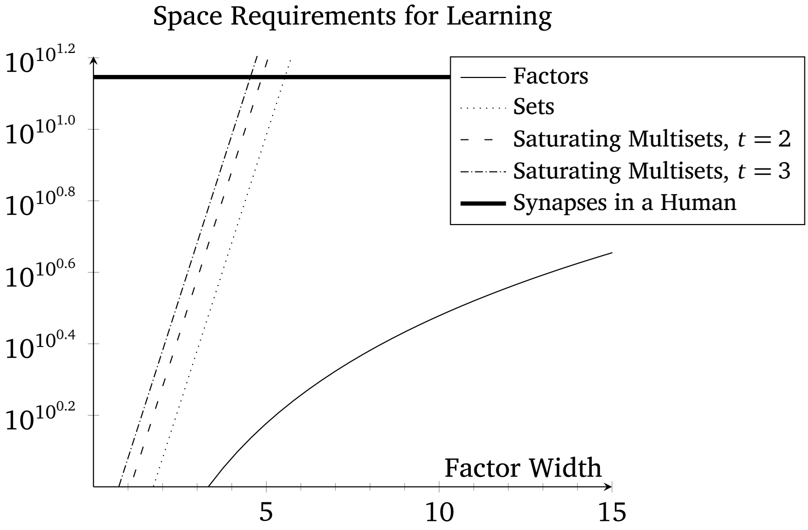


Figure 12: While gathering factors requires space exponential in terms of factor width, the requirements are doubly exponential for any of the larger structures we might employ. Here the space requirements are shown for just a binary alphabet

30 or 40, the average phoneme count in languages of the world. However, it should be noted that attempting to plot this graph for an alphabet of size 10 exceeded the numerical range of our plotting software.

Due to the enormous space requirements in terms of alphabet size of the higher-numbered algorithms, it seems in general unfeasible to learn patterns that lie strictly in their corresponding classes. That is, purely from learnability considerations alone, we would expect the typology of patterns in natural language to lie primarily within the region spanned by Algorithms I and II. This region is highlighted in Figure 11 by a dotted line. Further, we would expect any attested patterns to require relatively small values for the factor width parameter k, since that is the exponent of these singly and doubly exponential space complexities. This constraint on the learning algorithms is again agnostic to the representation, showing that the way the learner collects and stores the data matters.

The multiple-tier-based classes also require significant space, but in a different way. The classes of Algorithms III and IV admit exponentially more distinctions than Algorithms I and II, and thus require exponentially more space. The multiple-tier-based classes in contrast require many grammars of the same type: one per subset of Σ. This scales the space requirements by a multiplicative factor of $2^{|\Sigma|}$. This difference, while less pronounced, is still significant and will be taken into account later.

To briefly sum up, this section presented a suite of online learning algorithms that extract structural information based on the particular representations it is given. The combination of a particular algorithm and a model-theoretic signature define a range of classes of languages that can be learned. One model signature may be used by any of the learning algorithms, and any of the algorithms may use any of the signatures. In this way, we have organized the space of possible generalizations the typology can inhabit, which ultimately amounts to possible restrictions on the capacity of the learner. This provides a unifying perspective on previously studied subregular classes. Another contribution of this section is the introduction of Algorithm I, which naturally leads to "Complement" classes of the "Strict" ones.

However, there is a strong divergence between the space requirements of the two algorithms that make distinctions based solely on the presence of individual factors and those two algorithms that make distinctions based on sets or multisets of factors. For a feasible learner, then, it is advantageous to disprefer learning strategies that rely on an ability to make as many distinctions as these two more complex algorithms allow. Drawing a boundary for the language classes learned by the two simpler algorithms, we significantly reduce the possible typology available to the learner. Can there be any other restrictions? This is the topic of the next section.

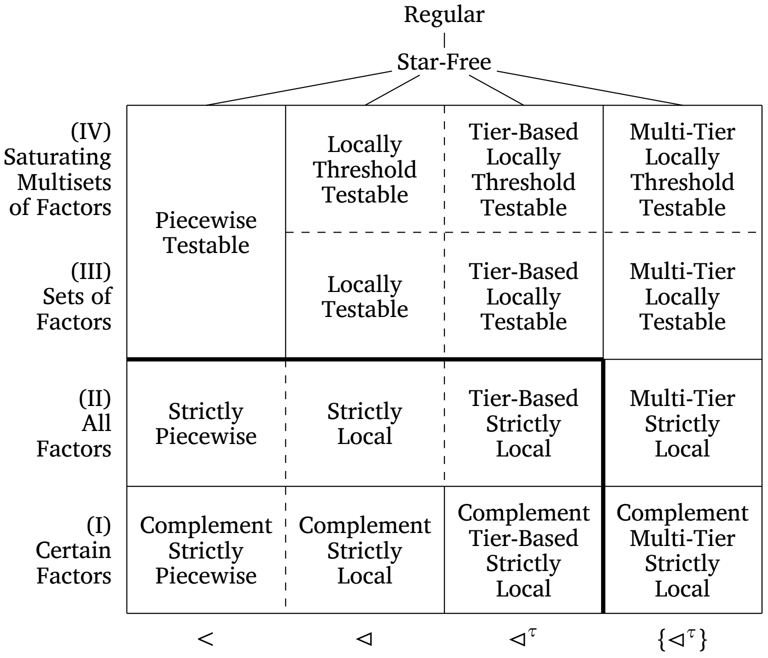
COMPLEXITY IN TIME 4

The Strictly Local (SL) class (McNaughton and Papert 1971) is learned by gathering the factors of simple adjacency. Under such a model, there exists at most a single window of size k at any given point. Thus for each index in the word, we can simply insert the contents of this single window into the grammar. Including the time it takes to insert a factor into a set, the class is learnable in $\mathcal{O}(nk \log|\Sigma|)$ time for input of size n, and since Σ and k are assumed constant this amounts to linear time. As discussed, this Algorithm II learner also uses constant space that is but singly exponential in the width of the factors.

For the Tier-Based Strictly Local (TSL) class (Heinz *et al.* 2011, see also Lambert and Rogers 2020), if the tier alphabet τ is known, then this approach applies directly to the projection of the word to τ. But generally we assume that τ is not known, and one might initially assume that a learner might need to construct grammars for all possibilities, which would result in increased resource requirements, be that in terms of time, space, or both. Per Jardine and McMullin (2017), maintaining the factors of width bounded above by $k+1$ is sufficient to determine the value of τ. But their approach seems to require a batch approach, first deciding the value of the τ parameter and then processing the (projections of) the input as for the Strictly Local class. But it turns out that, due to (inverse-)projection closure and the fact that in the Gold framework we assume a complete text, we can guarantee that any substring whose projection will appear on the tier will itself appear as a substring in some word. Since we still need to determine the value of τ, we do still require the factors of width bounded above by $k+1$, but nothing more. The exact learning algorithm used for SL_{k+1} will produce a grammar for TSL_k, and only the interpretation of the result is changed (Lambert 2021). These same properties hold true for the relativized variants of the Locally Testable (LT) (McNaughton and Papert 1971) and Locally Threshold Testable (LTT) (Beauquier and Pin 1989) classes as well, where the corresponding adjacency-based learners suffice to learn the relativized-adjancy classes (Lambert 2021).

The Strictly Piecewise (SP) class (Rogers *et al.* 2010, see also Haines 1969) is similar in that, one might expect a time complexity on the order of $\mathcal{O}(n^k)$ to find all of the subsequences of each word. Heinz and Rogers (2013) show that in fact a factored approach can use simply $\mathcal{O}(n|\Sigma|^k)$, but we can reduce this even further by taking advantage of this same property. Given a complete text, every attested subsequence will eventually occur as a substring due to the SP stringsets' closure under deletion. Again then, the same learning algorithm used for SL_k will produce a grammar for SP_k as well, where the difference lies only in interpretation.

Given this ability to learn the SP, SL, and TSL classes in linear time and in space only singly exponential in factor width, we can modify Figure 11 to indicate the boundary between the classes that are learnable within these resource bounds and those that are not. This boundary is indicated by a thick line in Figure 13, which also uses dashed lines to indicate where one algorithm may be used for multiple distinct classes. As discussed in Section 3.2, the multiple-tier-based classes do not fit within this low-resource region because, in general, exponentially many grammars must be learned.



	$<$	$\lhd$	$\lhd^{\tau}$	$\{\lhd^{\tau}\}$
(IV) Saturating Multisets of Factors	Piecewise Testable	Locally Threshold Testable	Tier-Based Locally Threshold Testable	Multi-Tier Locally Threshold Testable
(III) Sets of Factors		Locally Testable	Tier-Based Locally Testable	Multi-Tier Locally Testable
(II) All Factors	Strictly Piecewise	Strictly Local	Tier-Based Strictly Local	Multi-Tier Strictly Local
(I) Certain Factors	Complement Strictly Piecewise	Complement Strictly Local	Complement Tier-Based Strictly Local	Complement Multi-Tier Strictly Local

Figure 13: Dashed lines indicate that the classes on either side can be learned by exactly the same algorithm. The thick solid line denotes the barrier between linear-time $\mathcal{O}(|\Sigma|^k)$-space learning and more resource-intensive learning

Considering only the SP class of stringsets, there are at least three online learning algorithms of various complexities:

- Gather all factors under general precedence of each word.
    - $\mathcal{O}(n^k k \log|\Sigma|)$ time
    - $\mathcal{O}(|\Sigma|^k)$ space
    - Learns an SP least upper bound (lub) of the source text.

- Use factored learning as per Heinz and Rogers (2013).
  - $\mathcal{O}(n|\Sigma|^k)$ time
  - $\mathcal{O}(|\Sigma|^k)$ space
  - Learns an SP lub of the source text.
- Gather all adjacency factors of each word.
  - $\mathcal{O}(nk\log|\Sigma|)$ time
  - $\mathcal{O}(|\Sigma|^k)$ space
  - Only works if every permitted subsequence eventually occurs adjacently, which holds for SP targets.

One caveat is that these optimizations of the learning algorithms for the SP and TSL classes rely on certain properties of the input stringset. The nonoptimized variants are guaranteed to learn a smallest in-class superset of the input stringset, a property which is lost in this optimization. For example, a long distance sibilant harmony constraint (Heinz 2010a) will not be learned by the optimized SP learner if the text is drawn from a language that exemplifies both this constraint and a CV syllable structure, even though it would be learned by the nonoptimized variant. Other examples of this type may be found in the Appendix. This prompts a question regarding the learnability consequences of constraint interaction.

5 LEARNING INTERACTIONS

Most natural languages are describable not by a single subregular class but by an interaction of constraints from multiple such classes. The interaction of constraints from different classes might influence the learnability of each constraint individually, in which case time or space tradeoffs might be necessary.

For example, we might consider the default-to-opposite stress pattern of Chuvash (Krueger 1961), where primary stress falls on the rightmost heavy syllable if there is one, or on the initial syllable otherwise. One way of describing this invokes the conjunction of two constraints from two different classes, namely an SP constraint detailing a lack of:

- a heavy syllable anywhere after a stressed syllable,
- a stressed light syllable after any other syllable, or
- two stressed syllables in the same word,

and a coSP constraint that states that every word contains some stressed syllable.

The requirement that some stress must occur does not affect which substrings may appear in a word, and so the SP constraint may be learned by any of the three algorithms that have been discussed for that class so far, including the optimized substring-based learner. Further, the precedence restrictions do not prevent seeing the two words (light and heavy stressed monosyllables) required to learn the stress requirement. In other words, these constraints interact in such a way that learning is not hindered. This is not always the case.

Consider now the sibilant harmony of Samala (Applegate 1972), in which as 's' and an 'ʃ' may not appear in the same word. Since this constraint acts on the segment level rather than the syllable level, we might assume that it is isolated from any kind of stress constraint. But other segment-level constraints will certainly have the possibility of interaction. For example, imposing a CV syllable pattern restricts the substrings that may occur, in such a way that using an SL learner to infer the SP constraints is not a possibility. This means that one has to decide among the other possible SP learning algorithms, where time or space tradeoffs must be made.

In contrast, a tone plateauing constraint like that which occurs in Luganda (Hyman and Katamba 1993) is SP_3, which means that it could be learned directly alongside this sibilant harmony constraint without fear of interaction effects. Note that the word 'sásàʃá' that has been our running example violates both the harmony constraint and the tone plateauing constraint.

Given our space-based learnability considerations, we would assume that Algorithms III or IV are not practically learnable and would likely be unattested. In other words, we would expect linguistic typology to inhabit only the lower regions of the hierarchy, or at least be biased heavily toward this region. Rogers and Lambert (2019b) provide strong evidence that this is in fact the case when it comes to stress patterns. Their exhaustive analysis of the more than one hundred stress patterns in the StressTyp2 database (Goedemans *et al.* 2015) showed that each of these can be described as the interaction of constraints that can be learned by Algorithms I and II.

6 MODEL-THEORETIC REPRESENTATIONS OF NONLINEAR STRUCTURES

This model-theoretic formulation provides a distinct advantage when applied to various linguistic objects. It allows one to characterize the content of a particular linguistic representation, and in so doing, immediately guarantee that there are learning algorithms which can describe various constraints over those representations. This is important, because work describing nonlinear structures in syntax and phonology has proceeded in an ad-hoc way, by first defining constraints, and working backwards to the representations, often without any learning algorithms at all, or ones relativized to a particular structure.

The previous sections used various model signatures that characterized information based on a string data structure. This is because the subregular classes that were the central motivation for this paper are defined over strings, or model signatures based on strings, in the work of Büchi, Thomas, and others. The constructions considered to this point are not restricted to simple string models. Without modification, the algorithms may be applied to any relational model at all. They in fact apply to any structure that can be characterized as a graph. In this sense, strings are a special case, but the distinctions that each of the four learning algorithms pick out carry over onto these more general factors as well. In this section, we discuss some other linguistically-motivated models that one might consider.

6.1 *Autosegmental graphs*

An example of a nonlinear structure where the graph perspective is clearly relevant to linguistic research concerns autosegmental representations in phonology. Graphs were proposed to handle a variety of prosodic phenomena for which the string-based perspective was inadequate. Phonological processes affecting domains larger than two adjacent segments, such as tonal alternations in tonal languages, have temporal properties that do not always map consistently onto discrete vowel segments in a one-to-one fashion (Goldsmith 1976; Williams 1976). Goldsmith introduced a model of the phonological word where tonal features formed an independent string from the segmental string, called a tier. Segments on the two strings are linked via many-to-one relations, turning the structure into a graph.

In practice, encoding these adjustments into a word model involves adding more relational structure. Jardine (2017a,b, 2019) uses a binary relation $\alpha(x, y)$ to encode the association relation between autosegmental tiers. Augmenting the successor model signature used throughout this paper gives a signature as

$$\mathscr{M}^{\alpha,\lhd}(w) = \langle \mathscr{D}_w; \alpha_w, \lhd_w, \mathrm{s}_w, \int_w, \mathrm{a}_w, \mathrm{H}_w, \mathrm{L}_w \rangle.$$

Here, the domain is increased to accommodate the new autosegments, and the successor relation holds between elements on both tiers. The unary relations encoding vowels with tonal features have been split, into a relation ‘a’ for vowel information, and distinct ‘H’ and ‘L’ relations for tonal information. Under this signature, a word model for the example ‘sásàʃá’ is given in Figure 14.

Our notion of a factor is exactly a notion of a subgraph. The previous section showcased how this word violates a constraint on tone plateauing. The autosegmental model makes this information immediately accessible by encoding the ‘HLH’ structure as its own subgraph, shown on the bottom of Figure 14. Thus, the permissibility of tone sequences is liberated from the segmental elements that carry them.

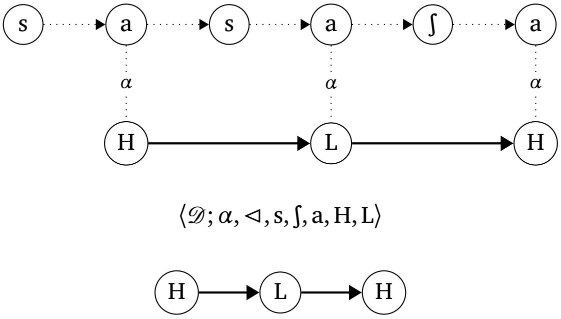


Figure 14:
The autosegmental successor model of ‘sásàʃá’, along with its 3-factor ‘HLH’. The α relation is shown without tips because it is symmetric

6.2 Tree models

The model-theoretic framework also allows describing tree structures (Rogers 1998), and opens the door to study parallels between phonological and syntactic constraints (Graf 2014). Rogers (2003) describes a model-theoretic characterization of trees of arbitrary dimensionality. In this framework, we specify the domain $\mathscr{D}$ as a Gorn tree domain (Gorn 1967). This is a hereditarily prefix closed set D of node addresses, that is to say, for every $d \in D$ with $d = \alpha i$, it holds that $\alpha \in D$, and if $i \neq 0$ then $\alpha(i-1) \in D$ as well. In this view, a string may be called a one-dimensional or unary-branching tree, since it has one axis along which its nodes are ordered. In a standard tree, on the other hand, the set of nodes is ordered as above by two relations, "dominance" and "immediate right-of". Suppose s is the mother of two nodes t and u in some standard tree, and also assume that t precedes u. Then we might say that s dominates the string tu.

While a Gorn tree domain as written encodes these dominance and adjacency relations implicitly, we may explicitly write them out model-theoretically so that a signature for a Σ-labelled two-dimensional tree T is $\mathscr{M}^{\triangleleft^{\downarrow},\triangleleft^{\rightarrow}} = \langle D; \triangleleft^{\downarrow}, \triangleleft^{\rightarrow}, R_{\sigma} \rangle_{\sigma \in \Sigma}$ where $\triangleleft^{\downarrow}$ is the immediate dominance relation and $\triangleleft^{\rightarrow}$ is the immediate right-of relation (see Figure 15). Model signatures that include the transitive closures of each of these relations have also been studied. Additionally, the anchored word models considered above for strings lift naturally

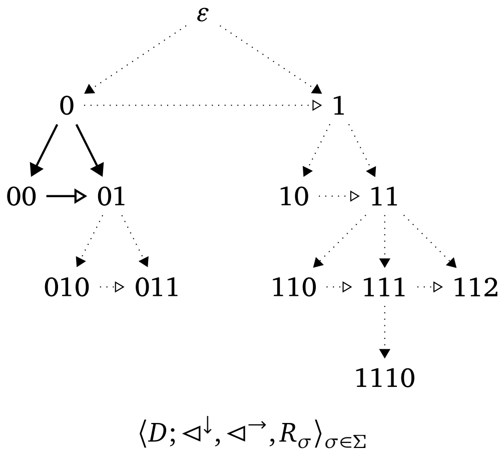

ε
0
1
00
01
10
11
010
011
110
111
112
1110
$\langle D; \triangleleft^{\downarrow}, \triangleleft^{\rightarrow}, R_{\sigma} \rangle_{\sigma \in \Sigma}$

Figure 15: A tree model. Nodes are organised by immediate dominance (black tip) and immediate right-of (white tip) relations. Labelling relations are omitted to show Gorn addresses. All edges are shown, with a particular factor noted with solid thick lines

to trees, where a root node is an anchor and each leaf is a separate anchor, or there is a single additional node which serves as the anchor for every leaf .

Recent work in syntax has synthesised the model-theoretic approach to trees with insights from the subregular approach to phonology. For instance, Graf and Shafiei (2019) hypothesise that the TSL class is sufficient to characterize syntactic constraints.

To sum up, this section has shown how the model-theoretic representations presented in Section 2 naturally apply to other linguistic representations.

FURTHER READING 7

The subregular classes considered here have been widely studied for decades. McNaughton and Papert (1971) introduce the Local hierarchy, with Beauquier and Pin (1989) adding the Locally Threshold Testable class. The Piecewise branch of the hierarchy stems from Simon (1975), with the Strictly Piecewise class only being integrated into the hierarchy in 2010 by Rogers *et al.*. (Languages closed under subsequence had been discussed by Haines 1969, though not in connection with other subregular classes.) The Tier-Based Strictly Local class was introduced by Heinz *et al.* (2011) and extended in various ways by De Santo and Graf (2019), Lambert and Rogers (2020), and Lambert (2021). Recent work in syntax has synthesized the model-theoretic perspective on trees with insights from the subregular program (Graf and Shafiei 2019; Graf 2020, 2014)

Provided a finite-state automaton, Caron (1998, see also Caron 2000) describe algorithms that decide whether the corresponding language is Locally or Piecewise Testable. An efficient algorithm for deciding SL is described by Edlefsen *et al.* (2008). Algorithms that extract SL and SP factors from a given language (and thus can also be used to decide class membership) are due to Rogers and Lambert (2019a), and these were extended to the TSL class by Lambert and Rogers (2020).

While this paper has so far focused on constraints, this work is easily extended to consider mappings between structures, expressed mathematically as Regular functions (Courcelle 1994; Courcelle and

Engelfriet 2012; Filiot 2015; Engelfriet and Hoogeboom 2001). The notion of strict locality has been generalized to functions and shown to be relevant for natural language phonology and morphology (Chandlee 2014, 2017). These local functions have been model-theoretically characterized and extended to consider nonlinear structures in phonology (Chandlee and Jardine 2019; Strother-Garcia 2019). Relativing input representations to consider multi-arity functions allows a notion of strictly local transducers expressed using multi-tape automata (Rawski and Dolatian 2020; Dolatian and Rawski 2020). Expressed as functions, these subregular characterizations have been extended to consider continuous functions over vector spaces and learning algorithms operating over them (Rawski 2019; Nelson *et al.* 2020).

There exist other learning algorithms alongside the string extension learners of Heinz (2010b) and Heinz *et al.* (2012). Garcia *et al.* (1990) demonstrate the learnability of SL. Heinz and Rogers (2013) provide learning algorithms for the SL and SP classes as well as their Testable correlates. Other approaches have directly incorporated phonological features into the models (Vu *et al.* 2018; Chandlee *et al.* 2019). Learning of TSL classes has been discussed by Jardine and Heinz (2016) and Jardine and McMullin (2017), while online learners for this class and the remaining single-tier-based hierarchy were proposed by Lambert (2021).

8 CONCLUSION

This paper showed how the nature of phonological typology emerges from simple representations and inference strategies. We discussed the nature of these representations in model-theoretic terms, forming a general notion of structural information (factors) that characterizes virtually any linguistic representation, from strings, to trees, to graphs. We also discussed a series of learning algorithms that work over any form of these factors, and are organised into a hierarchy of space complexity based on the distinctions they make with respect to structural information. We then derived the full hierarchical range of subregular formal language classes from the product of these different representations and inference strategies. Consideration of time complexity further parameterises this hierarchy, drawing equivalences and distinctions amongst the classes with respect to learning. We find that the scope of phonological typology is strongly biased into the range defined by the simplest learning algorithms and representations.

The relevance of these results for linguistic theory is clear. A learner, faced with dramatically sparse data, favours grammar induction strategies that limit the amount of necessary distinctions between structural forms in order to ensure that learning is possible and feasible. The requirement for learners to structure and limit their hypothesis spaces plays off the distinctions learners make and the representations they make them over. The results here, as well as typological and experimental evidence, suggest that a learner may fix a learning algorithm and allow representational primitives to vary. From this perspective, the requirement of parsing from a linguistic input to a particular linguistic form is of the utmost importance. Linguistic learning can be relativized over various representations, be they strings or graphs for phonology, or trees for syntax. In this way, natural language typology, considered through an algorithmic lens, can be shown to emerge from the interaction of simple learning algorithms and simple but wide-ranging notions of representation.

APPENDIX 9

This appendix offers a brief reference to the fifteen classes characterized by combinations of the learning algorithms and model signatures described in this text. Each class is accompanied by a sample of attested patterns that it can account for, with those accessible to a lower algorithm having backreferences. Each pattern is provided with a grammar given as a *plebby*-style expression[3] formatted in a way typical of the class.

[3] The Piecewise-Local Expression Builder Interpreter (*plebby*) is one component of the Language Toolkit, available from `https://github.com/vvulpes0/Language-Toolkit-2`, currently version 0.3.

9.1 *Expression syntax*

A complete formal description of the *plebby* expression language is available in the package documentation. An abridged summary follows here.

Expressions are built on factors, represented by sequences between angle brackets. For example $\langle \mathrm{a}\ \mathrm{b}, \mathrm{c}\ \mathrm{d}\rangle$ asserts the occurrence of four positions, say 1, 2, 3, and 4, that are respectively labelled by symbols in sets a, b, c, and d, where positions 1 and 2 are connected by the successor relation, as are positions 3 and 4, while positions 2 and 3 are connected by the general precedence relation. If a left (right) boundary symbol is prefixed to this notation, that means the leftmost (rightmost) position aligns with the left (right) edge of the word. For instance, $\rtimes\ltimes\langle \mathrm{a}\rangle$ asserts that all words consist of a single position labelled by an element of the symbolset a. Assignment of names to symbolsets is not discussed here.

More complex expressions are built from unary ($\otimes e$) or n-ary ($\otimes\{e_1, e_2, \ldots, e_n\}$) operations, where $\otimes$ is the operator and the various e are expressions. The Boolean 'and' ($\cap$) and 'or' ($\cup$) operations are n-ary and represent language intersection and union, respectively. The other n-ary operation is concatenation ($\bullet$). Complement ($\neg$) and projection ($[s_1, s_2, \ldots, s_n]$) are unary operations, where the projection operation asserts that the subexpression it operates over applies after a word has been projected to include only symbols in the union of symbolsets s_1 through s_n.

9.2 *Algorithm I*

Words must contain at least one element of some finite set of factors.

9.2.1 Complement Strictly Piecewise

Factors are subsequences.

- Minimum word length: $\cup\{\langle \overset{*}{\sigma}, \overset{*}{\sigma}\rangle\}$.
  Two syllables. More or fewer by adding or removing $\overset{*}{\sigma}$.
- Stress obligatoriness (Hyman 2009): $\cup\{\langle \acute{\sigma}\rangle\}$.

Complement Strictly Local 9.2.2

Factors are substrings.

- Minimum word length: $\cup\{\langle \overset{*}{\sigma}\ \overset{*}{\sigma} \rangle\}$.
  Two syllables. More or fewer by adding or removing $\overset{*}{\sigma}$.
- Stress obligatoriness: $\cup\{\langle \acute{\sigma} \rangle\}$.

Complement Tier-Based Strictly Local 9.2.3

Factors are substrings after projection to some subset of the alphabet.

- Anything complement strictly local.

Complement Multi-Tier Strictly Local 9.2.4

All words must satisfy at least one of a set of complement tier-based strictly local grammars.

- Anything complement tier-based strictly local.

Algorithm II 9.3

No word may contain any of a finite set of factors.

Strictly Piecewise 9.3.1

Factors are subsequences.

- Harmony, unblocked (Heinz 2010a): $\neg\cup\{\langle \mathrm{s}, \int \rangle, \langle \int, \mathrm{s} \rangle\}$.
  Symmetric. Asymmetric if only one factor were included.
- Stress culminativity (Hyman 2009): $\neg\cup\{\langle \acute{\sigma}, \acute{\sigma} \rangle\}$.
- Tone Plateauing (Hyman and Katamba 1993): $\neg\cup\{\langle \mathrm{H}, \mathrm{L}, \mathrm{H} \rangle\}$.

Strictly Local 9.3.2

Factors are substrings.

- AB alternation: $\neg\cup\{\langle \mathrm{A}\ \mathrm{A} \rangle, \langle \mathrm{B}\ \mathrm{B} \rangle\}$.
- Cambodian stress (Lambert and Rogers 2019):
  $\neg\cup\{\ltimes\langle \bar{\sigma} \rangle, \rtimes\ltimes\langle\rangle, \langle \acute{\sigma}\ \overset{*}{\sigma} \rangle, \langle \mathrm{H} \rangle, \langle \overset{*}{\mathrm{L}}\ \overset{*}{\mathrm{L}} \rangle . \rtimes\langle \overset{*}{\mathrm{L}} \rangle\}$.
- No light monosyllables (Lambert and Rogers 2019): $\neg\cup\{\rtimes\ltimes\langle \overset{*}{\mathrm{L}} \rangle\}$.

9.3.3 Tier-Based Strictly Local

Factors are substrings after projection to some subset of the alphabet.

- Anything strictly local.
- Dissimilation (Heinz *et al.* 2011): $[\mathrm{k},\mathrm{l},\mathrm{r}]\neg\cup\{\langle \mathrm{l}\ \mathrm{l}\rangle, \langle \mathrm{r}\ \mathrm{r}\rangle\}$.
- Harmony (Heinz 2010a): $[\mathrm{s},\int]\neg\cup\{\langle \mathrm{s}\ \int\rangle, \langle \int\ \mathrm{s}\rangle\}$.
  Unblocked. Blocked if other symbols project.
- Stress culminativity: $[\acute{\sigma}]\neg\cup\{\langle\acute{\sigma}\ \acute{\sigma}\rangle\}$.

9.3.4 Multi-Tier Strictly Local

Words must satisfy each member of a set of tier-based strictly local grammars.

- Anything tier-based strictly local.
- Bukusu harmony (Aksënova *et al.* 2020):
  $\cap\{[\mathrm{vowel}]\neg\cup\{\langle\mathrm{hi\ lo}\rangle, \langle\mathrm{lo\ hi}\rangle\}, [\mathrm{l},\mathrm{r}]\neg\cup\{\langle\mathrm{r\ l}\rangle\}\}$.

9.4 *Algorithm III*

The set of factors in a word must be a member of some finite set of factorsets.

9.4.1 Piecewise Testable

Factors are subsequences.

- Anything (complement) strictly piecewise.
- No light monosyllables: $\cup\{\neg\langle\overset{*}{\mathrm{L}}\rangle, \langle\overset{*}{\sigma}, \overset{*}{\sigma}\rangle\}$.
  See also strictly local, Algorithm II.

9.4.2 Locally Testable

Factors are substrings.

- Anything (complement) strictly local.
- Harmony, unblocked, symmetric: $\neg\cap\{\langle\mathrm{s}\rangle, \langle\int\rangle\}$.
  See also strictly piecewise, Algorithm II.

Tier-Based Locally Testable 9.4.3

Factors are substrings after projection to some subset of the alphabet.

- Anything (complement) tier-based strictly local.
- Anything locally testable.

Multi-Tier Locally Testable 9.4.4

Words must satisfy a Boolean network of tier-based locally testable grammars.

- Anything (complement) multi-tier strictly local.
- Anything tier-based locally testable.

Algorithm IV 9.5

The multiset of factors in a word must be a member of some finite set of multisets of factors. Note that while *plebby* has no intrinsic notion of multisets, concatenation can be used as in the expression •{⟨a b⟩, ⟨a b⟩} which asserts that ⟨a b⟩ occurs at least twice.

Locally Threshold Testable 9.5.1

Factors are substrings.

- Anything locally testable.
- Stress culminativity: ¬∪{•{⟨σ́⟩, ⟨σ́⟩}}.
  See also strictly piecewise, Algorithm II.

Tier-Based Locally Threshold Testable 9.5.2

Factors are substrings after projection to some subset of the alphabet.

- Anything tier-based locally testable.
- Anything locally threshold testable.
- Tone plateauing: [H, L]¬∪{•{⟨H L⟩, ⟨H L⟩}, ∩{⟨H L⟩, ⋉⟨H⟩}}.
  See also strictly piecewise, Algorithm II.

9.5.3 Multi-Tier Locally Threshold Testable

Words must satisfy a Boolean network of tier-based locally threshold testable grammars.

- Anything multi-tier locally testable.
- Anything tier-based locally threshold testable.

REFERENCES

Alëna Aksënova, Jonathan Rawski, Thomas Graf, and Jeffrey Heinz (2020), The Computational Power of Harmony, in Harry van der Hulst, editor, *Oxford Handbook of Vowel Harmony*, Oxford University Press, under review.

Martin Anthony and Norman Biggs (1992), *Computational Learning Theory*, Cambridge University Press.

Richard Brian Applegate (1972), *Ineseño Chumash Grammar*, Ph.D. thesis, University of California, Berkeley.

Frederico Augusto Casarsa Azevedo, Ludmila Ribeiro Bezerra Carvalho, Lea Tenenholz Grinberg, José Marcelo Farfel, Renata Eloah de Lucena Ferretti, Renata Elaine Paraizo Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel (2009), Equal Numbers of Neuronal and Nonneuronal Cells Make the Human Brain an Isometrically Scaled-Up Primate Brain, *The Journal of Comparative Neurology*, 513:532–541, doi:10.1002/cne.21974.

Danièle Beauquier and Jean-Éric Pin (1989), Factors of Words, in Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simonetta Ronchi Della Rocca, editors, *Automata, Languages and Programming: 16th International Colloquium*, volume 372 of *Lecture Notes in Computer Science*, pp. 63–79, Springer Berlin / Heidelberg, doi:10.1007/BFb0035752.

Julius Richard Büchi (1960), Weak Second-Order Arithmetic and Finite Automata, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, doi:10.1002/malq.19600060105.

Pascal Caron (1998), LANGAGE: A Maple Package for Automaton Characterization of Regular Languages, in Derick Wood and Sheng Yu, editors, *Automata Implementation*, volume 1436 of *Lecture Notes in Computer Science*, pp. 46–55, Springer Berlin / Heidelberg, doi:10.1007/BFb0031380.

Pascal Caron (2000), Families of Locally Testable Languages, *Theoretical Computer Science*, 242(1–2):361–376, doi:10.1016/S0304-3975(98)00332-6.

Jane CHANDLEE (2014), *Strictly Local Phonological Processes*, Ph.D. thesis, University of Delaware, `https://chandlee.sites.haverford.edu/wp-content/uploads/2015/05/Chandlee_dissertation_2014.pdf`.

Jane CHANDLEE (2017), Computational Locality in Morphological Maps, *Morphology*, 27(4):599–641, doi:10.1007/s11525-017-9316-9.

Jane CHANDLEE, Rémi 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*, pp. 91–101, Association for Computational Linguistics, doi:10.18653/v1/W19-5708.

Jane CHANDLEE and Adam JARDINE (2019), Autosegmental Input Strictly Local Functions, *Transactions of the Association for Computational Linguistics*, 7:157–168, doi:10.1162/tacl_a_00260.

Alexander CLARK and Shalom LAPPIN (2011), *Linguistic Nativism and the Poverty of the Stimulus*, Wiley-Blackwell.

Bruno COURCELLE (1994), Monadic Second-Order Definable Graph Transductions: A Survey, *Theoretical Computer Science*, 126(1):53–75, doi:10.1016/0304-3975(94)90268-2.

Bruno COURCELLE and Joost ENGELFRIET (2012), *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, volume 138, Cambridge University Press.

Colin DE LA HIGUERA (2010), *Grammatical Inference: Learning Automata and Grammars*, Cambridge University Press, doi:10.1017/CBO9781139194655.

Aniello DE SANTO and Thomas GRAF (2019), Structure Sensitive Tier Projection: Applications and Formal Properties, in Raffaella BERNARDI, Greg KOBELE, and Sylvain POGODALLA, editors, *Formal Grammar 2019*, volume 11668 of *Lecture Notes in Computer Science*, pp. 35–50, Springer Verlag, doi:10.1007/978-3-662-59648-7_3.

Hossep DOLATIAN and Jonathan RAWSKI (2020), Multi-Input Strictly Local Functions for Templatic Morphology, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 282–296, `https://scholarworks.umass.edu/scil/vol3/iss1/28`.

Matt EDLEFSEN, Dylan LEEMAN, Nathan MYERS, Nathaniel SMITH, Molly VISSCHER, and David WELLCOME (2008), Deciding Strictly Local (SL) Languages, in Jon BREITENBUCHER, editor, *Proceedings of the 2008 Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pp. 66–73.

Joost ENGELFRIET and Hendrik Jan HOOGEBOOM (2001), MSO Definable String Transductions and Two-Way Finite-State Transducers, *ACM Transactions on Computational Logic*, 2(2):216–254, doi:10.1145/371316.371512.

Emmanuel FILIOT (2015), Logic-Automata Connections for Transformations, in *Logic and Its Applications*, volume 8923 of *Lecture Notes in Computer Science*, pp. 30–57, Springer Berlin / Heidelberg, doi:10.1007/978-3-662-45824-2_3.

Sara FINLEY (2008), *Formal and Cognitive Restrictions on Vowel Harmony*, Ph.D. thesis, Johns Hopkins University.

Jerry Alan FODOR and Zenon Walter PYLYSHYN (1988), Connectionism and Cognitive Architecture: A Critical Analysis, *Cognition*, 28(1–2):3–71, doi:10.1016/0010-0277(88)90031-5.

Pedro GARCIA, Enrique VIDAL, and José ONCINA (1990), Learning Locally Testable Languages in the Strict Sense, in *Proceedings of the 1st International Workshop on Algorithmic Learning Theory*, pp. 325–338, `https://grfia.dlsi.ua.es/repositori/grfia/pubs/111/alt1990.pdf`.

R. W. N. GOEDEMANS, Jeffrey HEINZ, and Harry VAN DER HULST (2015), StressTyp2, `http://st2.ullet.net/`.

Edward Mark GOLD (1967), Language Identification in the Limit, *Information and Control*, 10(5):447–474, doi:10.1016/S0019-9958(67)91165-5.

John Anton GOLDSMITH (1976), *Autosegmental Phonology*, Ph.D. thesis, Swarthmore College, `https://dspace.mit.edu/bitstream/handle/1721.1/16388/03188555-MIT`.

Saul GORN (1967), Explicit Definitions and Linguistic Dominoes, in John HART and Satoru TAKASU, editors, *Systems and Computer Science*, pp. 77–115, University of Toronto Press, doi:10.3138/9781487592769.

Thomas GRAF (2011), Closure Properties of the Minimalist Derivation Tree Languages, in Sylvain POGODALLA and Jean-Philippe PROST, editors, *Logical Aspects of Computational Linguistics*, volume 6736 of *Lecture Notes in Computer Science*, pp. 96–111, Springer Berlin / Heidelberg, doi:10.1007/978-3-642-22221-4_7.

Thomas GRAF (2014), Beyond the Apparent: Cognitive Parallels Between Syntax and Phonology, in Carson T. SCHÜTZE and Linnaea STOCKALL, editors, *Connectedness: Papers by and for Sarah VanWagenen*, volume 18 of *UCLA Working Papers in Linguistics*, pp. 161–174, `http://phonetics.linguistics.ucla.edu/wpl/issues/wpl18/papers/graf.pdf`.

Thomas GRAF (2020), Curbing Feature Coding: Strictly Local Feature Assigment, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 362–371, doi:10.7275/f7y5-xz32.

Thomas GRAF and Nazila SHAFIEI (2019), C-Command Dependencies as TSL String Constraints, in *Proceedings of the Society for Computation in Linguistics*, volume 2, pp. 205–215, doi:10.7275/4rrx-x488.

Leonard H. HAINES (1969), On Free Monoids Partially Ordered by Embedding, *Journal of Combinatorial Theory*, 6(1):94–98, doi:10.1016/s0021-9800(69)80111-0.

Jeffrey HEINZ (2010a), Learning Long-Distance Phonotactics, *Linguistic Inquiry*, 41(4):623–661, doi:10.1162/ling_a_00015.

Jeffrey HEINZ (2010b), String Extension Learning, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 897–906, Association for Computational Linguistics, `https://www.aclweb.org/anthology/P10-1092`.

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*, chapter 27, pp. 633–663, Oxford University Press, doi:10.1093/oxfordhb/9780199601264.013.27.

Jeffrey HEINZ (2018), The Computational Nature of Phonological Generalizations, in Larry HYMAN and Frank PLANK, editors, *Phonological Typology*, volume 23 of *Phonetics and Phonology*, chapter 5, pp. 126–195, Mouton de Gruyter, doi:10.1515/9783110451931-005.

Jeffrey HEINZ, Anna KASPRZIK, and Timo KÖTZING (2012), Learning in the Limit with Lattice-Structured Hypothesis Spaces, *Theoretical Computer Science*, 457:111–127, doi:10.1016/j.tcs.2012.07.017.

Jeffrey HEINZ, Chetan RAWAL, and Herbert G. TANNER (2011), Tier-based Strictly Local Constraints for Phonology, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Short Papers*, volume 2, pp. 58–64, Association for Computational Linguistics, `https://aclweb.org/anthology/P11-2011`.

Jeffrey HEINZ and James ROGERS (2013), Learning Subregular Classes of Languages with Factored Deterministic Automata, in *Proceedings of the 13th Meeting on the Mathematics of Language*, pp. 64–71, Association for Computational Linguistics, `https://www.aclweb.org/anthology/W13-3007`.

Suzana HERCULANO-HOUZEL and Roberto LENT (2005), Isotropic Fractionator: A Simple, Rapid Method for the Quantification of Total Cell and Neuron Numbers in the Brain, *Journal of Neuroscience*, 25(10):2518–2521, doi:10.1523/JNEUROSCI.4526-04.2005.

Larry M. HYMAN (2009), How (Not) to Do Phonological Typology: The Case of Pitch-Accent, *Language Sciences*, 31(2–3):213–238, doi:10.1016/j.langsci.2008.12.007.

Larry M. HYMAN and Francis X. KATAMBA (1993), A New Approach to Tone in Luganda, *Language*, 69(1):34–67, doi:10.2307/416415.

Sanjay JAIN, Daniel OSHERSON, James S. ROYER, and Arun SHARMA (1999), *Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change)*, The MIT Press, 2nd edition.

Adam JARDINE (2017a), The Local Nature of Tone-Association Patterns, *Phonology*, 34(2):385–405, doi:10.1017/s0952675717000185.

Adam JARDINE (2017b), On the Logical Complexity of Autosegmental Representations, in *Proceedings of the 15th Meeting on the Mathematics of Language*, pp. 22–35, Association for Computational Linguistics, doi:10.18653/v1/W17-3403.

Adam JARDINE (2019), The Expressivity of Autosegmental Grammars, *Journal of Logic, Language and Information*, 28(1):9–54, ISSN 1572-9583, doi:10.1007/s10849-018-9270-x.

Adam JARDINE, Nick DANIS, and Luca IACOPONI (2021), A Formal Investigation of Q-Theory in Comparison to Autosegmental Representations, *Linguistic Inquiry*, 52(2):333–358, doi:10.1162/ling_a_00376.

Adam JARDINE and Jeffrey HEINZ (2016), Learning Tier-Based Strictly 2-Local Languages, *Transactions of the Association for Computation in Linguistics*, 4:87–98, doi:10.1162/tacl_a_00085.

Adam JARDINE and Kevin MCMULLIN (2017), Efficient Learning of Tier-Based Strictly k-Local Languages, in Frank DREWES, Carlos MARTÍN-VIDE, and Bianca TRUTHE, editors, *Language and Automata Theory and Applications: 11th International Conference*, volume 10168 of *Lecture Notes in Computer Science*, pp. 64–76, Springer, Cham, doi:10.1007/978-3-319-53733-7_4.

Kent JOHNSON (2004), Gold's Theorem and Cognitive Science, *Philosophy of Science*, 71:571–592.

Mark JOHNSON (1988), *Attribute-Value Logic and the Theory of Grammar*, Center for the Study of Language and Information.

Michael KEARNS and Umesh VAZIRANI (1994), *An Introduction to Computational Learning Theory*, MIT Press.

Paul J. KING (1989), *A Logical Formalism for Head-Driven Phrase Structure Grammar*, Ph.D. thesis, University of Manchester, `https://www.proquest.com/docview/2201235827`.

Gregory M. KOBELE (2011), Minimalist Tree Languages are Closed Under Intersection with Recognizable Tree Languages, in Sylvain POGODALLA and Jean-Philippe PROST, editors, *Logical Aspects of Computational Linguistics*, volume 6736 of *Lecture Notes in Computer Science*, pp. 129–144, Springer Berlin / Heidelberg, doi:10.1007/978-3-642-22221-4_9.

John Richard KRUEGER (1961), *Chuvash Manual: Introduction, Grammar, Reader, and Vocabulary*, volume 7 of *Uralic and Altaic Series*, Indiana University.

Regine LAI (2015), Learnable vs. Unlearnable Harmony Patterns, *Linguistic Inquiry*, 46(3):425–451, doi:10.1162/LING_a_00188.

Dakotah LAMBERT (2021), Grammar Interpretations and Learning TSL Online, in Adam JARDINE, Jane CHANDLEE, Jeffrey HEINZ, Menno VAN ZAANEN, and Rémi EYRAUD, editors, *Proceeedings of the Fifteenth International Conference on Grammatical Inference (ICGI 2020/21)*, PMLR: Proceedings of Machine Language Research, `http://proceedings.mlr.press/`, in press.

Dakotah LAMBERT and James ROGERS (2019), A Logical and Computational Methodology for Exploring Systems of Phonotactic Constraints, in *Proceedings of the Society for Computation in Linguistics*, volume 2, pp. 247–256, doi:10.7275/t0dv-9t05.

Dakotah LAMBERT and James ROGERS (2020), Tier-Based Strictly Local Stringsets: Perspectives from Model and Automata Theory, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 330–337, doi:10.7275/2n1j-pj39.

Shalom LAPPIN and Stuart Merrill SHIEBER (2007), Machine Learning Theory and Practice as a Source of Insight into Universal Grammar, *Journal of Linguistics*, 43(2):393–427, doi:10.1017/S0022226707004628.

Julie Anne LEGATE and Charles D. YANG (2002), Empirical Re-assessment of Stimulus Poverty Arguments, *The Linguistic Review*, 18(1–2):151–162, doi:10.1515/tlir.19.1-2.151.

Leonid LIBKIN (2004), *Elements of Finite Model Theory*, Texts in Theoretical Computer Science, Springer Berlin / Heidelberg, doi:10.1007/978-3-662-07003-1.

Kevin MCMULLIN and Gunnar Ólafur HANSSON (2019), Inductive Learning of Locality Relations in Segmental Phonology, *Laboratory Phonology: Journal of the Association for Laboratory Phonology*, 10(1):1–53, doi:10.5334/labphon.150, article 14.

Robert MCNAUGHTON and Seymour A. PAPERT (1971), *Counter-Free Automata*, MIT Press.

Tom Michael MITCHELL (1982), Generalization as Search, *Artificial Intelligence*, 18(2):203–226, doi:10.1016/0004-3702(82)90040-6.

Tom Michael MITCHELL (2017), Key Ideas in Machine Learning, in *Machine Learning: Second Edition*, McGraw Hill, `http://www.cs.cmu.edu/~tom/mlbook/keyIdeas.pdf`, forthcoming.

Meyhar MOHRI, Afshin ROSTAMIZADEH, and Ameet TALWALKAR (2012), *Foundations of Machine Learning*, MIT Press.

Max NELSON, Hossep DOLATIAN, Jonathan RAWSKI, and Brandon PRICKETT (2020), Probing RNN Encoder-Decoder Generalizations of Subregular Functions Using Reduplication, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 31–42, doi:10.7275/xd0r-pg04.

Partha NIYOGI (2006), *The Computational Nature of Language Learning and Evolution*, Cambridge, MA: MIT Press.

Martin Andreas NOWAK, Natalia L. KOMAROVA, and Partha NIYOGI (2002), Computational and Evolutionary Aspects of Language, *Nature*, 417(6889):611–617, doi:10.1038/nature00771.

Chris OAKDEN (2020), Notational Equivalence in Tonal Geometry, *Phonology*, 37(2):257–296, doi:10.1017/S0952675720000123.

Christopher POTTS and Geoffrey PULLUM (2002), Model Theory and the Content of OT Constraints, *Phonology*, 19:361–393.

Jonathan RAWSKI (2019), Tensor Product Representations of Subregular Formal Languages, in *Proceedings of the International Joint Conference on Artificial Intelligence workshop on Neural-Symbolic Learning and Reasoning*, pp. 36–42.

Jonathan RAWSKI and Hossep DOLATIAN (2020), Multi-Input Strictly Local Functions for Tonal Phonology, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 245–260, `https://scholarworks.umass.edu/scil/vol3/iss1/25`.

Jonathan RAWSKI and Jeffrey HEINZ (2019), No Free Lunch in Linguistics or Machine Learning: Response to Pater, *Language*, 93(1):e125–e135, doi:10.1353/lan.2019.0004.

James ROGERS (1998), *A Descriptive Approach to Language-Theoretic Complexity*, (Monograph.) Studies in Logic, Language, and Information, CSLI Publications.

James ROGERS (2003), Syntactic Structures as Multi-Dimensional Trees, *Research on Language and Computation*, 1(3–4):265–305, doi:10.1023/A:1024695608419.

James ROGERS, Jeff HEINZ, Margaret FERO, Jeremy HURST, Dakotah LAMBERT, and Sean WIBEL (2012), Cognitive and Sub-Regular Complexity, in Glyn MORRILL and Mark-Jan NEDERHOF, editors, *Formal Grammar 2012*, volume 8036 of *Lecture Notes in Computer Science*, pp. 90–108, Springer-Verlag, doi:10.1007/978-3-642-39998-5_6.

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 Christian EBERT, Gerhard JÄGER, and Jens MICHAELIS, editors, *The Mathematics of Language: Revised Selected Papers from the 10th and 11th Biennial Conference on the Mathematics of Language*, volume 6149 of *LNCS/LNAI*, pp. 255–265, FoLLI/Springer, doi:10.1007/978-3-642-14322-9_19.

James ROGERS and Dakotah LAMBERT (2019a), Extracting Subregular Constraints from Regular Stringsets, *Journal of Language Modelling*, 7(2):143–176, doi:10.15398/jlm.v7i2.209.

James ROGERS and Dakotah LAMBERT (2019b), Some Classes of Sets of Structures Definable Without Quantifiers, in *Proceedings of the 16th Meeting on the Mathematics of Language*, pp. 63–77, Association for Computational Linguistics, doi:10.18653/v1/W19-5706.

Marcel-Paul SCHÜTZENBERGER (1965), On Finite Monoids Having Only Trivial Subgroups, *Information and Control*, 8(2):190–194, doi:10.1016/s0019-9958(65)90108-7.

Imre SIMON (1975), Piecewise Testable Events, in Helmut BRAKHAGE, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pp. 214–222, Springer-Verlag, doi:10.1007/3-540-07407-4_23.

Kristina STROTHER-GARCIA (2019), *Using Model Theory in Phonology: A Novel Characterization of Syllable Structure and Syllabification*, Ph.D. thesis, University of Delaware.

Wolfgang THOMAS (1982), Classifying Regular Events in Symbolic Logic, *Journal of Computer and Systems Sciences*, 25:360–376, doi:10.1016/0022-0000(82)90016-2.

Leslie Gabriel VALIANT (1984), A Theory of the Learnable, *Communications of the ACM*, 27(11):1134–1142, doi:10.1145/1968.1972.

Iris VAN ROOIJ and Giosuè BAGGIO (2021), Theory Before the Test: How to Build High-Verisimilitude Explanatory Theories in Psychological Science, *Perspectives on Psychological Science*, doi:10.1177/1745691620970604.

Vladimir VAPNIK (1995), *The Nature of Statistical Learning Theory*, Springer.

Mai Ha 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):1–26, doi:10.3389/frobt.2018.00076.

Edwin Samuel WILLIAMS (1976), Underlying Tone in Margi and Igbo, *Linguistic Inquiry*, 7(3):463–484.

Charles YANG (2013), Who's Afraid of George Kingsley Zipf? Or: Do Children and Chimps Have Language?, *Significance*, 10(6):29–34, doi:10.1111/j.1740-9713.2013.00708.x.

Dakotah Lambert
0000-0002-7056-5950
dakotah.lambert@stonybrook.edu

Jeffrey Heinz
0000-0002-5954-3195
jeffrey.heinz@stonybrook.edu

Department of Linguistics
and Institute for Advanced
Computational Science,
Stony Brook University

Jonathan Rawski
0000-0003-3996-9815
jon.rawski@sjsu.edu

Department of Linguistics
and Language Development,
San José State University

Dakotah Lambert, Jonathan Rawski, and Jeffrey Heinz (2021), *Typology emerges from simplicity in representations and learning*, Journal of Language Modelling, 9(1):151–194
https://dx.doi.org/10.15398/jlm.v9i1.262