

# Learning Subregular Classes of Languages with Factored Deterministic Automata

## Abstract

This paper shows how factored finite-state representations of subregular language classes are identifiable in the limit from positive data by learners which are polytime iterative and optimal. These representations are motivated in two ways. First, the size of this representation for a given regular language can be exponentially smaller than the size of the minimal deterministic acceptor recognizing the language. Second, these representations (including the exponentially smaller ones) describe actual formal languages which successfully model natural language phenomenon, notably in the subfield of phonology.

## 1 Introduction

In this paper we show how to define certain subregular classes of languages which are identifiable in the limit from positive data (ILPD) by efficient, well-behaved learners with a lattice-structured hypothesis space (Heinz et al., 2012). It is shown that every finite *set* of DFAs defines such an ILPD class. In this case, each DFA can be viewed as one *factor* in the description of every language in the class. This factoring of language classes into multiple DFA can provide a *compact, canonical* representation of the grammars for *every* language in the class. Additionally, many subregular classes of languages can be learned by the above methods including the Locally  $k$ -Testable, Strictly  $k$ -Local, Piecewise  $k$ -Testable, and Strictly  $k$ -Piecewise languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2010). From a linguistic (and cognitive) perspective, these subregular classes are interesting because they appear to be sufficient for modeling phonotactic patterns in human language (Heinz, 2010; Heinz et al., 2011; Rogers et al., to appear).

## 2 Preliminaries

For any function  $f$  and element  $a$  in the domain of  $f$ , we write  $f(a) \downarrow$  if  $f(a)$  is defined, and  $f(a) \uparrow$  otherwise.  $\Sigma^*$  and  $\Sigma^k$  denote all sequences of any finite length, and of length  $k$ , over a finite alphabet  $\Sigma$ . The empty string is denoted  $\lambda$ . A language  $L$  is a subset of  $\Sigma^*$ .

For all  $x, y$  belonging to a partially-ordered set  $(S, \leq)$ , if  $x \leq z$  and  $y \leq z$  then  $z$  is an *upper bound* of  $x$  and  $y$ . For all  $x, y \in S$ , the *least upper bound* (lub)  $x \sqcup y = z$  iff  $x \leq z, y \leq z$ , and for all  $z'$  which upper bound  $x$  and  $y$ , it is the case that  $z \leq z'$ . An *upper semi-lattice* is a partially ordered set  $(S, \leq)$  such that every subset of  $S$  has a lub. If  $S$  is finite, this is equivalent to the existence of  $x \sqcup y$  for all  $x, y \in S$ .

A deterministic finite-state automaton (DFA) is a tuple  $(Q, \Sigma, q_0, F, \delta)$ . The states of the DFA are  $Q$ ; the input alphabet is  $\Sigma$ ; the initial state is  $q_0$ ; the final states are  $F$ ; and  $\delta : Q \times \Sigma \rightarrow Q$  is the transition *function*. The transition function's domain is extended to  $Q \times \Sigma^*$  in the usual way. The language of a DFA is written  $L(\mathcal{A})$  and equals  $\{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$ . A DFA is *trim* iff  $(\forall q \in Q)[\exists w, v \in \Sigma^* \mid \delta(q_0, w) = q \wedge \delta(q, v) \in F]$ . The empty DFA is  $\mathcal{A}_\emptyset = (\emptyset, \Sigma, q_0, \emptyset, \emptyset)$ .

The DFA product of  $\mathcal{A}_1 = (Q_1, \Sigma, q_{01}, F_1, \delta_1)$  and  $\mathcal{A}_2 = (Q_2, \Sigma, q_{02}, F_2, \delta_2)$  is

$$\otimes(\mathcal{A}_1, \mathcal{A}_2) = (Q, \Sigma, q_0, F, \delta)$$

where  $Q = Q_1 \times Q_2$ ,  $q_0 = (q_{01}, q_{02})$ ,  $F = F_1 \times F_2$  and

$$\begin{aligned} (\forall q \in Q)(\forall \sigma \in \Sigma) \left[ \right. \\ \delta((q_1, q_2), \sigma) = (q'_1, q'_2) \\ \left. \Leftrightarrow \delta_1(q_1, \sigma) = q'_1 \wedge \delta_2(q_2, \sigma) = q'_2 \right]. \end{aligned}$$

The DFA product of two DFA is also a DFA. If the two DFA are trim, then their product is as well.

The product operation is associative and commutative (up to isomorphism), and so it can be applied to a finite set  $S$  of DFA, in which case we write  $\bigotimes S = \bigotimes_{\mathcal{A} \in S} \mathcal{A}$  (letting  $\bigotimes \{\mathcal{A}\} = \mathcal{A}$ ). In this paper, grammars are finite *sequences* of DFAs  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$  and we also use the  $\bigotimes$  notation for the product of a finite sequence of DFAs:  $L(\vec{\mathcal{A}}) = L(\bigotimes_{\mathcal{A} \in \vec{\mathcal{A}}} \mathcal{A})$ . Sequences are used instead of sets in order to match factors in two grammars. Let  $\mathcal{DFA}$  denote the collection of finite sequences of DFAs.

Theorem 1 is well-known.

**Theorem 1** *Consider a finite set of  $S$  of DFA. Then  $L(\bigotimes_{\mathcal{A} \in S} \mathcal{A}) = \bigcap_{\mathcal{A} \in S} L(\mathcal{A})$ .*

An important consequence of Theorem 1 is that some languages are exponentially more compactly represented by their factors. The grammar  $\vec{\mathcal{A}} = \{\mathcal{A}_1 \cdots \mathcal{A}_n\}$  has  $\sum_{1 \leq i \leq n} \text{card}(Q_i)$  states, whereas the trimmed  $\bigotimes G$  can have as many as  $\prod_{1 \leq i \leq n} \text{card}(Q_i) \in \Theta(\max_{1 \leq i \leq n} (\text{card}(Q_i))^n)$  states.

## 2.1 Identification in the limit

A *positive text*  $T$  for a language  $L$  is a total function  $T : \mathbb{N} \rightarrow L \cup \{\#\}$  ( $\#$  is a ‘pause’) such that  $\forall w \in L, \exists n \in \mathbb{N}$  such that  $T(n) = w$ . Let  $T[i]$  denote the initial finite sequence  $T(0), T(1) \dots T(i-1)$ . Let SEQ denote the set of all finite initial portions of all positive texts for all possible languages. The *content* of an element  $T[i]$  of SEQ is  $\text{content}(T[i]) = \{w \in \Sigma^* \mid (\exists j \leq i-1)[T(j) = w]\}$ . In this paper, learning algorithms are programs:  $\phi : \text{SEQ} \rightarrow \mathcal{DFA}$ . A learner  $\phi$  *identifies in the limit from positive texts* a collection of languages  $\mathcal{L}$  if and only if for all  $L \in \mathcal{L}$ , for all positive texts  $T$  for  $L$ , there exists an  $n \in \mathbb{N}$  such that for all  $m \geq n$ ,  $\phi(T[m]) = \phi(T[n]) = G$  and  $L(G) = L$  (see Gold (1967; Jain et al. (1999)). A class of languages is ILPD iff it is identifiable in the limit by such a learner.

## 3 Classes of factorable-DFA languages

In this section, classes of factorable-DFA languages are introduced. The notion of sub-DFA is central to this concept. Pictorially, a sub-DFA is obtained from a DFA by removing zero or more states, transitions, and/or revoking the final status of zero or more final states.

**Definition 1** *For any DFA  $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ , a DFA  $\mathcal{A}' = (Q', \Sigma', q'_0, F', \delta')$  is sub-DFA of  $\mathcal{A}$ ,*

*written  $\mathcal{A}' \sqsubseteq \mathcal{A}$ , if and only if  $Q' \subseteq Q, \Sigma \subseteq \Sigma', q'_0 = q_0, F' \subseteq F, \delta' \subseteq \delta$ . The sub-DFA relation is extended to grammars (sequences of DFA). Let  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$  and  $\vec{\mathcal{A}}' = \langle \mathcal{A}'_1 \cdots \mathcal{A}'_n \rangle$ . Then  $\vec{\mathcal{A}}' \sqsubseteq \vec{\mathcal{A}} \iff (\forall 0 \leq i \leq n)[\mathcal{A}'_i \sqsubseteq \mathcal{A}_i]$ .*

Clearly, if  $\mathcal{A}' \sqsubseteq \mathcal{A}$  then  $L(\mathcal{A}') \subseteq L(\mathcal{A})$ .

Every grammar  $\vec{\mathcal{A}}$  determines a class of languages: those recognized by a sub-grammar of  $\vec{\mathcal{A}}$ . Our interest is not in  $L(\vec{\mathcal{A}})$ , itself. Indeed, this will generally be  $\Sigma^*$ . Rather, we our interest is in identifying languages relative to the class of languages that are recognizable by sub-grammars of  $\vec{\mathcal{A}}$ .

**Definition 2** *Let  $\mathcal{G}(\vec{\mathcal{A}}) \stackrel{\text{def}}{=} \{\vec{\mathcal{B}} \mid \vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}\}$ , the class of grammars that are sub-grammars of  $\vec{\mathcal{A}}$ .*

*Let  $\mathcal{L}(\vec{\mathcal{A}}) \stackrel{\text{def}}{=} \{L(\vec{\mathcal{B}}) \mid \vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}\}$ , the class of languages recognized by sub-grammars of  $\vec{\mathcal{A}}$ .*

*A class of languages is a factorable-DFA class iff it is  $\mathcal{L}(\vec{\mathcal{A}})$  for some  $\vec{\mathcal{A}}$ .*

The set  $\mathcal{G}(\vec{\mathcal{A}})$  is necessarily finite, since  $\vec{\mathcal{A}}$  is, so every class  $\mathcal{L}(\vec{\mathcal{A}})$  is trivially ILPD by a learning algorithm that systematically rules out grammars that are incompatible with the text, but this naive algorithm is prohibitively inefficient. Our goal is to establish that the efficient general learning algorithm given by Heinz et al. (2012) can be applied to every class of factorable-DFA languages, and that this class includes many of the well-known sub-regular language classes as well as classes that are, in a particular sense, mixtures of these.

## 4 A motivating example

This section describes the Strictly 2-Piecewise languages, which motivates the factorization which is at the heart of this analysis Strictly Piecewise (SP) languages are characterized in Rogers et al. (2010) and are a special subclass of the Piecewise Testable languages (Simon, 1975). In fact SP are exactly those languages closed under sub-sequence.

Every SP language is the intersection of a finite set of complements of principle shuffle ideals:

$$L \in \text{SP} \stackrel{\text{def}}{\iff} L = \bigcap_{w \in S} \overline{\text{SI}(w)}, \quad S \text{ finite}$$

where

$$\text{SI}(w) \stackrel{\text{def}}{=} \{v \in \Sigma^* \mid w = \sigma_1 \cdots \sigma_k \text{ and } (\exists v_0, \dots, v_k \in \Sigma^*)[v = v_0 \cdot \sigma_1 \cdot v_1 \cdots \sigma_k \cdot v_k]\}$$

So  $v \in \text{SI}(w)$  iff  $w$  occurs as a subsequence of  $v$  and  $L \in \text{SP}$  iff there is a finite set of strings for

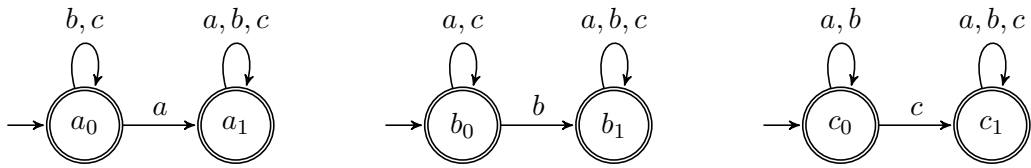


Figure 1: The sequence of DFA  $\vec{\mathcal{A}} = \langle A_a, A_b, A_c \rangle$ , where  $\Sigma = \{a, b, c\}$  and each  $A_\sigma$  accepts  $\Sigma^*$  and whose states distinguish whether  $\sigma$  has yet occurred.

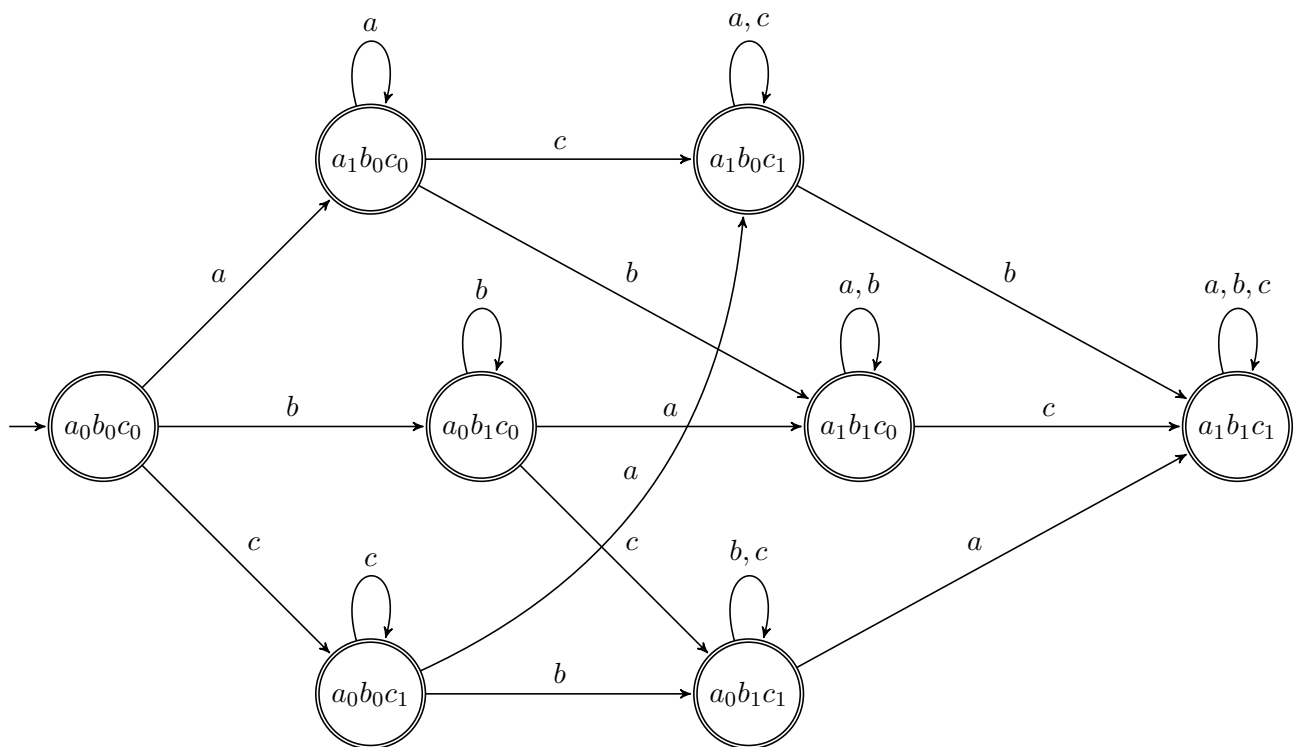


Figure 2: The product  $A_a \times A_b \times A_c$ .

which  $L$  includes all and only those strings that do not include those strings as subsequences.

A language is  $\text{SP}_k$  iff it is generated by a set of strings each of which is of length less than or equal to  $k$ . Clearly, every SP language is  $\text{SP}_k$  for some  $k$  and  $\text{SP} = \bigcup_{1 \leq k \in \mathbb{N}} [\text{SP}_k]$ .

If  $w \in \Sigma^*$  and  $|w| = k$ , then  $\overline{\text{SI}(w)} = L(\mathcal{A}_{\overline{w}})$  for a DFA  $\mathcal{A}_{\overline{w}}$  with no more than  $k$  states. For example, if  $k = 2$  and  $\Sigma = \{a, b, c\}$  and, hence,  $w \in \{a, b, c\}^2$ , then the minimal trim DFA recognizing  $\overline{\text{SI}(w)}$  will be a sub-DFA (in which one of the transitions from the  $\sigma_1$  state has been removed) of one of the three DFA of Figure 1.

Figure 1 shows  $\vec{\mathcal{A}} = \langle A_a, A_b, A_c \rangle$ , where  $\Sigma = \{a, b, c\}$  and each  $A_\sigma$  is a DFA accepting  $\Sigma^*$  whose states distinguish whether  $\sigma$  has yet occurred. Figure 2 shows  $\otimes \vec{\mathcal{A}}$ .

Note that every  $\text{SP}_2$  language over  $\{a, b, c\}$  is  $L(\vec{\mathcal{B}})$  for some  $\vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}$ . The class of grammars of  $\mathcal{G}(\vec{\mathcal{A}})$  recognize a slight extension of  $\text{SP}_2$  over  $\{a, b, c\}$  (which includes 1-Reverse Definite languages as well).

Observe that 6 states are required to describe  $\vec{\mathcal{A}}$  but 8 states are required to describe  $\otimes \vec{\mathcal{A}}$ . Let  $\vec{\mathcal{A}}_\Sigma$  be the sequence of DFA with one DFA for each letter in  $\Sigma$ , as in Figure 1. As  $\text{card}(\Sigma)$  increases the number of states of  $\vec{\mathcal{A}}_\Sigma$  is  $2 \times \text{card}(\Sigma)$  but the number of states in  $\otimes \vec{\mathcal{A}}_\Sigma$  is  $2^{\text{card}(\Sigma)}$ . The number of states in the product, in this case, is exponential in the number of its factors.

The Strictly 2-Piecewise languages are currently the strongest computational characterization<sup>1</sup> of long-distance phonotactic patterns in human languages (Heinz, 2010). The size of the phonemic inventories<sup>2</sup> in the world's languages ranges from 11 to 140 (Maddieson, 1984). English has about 40, depending on the dialect. The fact that there are about  $10^{11}$  neurons in human brains (Williams and Herrup, 1988), which is about  $2^{36.5}$  helps motivate interest in the more compact, parallel representation given by  $\vec{\mathcal{A}}_\Sigma$  as opposed to the singular representation of the DFA  $\otimes \vec{\mathcal{A}}_\Sigma$ .

<sup>1</sup>See Heinz et al. (2011) for competing characterizations.

<sup>2</sup>The mental representations of speech sounds are called phonemes, and the phonemic inventory is the set of these representations (Hayes, 2009).

## 5 Learning factorable classes of languages

In this section, classes of factorable-DFA languages are shown to be analyzable as finite lattice spaces. By Theorem 6 of Heinz et al. (2012), every such class of languages can be identified in the limit from positive texts.

**Definition 3 (Joins)** Let  $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ ,  $\mathcal{A}_1 = (Q_1, \Sigma, q_0, F_1, \delta_1)$ ,  $\mathcal{A}_2 = (Q_2, \Sigma, q_0, F_2, \delta_2)$  such that  $\mathcal{A}_1 \sqsubseteq \mathcal{A}$  and  $\mathcal{A}_2 \sqsubseteq \mathcal{A}$ . The join of  $\mathcal{A}_1$  and  $\mathcal{A}_2$  is  $\mathcal{A}_1 \sqcup \mathcal{A}_2 = (Q_1 \cup Q_2, \Sigma, q_0, F_1 \cup F_2, \delta_1 \cup \delta_2)$ .

Similarly, for all  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$ ,  $\vec{\mathcal{B}} = \langle \mathcal{B}_1 \cdots \mathcal{B}_n \rangle$ , and  $\vec{\mathcal{C}} = \langle \mathcal{C}_1 \cdots \mathcal{C}_n \rangle$  such that  $\vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}$  and  $\vec{\mathcal{C}} \sqsubseteq \vec{\mathcal{A}}$ , the join of  $\vec{\mathcal{B}}$  and  $\vec{\mathcal{C}}$  is  $\vec{\mathcal{B}} \sqcup \vec{\mathcal{C}} = \langle \mathcal{B}_1 \sqcup \mathcal{C}_1 \cdots \mathcal{B}_n \sqcup \mathcal{C}_n \rangle$ .

Note that the join of two sub-DFA of  $\mathcal{A}$  is also a sub-DFA of  $\mathcal{A}$ . Since  $\mathcal{G}(\vec{\mathcal{A}})$  is finite, binary join suffices to define join of any set of sub-DFA of a given DFA (as recursive binary joins). Let  $\bigsqcup[S]$  be the join of the set of sub-DFA  $S$ .

**Lemma 1** The set of sub-DFA of a DFA  $\mathcal{A}$ , ordered by  $\sqsubseteq$ ,  $(\{\mathcal{B} \sqsubseteq \mathcal{A}\}, \sqsubseteq)$ , is an upper semi-lattice with the least upper bound of a set of  $S$  sub-DFA of  $\mathcal{A}$  being their join.

Similarly the set of sub-grammars of a grammar  $\vec{\mathcal{A}}$ , ordered again by  $\sqsubseteq$ ,  $(\{\vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}\}, \sqsubseteq)$ , is an upper semi-lattice with the least upper bound of a set of sub-grammars of  $\vec{\mathcal{A}}$  being their join.

This follows from the fact that  $Q_1 \cup Q_2$  (similarly  $F_1 \cup F_2$  and  $\delta_1 \cup \delta_2$ ) is the lub of  $Q_1$  and  $Q_2$  (etc.) in the lattice of sets ordered by subset.

### 5.1 Paths and Chisels

**Definition 4** For a DFA  $\mathcal{A}$ , the path of a word  $w = \sigma_0 \sigma_1 \cdots \sigma_n \in \Sigma^*$  in  $\mathcal{A}$  is the sequence  $\pi(\mathcal{A}, w) = \langle (q_0, \sigma_0), (q_1, \sigma_1), \cdots (q_n, \sigma_n), (q_{n+1}, \lambda) \rangle$  where  $(\forall 0 \leq i \leq n)[q_{i+1} = \delta(q_i, \sigma_i)]$  if  $\delta(q_0, w) \downarrow$ ; otherwise  $\pi(\mathcal{A}, w) \uparrow$ . The set of states traversed in a path  $\pi$  will be denoted  $Q_\pi$ , the transitions traversed  $\delta_\pi$ , and if  $w \in L(\mathcal{A})$  then the set of final states in the path  $\pi$  is  $F_\pi = \{q_{n+1}\}$  and the empty set otherwise.

Next, for any DFA  $\mathcal{A}$ , we define the *chisel* of a word, which etches out a path within  $\mathcal{A}$  iff  $w \in L(\mathcal{A})$  and returns the sub-DFA exactly encompassing that path.

**Definition 5** For any DFA  $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$  and all  $w \in \Sigma^*$ . The chisel of  $w$

given  $\mathcal{A}$  is the sub-DFA  $C_{\mathcal{A}}(w) = (Q_{\pi(\mathcal{A},w)}, \Sigma, q_0, F_{\pi(\mathcal{A},w)}, \delta_{\pi(\mathcal{A},w)})$  if  $w \in L(\mathcal{A})$ ; otherwise  $C_{\mathcal{A}}(w) = \mathcal{A}_{\emptyset}$ . Consider any  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$  and any word  $w \in \Sigma^*$ . The chisel of  $w$  given  $\vec{\mathcal{A}}$  is  $C_{\vec{\mathcal{A}}}(w) = \langle C_{\mathcal{A}_1}(w) \cdots C_{\mathcal{A}_n}(w) \rangle$ .

Observe that for all words  $w$  and all  $\mathcal{A}$ ,  $C_{\mathcal{A}}(w) \sqsubseteq \mathcal{A}$  and that  $C_{\mathcal{A}}(w)$  is trim.

Using the join, the domain of the chisel is extended to sets of words:  $C_{\vec{\mathcal{A}}}(S) = \bigsqcup_{w \in S} C_{\vec{\mathcal{A}}}(w)$ . Note that  $\{C_{\vec{\mathcal{A}}}(w) \mid w \in \Sigma^*\}$  is finite, since  $\{\vec{\mathcal{B}} \mid \vec{\mathcal{B}} \sqsubseteq \vec{\mathcal{A}}\}$  is.

**Theorem 2** For any grammar  $\vec{\mathcal{A}}$ , let  $\mathcal{C}(\vec{\mathcal{A}}) = \{C_{\vec{\mathcal{A}}}(S) \mid S \subseteq \Sigma^*\}$ . Then  $(\mathcal{C}(\vec{\mathcal{A}}), \sqsubseteq)$  is an upper semi-lattice with the lub of two elements given by the join  $\sqcup$ .

**Proof** This follows immediately from the finiteness of  $\{C_{\vec{\mathcal{A}}}(w) \mid w \in \Sigma^*\}$  and Lemma 1.  $\square$

**Lemma 2** For all  $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ , there is a finite set  $S \subseteq \Sigma^*$  such that  $\bigsqcup_{w \in S} C_{\mathcal{A}}(w) = \mathcal{A}$ . Similarly, for all  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$ , there is a finite set  $S \subseteq \Sigma^*$  such that  $C_{\vec{\mathcal{A}}}(S) = G$ .

**Proof** For the first statement, let  $S$  be the set of  $u\sigma v$  where, for each  $q \in Q$  and for each  $\sigma \in \Sigma$ ,  $\delta(q_0, u) \downarrow = q$  and  $\delta(\delta(q, \sigma), v) \downarrow \in F$  such that  $u\sigma v$  has minimal length. By construction,  $S$  is finite. Furthermore, for every state and every transition in  $\mathcal{A}$ , there is a word in  $S$  whose path touches that state and transition. By definition of  $\sqcup$  it follows that  $C_{\mathcal{A}}(S) = \mathcal{A}$ . For proof of the second statement,  $\forall \mathcal{A}_i$  in  $\vec{\mathcal{A}}$ , construct  $S_i$  as stated and take their union.  $\square$

Heinz et al. (2012) define lattice spaces. For an upper semi-lattice  $V$  and a function  $f : \Sigma^* \rightarrow V$  such that  $f$  and  $\sqcup$  are (total) computable,  $(V, f)$  is called a *Lattice Space (LS)* iff, for each  $v \in V$ , there exists a finite  $D \subseteq \text{range}(f)$  with  $\bigsqcup D = v$ .

**Theorem 3** For all grammars  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$ ,  $(\mathcal{C}(\vec{\mathcal{A}}), C_{\vec{\mathcal{A}}})$  is a lattice space.

**Proof** For all  $\vec{\mathcal{A}}' \in \mathcal{C}(\vec{\mathcal{A}})$ , by Lemma 2, there is a finite  $S \subseteq \Sigma^*$  such that  $\bigsqcup_{w \in S} C_{\vec{\mathcal{A}}}(w) = \vec{\mathcal{A}}'$ .  $\square$

For Heinz et al. (2012), elements of the lattice are grammars. Likewise, here, each grammar  $\vec{\mathcal{A}} = \langle \mathcal{A}_1 \cdots \mathcal{A}_n \rangle$  defines a lattice whose elements are its sub-grammars. Heinz et al. (2012) associate

the languages of a grammar  $v$  in a lattice space  $(V, f)$  with  $\{w \in \Sigma^* \mid f(w) \sqsubseteq v\}$ . This definition coincides with ours: for any element  $\vec{\mathcal{A}}'$  of  $\mathcal{C}(\vec{\mathcal{A}})$  (note  $\vec{\mathcal{A}}' \sqsubseteq \vec{\mathcal{A}}$ ), a word  $w$  belongs to  $L(\vec{\mathcal{A}}')$  if and only if  $C_{\vec{\mathcal{A}}}(w)$  is a sub-DFA of  $\vec{\mathcal{A}}'$ . The class of languages of a LS is the collection of languages obtained by every element in the lattice. For every LS  $(\mathcal{C}(\vec{\mathcal{A}}), C_{\vec{\mathcal{A}}})$ , we now define a learner  $\phi$  according to the construction in Heinz et al. (2012):  $\forall T \in \text{SEQ}, \phi(T) = \bigsqcup_{w \in \text{content}(T)} C_{\vec{\mathcal{A}}}(w)$ .

Let  $\mathcal{L}_{(\mathcal{C}(\vec{\mathcal{A}}), C_{\vec{\mathcal{A}}})}$  denote the class of languages associated with the LS in Theorem 3. According to Heinz et al. (2012, Theorem 6), the learner  $\phi$  identifies  $\mathcal{L}_{(\mathcal{C}(\vec{\mathcal{A}}), C_{\vec{\mathcal{A}}})}$  in the limit from positive data. Furthermore,  $\phi$  is *polytime iterative*, i.e. can compute the next hypothesis in polytime from the previous hypothesis alone, and optimal in the sense that no other learner converges more quickly on languages in  $\mathcal{L}_{(\mathcal{C}(\vec{\mathcal{A}}), C_{\vec{\mathcal{A}}})}$ . In addition, this learner is *globally-consistent*, *locally-conservative*, *strongly-monotone*, and *prudent*. Formal definitions of these terms are given in Heinz et al. (2012) and can also be found elsewhere, e.g. (Jain et al., 1999).

## 6 Complexity considerations

The space of sub-grammars of a given sequence of DFAs is necessarily finite and, thus, identifiable in the limit from positive data by a naïve learner that simply enumerates the space of grammars. The lattice learning algorithm has better efficiency because it works bottom-up, extending the grammar minimally, at each step, with the chisel of the current string of the text. The lattice learner never explores any part of the space of grammars that is not a sub-grammar of the correct one and, as it never moves down in the lattice, it will skip much of the space of grammars that are sub-grammars of the correct one. The space it explores will be minimal, given the text it is running on. Generalization is a result of the fact that in extending the grammar for a string the learner adds its entire Nerode equivalence class to the language.

The time complexity of either learning or recognition with the factored automata may actually be somewhat worse than the complexity of doing so with its product. Computing the chisel of a string  $w$  in the product machine of Figure 2 is  $\Theta(|w|)$ , while in the factored machine of Figure 1 one must compute the chisel in each factor and its complexity is, thus,  $\Theta(|w| \text{card}(\Sigma)^{k-1})$ . But  $\Sigma$  and  $k$  are

fixed for a given factorization, so this works out to be a constant factor.

Where the factorization makes a substantial difference is in the number of features that must be learned. In the factored grammar of the example, the total number of states plus edges is  $\Theta(k \mathbf{card}(\Sigma)^{k-1})$ , while in its product it is  $\Theta(2(\mathbf{card}(\Sigma)^{k-1}))$ . This represents an exponential improvement in the space complexity of the factored grammar.

Every DFA can be factored in many ways, but the factorizations do not, necessarily provide an asymptotically significant improvement in space complexity. The canonical contrast is between sequences of automata  $\langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$  that count modulo some sequence of  $m_i \in \mathbb{N}$ . If the  $m_i$  are pairwise prime, the product will require  $\prod_{1 \leq i \leq n} [m_i] = \Theta((\max_i [m_i])^n)$  states. If on the other hand, they are all multiples of each other it will require just  $\Theta(\max_i [m_i])$ .

## 7 Examples

Factored grammars in which each factor recognizes  $\Sigma^*$ , as in the case of Figure 1, are of particular interest. Every sub-Star-Free class of languages in which the parameters of the class ( $k$ , for example) are fixed can be factored in this way.<sup>3</sup> If the parameters are not fixed and the class of languages is not finite, none of these classes can be identified in the limit from positive data at all.<sup>4</sup> So this approach is potentially useful at least for all sub-Star-Free. The learners for non-strict classes are practical, however, only for small values of the parameters. So that leaves the Strictly Local  $SL_k$  and Strictly Piecewise  $SP_k$  languages as the obvious targets.

The  $SL_k$  languages are those that are determined by the substrings of length no greater than  $k$  that occur within the string (including endmarkers). These can be factored on the basis of those substrings, just as the  $SP_k$  languages can, although the construction is somewhat more complex. (See the Knuth-Morris-Pratt algorithm (Knuth et al., 1977) for a way of doing this.) But  $SL_k$  is a case in

<sup>3</sup>We conjecture that there is a parameterized class of languages that is equivalent to the Star-Free languages, which would make that class learnable in this way as well.

<sup>4</sup>For most of these classes, including the Definite, Reverse-Definite and Strictly Local classes and their super classes, this is immediate from the fact that they are super-finite.  $SP$ , on the other hand, is not super-finite (since it does not include all finite languages) but it is, nevertheless not IPLD.

which there is no complexity advantage in factoring the DFA. This is because every  $SL_k$  language is recognized by a DFA that is a Myhill graph: with a state for each string of  $\Sigma^{<k}$  (i.e., of length less than  $k$ ). Such a graph has  $\Theta(\mathbf{card}(\Sigma)^{k-1})$  states, asymptotically the same as the number of states in the factored grammar, which is actually marginally worse.

Therefore, factored  $SL_k$  grammars are not, in themselves, interesting. But they are interesting as factors of other grammars. Let  $(SL + SP)_{k,l}$  (resp.  $(LT + SP)_{k,l}$ ,  $(SL + PT)_{k,l}$ ) be the class of languages that are intersections of  $SL_k$  and  $SP_l$  (resp.  $LT_k$  and  $SP_l$ ,  $SL_k$  and  $PT_l$ ) languages. Where  $LT$  ( $PT$ ) languages are determined by the *set* of substrings (subsequences) that occur in the string.

These classes are of particular interest in phonotactics. They are linguistically well-motivated approaches to modeling phonotactics and they are sufficiently powerful to model most phonotactic patterns. The results of Heinz (2007) and Heinz (2010) strongly suggest that nearly all segmental patterns are  $(SL + SP)_{k,l}$  for small  $k$  and  $l$ . Moreover, roughly 72% of the stress patterns that are included in Heinz’s database (Phonology Lab, 2012) of patterns that have been attested in natural language can be modeled with  $SL_k$  grammars with  $k \leq 6$ . Of the rest, all but four are  $LT_1 + SP_4$  and all but two are  $LT_2 + SP_4$ . Both of these last two are properly regular (Wibel et al., ).

## 8 Conclusion

We have shown how subregular classes of languages can be learned over factored representations, which can be exponentially more compact than representations with a single DFA. Essentially, words in the data presentation are passed through each factor, “activating” the parts touched. This approach immediately allows one to naturally “mix” well-characterized learnable subregular classes in such a way that the resulting language class is also learnable. While this mixing is partly motivated by the different kinds of phonotactic patterns in natural language, it also suggests a very interesting theoretical possibility. Specifically, we anticipate that the right parameterization of these well-studied subregular classes will cover the class of star-free languages. Future work could also include extending the current analysis to factoring stochastic languages, perhaps in a way that connects with earlier research on factored HMMs

(Ghahramani and Jordan, 1997).

## References

- Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden markov models. *Machine Learning*, 29(2):245–273.
- E.M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.
- Bruce Hayes. 2009. *Introductory Phonology*. Wiley-Blackwell.
- 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*, pages 58–64, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127, October.
- Jeffrey Heinz. 2007. *The Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles.
- Jeffrey Heinz. 2010. Learning long-distance phonotactics. *Linguistic Inquiry*, 41(4):623–661.
- 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.
- Donald Knuth, James H Morris, and Vaughn Pratt. 1977. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350.
- Ian Maddieson. 1984. *Patterns of Sounds*. Cambridge University Press, Cambridge, UK.
- Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.
- UD Phonology Lab. 2012. UD phonology lab stress pattern database. <http://phonology.cogsci.udel.edu/dbs/stress>. Accessed December 2012.
- James Rogers and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342.
- James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edleson, 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*, volume 6149 of *Lecture Notes in Artificial Intelligence*, pages 255–265. Springer.
- James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. to appear. Cognitive and sub-regular complexity. In *Proceedings of the 17th Conference on Formal Grammar*.
- Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages: 2nd Grammatical Inference conference*, pages 214–222, Berlin. Springer-Verlag.
- Sean Wibel, James Rogers, and Jeffrey Heinz. Factoring of stress patterns. In preparation.
- R.W. Williams and K. Herrup. 1988. The control of neuron number. *Annual Review of Neuroscience*, 11:423–453.