Learning Left-to-Right and Right-to-Left Iterative Languages

Jeffrey Heinz

University of Delaware, Newark DE 19716, USA
heinz@udel.edu

Abstract. The left-to-right and right-to-left iterative languages are previously unnoticed subclasses of the regular languages of infinite size that are identifiable in the limit from positive data. Essentially, these language classes are the ones obtained by merging final states in a prefix tree and initial states in a suffix tree of the observed sample, respectively. Strikingly, these classes are also transparently related to the zero-reversible languages because some algorithms that learn them differ minimally from the ZR algorithm given in Angluin (1982). Second, they are part of the answer to the challenge provided by Muggleton (1990), who proposed mapping the space of language classes obtainable by a general state-merging algorithm IM1. Third, these classes are relevant to a hypothesis of how children can acquire sound patterns of their language—in particular, the hypothesis that all phonotactic patterns found in natural language are neighborhood-distinct (Heinz 2007).

1 Introduction

One motivation behind the learning paradigm known as *identification in the limit from positive data* [1] is the observation that children encounter only positive examples of natural language [2]. Gold's (1967) result that no class of languages including all finite languages plus one infinite language is learnable this way—and hence the regular, context-free, and context-sensitive languages are not either—launched research to find learnable subclasses which crosscut the Chomsky Hierarchy [3,4,5,6,7,8,9,10,11,12,13]. This approach is also justifiable from a linguistic perspective because language typologists repeatedly observe that the extensive variation that exists in natural languages appears to be limited, though stating exact universals is difficult [14,15,16].

One focus of the grammatical inference community is finding learnable classes of languages which reach higher regions of the Chomsky Hierarchy [11,12,13]. This is partly because the most complex known natural language patterns are at least mildly context-sensitive [17,18], and partly because of the technical challenge. However, the hypothesis that all phonological patterns—i.e. sound patterns—are regular is well-supported [19,20,21]. In other words, although sentence well-formedness seems to necessitate mildly-context sensitive computations over words, word well-formedness seems only to require regular computations

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 84–97, 2008.

over individual sounds. Thus it remains an important open question what subclasses of the regular languages (if any) are identifiable in the limit from positive data and which properly include word well-formedness patterns observed cross-linguistically (see also [22]). This question is especially interesting in light of acquisition research which indicates infants acquire many word well-formedness patterns within one year of birth [23,24,25]; i.e. before they can talk.

This paper addresses this question by introducing and characterizing the left-to-right iterative (LRI) and right-to-left iterative (RLI) languages, which are relevant to natural language word well-formedness patterns (henceforth *phonotactic patterns*). Essentially, these language classes are the ones obtained by merging final states in a prefix tree and initial states in a suffix tree of the observed sample, respectively. They are interesting for at least four reasons.

First, [7] proposes a general state-merging algorithm IM1 for learning subclasses of the regular languages. Essentially, IM1 returns a nondeterministic finite-state acceptor by merging states according to some state equivalence criteria. To my knowledge, no systematic study of the language classes obtained in this way has been undertaken. This paper contributes to this research program, whose known results and open questions are summarized in §3. Second, this work introduces the concept of head-canonical acceptor which is the smallest reverse deterministic acceptor for a regular language (useful in understanding RLI languages). Third, these two language classes are closely related to—but incomparable with—the zero-reversible languages since algorithms that identify them are essentially the same as algorithm ZR in [5] with one line removed.

Finally, LRI and RLI languages are relevant to phonotactic patterns. When hundreds of phonotactic patterns are collected and placed in the Chomsky Hierarchy, they all fall into the class of the regular languages [21]. More specifically, they are Noncounting, a class known not to be identifiable in limit from positive data [26,22]. [21] hypothesizes all phonotactic patterns belong to a smaller class called neighborhood-distinct (defined in §3), which crosscuts the Subregular Hierarchy. Neighborhood-distinctness is a complex property, making it difficult to analyze. LRI and RLI languages are properly thought of as basic components of this more complex class, and therefore this paper also makes a step towards understanding neighborhood-distinctness.

This paper is organized as follows. §2 introduces notation and definitions. §3 discusses a general state-merging algorithm due to [7]. §4 presents the neighborhood-distinct hypothesis. §5 characterizes the LRI languages in automata- and language-theoretic terms and gives a procedure which identifies this class in the limit from positive data, and §6 does the same for RLI languages. §7 summarizes the contributions and open questions.

2 Preliminaries

This section establishes notation and basic definitions. A set π of nonempty subsets of S is a *partition* of S iff the elements of π are pairwise disjoint and their union equals S. Each *block* in π containing $x \in S$ is denoted $[x]_\pi$. A partition

π *refines* another partition π' iff every block of π' is a union of blocks of π. If π is a partition of a set S and $S' \subseteq S$, then the *restriction* of π to S' is the partition π' consisting of all sets B' that are nonempty and are the intersection of S' and some block of π. The *trivial partition* is the unique partition where $|\pi| = |S|$.

2.1 Strings and Languages

Σ denotes a fixed finite set of symbols, the *alphabet*. Let Σ^n, $\Sigma^{\leq n}$, Σ^* denote all sequences over this alphabet of length n, of length less than or equal to n, and of any finite length, respectively. For sequence s, $range(s)$ is the set of elements in s. λ denotes the empty string and $|w|$ denotes the length of string w. The reverse of string u is denoted u^r. A string u is a *prefix* (*suffix*) of w iff there exists v in Σ^* such that $w = uv$ ($w = vu$).

A language L is a subset of Σ^*. $L^r = \{u^r : u \in L\}$. $L_1 \cdot L_2 = \{uv : u \in L_1 \text{ and } v \in L_2\}$. Like strings above, let $L^0 = \{\lambda\}$, $L^{n+1} = L^n \cdot L$, and $L^* = \bigcup_{n \in \mathbb{N}} L^n$. Let $L^{|k|}, L^{\leq |k|}$ denote all strings in L with length exactly k and length less than or equal to k, respectively. The *length* of a finite language L is $length(L) = \Sigma_{w \in L}|w|$. The *prefixes* of language L are given by $Pref(L) = \{u : \exists v \text{ so that } uv \in L\}$. The *suffixes* of a language are defined as $Suff(L) = \{u : \exists v \text{ so that } vu \in L\}$. The *tails* of w given L, is denoted by $T_L(w) = \{u : wu \in L\}$. Also, the *heads*, of w given L, is denoted by $H_L(w) = \{u : uw \in L\}$. A language L naturally induces partitions π_{TL}: $[u]_{\pi_{TL}} = [v]_{\pi_{TL}}$ iff $T_L(u) = T_L(v)$ and π_{HL}: $[u]_{\pi_{HL}} = [v]_{\pi_{HL}}$ iff $H_L(u) = H_L(v)$. The Myhill-Nerode theorem states that a language is *regular* iff π_{TL} is finite.

2.2 Identification in the Limit

A positive text S of a language L is an infinite sequence such that $range(S) = L$. S_t denotes the first t elements of S. A learner ϕ is an algorithm which maps finite sequences of words to grammars. The learner ϕ *identifies a class of languages* $\mathcal{L}$ *in the limit from positive data* iff for all $L \in \mathcal{L}$, for all positive texts S for L, there is some $i \in \mathbb{N}$ such that for all $j > i$, $\phi(S_j)$ is a grammar recognizing L. A language class with such a ϕ is *identifiable in the limit from positive data*. If for each $L \in \mathcal{L}$, there is a finite language $S_L \subseteq L$ such that for all other $L' \in \mathcal{L}$ which contain S_L, $L \subseteq L'$, then $\mathcal{L}$ is identifiable in the limit from positive data and S_L is called a *characteristic sample* for L in $\mathcal{L}$ [4,5].

2.3 Finite State Acceptors

For a given Σ, a finite-state acceptor (FSA) is a quadruple $A = (Q, I, F, \delta)$ such that Q is finite, I and F are subsets of Q and $\delta : Q \times \Sigma \rightarrow 2^Q$. δ is extended recursively so that $\delta : 2^Q \times \Sigma^* \rightarrow 2^Q$. The language of an acceptor A is $L(A) = \{w \in \Sigma^* : \delta(I, w) \cap F \neq \emptyset\}$. Languages recognizable by FSAs are regular. We assume familiarity with regular expressions, which also define regular languages.[1] [27] provides other well-known characterizations of this class.

[1] See [34] for the regular expression notation used in this paper.

Consider two acceptors $A = (Q, I, F, \delta)$ and $A' = (Q', I', F', \delta')$. Acceptors A and A' are *equivalent* iff $L(A) = L(A')$. A and A' are *isomorphic* iff there is a bijection h from Q to Q' such that $h(I) = I'$, $h(F) = F'$, and for all $q \in Q$ and $a \in \Sigma$ it is the case that $h(\delta(q, a)) = \delta'(h(q), a)$. A' is a *subacceptor* of A iff $Q' \subseteq Q$, $I' \subseteq I$, $F' \subseteq F$, and for every $q' \in Q'$ and $a \in \Sigma$, $\delta'(q', a) \subseteq \delta(q', a)$. It follows that if A' is a subacceptor of A then $L(A') \subseteq L(A)$. The reverse of A is $A^r = (Q, F, I, \delta^r)$, where $\delta^r(q, a) = \{q' : q \in \delta(q', a)\}$ for all $a \in \Sigma, q \in Q$.

An acceptor $A = (Q, I, F, \delta)$ is *forward deterministic* iff $|I| \leq 1$ and each $q \in Q$ has at most one b-successor for all $b \in \Sigma$. A is *reverse deterministic* iff $|F| \leq 1$ and each $q \in Q$ has at most one b-predecessor for all $b \in \Sigma$. An acceptor which is both forward and reverse deterministic is called *zero-reversible* [5].

An acceptor is *trimmed* iff for all $q \in Q$, there are $u, v \in \Sigma^*$ such that $q \in \delta(I, u)$ and $\delta(q, v) \cap F \neq \emptyset$. An acceptor is *cyclic* iff it is trimmed and has at least one loop. An acceptor which is not cyclic is *acyclic*. Two important acyclic acceptors are *prefix* and *suffix trees*. The prefix (suffix) tree of a finite language S is forward (backward) deterministic, and is defined below.

PT(S)	ST(S)
$Q = Pref(S)$	$Q = Suff(S)$
$I = \{\lambda\}$	$I = S$
$F = S$	$F = \{\lambda\}$
$\delta(u, a) = ua$ iff $u, ua \in Q$	$\delta(au, a) = u$ iff $u, ua \in Q$

These acceptors are not mirror images of each other, though they are equivalent. It is easy to show for any finite language S, $PT^r(S)$ is isomorphic to $ST(S^r)$.

The *tail-canonical acceptor* for a regular language L is denoted $A_T(L)$ and is defined below. Acceptors isomorphic to the tail-canonical acceptor are *tail-canonical*. For a regular language L, a tail-canonical acceptor is the forward deterministic acceptor with the fewest states.The *head-canonical acceptor* for L is $A_H(L)$ is also defined below. Acceptors isomorphic to the head-canonical acceptor are called *head-canonical*. A head-canonical acceptor is the acceptor with the fewest states for a regular language L that is backward deterministic.

$A_T(L)$	$A_H(L)$
$Q = \{T_L(u) : u \in Pref(L)\}$	$Q = \{H_L(u) : u \in Suff(L)\}$
$I = \{T_L(\lambda)\}$	$I = \{H_L(w) : w \in L\}$
$F = \{T_L(w) : w \in L\}$	$F = \{H_L(\lambda)\}$
$\delta(T_L(u), a) = T_L(ua)$ iff $u, ua \in Pref(L)$	$\delta(H_L(au), a) = H_L(u)$ iff $u, au \in Suff(L)$

Theorem 1 shows how one can obtain an equivalent head canonical acceptor from a tail canonical one: reverse, determinize, minimize, and reverse again.

Theorem 1. *Let L be a regular language. Then $A_H^r(L)$ is isomorphic to $A_T(L^r)$.*

The proof is omitted (see [21]), but the needed bijection is $h(L) = L^r$. Head canonical acceptors allow another definition of zero-reversible languages: they are those languages whose tail and head canonical acceptors are isomorphic.

The following notions are used in §§3-6. The *k-leaders* of a state q are denoted $I_k(q) = \{u \in \Sigma^{\leq k} : \exists q' \in Q$ such that $q \in \delta(q', u)\}$. The *k-followers* of a state

are denoted $O_k(q) = \{u \in \Sigma^{\leq k} : \exists q' \in Q \text{ such that } q \in \delta(q', u)\}$. ($I$ and O invoke *incoming* and *outgoing*, respectively.) The function $final(q) = \mathbb{f}$ if $q \in F$, else q. The function $nonfinal(q) = \mathbb{nf}$ if $q \notin F$, else q. The function $start(q) = \mathbb{s}$ if $q \in I$, else q. The function $nonstart(q) = \mathbb{ns}$ if $q \notin I$, else q. For any $q \in Q$, $b \in \Sigma$, the *b-successors* of q are $\delta(q, b)$ and the *b-predecessors* of q are $\delta^r(q, b)$.

For some acceptor $A = (Q, I, F, \delta)$, a sequence $q_0 q_1 \dots q_k$ is a *path* in A iff for all $0 \leq i \leq k-1$, there is $a \in \Sigma$ such that $q_{i+1} \in \delta(q_i, a)$. Paths $q_0 q_1 \dots q_k$ such that $q_0 = q_k$ and for $1 \leq i \leq k-1$, $q_i \neq q_0$ are called *loops*. Let $loops(A)$ denote the set of loops in A. Also, if $p = q_0 q_1 \dots q_k$ is a path then $strings(p) = \{u : u = a_0 a_1 \dots a_{k-1} \text{ where for all } 0 \leq\ i\ \leq k-1,\ q_{i+1} \in \delta(q_i, a_i)\}$.

3 Partitioning Acceptors

Let $A = (Q, I, F, \delta)$ be any acceptor. Any partition π of Q, defines another acceptor $A/\pi = (Q', I', F', \delta')$ defined as follows:

$$\begin{aligned} Q' &= \{B : [q]_\pi \text{ such that } q \in Q\} \\ I' &= \{B : [q]_\pi \text{ such that } q \in I\} \\ F' &= \{B : [q]_\pi \text{ such that } q \in F\} \\ \delta'([q]_\pi, a) &= \{[q']_\pi : q' \in \delta(q, a)\} \end{aligned}$$

For any acceptor A and π over the states of A it follows that if $p \in \delta(q, u)$ then $[p]_\pi \in \delta'([q]_\pi, u)$. Hence $L(A/\pi)$ includes all strings in $L(A)$, possibly more.

Remark 1. For any A and π over Q, if $p = q_0 \dots q_k$ is a path in A which is not a loop and $[q_0]_\pi = [q_k]_\pi$ then $p' = [q_0]_\pi \dots [q_k]_\pi$ is a *new loop* in A/π.

Remark 2. Consider $PT(S)$ and any π over the states of PT, and consider any state B in $PT(S)/\pi$. If $|B| = 1$ then $I_1(B) \leq 1$. In other words, states which are not merged with others have at most one 1-leader. Similarly for any state B in $ST(S)/\pi$, if $|B| \leq 1$ then $O_1(B) \leq 1$.

It is also known that if a sample of words of some regular language L is sufficient—that is if generated by $A_T(L)$ then every transition in $A_T(L)$ would be exercised—then there exists some partition π_T of $PT(S)$ such that $PT(S)/\pi_T$ is isomorphic to $A_T(L)$ [5]. Similarly, it follows that if generating S exercises every transition in the head canonical acceptor, that there is some some partition π_H of $ST(S)$ such that $ST(S)/\pi_H$ is isomorphic to $A_H(L)$.

The merging procedure above is independent of the decision of state-equivalence. The latter aspect determines the generalization strategy of the learner, and ultimately the class of languages that can be learned (if any). [7] proposes a general state-merging algorithm IM1 based partly on this observation, and functions $f : Q \to A$, which naturally induce a partition π_f over Q: for all $q_1, q_2 \in Q$, $[q_1]_{\pi_f} = [q_2]_{\pi_f}$ iff $f(q_1) = f(q_2)$. IM1 computes $PT(S)/\pi_f$.

Here IM1 is generalized to compute $M(S)/\pi_f$, where $M(S)$ is any well-defined FSA recognizing the sample. This study limits M to prefix and suffix trees. It

follows that the class of languages obtained by an algorithm which computes $M(S)/\pi_f$ depends not only on f but on M as well.

For example, if $M = PT$ and $f = I_k$ then the language class obtained is the Locally $(k+1)$ Testable in the Strict Sense (LTSS) [6]. If only start states are merged in the prefix tree (i.e $M = PT$ and $f = start$), or only final states in the suffix tree ($M = ST$ and $f = final$), it is easy to see that $\mathcal{L}_{fin}$ is the class of languages obtained (since no states are actually merged). In §§5-6 it is shown that the left-to-right iterative languages (LRI) and right-to-left iterative languages (RLI) are the language classes obtained by merging final states in the prefix tree and start states in the suffix tree, respectively. Table 1 summarizes the known language classes identifiable in the limit from positive data by varying parameters M and f. The fact that language classes of some cells are the reverse of language classes of other cells reveals an underlying algebra, whose exact properties await future discovery.

4 The Neighborhood-Distinct Hypothesis

Part of the motivation for filling in Table 1 comes from the hypothesis that all natural language phonotactic patterns are *1-1 neighborhood-distinct* [21]. The *j-k neighborhood* of a state is the tuple

$$nd_k^j(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

For example, the 1-1 neighborhood of state 5 in Fig. 1 is $(\{g\}, \{h, i\}, 0, 0)$. An acceptor is *j-k neighborhood-distinct* iff every state has a unique j-k neighborhood. The languages of such acceptors are called *j-k neighborhood-distinct* (j-k ND).

[21] shows three main classes of phonotactic patterns—patterns over adjacent segments, long distance patterns such as vowel and consonantal harmony, and rhythmic patterns—are 1-1 ND. These patterns crosscut the Subregular Hierarchy. For example, Navajo and Sarcee have sibilant harmony patterns [28]. In Navajo, this pattern is symmetric: the sibilant sound [s] may not precede [ʃ] (sounds *s* and *sh*, respectively) in a word, even if they are separated by arbitrarily many other sounds, and vice versa. In Sarcee, the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]. The Navajo pattern is Locally 1-Testable since one can decide if a string obeys the sibilant harmony pattern by

Table 1. Language Classes Obtained by Merging States in Prefix and Suffix Trees

f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
$final$	LRI	$\mathcal{L}_{fin}$
$start$	$\mathcal{L}_{fin}$	RLI
$nonfinal$	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
$nonstart$	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?

checking whether [s] and [ʃ] are both present in the string. This procedure does not work for Sarcee since the order matters; hence it is Noncounting (also called Locally Testable with Order [26]).[2] These patterns are shown in Table 2 where C represents any consonant except sibilants, *s* sibilants like [s], *V* any vowel, and ʃ sibilants like [ʃ].

More examples comes from the different kinds of ways languages stress syllables in words [29]. In Sierra Miwok, main stress falls on the initial syllable if it is 'heavy', else on the peninitial syllable.[3] Secondary stress falls on all other heavy syllables. On the other hand, in Kwakwala, main stress falls on the leftmost 'heavy' syllable, but if there are none, on the final syllable. Patterns like Sierra Miwok are called *bounded* and are mostly 3-LTSS. Patterns like Kwakwala are called *unbounded* and Noncounting. Table 2 shows regular expressions for these patterns. The symbols *L* and *H* indicate light and heavy syllables, and acute and grave accents main and secondary stress, respectively. It is easy to verify that the examples in Table 2 are all 1-1 ND by drawing acceptors for them.

Table 2. Phonotactic Patterns

Pattern	Example Language	Regular Expression
Symmetric Harmony	Navajo	(C+V+ʃ)*+(C+V+s)*
Asymmetric Harmony	Sarcee	(C+V+ʃ)*(C+V+s)*
Bounded Stress	Sierra Miwok	(LH́ + LĹ + H́)(H̀+L)*
Unbounded Stress	Kwakwala	(L*H́(H+L)*) + (L*L)

There is currently no language-theoretic characterization of the j-k ND class due partly to its complexity. Also, [30,21,31] present an algorithm like IM1 which returns the intersection of the acceptors obtained by merging same-1-1-neighborhood states in prefix and suffix trees of the observed sample. While this algorithm appears to identify many 1-1 ND patterns (including almost all attested phonotactic patterns), it does not identify the class.[4] It remains an open question exactly what class is identified by this procedure. The line of research here aims to understand the primitive components that make up the neighborhood (see Table 1). The idea is that this language class is some composition of language classes obtained by simpler learners like the ones in Table 1 acting in concert (cf. [33]). Classes LRI and RLI are a small step towards this goal since

[2] [21] defines the precedence languages which contain long-distance harmony patterns. This class is identifiable in the limit from positive data with a string extension learner.

[3] A heavy syllable is typically more sonorous than a light syllable. Languages may make a heavy/light syllable distinction in different ways. See [29].

[4] For fixed j-k, the j-k ND class is finite, and so there are many learners which can learn this class [32]. However it is interesting to consider learners which generalize on the basis of hypothesized universal properties of natural language patterns, see [16] for discussion.

they relate to the indicator functions $[\mathbf{q} \in \mathbf{F}]$ and $[\mathbf{q} \in \mathbf{I}]$, which are boolean compositions of the functions $final$, $nonfinal$ and $start$, $nonstart$, respectively.

5 Left-to-Right Iterative Languages

LRI languages are defined as the intersection of the following two classes:

1. $\mathcal{L}_{1fd} = \{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$.
2. $\mathcal{L}_{LL^*fin} = \{L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}\}$

LRI languages are so named because words fix some strings to the left edge and then may iterate other strings rightward. As shown below, *LRI* languages are exactly the ones recognizable by acceptors which are forward deterministic, have at most one final state, and whose loops, if there are any, pass through the final state, if there is one. A schematic is given in Fig. 1.

Additionally, *LRI* is identifiable in the limit by a learner which computes $PT(S)/\pi_{final}$. Although this acceptor is not necessarily deterministic, it has one final state, all of its loops pass through this final state, and it can be made deterministic without altering those properties or the language recognized.

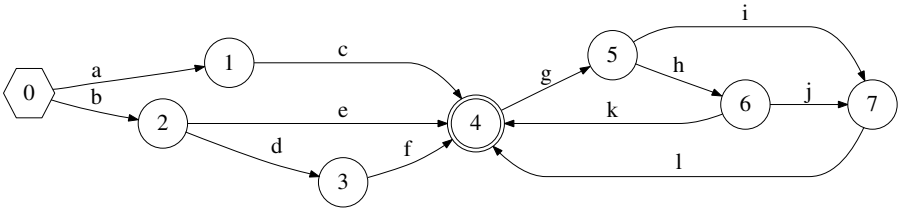

Fig. 1. Schematic of acceptors recognizing *LRI* languages

Call the class of languages recognized by acceptors which are forward deterministic with at most one final state *1-final-deterministic* and denote this class with $\mathcal{L}_{1fd}$. The proof of Theorem 2 is straightforward and thus omitted.

Theorem 2. *L is 1-final-deterministic iff whenever $u, v \in L$, $T_L(u) = T_L(v)$.*

Note that neither $\mathcal{L}_{LL^*fin}$ nor $\mathcal{L}_{1fd}$ are identifiable in the limit from positive data. $\mathcal{L}_{LL^*fin}$ contains all finite languages and at least one infinite language. As for $\mathcal{L}_{1fd}$, a limit point proof [32] establishes this claim, which is sketched here. Since $\{abc\}$, $\{abc, abbc\}$, $\{abc, abbc, abbbc\}$, $\ldots ab^*c$ all belong to $\mathcal{L}_{1fd}$, no learner can identify this subset in the limit from positive data—and hence not $\mathcal{L}_{1fd}$. Although neither $\mathcal{L}_{LL^*fin}$ nor $\mathcal{L}_{1fd}$ is identifiable in the limit, their intersection (LRI) is. Below is an automata-theoretic characterization of LRI.

Theorem 3. *$L \in LRI$ iff (1) $A_T(L) = (Q, I, F, \delta)$ is 1-final-deterministic and (2) if L is infinite, then every loop in $A_T(L)$ passes through the final state, i.e. $F \subseteq \bigcap_{p \in loops(A)} range(p)$.*

Proof. Consider any $L \in LRI$. Both directions of (1) follow from Theorem 2. Now assume L is infinite and let q_f denote the unique final state. Note $L = L_1 \cdot L_2^*$ where $L_1, L_2 \in \mathcal{L}_{fin}$. Suppose there is a loop $p = q_0 q_1 \ldots q_k q_0$ in A such that no $q_i = q_f$. Let $v \in strings(p)$. Since $A_T(L)$ is trimmed, there are u, w such that $\delta(I, u) = q_0$ and $\delta(q_0, w) = q_f$. It follows that $uv^*w \subseteq L$. Since $L = L_1 \cdot L_2^*$ and since p does not contain q_f, either $uv^* \subseteq L_1$ or $v^*w \subseteq L_2$. But this is a contradiction, since $L_1, L_2 \in \mathcal{L}_{fin}$. Thus every loop must pass through q_f. It remains to be shown that if all loops pass through the unique final state of $A_T(L)$, that $L = L_1 \cdot L_2^*$, $L_1, L_2 \in \mathcal{L}_{fin}$. It can be shown that L_1 is the language recognized by the largest trimmed acyclic subacceptor of $A_T(L)$ and L_2 is the union of $strings(p)$ for all loops $p = q_f q_1 \ldots q_k q_f$. □

Remark 3. It follows if $L \in LRI$ then the language of any subacceptor of $A_T(L)$ is also left-to-right iterative.

Remark 4. In terms of regular expressions, it follows that a language L belongs to LRI if $L = (a_0 + a_1 + \ldots a_n)(b_0 + b_1 + \ldots b_m)^*$ for $n, m \in \mathbb{N}$, $a_i, b_i \in \Sigma^*$ and for all i, j, no a_i (b_i) is a proper prefix of a_j (b_j).

With Theorem 3 and Remark 4 it is easy to see that the stress pattern of Miwok is in LRI, but not the other patterns in Table 2. LRI languages do not appear to characterize phonotactic patterns in general, even though they are related to them via the composition which constructs the neighborhood.

Also, it can be shown that LRI languages are incomparable with the ZR languages. In Fig. 1 if $j = i$ then the acceptor still accepts a language in LRI, but it is no longer backward deterministic and therefore its language is not in ZR. Similarly if instead we add a transition from state 1 to itself labeled g, then the language of the acceptor belongs to ZR, but not LRI.

The next lemma illustrates how generalization takes place, and is used to establish the fact that every language in LRI has a characteristic sample.

Lemma 1. *Let $L \in \mathcal{L}_{1fd}$ and let $x, y_i \in \Sigma^*$ for $0 \leq i \leq k$ such that $x, xy_i \in L$. Then $x(y_1 + y_2 + \ldots + y_k)^* \subseteq L$.*

Proof. For some $k \in \mathbb{N}$, let $x, xy_1, xy_2, \ldots xy_k \in L$. By induction on n, it is shown $x(y_1 + y_2 + \ldots + y_k)^n \subseteq L$. Clearly when $n = 0$, $x \in L$. Now assume for some $n \in \mathbb{N}$, if $x, xy_1, xy_2, \ldots xy_k \in L$ then $x(y_1 + y_2 + \ldots + y_k)^n \subseteq L$. It remains to be shown that for all $1 \leq i \leq k$, $x(y_1 + y_2 + \ldots + y_k)^n y_i \subseteq L$. For any $w \in L$, $y_i \in T_L(w)$ because $x, xy_i \in L$ and $L \in \mathcal{L}_{1fd}$ so by Theorem 2, $T_L(w) = T_L(x)$. By the inductive hypothesis, for all $w \in x(y_1 + y_2 + \ldots + y_k)^n$, $w \in L$ and so therefore $wy_i \in L$. It follows that $x(y_1 + y_2 + \ldots + y_n)^{n+1} \subseteq L$. □

Theorem 4. *For $L \in LRI$, there exists a characteristic sample S_L.*

Proof. For any $L \in LRI$, let $L_1, L_2 \in \mathcal{L}_{fin}$ such that $L = L_1 \cdot L_2^*$. Then $S = L_1 \cup L_1 \cdot L_2$ is characteristic. Note S is finite. Let $L' \in LRI$ containing S. Consider any $w \in L$. It is sufficient to show that $w \in L'$. Since $w \in L$ and $L \in LRI$, there is some $k \in \mathbb{N}$ such that $w = xy_1y_2 \ldots y_k$ where $x \in L_1$ and for any $1 \leq i \leq k$, $y_i \in L_2$. It is also the case that $x, xy_i \in S$. Since $S \subseteq L'$, $x(y_1 + y_2 + \ldots + y_k)^* \subseteq L'$ by Lemma 1. Clearly $w \in x(y_1 + y_2 + \ldots + y_k)^*$ and thus is in L'. □

As an example, if $L = L_1 \cdot L_2^*$ is in LRI and $L_1 = \{u, v\}$ and $L_2 = \{x, y, z\}$ then $S_L = \{u, v, ux, uy, uz, vx, vy, vz\}$. Since a characteristic sample S_L for any $L \in LRI$ exists, a learner guessing L after exposure to S_L is picking the smallest LRI language consistent with S.

For all L in LRI, the size of S_L grows polynomially with respect to the size of $A_T(L)$ (usually measured in states, see [35]). This is primarily because $L = L_1 \cdot L_2$ where L_1 and L_2 are finite languages, and thus the size of $A_T(L)$ is in the worst case approximates $length(L_1) + length(L_2)$. Now the length of S_L equals $(|L_2|+1)length(L_1)+|L_1|length(L_2)$. Since for finite L, $|L| \leq length(L)+1$, the size of S_L is bounded by a quadratic function over $length(L_1)$ and $length(L_2)$.

$PT(S)/\pi_{final}$ is not necessarily deterministic. We show that application of S-UPDATE, an algorithm which merges states that are b-successors of some state, returns an equivalent acceptor which is deterministic.

Algorithm 1. Pseudo-code for S-UPDATE (forward determinize)

Input: an acceptor A.
Output: a forward deterministic acceptor A'.
Initialization
Let $A_0 = (Q_0, I_0, F_0, \delta_0)$, π_0 the trivial partition of Q_0, and $i = 0$.
Let LIST contain all pairs (q_1, q_2) such that q_1 and q_2 are distinct b-successors of some $q_0 \in Q$ for all $b \in \Sigma$.
Merging
while LIST $\neq \emptyset$ **do**
  Remove some element (q_1, q_2) from LIST.
  Let π_{i+1} be the one obtained by merging blocks $[q_1]_{\pi_i}$ and $[q_2]_{\pi_i}$
  For all $[q], [r], [s]$ in π_{i+1}, $b \in \Sigma$, add (r, s) to LIST iff $[r]$ and $[s]$ are distinct b-successors to $[q]$.
  Increase i by one.
end while
Termination: Let $f = i$ and output the acceptor A_0/π_f.

In Fig. 1, for example, if $h = i$ then S-UPDATE removes this nondeterminism by merging states 6 and 7. Note this merge does not alter the relevant character of the acceptor or the language recognized by the acceptor. If merging two states creates additional nondeterminism (e.g. $l = k$ in Fig. 1), then the b-successors of the source of non-determinism are added to LIST.

Theorem 5. *Let S be any finite sample. Then $L(PT(S)/\pi_{final}) \in LRI$.*

Proof. Let $A = PT(S)/\pi_{final}$. Let A' be the acceptor obtained by submitting A to S-UPDATE which removes sources of nondeterminism by merging states. It is sufficient to show $L(A') \in LRI$ and $L(A') = L(A)$. The proof is by induction. The assumptions here are inductive hypothesis. Assume there is a partition of π_i such that $A/\pi_i = (Q_i, I_i, F_i, \delta_i)$ where

1. there is only one final state $[q_f]_{\pi_i}$,
2. all loops in A pass through $[q_f]/\pi_i$,

3. no state other than $[q_f]_{\pi_i}$ has more than one 1-leader,
4. $L(A/\pi_i) = L(A)$.
5. if there exist distinct $q_0, q_1, q_2 \in Q_i$, and $b \in \Sigma$ such that $[q_1]_{\pi_i}$ and $[q_2]_{\pi_i}$ are b-successors to $[q_0]_{\pi_i}$, then any path connecting $[q_1]_{\pi_i}$ to $[q_2]_{\pi_i}$ (or vice versa) goes through $[q_0]_{\pi_i}$

Note if the conditional in (5) is false and (1-5) holds, then $L(A/\pi_i)$ is deterministic and hence A/π_i belongs to LRI.

If the conditional in (5) is true, we show the acceptor obtained by merging $[q_1]_{\pi_i}$ and $[q_2]_{\pi_i}$ eliminates this nondeterminism but maintains properties (1-4). Let π_{i+1} obtain by merging $[q_1]\pi_i$ and $[q_2]_{\pi_i}$. Since π_i has only one final state, π_{i+1} must as well (1). If $[q_1]\pi_i \neq [q_f]\pi_i$ and $[q_2]\pi_i \neq [q_f]\pi_i$ then it follows that the 1-leaders of $[q_1]\pi_i$ is $\{b\}$, as it is for for $[q_2]_{\pi_i}$. Therefore $[q_1]\pi_{i+1} = [q_2]\pi_{i+1}$ also has one 1-leader, namely $\{b\}$. On the other hand, if either $[q_1]\pi_i$ or $[q_1]_{\pi_i}$ equals $[q_f]_{\pi_i}$ then the b-successors of $[q_f]_{\pi_{i+1}} = [q_f]_{\pi_i}$. In any situation, it follows that no state other than $[q_f]_{\pi_{i+1}}$ has more than one 1-leader (3). Thus merging $[q_1]/\pi_i$ and $[q_2]/\pi_i$ creates no loops that do not go through $[q_f]/\pi_{i+1}$. It follows that all loops in A/π_{i+1} pass through $[q_f]/\pi_{i+1}$ (2). Finally it follows that $L(A/\pi_{i+1}) = L(A/\pi_i)$ since any path connecting $[q_1]_{\pi_i}$ and $[q_2]_{\pi_i}$ (or vice versa) goes through $[q_0]_{\pi_i}$ (4). Thus this merging eliminates the nondeterminism at $[q_0]_{\pi_i}$ but maintains properties (1-4).

The base of the induction is established with π_0 the trivial partition of A. Since only final states are merged in $PT(S)$, A contains only 1 final state (1). Also, since $PT(S)$ is acyclic, all loops in A pass through q_f as any loops are the result of merging states (Remark 1) (2). No state other than q_f has more than one 1-leader (Remark 2) (3). Also, clearly $L(A/\pi_0) = L(A)$ (4). This completes the induction and so the final partition obtained by S-UPDATE π_f is such that $L(A) = L(A/\pi_f) \in LRI$. □

Supplementing $PT(S)/\pi_{final}$ with Algorithm 1 provides an algorithm almost identical to the algorithm ZR in [5]. In ZR, pairs of states are placed on LIST if they are to be merged. LIST is initialized to include all final states, and pairs of states are added if they are a source of non-forward-determinism or non-reverse-determinism. Adding S-UPDATE to the computation of $PT(S)/\pi_{final}$ is exactly the same except there is no procedure for updating LIST when two blocks share the same b-predecessors (i.e. reverse-determinism is not enforced). Because it does strictly less than ZR, which is tractable, supplementing $PT(S)/\pi_{final}$ with Algorithm 1 is also tractable.

The lemma and theorems below establish that a learner which computes $PT(S)/\pi_{final}$ at each point in the text identifies LRI in the limit.

Lemma 2. *Let S be any nonempty positive sample, $PT(S)$ the prefix tree for S, and π_f the final partition found by applying Algorithm 1 to $PT(S)/\pi_{final}$. Then π_f is the finest partition π such that $(PT(S)/\pi_{final})/\pi_f$ is LRI.*

Proof. The proof (by induction) is essentially identical to the second part of Lemma 25 in [5]. □

Theorem 6. *Let S be any nonempty finite sample. Then $L(PT(S)/\pi_{final})$ is the smallest language in LRI which contains S.*

Proof. Theorem 5 establishes that $L(PT(S)/\pi_{final}) \in LRI$. Let L be any LRI language containing S and let π be the restriction of the partition $\pi_L$to the elements of $Pref(S)$. Lemma 1 shows that $PT(S)/\pi$ is isomorphic to a subacceptor of $A_T(L)$, and it follows that $L(PT(S)/\pi)$ is contained in L. Theorem 3 shows that $A_T(L)$ is LRI, and thus $PT(S)/\pi$ is LRI, by Remark 3. By Lemma 2, π_f therefore refines π. Hence, $L(PT(S)/\pi)$ is contained in L, and $L(PT(S)/\pi_{final})$ is the smallest LRI language containing S. □

Theorem 7. *LRI is identifiable in the limit from positive data.*

Proof. Let $\phi(t) = PT(S_t)/\pi_{final}$. By Theorem 4, L contains a characteristic sample S_0. For any text T for $L \in LRI$, there is a i such that $S_0 \subseteq range(T_i)$. For $n \geq i$, $L(PT(T_i)/\pi_f)$ is the smallest LRI language containing T_i by Theorems 5 and 6. Since T_i contains characteristic S_0, this is L. Thus ϕ converges to L. □

6 Right-to-Left Iterative Languages

RLI languages are defined as the intersection of the one-start-reverse deterministic languages ($\mathcal{L}_{1srd}$) and the reverse of $\mathcal{L}_{LL^*fin}$:

1. $\mathcal{L}_{1srd} = \{L : \text{whenever} u, v \in L, H_L(u) = H_L(v)\}$.
2. $\mathcal{L}_{L^*Lfin} = \{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

RLI languages are the reverse of languages in LRI. RLI languages are exactly the ones recognizable by acceptors which are reverse deterministic, have at most one start state, and whose loops, if there are any, pass through the start state, if there is one. A schematic is obtained by reversing the acceptor in Fig. 1. RLI is is identifiable in the limit from a process which essentially merges initial states in suffix trees. The characteristic sample for language $L_1^* \cdot L_2 \in RLI$ is $L_1 \cdot L_2 \cup L_2$. The theorems establishing these results parallel exactly those of §5. Simply subsitute $ST(S)$, $start$, $A_H(L)$, b-predecessors, P-UPDATE and so on for $PT(S)$, $final$, $A_T(L)$, b-successors, and S-UPDATE, respectively. (P-UPDATE eliminates reverse determinisim by merging states with the same b-predecessors.)

Finally, [10] introduces a family of function-distinguishable language classes, of which ZR is one such class. It would be interesting to relate the results here to the ones obtained in that work.

7 Conclusion

LRI and RLI languages are previously unnoticed language classes which are infinite in size, identifiable in the limit from positive data, closely related to the zero-reversible languages, and relevant to a hypothesis regarding a universal property of phonotactic patterns. Understanding these classes not only begins

to shed light on the neighborhood-distinct hypothesis, but also on the algebra underlying the state-merging operations, the reverse operator, prefix and suffix trees, and tail and head canonical acceptors. I hope this algebra is soon made clear, that the question marks in Table 1 are soon filled, and that future research investigates complex functions, like the neighborhood function, which are defined compositionally in terms of simpler functions. Finally, the day is not far off for three communities with overlapping interests to come together to develop a successful research program: the grammatical inference community which understands how different kinds of logically possible patterns could be learned, linguists who are familiar with natural language patterns, and acquisitionists who are experimenting with models of how children acquire grammar.

References

1. Gold, E.: Language identification in the limit. Information and Control 10, 447–474 (1967)
2. Marcus, G.: Negative evidence in language acquisition. Cognition 46, 53–85 (1993)
3. Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62 (1980)
4. Angluin, D.: Inductive inference of formal languages from positive data. Information Control 45, 117–135 (1980)
5. Angluin, D.: Inference of reversible languages. Journal for the Association of Computing Machinery 29(3), 741–765 (1982)
6. Garcia, P., Vidal, E., Oncina, J.: Learning locally testable languages in the strict sense. In: Proceedings of the Workshop on Algorithmic Learning Theory, pp. 325–338 (1990)
7. Muggleton, S.: Inductive Acquisition of Expert Knowledge. Addison-Wesley, Reading (1990)
8. Kanazawa, M.: Identification in the limit of categorical grammars. Journal of Logic, Language, and Information 5, 115–155 (1996)
9. Denis, F., Lemay, A., Terlutte, A.: Some classes of regular languages identifiable in the limit from positive data. In: Adriaans, P.W., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 63–76. Springer, Heidelberg (2002)
10. Fernau, H.: Identification of function distinguishable languages. Theoretical Computer Science 290, 1679–1711 (2003)
11. Yokomori, T.: Polynomial-time identification of very simple grammars from positive data. Theoretical Computer Science 298(1), 179–206 (2003)
12. Oates, T., Armstrong, T., Bonache, L.B.: Inferring grammars for mildly context-sensitive languages in polynomial-time. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 137–147. Springer, Heidelberg (2006)
13. Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. Journal of Machine Learning Research 8, 1725–1745 (2007)
14. Greenberg, J.: Some universals of grammar with particular reference to the order of meaningful elements. In: Universals of Language, pp. 73–113. MIT Press, Cambridge (1963)
15. Mairal, R., Gil, J. (eds.): Linguistic Universals. Cambridge University Press, Cambridge (2006)

16. Stabler, E.P.: Computational models of language universals: Expressiveness, learnability and consequences. In: Cornell Symposium on Language Universals (2007)
17. Shieber, S.: Evidence against the context-freeness of natural language. Linguistics and Philosophy 8, 333–343 (1985)
18. Kobele, G.: Generating Copies: An Investigation into Structural Identity in Language and Grammar. PhD thesis, University of California, Los Angeles (2006)
19. Johnson, C.D.: Formal Aspects of Phonological Description. Mouton, The Hague (1972)
20. Kaplan, R., Kay, M.: Regular models of phonological rule systems. Computational Linguistics 20(3), 331–378 (1994)
21. Heinz, J.: The Inductive Learning of Phonotactic Patterns. PhD thesis, University of California, Los Angeles (2007)
22. Pullum, G., Rogers, J.: Aural pattern recognition experiments and the subregular hierarchy. In: Kracht, M. (ed.) Proceedings of 10th Mathematics of Language Conference, pp. 1–7. University of California, Los Angeles (2007)
23. Jusczyk, P., Cutler, A., Redanz, N.: Infants' preference for the predominant stress patterns of english words. Child Development 64, 675–687 (1993)
24. Jusczyk, P., Luce, P., Charles-Luce, J.: Infants' sensitivity to phonotactic patterns in the native language. Journal of Memory and Language 33, 630–645 (1994)
25. Mattys, S., Jusczyk, P.: Phonotactic cues for segmentation of fluent speech by infants. Cognition 78, 91–121 (2001)
26. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
27. Kracht, M.: The Mathematics of Language. Mouton de Gruyter, Berlin (2003)
28. Hansson, G.: Theoretical and typological issues in consonant harmony. PhD thesis, University of California, Berkeley (2001)
29. Hayes, B.: Metrical Stress Theory. Chicago University Press (1995)
30. Heinz, J.: Learning quantity insensitive stress systems via local inference. In: Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology at HLT-NAACL, New York City, USA, pp. 21–30 (2006)
31. Heinz, J.: On the role of locality in learning stress patterns. Phonology (to appear)
32. Jain, S., Osherson, D., Royer, J.S., Sharma, A.: Systems That Learn: An Introduction to Learning Theory, 2nd edn. The MIT Press, Cambridge (1999)
33. Case, J., Moelius, S.: Parallelism increases iterative learning power. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 49–63. Springer, Heidelberg (2007)
34. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (2001)
35. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning 27, 125–138 (1997)