On Languages Piecewise Testable in the Strict Sense

James Rogers[1], Jeffrey Heinz[2,⋆], Gil Bailey[1], Matt Edlefsen[1], Molly Visscher[1], David Wellcome[1], and Sean Wibel[1]

[1] Dept. of Computer Science, Earlham College
[2] Dept. of Linguistics and Cognitive Science, University of Delaware

Abstract. In this paper we explore the class of Strictly Piecewise languages, originally introduced to characterize long-distance phonotactic patterns by Heinz [7] as the Precedence Languages. We provide a series of equivalent abstract characterizations, discuss their basic properties, locate them relative to other well-known subregular classes and provide algorithms for translating between the grammars defined here and finite state automata as well as an algorithm for deciding whether a regular language is Strictly Piecewise.

1 Introduction

From the beginning of the generative linguistics program, long-distance dependencies in natural language have attracted considerable interest. For example, [2] establishes that the long-distance dependencies necessary to describe sentence well-formedness are beyond the reach of finite state methods, and later work continues to characterize the kinds of non-local dependencies in natural language in ways which require increasingly expressive formalisms [11,21,12].

Although many long-distance dependencies in natural language require expressive formalisms that are at least context-free [2,11,21,12], some non-local patterns in natural language do not. An example from Heinz [7,8] comes from the sibilant harmony process of Sarcee, where [-anterior] sibilants like [ʃ] and [ʒ] regressively require [+anterior] sibilants like [s] and [z] to assimilate in anteriority, but not vice versa [3,4].[1] As a consequence of this phonological rule, there are no words in Sarcee where [-anterior] sibilants may follow [+anterior] sibilants as in (1b), though the reverse is possible (1a) (data from Cook [3]). In the examples in (1), witness that words are well-formed when the [+anterior] sibilant [z] follows a [-anterior] sibilant like [ʃ] in (a) 'my duck,' but there are no words in Sarcee where [-anterior] sibilants like [ʃ] may follow [+anterior] sibilants like [s], as in the hypothetical example in (1c).

⋆ The author acknowledges the support of a 2008-2009 University of Delaware Research Fund Grant.

[1] Linguistic descriptions of Sarcee (and many other languages with consonantal harmony [6,18]) are clear that agreeing consonants can be arbitrarily distant.

C. Ebert, G. Jäger, and J. Michaelis (Eds.): MOL 10/11, LNAI 6149, pp. 255–265, 2010.

1. a. /si-tʃiz-aʔ/ → **ʃítʃídzàʔ** 'my duck' c. cf. ***sítʃídzàʔ**
   b. /na-s-ɣatʃ/ → **nāʃɣátʃ** 'I killed them again'

Heinz [7,8] observes that these kinds of long-distance dependencies can be described according to the well-formedness of subsequences: in Sarcee discontiguous subsequences like [ʃz] and [ʃs] are well-formed but discontiguous subsequences like [zʃ] and [sʃ] are not (since the phonological rule requires the [s] to become [ʃ] when followed by [ʃ]).

The Piecewise Testable (PT) languages [22] are a subclass of the regular languages that can describe this kind of non-local pattern. These languages are similar, in many respects, to the Locally Testable (LT) languages [16,17] except that the two classes differ in how they determine an expression's well-formedness: for LT languages, an expression's well-formedness depends entirely on the set of *contiguous* subsequences (up to some length k, known as *k-factors*) in the expression, whereas for PT languages, an expression's well-formedness depends entirely on the set of subsequences (not necessarily contiguous and up to some length k) found within the expression.

In fact, Sarcee-like non-local patterns are describable by a proper subclass of the PT languages, which we call Strictly Piecewise (SP). This name reflects the fact that the relationship between the SP languages and the PT languages is precisely analogous to the the relationship between the Strictly Local (SL) languages and the LT languages [16,17]. The SP class completes a *dual hierarchy* of subregular language classes with the Local branch being characterized by immediate adjacency (successor) and the Piecewise branch by precedence (less-than):

- SP and LT are the languages definable as intersections of certain simple negative constraints, i.e., as conjunctions of complements of atomic formulae which are satisfied by strings that contain a specified subsequence or, respectively, factor (the so called forbidden subsequences/factors).
- PT and LT are the languages definable by arbitrary propositional formulae of this sort.
- The Star Free (SF) languages and the Locally Threshold Testable (LTT) languages are the languages that First-Order (FO) definable over sequences with less-than and successor, respectively. Since successor is FO definable from less-than, LTT is a subclass of SF.
- The Regular languages are those that are Monadic Second-Order (MSO) definable over sequences with either less-than or successor.

Strikingly, SP turns out to be exactly the class of languages which are closed under subsequence.

The structure of the paper is as follows. Section 2 defines basic notation. Section 3 reviews the Piecewise Testable languages. Most of the results in this section are well-known (see [22], [19], [15], [13], [23]); the rest are probably best attributed to folklore. The primary contributions of this paper are in Sections 4 and 5. Section 4 defines the Strictly Piecewise Testable languages, explores some of their properties and provides a number of abstract characterizations of the

class. Section 5 presents algorithms for extracting a SP_k grammar from a minimal Deterministic Finite-State Automaton (DFA) recognizing a SP_k language and for constructing a minimal DFA recognizing an SP_k language from its grammar. Together these provide an algorithm for deciding if an arbitrary regular language is SP and, if it is, for determining the least k for which it is SP_k. In section 6 we consider the parallels between the Piecewise Testable and Locally Testable hierarchies from a descriptive perspective.

2 Preliminaries

We start with some mostly standard notation. $\mathcal{P}(S)$ denotes the power set of the set S; $S_1 - S_2$ set-theoretic difference. Σ denotes a finite set of symbols and a string over Σ is a finite sequence of symbols drawn from that set. Σ^k, $\Sigma^{\leq k}$, Σ^* denote all strings over this alphabet of length k, of length less than or equal to k, and of any finite length, respectively. ϵ denotes the empty string. $|w|$ denotes the length of string w and $|w|_\sigma$ denotes the number of occurrences of $\sigma \in \Sigma$ in w. A language L is a subset of Σ^*; $\overline{L}$ its complement relative to Σ^*. Concatenation of sets of strings is denoted $L_1 \cdot L_2 = \{uv \mid u \in L_1 \text{ and } v \in L_2\}$. When discussing partial functions, we use the notation $\uparrow$ and $\downarrow$ to indicate that the function is undefined, respectively is defined, for some particular arguments.

A *Deterministic Finite-state Automaton.* (DFA) is a tuple $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F\rangle$ where Q is the state set, Σ is the alphabet, q_0 is the start state, δ is a deterministic transition function and F is the set of accepting states. Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be the (partial) path function of $\mathcal{M}$, i.e., $\hat{\delta}(q, w)$ is the (unique) state reachable from state q via the sequence w, if any, or $\hat{\delta}(q, w)\uparrow$ otherwise. The language recognized by a DFA $\mathcal{M}$ is $L(\mathcal{M}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(q_0, w)\downarrow \in F\}$.

Two strings w and v over Σ are *distinguished* by a DFA $\mathcal{M}$ iff $\hat{\delta}(q_0, w) \neq \hat{\delta}(q_0, v)$. They are *Nerode equivalent* with respect to a language L if and only if $wu \in L \iff vu \in L$ for all $u \in \Sigma^*$. All DFAs which recognize L must distinguish strings which are inequivalent in this sense, but no DFA recognizing L necessarily distinguishes any strings which are equivalent. Hence the number of equivalence classes of strings over Σ modulo Nerode equivalence with respect to L gives a (tight) lower bound on the number of states required to recognize L. A DFA is *minimal* if the size of its state set is minimal among DFAs accepting the same language. A minimal DFA is *trimmed* if the (unique) sink state has been removed. The reader is referred to [10] for details.

The relation between strings which is fundamental to Piecewise Testability is the *subsequence* relation, which is a partial order on Σ^*:

$$w \sqsubseteq v \stackrel{\text{def}}{\iff} w{=}\varepsilon \text{ or } w{=}\sigma_1 \cdots \sigma_n \text{ and } (\exists w_0, \ldots, w_n \in \Sigma^*)[v{=}\, w_0\sigma_1 w_1 \cdots \sigma_n w_n].$$

in which case we say w is a *subsequence* of v. The subsequence relation is compatible with concatenation: $w_1 \sqsubseteq v_1$ and $w_2 \sqsubseteq v_2$ implies that $w_1 w_2 \sqsubseteq v_1 v_2$.

For $w \in \Sigma^*$, let

$$\mathrm{P}_k(w) \stackrel{\text{def}}{=} \{v \in \Sigma^k \mid v \sqsubseteq w\} \text{ and } \mathrm{P}_{\leq k}(w) \stackrel{\text{def}}{=} \{v \in \Sigma^{\leq k} \mid v \sqsubseteq w\},$$

the set of subsequences of length k, respectively length no greater than k, of w. Let $\mathrm{P}_k(L)$ and $\mathrm{P}_{\leq k}(L)$ be the natural extensions of these to sets of strings. Note that $\mathrm{P}_0(w) = \{\varepsilon\}$, for all $w \in \Sigma^*$, that $\mathrm{P}_1(w)$ is the set of symbols occurring in w and that $\mathrm{P}_{\leq k}(L)$ is finite, for all $L \subseteq \Sigma^*$.

3 Piecewise Testable Languages

The class of Piecewise Testable languages (PT) was introduced by Simon [22] as the Boolean closure of the class of languages of the form $\Sigma^*\sigma_1\Sigma^* \cdots \Sigma^*\sigma_n\Sigma^*$ where $\sigma_1 \cdots \sigma_n$ is a possibly empty word in Σ^*.

Following Sakarovitch and Simon [19], Lothaire [15] and Kontorovich, et al., [13] we call the set of strings that contain w as a subsequence the *(principal) shuffle ideal*[2] of w:

$$\mathrm{SI}(w) = \{v \in \Sigma^* \mid w \sqsubseteq v\}.$$

Then the class of Piecewise Testable (PT) languages is the smallest class of languages including $\mathrm{SI}(w)$ for all $w \in \Sigma^*$ and closed under Boolean operations. Similarly, the class of k-Piecewise Testable (PT_k) languages is the smallest class of languages including $\mathrm{SI}(w)$ for all $w \in \Sigma^{\leq k}$ and closed under Boolean operations.

Extending the notion of shuffle ideal to languages, $\mathrm{SI}(L)$ is the closure of L under inverse $\sqsubseteq$:

$$\mathrm{SI}(L) = \{v \in \Sigma^* \mid \exists w \in L, w \sqsubseteq v\}$$

From a model-theoretic perspective, PT_k is the class of languages definable by propositional formulae in which the atomic formulae are strings in $\Sigma^{\leq k}$ with a string $w \in \Sigma^*$ satisfying a formula $p \in \Sigma^{\leq k}$ iff $w \in \mathrm{SI}(p)$. PT is the class of languages definable by arbitrary finite formulae of this type.

The class of Piecewise Testable languages has a well-known characterization (sometimes taken to be its definition):

Theorem 1. *$L \subseteq \Sigma^*$ is in the class* PT *iff there exists some k such that, for all $w, v \in \Sigma^*$,*

$$P_{\leq k}(w) = P_{\leq k}(v) \Rightarrow (w \in L \Leftrightarrow v \in L).$$

Since $P_{\leq k}(\Sigma^*)$ is finite for all k and Σ, one consequence of this characterization is that a language is in PT iff it is the union of a finite set of equivalence classes modulo the relation $w \equiv_k v \overset{\text{def}}{\iff} P_{\leq k}(w) = P_{\leq k}(v)$. Given this, we can take a PT_k language to be generated by a finite set of subsets of $\Sigma^{\leq k}$.

Definition 1 (PT_k Grammar). *A PT_k grammar is a pair $\mathcal{G} = \langle \Sigma, T \rangle$ where $T \subseteq \mathcal{P}(\Sigma^{\leq k})$. The language licensed by a PT_k grammar is*

$$L(\mathcal{G}) \overset{def}{=} \{w \in \Sigma^* \mid P_{\leq k}(w) \in T\}.$$

[2] Properly $\mathrm{SI}(w)$ is the principal ideal generated by $\{w\}$ wrt the inverse of $\sqsubseteq$.

Note that $L(\mathcal{G}) = \emptyset$ iff $T = \emptyset$ and $\varepsilon \in L(\mathcal{G})$ iff $\{\varepsilon\} \in T$.

Theorem 2. *The classes* PT_k *form a proper hierarchy in* k*:* $(\forall k)[\mathrm{PT}_k \subsetneq \mathrm{PT}_{k+1}]$.

The inclusion follows from the fact that $\equiv_{k+1}$ is a refinement of $\equiv_k$. To see that it is proper, let

$$L_{\leq kb} \stackrel{\text{def}}{=} \{w \in \{a,b\}^* \mid |w|_b \leq k\}.$$

This is not in PT_k since $P_{\leq k}(b^k) = P_{\leq k}(b^{k+1})$. On the other hand, it is in PT_{k+1} since it is the complement of $\mathrm{SI}(b^{k+1})$.

Theorem 3. *The class of finite languages is a proper subset of the class of Piecewise Testable languages.*

Any singleton set $\{w\}$ is $\mathrm{PT}_{|w|+1}$, being the intersection of $\mathrm{SI}(w)$ and all $\overline{\mathrm{SI}(v)}$ for $v \in \Sigma^{|w|+1}$. Hence every finite set L is in PT_k for every k greater than the longest string in L. On the other hand, there is no k for which the class of finite languages is included in PT_k.

Theorem 4. PT *and* PT_k*, for any* $k > 0$*, are not closed under concatenation.*

The languages $L_{\leq kb} = L_{\leq (k-1)b} \cdot L_{\leq 1b}$ witness that PT_k is not closed under concatenation. For the general case consider the language $L_{awb} = \{a\} \cdot \Sigma^* \cdot \{b\}$. This is the concatenation of three PT_2 languages, but it is not, itself, PT. Suppose, by way of contradiction, that it was PT. Then it would be PT_k for some k. But then the string $(ab)^k \in L_{awb}$ while $(ab)^k a \notin L_{awb}$ despite the fact that $P_{\leq k}((ab)^k) = P_{\leq k}(\{a,b\}^*) = P_{\leq k}((ab)^k a)$, contradicting Theorem 1.

Theorem 5 (Simon 1975). *The class of Piecewise Testable languages is a proper subset of the class of Star-Free languages:* $\mathrm{PT} \subsetneq \mathrm{SF}$.

$\mathrm{SI}(w)$, where $w = \sigma_1\sigma_2\cdots\sigma_{|w|}$, is denoted by the SF expression

$$\overline{\emptyset} \cdot \sigma_1 \cdot \overline{\emptyset} \cdot \cdots \cdot \overline{\emptyset} \cdot \sigma_{|w|} \cdot \overline{\emptyset}.$$

and SF is closed under Boolean operations. That the inclusion is proper is witnessed by the fact that $L_{awb} \in \mathrm{SF}$.

Theorem 6. *The class of Star-Free languages is the closure of the class of Piecewise Testable languages under concatenation and Boolean operations.*

SF is the closure of the class of Finite languages under union, concatenation and complement (hence concatenation and Boolean operations).

4 Strictly Piecewise Languages

Languages that are *Locally Testable in the Strict Sense* (Strictly Local, SL) are defined only in terms of the set of k-factors which are licensed to occur in the string (equivalently the complement of that set with respect to $\Sigma^{\leq k}$, the *forbidden factors*) [16]. In this section we introduce the class of languages obtained by the analogous restriction to PT, which we call *Piecewise Testable in the Strict Sense* (Strictly Piecewise, SP).

Definition 2 (SP_k Grammar). *A SP_k grammar is a pair $\mathcal{G} = \langle \Sigma, T \rangle$ where $T \subseteq \Sigma^k$. The language licensed by a SP_k grammar is*

$$L(\mathcal{G}) \stackrel{def}{=} \{w \in \Sigma^* \mid P_{\leq k}(w) \subseteq P_{\leq k}(T)\}.$$

A language is SP_k iff it is $L(\mathcal{G})$ for some SP_k grammar $\mathcal{G}$. It is SP iff it is SP_k for some k.

The SP languages have a variety of characteristic properties.

Theorem 7. *The following are equivalent:*

1. $L = \bigcap_{w \in S} [\overline{\mathrm{SI}(w)}]$, *$S$ finite,*
2. $L \in \mathrm{SP}$,
3. $(\exists k)[P_{\leq k}(w) \subseteq P_{\leq k}(L) \Rightarrow w \in L]$,
4. *$w \in L$ and $v \sqsubseteq w \Rightarrow v \in L$ (L is* subsequence closed*),*
5. $L = \overline{\mathrm{SI}(X)}$, *$X \subseteq \Sigma^*$ (L is the complement of a shuffle ideal).*

Proof. These are each almost immediate consequences of their predecessors.

That 1 implies 2 is witnessed by the SP_k grammar $\langle \Sigma, \Sigma^{\leq k} - S \rangle$, where k is the maximum length of the strings in S.

To see that 2 implies 3, suppose that $L \in \mathrm{SP}$. Then $L \in \mathrm{SP}_k$ for some k and there is some Σ and $T \subseteq \Sigma^{\leq k}$ for which $L = \{w \in \Sigma^* \mid P_{\leq k}(w) \subseteq P_{\leq k}(T)\}$. Then $P_{\leq k}(L) = \bigcup_{w \in L} [P_{\leq k}(w)] \subseteq P_{\leq k}(T)$. That L is closed under Property 3 follows immediately.

That 3 implies 4 follows from the fact that $v \sqsubseteq w \in L \Rightarrow P_{\leq k}(v) \subseteq P_{\leq k}(w) \subseteq P_{\leq k}(L)$.

That 4 implies 5 follows immediately from the definition of $\mathrm{SI}(X)$ since closure of L under subsequence implies that $\overline{L}$ is closed under inverse subsequence.

Finally, 5 implies 1 by DeMorgan's theorem and the fact that every shuffle ideal is finitely generated, which is a consequence of the fact that there are no infinite sequences of strings over a fixed alphabet which are pairwise unrelated by subsequence.[3]

Corollaries: If $L \in \mathrm{SP}_k$ then:

1. $wv \in L \Rightarrow w, v \in L$ (Prefix and Suffix closure),
2. $P_1(L) \subseteq L$ (Unit strings) and
3. $L \neq \emptyset \Rightarrow \varepsilon \in L$ (Empty string).

Theorem 8 (Proper Hierarchy). $(\forall k)[\mathrm{SP}_k \subsetneq \mathrm{SP}_{k+1}]$.

Inclusion follows from Property 3 of Theorem 7 along with the fact that $P_{\leq k}(w) = P_{\leq k}(P_{\leq k+1}(w))$.[4] The same sequence of languages that witnesses separation of the PT_k classes witnesses separation of the SP_k classes.

[3] This is Theorem 6.12 of Lothaire [15], although Lothaire attributes the general principle to Higman [9].

[4] It should be noted, though, that the language licensed by $\langle \Sigma, T \rangle$ as an SP_{k+1} grammar is not equal to that licensed by $\langle \Sigma, T \rangle$ as an SP_k grammar, since $T \subseteq \Sigma^k$ implies that no string of length greater than k will be licensed in the SP_{k+1} sense.

Theorem 9. SP *and* SP_k*, for any* $k > 0$*, are closed under intersection and (in a trivial sense) Kleene closure.* SP_k *is not closed under union or concatenation, although* SP *is closed under both. Neither* SP *nor* SP_k *are closed under complement or intersection with Regular languages.*

Proof. Closure under intersection follows immediately from Property 1 of Theorem 7.

Non-closure of SP_k under union is witnessed by the language $L = L_{<ka} \cup L_{<kb} = \{w \in \Sigma^* \mid |w|_a < k \text{ or } |w|_b < k\}$ This is the union of two SL_k sets, but it is not, itself, SL_k since $P_{\leq k}(a^k b^k) \subseteq P_{\leq k}(L)$. For concatenation, note that $L_{\leq (k-1)b}$ and $L_{\leq 1b}$ are both SL_k but $L_{\leq (k-1)b} \cdot L_{\leq 1b} = L_{\leq kb} \notin \mathrm{SL}_k$.

Closure of SP under union, intersection and concatenation follows nearly immediately from the equivalence of SP with the class of languages that are closed under subsequence.

Since all SP_k languages include all symbols occurring in the strings of the set as unit strings, if $L \in \mathrm{SP}_k$ then $L^* = P_1(L)^*$, i.e., L^* is the set of all strings over that subset of the alphabet which actually occurs in strings in L.

Non-closure of SP (hence SP_k) under complement follows from the fact that no non-trivial shuffle ideal is closed under subsequence. Non-closure under intersection with Regular sets is witnessed by the fact that no non-empty subset of $\overline{\{\varepsilon\}}$ is closed under subsequence.

Theorem 10. PT *(respectively,* PT_k*) is the closure of* SP *(respectively,* SP_k*) under Boolean operations.*

That every SP_k language is PT_k follows from Theorem 1 and Property 3 of Theorem 7. That every PT_k language is a Boolean combination of SP_k languages follows from the fact that $\mathrm{SI}(w)$, for any $w \in \Sigma^*$ is the complement of a $\mathrm{SP}_{|w|}$ language.

It's striking that, while PT is the closure of SP under Boolean operations, the latter is closed under concatenation while the former is not.

5 SP and the Regular Languages

Since SP $\subseteq$ PT $\subseteq$ Star-Free $\subseteq$ Regular every SP language is recognizable by a Deterministic Finite-State Automaton (DFA). Theorem 7 has a number of consequences for the structure of the trimmed, minimal DFAs which recognize SP languages. In particular, let $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F \rangle$ be a trimmed minimal DFA for which $L(\mathcal{M}) \in \mathrm{SP}_k$. Then:

- All states of $\mathcal{M}$ are accepting states: $F = Q$.
- For all $q_1, q_2 \in Q$ and $\sigma \in \Sigma$, if $\hat{\delta}(q_1, \sigma)\uparrow$ and $\hat{\delta}(q_1, w) = q_2$ for some $w \in \Sigma^*$ then $\hat{\delta}(q_2, \sigma)\uparrow$. (Missing edges propagate down.)
- All cycles are self-edges.
- For all $q_1, q_2 \in Q$ and $u, v, w \in \Sigma^*$, if $\hat{\delta}(q_0, w) = q_1$, $\hat{\delta}(q_1, v) = q_2$ and $q_1 \neq q_2$ then:

- $(\exists u \in \Sigma^*)[\hat{\delta}(q_0, wu)\downarrow$ and $\hat{\delta}(q_0, wvu)\uparrow]$ and
- $(\forall u \in \Sigma^*)[\hat{\delta}(q_0, wvu)\downarrow \Rightarrow \hat{\delta}(q_0, wu)\downarrow]$

Lemma 1. *Let $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F\rangle$ be a trimmed, minimal DFA for which $L(\mathcal{M}) \in \mathrm{SP}_k$. Then $P_{\leq k}(L(\mathcal{M})) = \{w \in \Sigma^{\leq k} \mid \hat{\delta}(q_0, w)\downarrow\}$.*

This follows from closure under subsequence.

Lemma 1 provides an algorithm which, given a DFA that recognizes an SP_k language, constructs the SP_k grammar for that language. One simply does a search of the transition graph of the DFA with the depth limited to k, recording the strings labeling the paths traversed. The time complexity of this algorithm is $\Theta(\mathbf{card}(\Sigma)^k)$. Note that this construction will yield some SP_k grammar given any $\mathcal{M}$; that grammar will license $L(\mathcal{M})$ iff $L(\mathcal{M})$ is SP_k.

Lemma 2. *Suppose $w \in \Sigma^k$, $w = \sigma_1 \cdots \sigma_k$.*
Let $\mathcal{M}_{\overline{\mathrm{SI}(w)}} = \langle Q, \Sigma, q_0, \delta, F\rangle$, where $Q = \{i \mid 1 \leq i \leq k\}$, $q_0 = 1$, $F = Q$ and for all $q_i \in Q, \sigma \in \Sigma$:

$$\delta(q_i, \sigma) = \begin{cases} q_{i+1} & \text{if } \sigma = \sigma_i \text{ and } i < k, \\ \uparrow & \text{if } \sigma = \sigma_i \text{ and } i = k, \\ q_i & \text{otherwise.} \end{cases}$$

Then $\mathcal{M}_{\overline{\mathrm{SI}(w)}}$ is a minimal, trimmed DFA that recognizes the complement of $\mathrm{SI}(w)$, i.e., $\overline{\mathrm{SI}(w)} = L(\mathcal{M}_{\overline{\mathrm{SI}(w)}})$.

Lemma 2 provides the foundation for an algorithm which, given an SP_k grammar $\langle \Sigma, T\rangle$ for a language L, constructs a minimal, trimmed DFA which recognizes L. One constructs the trimmed, minimal DFA for $\overline{\mathrm{SI}(w)}$ for each $w \in \Sigma^{\leq k} - P_{\leq k}(T)$, and then constructs the trimmed, minimal DFA for their intersection. The complexity of this algorithm is $\Theta(\mathbf{card}(\Sigma)^k)$ (since $\mathbf{card}(T) = \Theta(\mathbf{card}(\Sigma)^k$, worst case).

Together, Lemmas 1 and 2, applied alternately for increasingly large k provide a mechanism for determining the least k for which $L(\mathcal{M}) \in \mathrm{SP}_k$ if, in fact, there is such a k. All that remains is to determine a bound on the size of the k for which $L(\mathcal{M})$ could be SP_k.

Lemma 3. *Suppose $L \in \mathrm{SP}_k - \mathrm{SP}_{k-1}$. Then every DFA that recognizes L has at least k states.*

$L \in \mathrm{SP}_k - \mathrm{SP}_{k-1}$ implies that there is at least one $w \in \Sigma^k$ such that $\mathrm{SI}(w) \cap L = \emptyset$ but for all proper subsequences v of w it is the case that $\mathrm{SI}(v) \cap L \neq \emptyset$. In fact, since SP_k languages are closed under subsequence v itself must be in L.

Suppose v_1 and v_2 are distinct proper prefixes of w, $|v_1| < |v_2|$. Then there is some u such that $v_2 u = w \notin L$. On the other hand, $v_1 u$ is a proper subsequence of w and thus, by choice of w, $v_1 u \in L$.

Hence none of the k proper prefixes of w are Nerode equivalent (with respect to L) to each other or to w and any DFA recognizing L will need to distinguish at least $k + 1$ classes of strings, hence have at least k states.

Theorem 11. *There is an algorithm which, given any Regular language L, decides if L is* SP *and, if it is, determines the least k for which L is* SP_k *and returns an* SP_k *grammar for L.*

Assume, wlog, that L is presented as a trimmed, minimal DFA. The algorithm constructs potential SP_k grammars for increasing k using Lemma 1 and constructs trimmed, minimal DFAs for each using Lemma 2. The first grammar constructed in this way that is isomorphic to the DFA for L will be an SP_k grammar for the least k for which L is SP_k. By Lemma 3, if no such DFA is found for $k \leq \mathbf{card}(Q)$ then L is not SP_k for any k.

The time complexity of this algorithm is $\Theta(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)})$, i.e., it is exponential time. This, however, turns out to be optimal for algorithms that actually construct an SP_k grammar for L.

Theorem 12. *Suppose $L \in$* SP*. Let $\boldsymbol{card}(Q)$ be the size of the state set of a trimmed, minimal DFA recognizing L. Then the worst case size of the grammar for L is $\Theta(\boldsymbol{card}(\Sigma)^{\boldsymbol{card}(Q)})$.*

By Lemma 3, no grammar for an SP language can be larger than $\mathbf{card}(\Sigma)^{\mathbf{card}(Q)}$, where Q is the state set of a trimmed minimal DFA recognizing that language. By Lemma 2, grammars of that size do exist.

6 Dual Subregular Hierarchies

The hierarchy of Local classes of languages has a very attractive model-theoretic characterization. The class of Strictly Local languages is properly extended by the class of Locally Testable languages, which is the class of languages definable by propositional formulae in which the atomic formulae are blocks of symbols interpreted as factors of the string. This is properly extended by the class of Locally Threshold Testable languages, which is the class of languages definable by First-Order formulae with adjacency (successor) but not precedence (less-than). This is properly extended by the class of Regular languages, which is the class of languages definable by Monadic Second-Order formulae with either adjacency or precedence, equivalently with both (since they are MSO definable from each other).

As we have seen here, the Piecewise classes provide a parallel sequence of classes. The class of SP languages corresponds to the Strictly Local languages, except that they are defined in terms of subsequences rather than factors. This is extended by the class of PT languages, which is the class of languages definable over propositional formulae in which the atomic formulae are blocks of symbols interpreted as subsequences of the string. Hence PT corresponds to LT except that, again, it is defined in terms of subsequences rather than factors. The class of languages definable by First-Order formulae with precedence, corresponding to LTT on the adjacency side, is the class of Star-Free sets. Since adjacency is FO definable from precedence, LTT is actually a (proper) subclass of SF. Finally, the

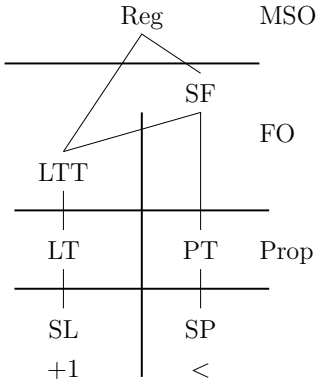

Fig. 1. Parallel Sub-regular Hierarchies

two branches become indistinguishable at the MSO level in the class of Regular languages.[5]

7 Conclusion

We have characterized the Strictly Piecewise languages and presented their basic properties. Additionally, it was shown that SP languages complete a subregular hierarchy based on precedence in the same way the Local classes form a hierarchy based on adjacency. We have also provided algorithms for translating between the SP grammars defined here and finite state automata, as well as an algorithm for deciding if some regular language is SP.

The theoretical contributions above provide a better understanding of linguistic, cognitive, and natural language processing models. We have already mentioned the capability of the SP languages to describe certain kinds of phonotactic patterns [7,8]. The introductory Sarcee pattern, for example, is given by a SP_2 grammar which only prohibits subsequences consisting of a [+anterior] sibilant followed by a [-anterior] sibilant. Interestingly, SP languages also appear in models of reading comprehension [5,24] as well as in text classification [14,1] (see also Shawe-Taylor [20, chap. 11]). We hope that this paper continues to spur interest in the utility and beauty of languages described piecewise.

References

1. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word-sequence kernels. Journal of Machine Learning Research 3, 1059–1082 (2003)
2. Chomsky, N.: Three models for the description of language. I.R.E. Transactions on Information Theory IT-2, 113–123 (1956); reprinted in Readings in Mathematical Psychology. In: Duncan Luce, R., Bush, R.R., Galanter, E. (eds.), vol. II, pp. 113–123. John Wiley & Sons, New York (1965)

[5] Interestingly, it can also be said that they join at the bottom of the hierarchy as well, since SP_1 and SL_1 both contain just $\emptyset$ and Γ^* for each $\Gamma \subseteq \Sigma$.

3. Cook, E.D.: The synchronic and diachronic status of Sarcee g^y. International Journal of American Linguistics 4, 192–196 (1978)
4. Cook, E.D.: A Sarcee Grammar. University of British Columbia Press (1984)
5. Grainger, J., Whitney, C.: Does the huamn mnid raed wrods as a wlohe? Trends in Cognitive Science 8, 58–59 (2004)
6. Hansson, G.: Theoretical and typological issues in consonant harmony. Ph.D. thesis, University of California, Berkeley (2001)
7. Heinz, J.: The Inductive Learning of Phonotactic Patterns. Ph.D. thesis, University of California, Los Angeles (2007)
8. Heinz, J.: Learning long distance phonotactics (2008) (submitted manuscipt)
9. Higman, G.: Ordering by divisibility in abstract algebras. Proceedings of the London Mathmatical Society 2, 326–336 (1952)
10. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory. In: Languages, and Computation. Addison-Wesley, Reading (2001)
11. Joshi, A.K.: Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press, Cambridge (1985)
12. Kobele, G.: Generating Copies: An Investigation into Structural Identity in Language and Grammar. Ph.D. thesis, University of California, Los Angeles (2006)
13. Kontorovich, L.A., Cortes, C., Mohri, M.: Kernel methods for learning languages. Theoretical Computer Science 405(3), 223–236 (2008)
14. Lodhi, H., Cristianini, N., Shawe-Taylor, J., Watkins, C.: Text classification using string kernels. Journal of Machine Language Research 2, 419–444 (2002)
15. Lothaire, M. (ed.): Combinatorics on Words. Cambridge University Press, Cambridge (1997)
16. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
17. Rogers, J., Pullum, G.: 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)
18. Rose, S., Walker, R.: A typology of consonant agreement as correspondence. Language 80(3), 475–531 (2004)
19. Sakarovitch, J., Simon, I.: Subwords. In: Lothaire, M. (ed.) Combinatorics on Words, Encyclopedia of Mathematics and Its Applications, vol. 17, ch. 6, pp. 105–134. Addison-Wesley, Reading (1983)
20. Shawe-Taylor, J., Christianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2005)
21. Shieber, S.: Evidence against the context-freeness of natural language. Linguistics and Philosophy 8, 333–343 (1985)
22. Simon, I.: Piecewise testable events. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 214–222. Springer, Heidelberg (1975)
23. Trahtman, A.: Piecewise and local threshold testability of DFA. In: Freivalds, R. (ed.) FCT 2001. LNCS, vol. 2138, pp. 347–358. Springer, Heidelberg (2001)
24. Whitney, C., Cornelissen, P.: SERIOL reading. Language and Cognitive Processes 23, 143–164 (2008)