Learning with Partially Ordered Representations

Jane Chandlee
Tri-Co Department of Linguistics
Haverford College
jchandlee@haverford.edu

Rémi Eyraud
QARMA team, LIS
Aix-Marseille University
remi.eyraud@lis-lab.fr

Jeffrey Heinz
Department of Linguistics
Institute for Advanced Computational Science
Stony Brook University
jeffrey.heinz@stonybrook.edu

Adam Jardine
Department of Linguistics
Rutgers University
adam.jardine@rutgers.edu

Jonathan Rawski
Department of Linguistics
Institute for Advanced Computational Science
Stony Brook University
jonathan.rawski@stonybrook.edu

Abstract

This paper examines the characterization and learning of grammars defined with enriched representational models. Model-theoretic approaches to formal language theory traditionally assume that each position in a string belongs to exactly one unary relation. We consider unconventional string models where positions can have multiple, shared properties, which are arguably useful in many applications. We show the structures given by these models are partially ordered, and present a learning algorithm that exploits this ordering relation to effectively prune the hypothesis space. We prove this learning algorithm, which takes positive examples as input, finds the most general grammar which covers the data.

1 Introduction

Foundational connections between formal languages, finite-state automata, and logic have been known for decades (Büchi, 1960; Thomas, 1997). Logical approaches are advantageous since they flexibly admit different representations. In many domains, such as biological sequencing or linguistics, shared properties of symbols in sequences provide information currently ignored by string-based inference algorithms, which largely focus on learning automata (de la Higuera, 2010). Here we explore the idea that domain-specific knowledge can be encoded representationally via model theory (Libkin, 2004), and shows how these representations can facilitate pattern learning.

This paper synthesizes results in grammatical inference and model theory to present a novel algorithm which learns classes of formal languages using enriched representations of strings. In fact, our model-theoretic approach immediately generalizes these results to arbitrary data structures. Here we are concerned with the learning of those formal languages which can be defined via a set of structural constraints, such as the Strictly k-Local and Strictly k-Piecewise languages (Rogers and Pullum, 2011; Rogers et al., 2010). Models of strings in the languages must not contain these forbidden structures (Rogers et al., 2013). Specifically, we define a learner whose hypothesis space is structured as a partial order by the relational signature of the particular model theory. We show how to traverse this space bottom-up from positive data to find a grammar which covers the data with the most general constraints.

The paper is structured as follows: Section 2 provides mathematical preliminaries in model theory. Section 3 characterizes ordering relations over these structures. Section 4 generalizes the grammars employed in string extension and lattice-based learning (Heinz, 2010; Heinz et al., 2012) to show how these model theoretic structures can define classes of formal languages. Section 5 discusses some entailments our learning al-
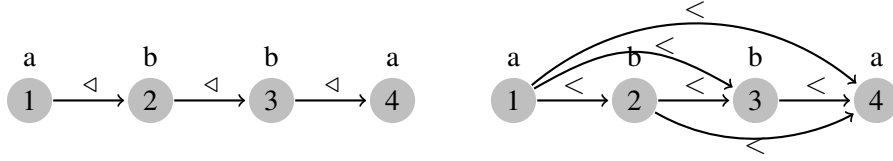
Figure 1: Visualizations of the successor (left) and precedence (right) models of *abba*.

gorithm takes advantage of. Section 6 defines a learning problem and criteria for selecting adequate solutions. Section 7 presents a general-to-specific, bottom-up algorithm which provably satisfies the learning criteria. Section 8 concludes the paper.

2 Preliminaries

2.1 Elements of Language Theory

The set of all possible finite strings of symbols from a finite alphabet Σ and the set of strings of length $\leq n$ are Σ^* and $\Sigma^{\leq n}$, respectively. The unique empty string is represented with λ. The length of a string w is $|w|$, so $|\lambda| = 0$. If u and v are two strings then we denote their concatenation with uv. If w is a string and σ is the ith symbol in w then $w_i = \sigma$, so $abcd_2 = b$.

The set of prefixes of w, $\texttt{Pref}(w)$, is $\{p \in \Sigma^* \mid (\exists s \in \Sigma^*)[w = ps]\}$, the set of suffixes of w, $\texttt{Suff}(w)$, is $\{s \in \Sigma^* \mid (\exists p \in \Sigma^*)[w = ps]\}$, the set of substrings, $\texttt{Substr}(w)$, is $\{u \in \Sigma^* \mid (\exists l, r \in \Sigma^*)[w = lur]\}$, and the set of subsequences, $\texttt{Subseq}(w) = u_1 u_2 \cdots u_n \mid \exists v_0 \cdot v_1 \cdots v_n \in \Sigma^*[w = v_0 u_1 v_1 \cdots u_n v_n]$

2.2 Elements of Finite Model Theory

Model theory, combined with logic, provides a powerful way to study and understand mathematical objects with structures (Enderton, 2001). In this paper we only consider finite relational models (Libkin, 2004) of strings in Σ^*.

Definition 1 (Models). *A* model signature *is a tuple $S = \langle D; R_1, R_2, \ldots, R_m \rangle$ where the domain D is a finite set, and each R_i is a n_i-ary relation over the domain. A* model *for a set of objects Ω is a total, one-to-one function from Ω to structures whose type is given by a model signature.*

For example, a conventional model for strings in Σ^* is given by the signature $\Gamma^{\lhd} \overset{def}{=} \langle D; \lhd, [R_\sigma]_{\sigma \in \Sigma} \rangle$ and the function $M^{\lhd}$: $\Sigma^* \to \Gamma^{\lhd}$ such that $M^{\lhd}(w) \overset{def}{=} \langle D^w; \lhd, [R_\sigma^w]_{\sigma \in \Sigma} \rangle$ where $D^w \overset{def}{=} \{1, \ldots, |w|\}$ is the domain,

$\lhd \overset{def}{=} \{(i, i+1) \in D \times D\}$ is the successor relation which orders the elements of the domain, and $[R_\sigma^w]_{\sigma \in \Sigma}$ is a set of $|\Sigma|$ unary relations such that for each $\sigma \in \Sigma$, $R_\sigma^w \overset{def}{=} \{i \in D^w \mid w_i = \sigma\}$. We will usually omit the superscript w since it will be clear from the context.

For example, with $\Sigma = \{a, b, c\}$ and the model above for strings, we have $M^{\lhd}(abba) = \langle D = \{1, 2, 3, 4\}; \lhd = \{(1, 2), (2, 3), (3, 4)\}, R_a = \{1, 4\}, R_b = \{2, 3\}, R_c = \emptyset \rangle$.

Figure 1 illustrates $M^{\lhd}(abba)$ on the left.

Another conventional model is the precedence model, with the signature $\Gamma^< \overset{def}{=} \langle D; <, [R_\sigma]_{\sigma \in \Sigma} \rangle$. It differs from the successor model only in that the order relation is defined with general precedence $< \overset{def}{=} \{(i, j) \in D \times D \mid i < j\}$ (Büchi, 1960; McNaughton and Papert, 1971; Rogers et al., 2013). Under this signature, the string *abba* has the following model.

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

Figure 1 illustrates $M^<(abba)$ on the right.

The model-theoretic framework is not unique to strings. It extends to arbitrary data structures by expanding parts of the model signature. For example, Rogers (2003) describes a model-theoretic characterization of trees of arbitrary dimensionality where the domain D is 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 $d = \alpha i$, where $i \in \mathbb{N}$, $\alpha \in \mathbb{N}^*$ it holds that $\alpha \in D$, and for every $d \in D$ with $d = \alpha i \neq \alpha 0$, then $\alpha(i-1) \in D$.

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 mdoel signature, the set of nodes is ordered by two binary relations, "dominance" and "immediate left-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. Standard or

two-dimensional trees, then, relate nodes to one-dimensional trees (strings) by immediate dominance. A three-dimensional tree relates nodes to two-dimensional, i.e. standard trees, corresponding to Tree-Adjoining Grammar derivations. In general, a d-dimensional tree is a set of nodes ordered by d dominance relations such that the n-th dominance relation relates nodes to $(n-1)$-dimensional trees (for $d = 1$, single nodes are zero-dimensional trees).

While a Gorn tree domain as written encodes these dominance and precedence relations implicitly, we may explicitly write them out model-theoretically so that a signature for a Σ-labeled 2-d tree is $\Gamma^{\lhd \prec} = \langle D; \lhd, \prec, [R_\sigma]_{\sigma \in \Sigma} \rangle$ where $\lhd$ is the "immediate dominance" relation and $\prec$ is the "immediate left-of" relation. Model signatures that include transitive closure relations of each of these have also been studied.
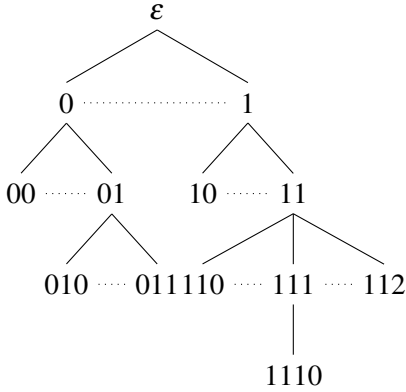

Figure 2: 2-dimensional tree model. Dominance and precedence relations shown with solid/dashed and dotted lines, respectively

2.3 Unconventional Word Models

Whereas Rogers (2003) generalized conventional word models to trees, here we generalize word models in a different way. Conventional string models are the successor and precedence models introduced previously. What makes these models conventional is the unary relations which essentially label each domain element with a single, mutually exclusive, property: the property of being some $\sigma \in \Sigma$.

In contrast, unconventional models for strings recognize that distinct alphabetic symbols may share properties, and expands the model signature by including these properties as unary relations (Strother-Garcia et al., 2016; Vu et al., 2018). For example, a conventional model of

$\Sigma = \{\mathtt{a}, \ldots, \mathtt{z}, \mathtt{A}, \ldots, \mathtt{Z}\}$ would include 52 unary relations, one for each lowercase and capital letter. On the other hand, an unconventional model might only include 27: 26 for the letters, and one unary relation $\mathtt{Capital}$. Then, letters $\mathtt{A}$ and $\mathtt{a}$ share the 'A' property and $\mathtt{A}$ additionally has the property of being $\mathtt{Capital}$.

In linguistics, speech sounds are commonly decomposed into binary features based on their phonetic properties. So the set of segments $\{\mathtt{z}, \mathtt{ʒ}, \mathtt{d}, \mathtt{b}, \mathtt{g}, \ldots\}$ all share the property $\mathtt{+Voice}$, meaning the vocal cords are activated, while the segments $\{\mathtt{s}, \mathtt{ʃ}, \mathtt{t}, \mathtt{p}, \mathtt{k}, \ldots\}$ share the property $\mathtt{-Voice}$, meaning the vocal cords are not activated. Thus unconventional models may refer to individual features in defining grammatical constraints, rather than each individual segment.

Different representations of strings and trees provide a unified perspective on well-known subclasses of the regular languages from a model-theoretic and logical perspective (Thomas, 1997; Rogers et al., 2013). However, they also open up new doors for grammatical inference by allowing one to consider other models for strings (Strother-Garcia et al., 2016; Vu et al., 2018).

3 Subfactors, Superfactors, Ideals and Filters

We sometimes refer to the model of a string w as a *structure*. However, structures are more general in that they correspond to any mathematical structure conforming to the model signature. As such, while a model of a string w will always be a structure, a structure will not always be a model of a string w. The *size* of a structure S, denoted $|S|$, coincides with the cardinality of its domain.

We next wish to introduce a partial ordering over structures. To do so, we must define the terms *connected, restriction*, and *factor*. For each structure $S = \langle D; \lhd, R_1, \ldots R_n \rangle$ let the binary "connectedness" relation C be defined as follows.

$C \overset{def}{=} \{(x,y) \in D \times D \mid \exists i \in \{1 \ldots n\}, \exists (x_1 \ldots x_m) \in R_i, \exists s,t \in \{1 \ldots m\}, x = x_s, y = x_t\}$

Informally, domain elements x and y belong to C provided they belong to some non-unary relation. Let C^* denote the symmetric transitive closure of C.

Definition 2 (Connected structure). *A structure $S = \langle D; \lhd, R_1, R_2, \ldots, R_n \rangle$ is connected iff for all $x, y \in D$, $(x, y) \in C^*$.*

For example, $M^\lhd(abba)$ above is a connected structure. However, the structure $S_{ab,\,ba}$ shown below which is identical to $M^\lhd(abba)$ except it omits the pair $(2,3)$ from the order relation is not connected since none of $(1,3),(1,4),(2,3)$ nor $(2,4)$ belong to C^*. $S_{ab,\,ba} = \langle D = \{1,2,3,4\}; \lhd = \{(1,2),(3,4)\}, R_a = \{1,4\}, R_b = \{2,3\}, R_c = \emptyset \rangle$

Note that no string in Σ^* has structure $S_{ab,\,ba}$ as its model.

Definition 3. $A = \langle D^A; \lhd, R_1^A, \dots, R_n^A \rangle$ is a restriction of $B = \langle D^B; \lhd, R_1^B, \dots, R_n^B \rangle$ iff $D^A \subseteq D^B$ and for each m-ary relation R_i, we have $R_i^A = \{(x_1 \dots x_m) \in R_i^B \mid x_1, \dots, x_m \in D^A\}$.

Informally, one identifies a subset A of the domain of B and strips B of all elements and relations which are not wholly within A. What is left is a restriction of B to A.

Definition 4. *Structure A is a* subfactor *of structure B ($A \sqsubseteq B$) if A is connected, there exists a restriction of B denoted B', and there exists $h : A \to B'$ such that for all $a_1, \dots a_m \in A$ and for all R_i in the model signature: if $h(a_1), \dots h(a_m) \in B'$ and $R_i(a_1, \dots a_m)$ holds in A then $R_i(h(a_1), \dots h(a_m))$ holds in B'. If $A \sqsubseteq B$ we also say that B is a* superfactor *of A.*

In other words, properties that hold of the connected structure A also hold in a related way within B.

If $A \sqsubseteq B$ and $|A| = k$ then we say A is a k-subfactor of B. For all $w \in \Sigma^*$, and for any model M of Σ^*, let the subfactors of w be $\texttt{Subfact}(M,w) = \{A \mid A \sqsubseteq M(w)\}$ and the k-subfactors of w be $\texttt{Subfact}_k(M,w) = \{A \mid A \sqsubseteq M(w), |A| \leq k\}$. We also define $\texttt{Subfact}(M,\Sigma^*)$ to be $\bigcup_{w \in \Sigma^*} \texttt{Subfact}(M,w)$ and $\texttt{Subfact}_k(M,\Sigma^*)$ to be $\bigcup_{w \in \Sigma^*} \texttt{Subfact}_k(M,w)$. When M is understood from context, we write $\texttt{Subfact}(w)$ instead of $\texttt{Subfact}(M,w)$. We define the sets of superfactors $\texttt{Supfact}(M,w)$ and $\texttt{Supfact}(M,\Sigma^*)$ similarly.

Observe that $(\texttt{Subfact}(M,w), \sqsubseteq)$ is a partially ordered set (poset). The next definition and lemma establishes that models of strings are principal elements of ideals and filters.

Definition 5 (Ideals). *A subset I of a poset is an Ideal if*

- *I is non-empty*

- *for every x in I, $y \leq x$ implies that y is in I*

- *for every x,y in I, there exists some element z in I, such that $x \leq z$ and $y \leq z$.*

The dual of an ideal is a filter.

Definition 6 (Filters). *A subset F of a poset is a filter iff*

- *F is non-empty*

- *for every x in F, $x \leq y$ implies that y is in F*

- *for every x,y in F, there exist some element z in F, such that $z \leq x$ and $z \leq y$.*

Definition 7 (Principal Ideals, Filters and Elements). *For any poset $\langle X, \leq \rangle$, the smallest filter containing $x \in X$ is a* principal filter *and x is the* principal element *of this filter. Similarly, the smallest ideal containing $x \in X$ is a* principal ideal *and x is the* principal element *of this ideal.*

Remark 1. *Given a model M of Σ^* and $k > 0$, $\texttt{Subfact}_k(M,w)$ is a principal ideal in $\texttt{Subfact}(M,\Sigma^*)$ whose principal element is $M(w)$. $\texttt{Supfact}_k(M,w)$ is a principal filter in $\texttt{Supfact}(M,\Sigma^*)$ whose principal element is $M(w)$. The empty structure $\langle \emptyset; \emptyset, \dots \emptyset \rangle$ is a subfactor of every structure in $\texttt{Subfact}(M,\Sigma^*)$.*

The next two propositions show how this representational perspective unifies the treatment of substrings and subsequences. They are subfactors under the successor and precedence models, respectively. A string $x = x_1 \cdots x_n$ is a substring of y iff there exists l, r such that $y = lxr$. String x is a subsequence of y iff there exists $v_0, v_1, \dots v_n$ such that $w = v_0 x_1 v_1 \cdots x_n v_n$.

Proposition 1 (Substrings are subfactors under $M^\lhd$). *For all strings $x, y \in \Sigma^*$, x is a substring of y iff $M^\lhd(x) \sqsubseteq M^\lhd(y)$.*

Proof. Note that the result trivially holds for $x = \lambda$: we restrict ourselves to the case $x \neq \lambda$. Let $M^\lhd(x) = \langle D^x; \lhd, [R_\sigma^x] \rangle$ and $M^\lhd(y) = \langle D^y; \lhd, [R_\sigma^y] \rangle$
($\Rightarrow$). Suppose x is a substring of y: it exists l, r such that $y = lxr = \sigma_1 \dots \sigma_{|l|} \sigma_{|l|+1} \dots \sigma_{|l|+|x|} \sigma_{|l|+|x|+1} \dots \sigma_{|l|+|x|+|r|}$. This implies that, for all i, $1 \leq i \leq |x|$, $d \in R_{\sigma_{|l|+i}}^y$ iff $d \in R_{\sigma_i}^x$. Thus, if we set the isomorphism ϕ to be such that $\phi(i) = |l|+i$ for $1 \leq i \leq |x|$, we have $\phi(M^\lhd(x))$ that is a restriction of $M^\lhd(y)$, and therefore $M^\lhd(x) \sqsubseteq M^\lhd(y)$ by definition.

($\Leftarrow$). Let y be the sequence of letters $\sigma_1 \ldots \sigma_{|y|}$ and suppose $M^\lhd(x) \sqsubseteq M^\lhd(y)$: there exists a isomorphism $\phi : \{1,\ldots,|x|\} \to \{1,\ldots,|y|\}$ such that $\phi(M^\lhd(x))$ is a restriction of $M^\lhd(y)$. This means that $\phi(D^x) \subseteq D^y$ and for all σ: $\phi(R^x_\sigma) = \{\phi(i) \in R^y_\sigma \mid \phi(i) \in \phi(D^x)\}$ (Definition 3). This implies that $x = \sigma_{\phi(1)} \ldots \sigma_{\phi(|x|)}$. Given that $\lhd = \{(i, i+1) \in D \times D\}$, we have $\phi(i+1) = \phi(i) + 1$ and thus there exist l and r in Σ^* such that $y = l\sigma_{\phi(1)} \ldots \sigma_{\phi(|x|)} r = lxr$. $\qquad\square$

Proposition 2 (Subsequences are subfactors under $M^<$). *For all strings $x, y \in \Sigma^*$, x is a subsequence of y iff $M^<(x) \sqsubseteq M^<(y)$.*

Proof. We leave this proof to the Reader since it is of similar nature to the previous one. $\qquad\square$

4 Grammars, Languages, and Language Classes

Factors can define grammars, formal languages, and classes of formal languages. Usually a model signature provides the vocabulary for some logical language. Sentences in this logical language define sets of strings as follows. The language of a sentence ϕ is all and only those strings whose models satisfy ϕ. Within the regular languages, many well-known subregular classes can be characterized logically in this way (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013; Thomas, 1997).

Intuitively, the grammars we are interested in consist of a finite list of *forbidden* subfactors, whose largest size is bounded by k. Strings in the language of this grammar are those which do not contain any forbidden subfactors. In this way these grammars are like logical expressions which are "conjunctions of negative literals" (Rogers et al., 2013) where the negative literals are played by the the forbidden factors.

Each forbidden subfactor is a principal element of a filter and the language is all strings whose models are not in any of these filters. For each k, there is a class of languages including all and only those languages that can be defined in this way. For example, the Strictly k-Local (SL$_k$) and Strictly k-Piecewise languages can be defined in this way; they are languages which forbid finitely many substrings or subsequences, respectively (Garcia et al., 1990; Rogers et al., 2010). Formally:

Definition 8. *Let k be some positive integer, and M a model of Σ^* with signature Γ. A grammar G is a subset of $\mathtt{Subfact}_k(M, \Sigma^*)$. The language of G is $L(G) = \{w \in \Sigma^* \mid \mathtt{Subfact}_k(M, w) \cap G = \emptyset\}$. The class of languages $\mathscr{L}(M, k) = \{L \mid \exists G \subseteq \mathtt{Subfact}_k(M, \Sigma^*), L(G) = L\}$.*

The elements of G are principal elements of filters, and are called forbidden subfactors.

As an example, let $\Sigma = \{a, b, c\}$ and consider $G = \{M^\lhd(aa), M^\lhd(bb), M^\lhd(c)\}$. $L(G)$ includes the strings $(ab)^+$ and $(ba)^+$ and no other strings, because the substrings aa, bb, and c are all forbidden. This language belongs to $\mathscr{L}(M^\lhd, 2)$.

Proposition 3. *For each $w \in L(G)$ and each $g \in G$, $\mathtt{Subfact}(M, w)$ has a zero intersection with $\mathtt{Supfact}(g)$.*

Proof. Suppose there exists $A \in \mathtt{Subfact}_k(\Sigma^*)$ such that $A \sqsubseteq M(w)$ and $g \sqsubseteq A$. This implies that $g \sqsubseteq M(w)$ and thus that $\mathtt{Subfact}_k(M, w) \cap G \neq \emptyset$ which contradicts Definition 8. $\qquad\square$

In other words, the principal ideal of $M(w)$ is disjoint from the principal filters of the elements of G.

5 Grammatical Entailments

Given a grammar G, we call a subfactor s in $\mathtt{Subfact}(\Sigma^*)$ *ungrammatical* if it belongs to a principal filter of any element of G. Subfactors that are not ungrammatical are called *grammatical*. Lemma 14 ensures that grammaticality is downward entailing, in the sense that if a model of the word $M(w)$ is not contained in the principal filters of the elements of the grammar, then neither are the subfactors of $M(w)$. But it also ensures that ungrammaticality is upward entailing: if a model of the word $M(w)$ belongs to the principal filters of the elements of the grammar, then all of the superfactors of $M(w)$ in that filter are likewise contained.

In this way, the ideals and filters within a a particular model noted above give rise to these entailment properties of grammaticality with respect to the hypothesis space. If the learner constructs filters, then the grammar G will allow structures such that language membership is downward entailing with respect to the grammar G, and language non-membership is upward entailing with respect to the grammar G.

X... X... X...

X [capital, a][]   X [a][capital, a]   X [a][a]

✓ [capital, a]   X [a][]

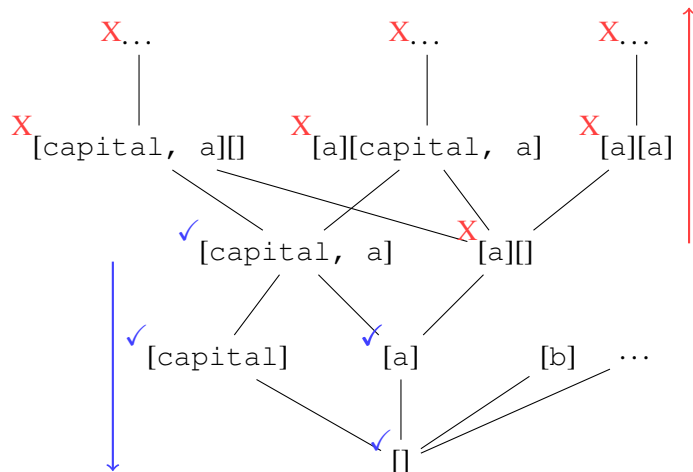✓ [capital]   ✓ [a]   [b] ...

✓ []

Figure 3: The Structure ideals(blue) and filters(red) for a capitalized letter model.

5.1 Example: Text Capitalization

As an example, consider capitalized letters as discussed above. In an unconventional word model, each capital letter at some position x is represented as satisfying one of the relations $R \in \{\texttt{a}(x), \texttt{b}(x), \ldots, \texttt{z}(x)\}$ as well as the unary relation $\texttt{capital}(x)$. Thus the relation $\texttt{a}(x)$ is true of both lowercase a and uppercase A, but $\texttt{a}(x) \wedge \texttt{capital}(x)$ is only true of uppercase A. Note also that in this model no position x of a structure can satisfy both predicates $\texttt{a}(x)$ and $\texttt{b}(x)$. We return to this point in §7.

Figure 2 showcases the relationship among these structures under a model M. The structure for A, [capital, a], contains as subfactors [capital], [a], [], and the empty structure (not shown). The empty structure is a subfactor of [], and [] in turn is a subfactor of [capital] and [a]. The subfactor [a] contains the subfactor [], the domain element with no relations, but has superfactors [capital, a], which has one domain element and two relations, and [a][], which has two domain elements, and the first satisfying the property a. Subfactors and superfactors are listed above and below each other, respectively, with lines between them. Members of one ideal are noted with a blue checkmark, and members of a filter are noted by a red asterisk.

Applying this to the example in Figure 3, if the structure [capital, a] is grammatical, then all of its subfactors, such as [capital] and [a], and [] are grammatical. Since those are grammatical, each of their subfactors is also grammatical, which in this case is just [∅], shown in blue in Figure 3. Conversely, if the structure [a][] is known to be ungrammatical, then any structure which has it as a subfactor is also ungrammatical (in this example, [capital, a][], shown in Red in Figure 3. To see the importance, consider a string with only lowercase letters. In a connected model, the grammar would ban 26 forbidden factors (A,B,C,...), but the "capital" model bans just one, [capital].

5.2 Example: Long Distance Linguistic Dependencies

As another example, sequences of speech sounds as mentioned earlier may be decomposed into binary features based on their phonetic properties like anteriority ($\pm\texttt{ant}$ — whether it occurs in the anterior of the vocal tract), stridency ($\pm\texttt{str}$ — whether it produces a high-intensity fricative noise), or voicing ($\pm\texttt{voi}$ — whether it activates the vocal chords), among others (Hayes, 2009). Each sound at some position x is represented as satisfying relations $R \in \{\pm\texttt{voi}(x), \pm\texttt{str}(x), \ldots, \pm\texttt{ant}(x)\}$. Thus the relation $+\texttt{str}(x)$ is true of both the sound s as in the first sound of "sue" and ʃ, as in "shoe", but $+\texttt{str}(x) \wedge -\texttt{ant}(x)$ is only true of ʃ.

Note also that in this model no position x of a structure can satisfy both predicates $+\texttt{str}(x)$ and $-\texttt{str}(x)$. We return to this point in §7 below. We again use square brackets to delimit the domain elements and write the unary features within them, so a model representation like $\begin{bmatrix} +\texttt{str} \\ +\texttt{ant} \end{bmatrix} \begin{bmatrix} +\texttt{str} \\ -\texttt{ant} \end{bmatrix}$ has the following visual representation:
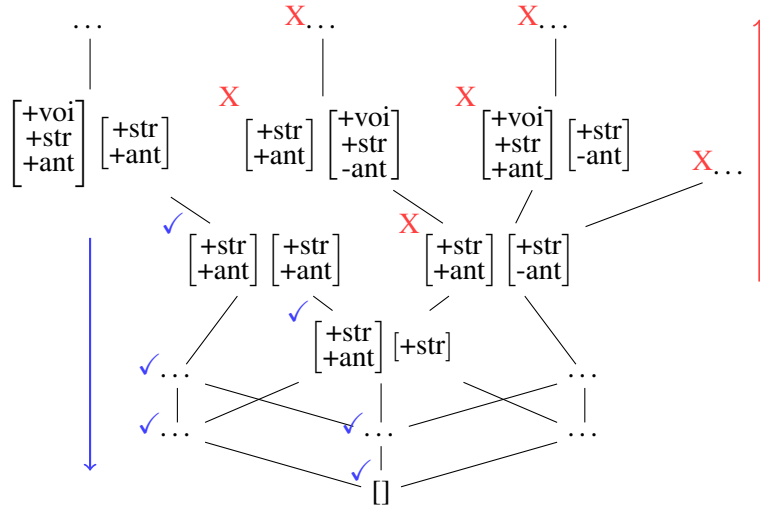
+str   +str
+ant   -ant

<   →
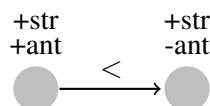
Figure 4: Structure ideals(blue) and filters(red) for a phonological precedence model.

To ease the exposition, we will use square brackets to delimit the domain elements and write the unary relations within them instead of specifying the model in mathematical detail. In an unconventional subsequence word model, then, one possible structure of the subsequence s...ʃ is written $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\-\text{ant}\end{smallmatrix}\right]$.

In many languages, the presence of certain segments is dependent on the presence of another segment. In Samala, subsequences like s...s are allowed but s...ʃ are not, so words like ha**s**xintilawa**s** are allowed but words like ha**s**xintilawaʃ are not (Hansson, 2010). In an unconventional model, banning structures of the form [+str][+str] is insufficient, since all these segments share that stridency property, while a structure like $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\-\text{ant}\end{smallmatrix}\right]$ will distinguish them, since they disallow stridents which disagree on the $\pm\text{ant}(x)$ relations. The structure [+ant][−ant] however, is insufficient, since consonants like p,b,m have that feature, and would incorrectly ban acceptable strings. To see the importance, a conventional string model must ban multiple sibilant factors sʃ,zʃ,sʒ,zʒ, while an unconventional model must just ban one, $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\-\text{ant}\end{smallmatrix}\right]$.

Figure 4 showcases the relationship among these structures under a precedence model $M^<$. The structure for $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]$[+str] contains as subfactors (among others) [+str][+str], [+str][], [], and the empty structure (not shown). The empty structure is a subfactor of [], and [] in turn is a subfactor of [+**ant**] and [−str], and so on.

If the structure $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]$ is grammatical, then all of its subfactors, are grammatical, and so are their subfactors, in turn. Conversely, if the structure $\left[\begin{smallmatrix}+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\-\text{ant}\end{smallmatrix}\right]$ is known to be ungrammatical, then any structure which has it as a subfactor is also ungrammatical (for example, $\left[\begin{smallmatrix}+\text{voi}\\+\text{str}\\+\text{ant}\end{smallmatrix}\right]\left[\begin{smallmatrix}+\text{str}\\-\text{ant}\end{smallmatrix}\right]$, where the first segment is also voiced +voi), shown in Red in Figure 4.

The structure filters give the learner an advantage when confronting hypothesis spaces under a particular model. In particular, it allows the learner to prune vast swathes of the hypothesis space as it reaches for principal elements of features. If a learner identifies one structure as being grammatical, the learner may infer that all of its subfactors are also grammatical and not have to consider them. Alternatively, if the learner knows a structure is ungrammatical, it may infer that the ideals above it are also ungrammatical.

Generally, these reductions can be exponential: an alphabet of size 2^n can be represented with n unary relations in the model signature. However, this exponential reduction does not necessarily make learning any easier. The reason for this is that the size of $\texttt{Subfact}_k(M,\Sigma^*)$ equals $\sum_{i=1}^{k}(2^n)^i$ where n is the number of unary relations. Since a grammar is defined as a subset of $\texttt{Subfact}_k(M,\Sigma^*)$, the number of considered grammars is thus very large. Therefore, the problem of how to search this space effectively is paramount.

6 The Learning Problem

For some M,k, is $\mathscr{L}(M,k)$ learnable from positive data? The short answer is Yes (Heinz, 2010; Heinz et al., 2012). The solution presented in these papers can be thought of as using the function $\texttt{Subfact}_k(M,w)$ to identify permissible k-factors in words w in the positive data. The k-factors that are not permissible are forbidden. With sufficient positive data, such a learning algorithm will converge to a grammar that generates any target language in the class. While this solution is sound in theory, when the space of k-factors is very large, it is not practical. Here, we make clear the problem the learning algorithm solves.

We state the learning problem not in terms of converging to a correct grammar in the limit as previously studied, but instead of returning an 'adequate' grammar given a finite positive sample. Determining what counts as an adequate grammar is what (De Raedt, 2008) calls a Quality Criterion.

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

1. G is consistent, that is, it covers the data: $D \subseteq L(G)$;

2. $L(G)$ is a smallest language in $\mathscr{L}$ which covers the data: so for all $L \in \mathscr{L}$ where $D \subseteq L$, we have $L(G) \subseteq L$; and

3. G includes structures S that are restrictions of structures S' included in other grammars G' that also satisfy (1) and (2): for all G' satisfying the first two criteria for all $S' \in G'$, there exists $S \in G$ such that $S \sqsubseteq S'$.

The first criterion is self-explanatory. The second criterion is motivated by Angluin's (1980) analysis of identification in the limit. The third criterion requires that the grammar contain the most "general" subfactors. An example will help illustrate this criterion.

Consider again the grammar $G = \{M^{\lhd}(aa), M^{\lhd}(bb), M^{\lhd}(c)\}$ with $\Sigma = \{a,b,c\}$. $L(G)$ is the same as $L(H)$ where $H = \{M^{\lhd}(aa), M^{\lhd}(bb), M^{\lhd}(ac), M^{\lhd}(bc), M^{\lhd}(cc), M^{\lhd}(ca), M^{\lhd}(cb)\}$. In H all the forbidden subfactors are of size 2, whereas G encapsulates all of the 2-factors in H which include c with a single 1-factor $M^{\lhd}(c)$. Both grammars G and H may satisfy criteria (1) and (2) but H would not satisfy criterion (3) because of G.

7 A Bottom-Up Learning Algorithm

(De Raedt, 2008) identifies two directions of inference: specific-to-general (i.e., 'top-down') and general-to-specific (i.e., 'bottom-up'). Since we are trying to find the most general subfactors, top-down inference has the potential to consider exponentially many more subfactors than bottom-up inference. It makes mores sense to traverse bottom-up, that is, from the most general subfactors possible to the most specific. Additionally, once a subfactor is identified as an element of the grammar, none of its superfactors (elements of its principal filter) need to be considered further.

A bottom-up learner is shown in Algorithm 1. Its input is a positive data sample D and an integer k that identifies the upper bound on the size of the subfactors.

Data: positive sample D, empty structure s_0, max constraint size k
Result: G, a set of constraints
$Q \leftarrow \{s_0\}$;
$G \leftarrow \emptyset$;
$V \leftarrow \emptyset$;
while $Q \neq \emptyset$ **do**
    $s \leftarrow Q.\texttt{dequeue}()$;
    $V \leftarrow V \cup \{s\}$;
    if $\exists x \in D$ such that $s \sqsubseteq x$ **then**
        $S \leftarrow \texttt{NextSupFact}(s)$;
        $S' \leftarrow \{s \in S \mid |s| \leq k \wedge (\neg \exists g \in G)[g \sqsubseteq s] \wedge s \notin V\}$;
        $Q.\texttt{enqueue}(S')$;
    end
    else
        $G \leftarrow G \cup \{s\}$;
    end
end
return G;

Algorithm 1: Bottom-up learning algorithm for lattice-structured constraints
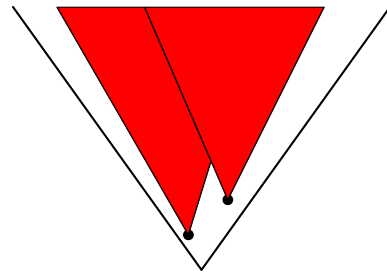

Figure 5: Pruning the hypothesis space

The algorithm makes use of a queue Q, which is initialized to contain just the empty structure s_0. It also initializes two empty sets: G, the grammar that will ultimately be returned, and V, the set of 'visited subfactors'. The subfactors in Q are con-

sidered one at a time, in order, and as each subfactor s is considered it is added to V. If s is not a subfactor of the model of any word in the positive sample D (i.e., not contained by any data point in D), then it is added to the grammar G.

If s is a subfactor of the sample, it is sent to the function `NextSupFact`, which returns a set of *least* superfactors for s. For concreteness, `NextSupFact`(s) may be defined formally as follows:

`NextSupFact`$(s) = \{S \in$ `Subfact`$_k(\Sigma^*) \mid s \sqsubseteq S, \neg\exists S'[s \sqsubseteq S' \sqsubseteq S]\}$.

Practically `NextSupFact` will be defined constructively so that each subfactor in `Subfact`$_k(\Sigma^*)$ is constructed only once as needed. Thus, not only will it not be needed to store the whole set `Subfact`$_k(\Sigma^*)$ in memory, but the set V may be excluded from the algorithm as well.

This set of superfactors is then filtered by the following criteria: they must be smaller than $k+1$, they must contain no element of G as a subfactor, and they must not have been previously considered (i.e., they cannot be in V). Those structures that survive this filter are added to Q. This procedure continues until there are no more structures left to consider in Q.

Theorem 1. *For any $L \in \mathscr{L}(M, k)$, and any finite set $P \subseteq L$ provided as input to Algorithm 1, it returns a grammar G satisfying Definition 9.*

Proof. Consider any $x \in D$. Algorithm 1 only adds elements to G that are not subfactors of x, so $x \notin$ `Supfact`(G). Thus $x \in L(G)$ and $D \subseteq L(G)$, satisfying Condition (1).

Consider any $L' \in \mathscr{L}$ with $D \subseteq L'$. To show $L = L(G) \subseteq L'$, consider any $w \in L$. Then `Subfact`$(w) \subseteq$ `Subfact`(D) and `Subfact`$(D) \subseteq$ `Subfact`(L') since $D \subseteq L$. Then `Subfact`$(w) \subseteq$ `Subfact`(L'). Hence, $w \in L'$, and so $L \subseteq L'$, satisfying Condition (2).

For condition (3), we use the fact that elements in the grammar G were in Q at some point. Suppose s, s' are subfactors such that $s \in G$, $s' \sqsubseteq s$, and $(\neg\exists x \in D)[s' \sqsubseteq M(x)]$. Since $s \in G$, then at some point $s \in Q$.

If $s' \sqsubseteq s$ then s' will be added to Q before s is generated by `NextSupFact`. Because Q is a queue, s' will also be removed from Q before s is generated by `NextSupFact`. Since s' is not contained by any $M(x)$ with $x \in D$, it will be added to G. When s is generated by `NextSupFact`, it will

not pass the filter because it fails the second criterion since $s' \sqsubseteq s$ and $s' \in G$. Then s is never added to Q, and therefore $s \notin G$, contra our original assumption. Thus Condition (3) is satisfied. $\square$

One aspect of the algorithm to highlight is that when a subfactor g is added to G, it is not added to Q. Consequently, `NextSupFact`(g) is never added to Q. In this way, finding elements of G prunes the remainder of the space to be searched (see figure 5). In general, it is not the case that every element in the principal filter of g will not be generated by `NextSupFact` since some of these elements may belong to `NextSupFact`(x) for other subfactors x on the Q. We expect subfactors on the 'border' of `Supfact`(g) to be generated in this way (and then they are filtered out). This pruning, especially when the subfactors are quite general, can significantly reduce the remaining space to be traversed.

In regard to efficiency, in the worst case, the elements of G are all very specific subfactors and are greatest elements of `Subfact`$_k(\Sigma^*)$. In this case, every subfactor `Subfact`$_k(\Sigma^*)$ will be added to Q and the time complexity is thus exponential. However, we are primarily interested in the case when `Subfact`$_k(D)$ are a small proportion of `Subfact`$_k(\Sigma^*)$. This constitutes an example of data sparsity. In this case, we believe the elements of the target grammar will be much 'lower' in the partial order and thus will be found much more quickly. Determining what conditions on `Subfact`$_k(D)$ and `Subfact`$_k(\Sigma^*)$ result in a polynomial time run in the size of D is a focus of current research activity.

Another area of active research is developing a recipe for the `NextSupFact` function for models with a successor or precedence order relation and arbitary unary relations. The basic idea underlying the bottom-up algorithm is to develop a spanning tree for the poset `Subfact`(Σ^*) and to traverse this tree in a breadth-first manner. The function `NextSupFact` helps control this search. Ideally, `NextSupFact` would only generate each subfactor once, which obviates the need to store visited subfactors in V. This can be accomplished to some extent in different ways. For incompatible unary relations, like `a` and `b` in our capitalization example, `NextSupFact` can be defined to prevent adding property `a` to a position that already satisfies property `b`.

For compatible unary relations, like `a` and

`capital` in our capitalization example, an ordering over the unary relations such as `a < b < capital` can help eliminate generating the same subfactor in different ways. For example, if `NextSupFact` is defined to only add 'lesser' unary relations to positions that already have them then it would only output `[capital,a]` given the subfactor `[a]` as input. On the other hand, when given as input the subfactor `[capital]`, it could not add any unary relation to this position.

8 Conclusion

In this paper, we considered the problem of learning formal languages defined as the complement of the union of finitely many principal filters, whose principal elements make up the grammar. This is one way to characterize the Strictly k-Local and Strictly k-Piecewise languages, but the generalization here lets us consider enriched representations of strings where different elements in a string can be said to share properties. it also lets us learn the shortest forbidden substrings in SL_k (Ron et al., 1996) This is useful in many applications where domain-specific knowledge is available and should be taken advantage of. Such enriched representations, however, have a drawback. The number of subfactors is large which makes identifying the principal elements of the filters difficult. This paper showed that the partial ordering of the subfactors motivates a bottom-up learning algorithm which finds the least subfactors whose filters do not include the positive data.

Acknowledgments

References

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

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

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

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

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

Saul Gorn. 1967. Explicit definitions and linguistic dominoes. In *Systems and Computer Science*, pages 77–115, Toronto. University of Toronto Press.

Gunnar Hansson. 2010. *Consonant Harmony: Long-Distance Interaction in Phonology*. Number 145 in University of California Publications in Linguistics. University of California Press, Berkeley, CA. Available on-line (free) at eScholarship.org.

Bruce Hayes. 2009. *Introductory Phonology*. Wiley-Blackwell.

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

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

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer.

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

James Rogers. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1(3-4):265–305.

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

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

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

Dana Ron, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149.

Kristina Strother-Garcia, Jeffrey Heinz, and Hyun Jin Hwangbo. 2016. Using model theory for grammatical inference: a case study from phonology. In *Proceedings of The 13th International Conference on Grammatical Inference*, volume 57 of *JMLR: Workshop and Conference Proceedings*, pages 66–78.

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

Mai Ha Vu, Ashkan Zehfroosh, Kristina Strother-Garcia1, Michael Sebok, Jeffrey Heinz, and Herbert G. Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics and AI*.