Extracting Subregular constraints from Regular stringsets

James Rogers and Dakotah Lambert
Dept. of Computer Science, Earlham College, Richmond, IN, USA

ABSTRACT

We introduce algorithms that, given a finite-state automaton (FSA), compute a minimal set of forbidden local factors that define a Strictly Local (SL) tight approximation of the stringset recognised by the FSA and the set of forbidden piecewise factors that define a Strictly Piecewise (SP) tight approximation of that stringset, as well as a set of co-SL factors that, together with the SL and SP factors, provide a set of purely conjunctive literal constraints defining a minimal superset of the stringset recognised by the automaton.

Using these, we have built computational tools that have allowed us to reproduce, by nearly purely computational means, the work of Rogers and his co-workers (Rogers *et al.* 2012) in which, using a mix of computational and analytical techniques, they completely characterised, with respect to the Local and Piecewise Subregular hierarchies, the constraints on the distribution of stress in human languages that are documented in the StressTyp2 database.

Our focus, in this paper, is on the algorithms and the method of their application. The phonology of stress patterns is a particularly good domain of application since, as we show here, they generally fall at the very lowest levels of complexity. We discuss these phonological results here, but do not consider their consequences in depth.

Keywords: regular languages, finite-state automata, local languages, piecewise languages

1 INTRODUCTION

That phonology is finite-state – characterised by patterns and functions that can be recognised by finite-state automata of varying types –

is uncontroversial. Over the last several years, a growing body of work has emerged characterising the complexity of phonological phenomena within a more finely resolved hierarchy of Subregular stringsets[1] and functions, focusing, in particular, on the lowest levels of the hierarchy. (For a comprehensive survey, see Heinz 2018.)

In this paper we present two main results. One is a set of algorithms that are capable of automatically analysing finite-state acceptors in terms of these classes. We have incorporated these into a computational workbench for exploring systems of Subregular constraints both automatically and interactively.

Using that, we have completed a longterm program of characterising the complexity of suprasegmental stress patterns in human languages. This yields our second result, which strengthens previous characterisations and places these patterns almost exclusively at the very bottom levels of the hierarchy: the Strictly Local, Locally Testable, Strictly Piecewise and Piecewise Testable stringsets. These classes are significant cognitively because they depend only on information that is explicit in the string itself without requiring inference of additional structure.[2]

A stringset L is *Strictly k-Local* (SL_k) if and only if (iff) it is strictly determined by its local k-factors: the *substrings* of length at most k that

[1] Following Sampson (1975), we distinguish between stringsets and languages when our topic extends to both formal and natural languages. In these situations, the formal languages serve as models for aspects of natural languages. To fail to distinguish the stringsets (or others sets of structures) that are the models from the phenomena they are modelling is falling into one of the most basic fallacies of mathematical modelling. Conclusions that are valid in the realm of the models are only valid in the realm of the phenomena to the extent that the models are faithful. That their faithfulness is limited is the very essence of the modelling process.

This is not just a pedantic issue: the history of formal linguistics is peppered with examples in which the failure to distinguish the two has led to erroneous conclusions. For discussion of specific examples see, *inter alia*, Pullum and Scholz (2001) or Rogers (1996). As the scope of this journal is, specifically, language modelling, it seems appropriate to us to be careful in maintaining the distinction.

[2] Regular stringsets, for example, from a purely declarative perspective, require a mechanism to infer a sequence of states (abstract categories) in parallel with a string, corresponding to a run of an automaton, and then classify the string based on that run.

occur in strings $\rtimes \cdot w \cdot \ltimes$ for $w \in L$. (The '$\rtimes$' and '$\ltimes$' are endmarkers.) By "strictly determined" we mean that L contains all and only the strings that are generated by the (inverse) substring relation from that set of k-factors.[3] A string is in the set as long as its k-factors are a subset of the generating set. The class of stringsets that are Strictly k-local for some k is known as SL.

A stringset is *k-Locally Testable* (LT_k) iff it is a Boolean combination of SL_k stringsets. While they are also determined by the set of local k-factors that occur in the strings in the set, the stringset is not generated by the substring relation from a single set of factors. Rather, there are multiple SL_k stringsets that interact in complex ways. Nevertheless, constraints of this form depend only on the information that is explicit in the string, in this case the particular subset of k-factors that occurs; various combinations of factors can be forbidden, permitted or required.

A stringset L is *Strictly k-Piecewise* (SP_k) iff it is strictly determined by its k-pieces: the *subsequences* of length at most k that occur in strings w for $w \in L$, where v is a subsequence of w iff the symbols in v occur in w in order, but not necessarily adjacently. That is to say, L contains all and only the strings that are generated by the (inverse) subsequence relation from that set of k-pieces. The class of stringsets that are Strictly k-Piecewise for some k is known as SP.

The class of *k-Piecewise Testable* (PT_k) stringsets is analogous to the class of LT_k stringsets, but based on subsequences rather than substrings.

These classes are at the bottom of the local and piecewise sides of a collection of classes of stringsets, all strict subclasses of the class of Regular stringsets, which are hierarchically related and are characterised by finite sets of either substrings (the Local Hierarchy) or subsequences (the Piecewise Hierarchy) or by combinations of the two. The Local hierarchy was established primarily by the work of McNaughton and Papert (1971) and Thomas (1982) (with many others); the Piecewise Hierarchy was established primarily by the work of Simon (1975), Lothaire (1983), Rogers *et al.* (2010) and Fu *et al.* (2011) (and many others). Rogers *et al.* (2012) argue that these

[3] Specifically, all and only those strings that include no substrings other than those in the given set.

hierarchies provide a robust notion of cognitive complexity for constraints on strings.

Our work here was inspired by the work carried out by the Theory Group at Earlham College in which they characterised all of the stress patterns collected in Goedemans *et al.* (2015) – a wide-coverage database of stress patterns occurring in human languages – with respect to this hierarchy. In Edlefsen *et al.* (2008), they established that roughly 75% of these patterns are SL_k for $k \leq 6$ and that half are SL_k for $k \leq 3$. Subsequently, they derived a set of "primitive" constraints sufficient to define all of the non-SL patterns by co-occurrence and classified them into abstract categories (Fero *et al.* 2014). Most of these constraints were, in fact, SL, and their main result was that all of the patterns could be defined by co-occurrence of constraints at the bottom two levels of the hierarchies – the Strict and Testable levels described above. Recent work by Heinz and his co-workers (Heinz 2018, 2010; Chandlee 2014; Jardine 2016) suggests that much of phonology may be characterisable by correspondingly simple sets of structures or functions.

The work on primitive constraints, however, did not provide the factors of the SL stringsets because the algorithm for determining if a given finite-state automaton recognises an SL stringset, and determining k if it does, does not yield the set of k-factors that define the stringset. We resolve that problem in this work. Moreover, the work on non-SL constraints was largely based on the English glosses of the constraints included in the database and were tailored specifically to capturing those specific non-SL lects. This potentially misses SP constraints that may not be explicit in these glosses.

In this paper we specifically address the piecewise classes as well as the class of co-SL constraints, complements of SL constraints. Combinations of SL and co-SL constraints define stringsets in terms of both forbidden and required local factors, but remain a weak fragment of LT. Working with this range of constraints we can sharpen the earlier result, which established LT + SP as an upper-bound for all but two lects in the database: 98 of the 106 lects are SL+co-SL+SP, another six require combinations of three properly LT constraints (rather than the nine identified in the earlier work) and the two properly Regular lects share a single Regular constraint, which entails counting modulo two.

While our working domain in developing these algorithms has been phonotactics, and stress patterns in particular, the algorithms are applicable to any Regular stringset. On the other hand, the algorithms are of relatively high complexity, exponential in the size of the automaton if it recognises a Strictly Local stringset and doubly exponential for the SL approximations of non-strict stringsets; the Piecewise algorithms are singly exponential in either case. But these are optimal for algorithms that return the set of forbidden factors of the stringset. In our corpus all of the automata are of moderate size – the largest has 33 states – and they are quite feasible; running on hardware that is unremarkable for modern desktop computers, without aggressive optimisation, it takes less than one minute to process all of the 106 lects of the StressTyp2 corpus of automata.

2 OVERVIEW OF THIS PAPER

In the next section we introduce our notation and basic formal definitions. In Section 3.1 we introduce the notion of local and piecewise factors and in Section 3.2 we consider stringsets from a model-theoretic perspective, which exposes the underlying relationship between these. From that perspective both the substring and the subsequence relations are just restricted variants of a more general is-a-factor-of relation. Henceforth, we will refer to substrings and subsequences as factors of either the local or piecewise type, except when the type is clear from the context. (In Sections 4 and 5, if the type is not specified, "factor" refers to local factors; in Sections 6 and 7 it refers to piecewise factors.)

In Section 4 we formally define Strictly Local stringsets and discuss their formal properties. In Sections 4.1–4.3 we distinguish five types of forbidden local factors – factors in the complement of the set of factors that generate the stringset – and develop the foundations of our algorithms for extracting those local factors given a finite-state automaton. We give details of these algorithms in the remainder of Section 4.

We adapt these algorithms to work on non-SL stringsets in Section 5. While the sets of factors we extract are, of course, insufficient to generate the stringset, they do generate an SL approximation of the stringset. We then introduce the notion of a *residue set*, the difference

between the original stringset and our approximation. This becomes the basis of our further computational analysis.

In Section 6 we formally define Strictly Piecewise stringsets, discuss their formal properties and develop our algorithms for extracting forbidden piecewise factors given a finite-state automaton that recognises an SP stringset. In Section 7 we adapt these to obtain optimal SP approximations of non-SP Regular sets.

In Section 8 we combine these algorithms in a way that allows us to extract co-SL constraints, which enables us to fully characterise 92% of the lects in StressTyp2 purely computationally.

We close by summarising our results, discussing complexity issues and sketching plans for recoding the factors we collect in a way that is more useful than their current form, a flat enumeration.

3 FORMAL PRELIMINARIES

Let Σ be an alphabet. For strings $v, w \in \Sigma^*$ we say v is a *substring* of w ($v \preccurlyeq w$) iff $w = u_1 v u_2$ for $u_1, u_2 \in \Sigma^*$. We say v is a *subsequence* of w ($v \sqsubseteq w$) iff the symbols of v occur in w in order, but not necessarily adjacently:

$$\text{if } v = \sigma_1\sigma_2\ldots\sigma_n \text{ then } v \sqsubseteq w \overset{\text{def}}{\iff} w = u_0\sigma_1u_1\sigma_2u_2\ldots\sigma_nu_n \mid u_i \in \Sigma^*.$$

We denote the *reversal* of w by w^{R}. We use this same notation for the reversal of a stringset.

A finite-state automaton (FSA) is an edge-labelled directed graph with distinguished vertices, that we will represent by a five-tuple $\langle \Sigma, Q, \delta, I, F\rangle$ where Σ is the alphabet of the language of the automaton, Q is the set of states, $\delta \subseteq (\Sigma \times Q \times Q)$ is a transition relation where $\langle \sigma, q_1, q_2\rangle \in \delta$ iff there is an edge labelled σ from q_1 to q_2, I is the set of initial states, and F is the set of accepting states. Let $\mathscr{A} = \langle \Sigma, Q, \delta, I, F\rangle$.

Let $w = \sigma_1\sigma_2\ldots\sigma_n \in \Sigma^*$ be a string and let $q_1, q_{n+1} \in Q$. Then there is a path $q_1 \overset{w}{\rightsquigarrow} q_{n+1}$ iff there exists some sequence of edges:

$$\langle \langle \sigma_i, q_i, q_{i+1}\rangle \in \delta \mid \quad 0 < i \leq n, \\ w = \sigma_1\sigma_2\ldots\sigma_n\rangle.$$

This is an accepting path on w if q_1 is in I and q_{n+1} is in F, else it is a non-accepting path.

The automaton $\mathscr{A}$ is *total* iff for every symbol $\sigma \in \Sigma$ and for every state $q \in Q$, there exists some q' such that $\langle \sigma, q, q' \rangle \in \delta$. It is (partial) *functional* iff δ is functional in its first two places. That is, given a state $q \in Q$ and a symbol $\sigma \in \Sigma$, there is at most one $q' \in Q$ such that $\langle \sigma, q, q' \rangle \in \delta$. An FSA is (fully) *deterministic* (a proper DFA) iff it has exactly one initial state and it is both total and functional.

An automaton is *trim* iff for all states $q \in Q$ there is some accepting path from q. Though trim automata may not be total, we still consider them to be deterministic if they have a single start state and are partial functional.

An automaton is *minimal* iff it is deterministic and no two states are Nerode-equivalent.[4] Further, it is *normalised* iff it is both minimal and trim.

The *reversal* of $\mathscr{A}$, that is, an automaton that accepts a string iff the reversal of that string is accepted by $\mathscr{A}$, is denoted $\mathscr{A}^{\mathrm{R}}$.

The powerset graph of the automaton $\mathscr{A}$, $\mathrm{PSG}(\mathscr{A}) = \langle V, E \rangle$, is another edge-labelled directed graph where:

$$\begin{array}{lcl} V & = & \mathscr{P}(Q) \quad \text{and} \\ E & = & \{\langle \sigma, S_1, S_2 \rangle \mid \sigma \in \Sigma, \\ & & \quad S_2 = \{q' \in Q \mid (\exists q \in S_1)[\langle \sigma, q, q' \rangle \in \delta]\}\}. \end{array}$$

Often we are interested only in the subgraph of this generated from a given set of initial subsets.

Lemma 1. *If $\mathscr{A}$ is deterministic, then the sizes of the sets along any path in* $\mathrm{PSG}(\mathscr{A})$ *are monotonically non-increasing.*

This is because if $\mathscr{A}$ is deterministic δ maps each state in S_1 to at most one state in S_2.

Corollary 1. *All sets in any cycle are equal in size.*

Corollary 2. *All in-edges to Q and all out-edges from $\emptyset$ are self-edges.*

3.1 *Local and piecewise factors*

Let Σ be the alphabet of L and let $\Sigma^k = \{v \in \Sigma^* \mid |v| = k\}$ and $\Sigma^{\leq k} = \bigcup_{1 \leq i \leq k} [\Sigma^k]$.

[4] States q_1 and q_2 are Nerode-equivalent iff for all strings w, there is an accepting path on w from q_2 iff there is an accepting path on w from q_1 (Hopcroft and Ullman 1979).

Definition 1 (Local k-factors)**.** For any string $w \in \Sigma^*$, the local k-factors of w are:

$$\mathrm{F}_k^{\lhd}(w) = \begin{cases} \{w\} & \text{if } |w| \leq k, \\ \{v \in \Sigma^k \mid v \preccurlyeq w\} & \text{otherwise.} \end{cases}$$

Similarly for $\mathrm{F}_{\leq k}^{\lhd}(w)$. This lifts to sets of strings in the obvious way.

Definition 2 (Shuffle ideal)**.** Let $v = \sigma_1\sigma_2\ldots\sigma_n \in \Sigma^*$. The *shuffle ideal* of v is defined as the set $\mathrm{SI}(v) = \{w \in \Sigma^* \mid v \sqsubseteq w\}$.[5]

Definition 3 (Piecewise k-factors)**.** For any string $w \in \Sigma^*$, the piecewise k-factors of w are:

$$\mathrm{F}_k^{<}(w) = \{v \in \Sigma^* \mid v \sqsubseteq w \wedge |v| = k\}.$$

Similarly for $\mathrm{F}_{\leq k}^{<}(w)$. Again, this lifts to sets of strings in the obvious way.

3.2 *A unifying perspective*

There is a fundamental regularity between the Local and Piecewise hierarchies that becomes apparent if one looks at strings as ordinary first-order structures. From this perspective, a string is just a labelled finite discrete linear order:

$$\mathscr{W} = \langle \mathscr{D}, \lhd, <, P_\sigma \rangle_{\sigma\in\Sigma}.$$

Where $\mathscr{D}$ is a finite domain, $\lhd$ is the successor relation (as well as the relation symbol denoting it), $<$ is the less-than relation (and symbol) and the P_σ are unary relations picking out the subset of the positions in the domain at which the symbol σ appears.

From this perspective a local factor is just a structure generated by a subset of the domain that is connected, in the graph-theoretic sense, by the $\lhd$ relation and a piecewise factor is just a similar structure connected by the $<$ relation. The size of the factor is just the size of the subset.

The testable classes of local and piecewise stringsets turn out to be the class of all and only those stringsets that are definable

[5] The term "shuffle ideal" appears to have been coined by J. Sakarovitch and I. Simon in Lothaire (1983).

in a propositional logic in which the atomic propositions are factors (Rogers *et al.* 2012). The strict classes are the class of stringsets definable by conjunctions of negative literals in this same logic.

This model-theoretic machinery extends to factors that incorporate both successor and less-than as well as to any class of relational structures, although here we will consider only local and piecewise formulae along with conjunctions of the two. It is this model-theoretic perspective that leads us to refer to both substrings and subsequences as factors, local factors and piecewise factors, respectively.

4 STRICTLY LOCAL STRINGSETS

An *anchored* string is one that has been augmented with one or both of the endmarkers '⋊' (left end) and '⋉' (right end).

A stringset L is *Strictly k-Local* ($L \in \mathrm{SL}_k$) iff it is strictly determined by the local k-factors of its fully anchored strings.

Let $\Sigma^*_{⋊⋉} = \{⋊\} \cdot \Sigma^* \cdot \{⋉\}$.

Let $\mathrm{G}^{\lhd}_{L,k} \subseteq \mathrm{F}^{\lhd}_{\leq k}(\Sigma^*_{⋊⋉})$ be the set of local factors that occur in fully anchored strings in L. Then the stringset generated by $\mathrm{G}^{\lhd}_{L,k}$ is:

$$L(\mathrm{G}^{\lhd}_{L,k}) = \left\{ w \in \Sigma^* \mid \mathrm{F}^{\lhd}_{\leq k}(⋊ \cdot w \cdot ⋉) \subseteq \mathrm{G}_{L,k} \right\}.$$

$L(\mathrm{G}^{\lhd}_{L,k})$ is, by definition, SL_k. If L is also SL_k then $L(\mathrm{G}^{\lhd}_{L,k}) = L$.

Since Σ is assumed to be finite, $\mathrm{F}^{\lhd}_{\leq k}(\Sigma^*)$ is also finite, and an SL_k language can equivalently be defined in terms of its forbidden factors: $\overline{\mathrm{G}^{\lhd}_{L,k}} = \mathrm{F}^{\lhd}_{\leq k}(\Sigma^*_{⋊⋉}) - \mathrm{G}^{\lhd}_{L,k}$. This is more natural in many applications, including many linguistic ones (as in "no pair of unstressed syllables occur adjacently").

A stringset is said to be SL if it is SL_k for any finite k.

The following proposition characterises SL_k.

Proposition 1 (Suffix Substitution Closure)**. (SSC)**

$$\begin{aligned} &L \in \mathit{SL}_k \textit{ iff} \\ &\quad (\forall x \in F^{\lhd}_{k-1}(\Sigma^*_{⋊⋉}))[\quad \textit{if } w_1 = u_1 \cdot x \cdot v_1 \in \{⋊\} \cdot L \cdot \{⋉\} \\ &\qquad\qquad\qquad\quad \textit{and } w_2 = u_2 \cdot x \cdot v_2 \in \{⋊\} \cdot L \cdot \{⋉\} \\ &\qquad\qquad\qquad\quad \textit{then } u_1 \cdot x \cdot v_2 \in \{⋊\} \cdot L \cdot \{⋉\} \qquad]. \end{aligned}$$

This is because if a symbol σ can follow x in some string of $L(\mathscr{A})$ then $x \cdot \sigma$ is a permitted local factor and σ can follow x in any string of L.

One consequence of this is that if $L(\mathscr{A}) \in \mathrm{SL}_k$ and $\mathscr{A}$ is deterministic, then for each length $k-1$ string x, all states in the set

$$\left\{q' \in Q \mid (\exists q \in Q)[q \overset{x}{\rightsquigarrow} q']\right\}$$

are Nerode-equivalent. If $\mathscr{A}$ is minimal as well, then all paths that end with the same $(k-1)$-factor lead to the same state. The computations of the automaton synchronise after at most $k-1$ steps.

This is the basis of the algorithm used by Edlefsen *et al.* (2008) to determine if a given $\mathscr{A}$ recognises an SL stringset and, if it does, to find the parameter k.

Proposition 2. *Suppose $\mathscr{A}$ is a normalised DFA. Then $L(\mathscr{A}) \in SL_k$ iff every path from Q in $\mathrm{PSG}(\mathscr{A})$ that is of length $k-1$ leads to a vertex that is either a singleton subset of Q or empty. If that is the case, then k is one plus the length of the longest path from Q to a singleton (that does not include other singletons). If there is no such longest path (i.e., there is an infinite path) then there is some cycle of non-singleton vertices, in which case $L(\mathscr{A})$ does not satisfy SSC for any k and it is not SL.*

In practice, it is not necessary to build even just the subgraph of $\mathrm{PSG}(\mathscr{A})$ generated by Q. All that one needs for a counter-example to SSC is a single pair of strings in which SSC fails. So it suffices to just explore the subgraph of $\mathrm{PSG}(\mathscr{A})$ that is generated by doubleton subsets of Q. The size of this subgraph is only $\Theta(\mathbf{card}(Q)^2)$, in contrast to the subgraph generated by Q, which is $\Theta(2^{\mathbf{card}(Q)})$.[6]

The following is an immediate consequence of this proposition.

Corollary 3. *If $\mathscr{A}$ is a normalised DFA and $L(\mathscr{A}) \in SL_k$ then all cycles in $\mathrm{PSG}(\mathscr{A})$ are cycles of singletons.*

4.1 *Classes of forbidden local factors*

Local factors may or may not include a left-end marker at the beginning or a right-end marker at the end or both. In the case that a factor contains neither, it can occur anywhere in a string (including, possibly, at the beginning or end) and we say that it is a *free factor* or, if forbidden, *free forbidden factor*. If the length of a free forbidden factor

[6] The pair-graph algorithm appears to have been first published in Caron (2000).

is one, then it has somewhat different status than free forbidden factors of greater length; it is, in essence, a restriction to the alphabet. We will refer to these as *forbidden units*. If the first symbol of a forbidden factor is '⋊', then it can only occur at the left end of the word; this is an *initial forbidden factor*. If the last symbol is '⋉', then it can only occur at the right end of the word; it is a *final forbidden factor*. Note that the length of the string that these anchored factors match is $k-1$. An SL_k definition can restrict prefixes and suffixes of length up to $k-1$, but not, in general length k prefixes and suffixes.[7] Finally, if a factor contains both endmarkers it is a *forbidden word*, where the (unanchored) word it forbids is actually of length $k-2$.

4.2 *Free forbidden factors*

Suppose $\mathscr{A}$ is a DFA. A factor w is a free forbidden factor of $L(\mathscr{A})$ iff there is no path in the transition graph of $\mathscr{A}$ from q_0 to an accepting state that includes w as a substring. If $\mathscr{A}$ is normalised, this will be the case iff there is no path at all that is labelled w from any state of $\mathscr{A}$, as all such paths would necessarily lead to the sink state which has been trimmed. Thus, in $\mathrm{PSG}(\mathscr{A})$ the path from Q that is labelled w leads to $\emptyset$. Again, the converse holds.

So the members of the set of all labels of paths from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ are free forbidden factors of $L(\mathscr{A})$. Moreover, that set includes *all* free forbidden factors of $L(\mathscr{A})$. Since in general $\mathrm{PSG}(\mathscr{A})$ may include cycles and even in the case that $L(\mathscr{A})$ is SL it may include cycles of singleton vertices, in general this set of paths will be infinite. (In fact, since $\mathrm{PSG}(\mathscr{A})$ invariably includes a trivial cycle on $\emptyset$ for each $\sigma \in \Sigma$, it will *always* be infinite.) The paths including trivial cycles on $\emptyset$ are labelled with strings in $w \cdot \Sigma^*$, where w is a free forbidden factor. We are interested in the set of paths that are minimal in the sense that the label of the path does not include the label of any other such path as a substring.

Note that, by Corollary 2, any such path that includes an in-edge to Q or an out-edge from $\emptyset$ includes another path from Q to $\emptyset$ that is

[7] In the original definition of SL_k (McNaughton and Papert 1971) prefix and suffix factors and forbidden words could be of length k. But the definition we use is equivalent in all significant aspects and accounts for the information contained in an anchored factor; it has become the prevailing definition in most of the literature.

strictly shorter. Thus none of those paths are minimal free forbidden factors. Note, also, that if $L(\mathscr{A}) \in \mathrm{SL}$, then there are no cycles on Q, although there will always be trivial cycles on $\emptyset$ for each $\sigma \in \Sigma$.

The next two lemmas establish that if $L(\mathscr{A})$ is SL then there is some bound such that all cyclic paths from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ with length greater than that bound will be labelled with a string that includes, as a suffix, the label of an acyclic path from Q to $\emptyset$. Thus the set of minimal free forbidden factors of $L(\mathscr{A})$ is just the set of labels from paths from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ that do not include the label of any other such path as a suffix and that do not include self-edges on $\emptyset$. This allows us to collect forbidden factors with a breadth-first bottom-up traversal of $\mathrm{PSG}(\mathscr{A})$.

Lemma 2. *If v and w label paths from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ that do not include loops on $\emptyset$ and $v \preccurlyeq w$, then $w = uv$ for some $u \in \Sigma^*$.*

Proof. $v \preccurlyeq w$ iff, by definition, $w = uvx$ for some $u, x \in \Sigma^*$. Since $Q \overset{v}{\rightsquigarrow} \emptyset$ and all vertices S of $\mathrm{PSG}(\mathscr{A})$ are subsets of Q, for all vertices S, $S \overset{v}{\rightsquigarrow} \emptyset$ as well, and, in particular, $Q \overset{u}{\rightsquigarrow} S \overset{v}{\rightsquigarrow} \emptyset$. Hence x is either ε or the path it labels is a self-loop on $\emptyset$. ∎

Lemma 3. *Let $\mathscr{A}$ be a DFA such that $L(\mathscr{A}) \in$ SL. If a path from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ includes a cycle other than a trivial cycle on Q or $\emptyset$, then there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to $\emptyset$ as a suffix.*

Proof. Since $L(\mathscr{A})$ is SL, any cycle must be a cycle of singletons. Suppose then that there is a path:

$$Q \overset{u}{\rightsquigarrow} \{q_0\} \overset{v}{\rightsquigarrow} \{q_1\} \overset{w}{\rightsquigarrow} \{q_0\} \overset{x}{\rightsquigarrow} \emptyset$$

where, possibly, v may be a prefix of x. Since $q_0, q_1 \in Q$ there must be a path:

$$Q \overset{u}{\rightsquigarrow} S_0 \overset{v}{\rightsquigarrow} S_1 \overset{w}{\rightsquigarrow} S_2 \overset{v}{\rightsquigarrow} S_3 \cdots$$

where $q_0 \in S_{2i}$ and $q_1 \in S_{2i+1}$ for $i \geq 0$. Since there are no cycles of non-singletons, by Lemma 1 the sequence of S_is must ultimately be decreasing in size. Thus, for some n it resolves to:

$$Q \overset{v}{\rightsquigarrow} S_1 \overset{w}{\rightsquigarrow} S_2 \overset{v}{\rightsquigarrow} S_3 \cdots \overset{w}{\rightsquigarrow} S_{2n} = \{q_0\} \overset{x}{\rightsquigarrow} Q$$

So $(vw)^n x$ labels a path from Q to $\emptyset$ and will be a suffix of all paths Q to $\emptyset$ that take the $\{q_0\} \rightsquigarrow \{q_1\}$ cycle at least $2n$ times. ∎

An example of this lemma, taken from the PSG of the canonical automaton for Cairene Arabic, is shown in Figure 1.

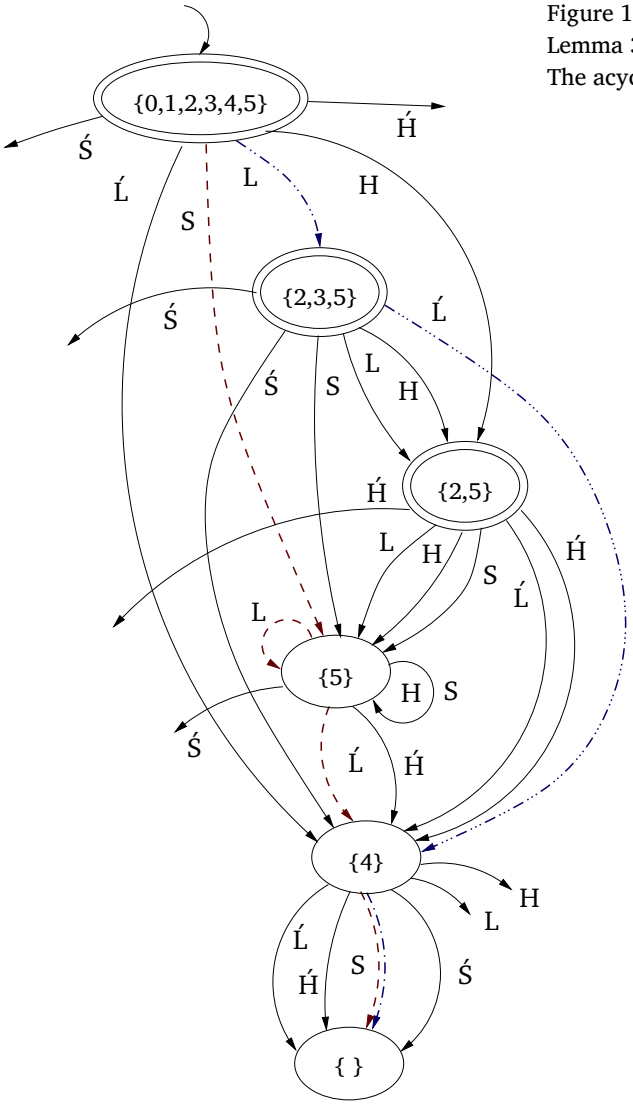

Figure 1:
Lemma 3: The cyclic path is labelled S(L)*ĹS. The acyclic path is labelled LĹS

Theorem 1. *If $L(\mathscr{A}) \in SL$ then a string w is a free forbidden factor of $L(\mathscr{A}) \in SL$ iff it labels a path in* PSG($\mathscr{A}$) *from Q to $\emptyset$. It is minimal if that path does not include any cycles other than cycles of singletons and w does not include the label of any other such path as a suffix.*

Note that if $L(\mathscr{A}) \in \mathrm{SL}$ then the only cycles of non-singletons will be trivial cycles on $\emptyset$. Labels of paths including these will include some free forbidden factor as a prefix and are thus not minimal.

Paths including cycles of singletons are necessary since none of the paths labelled $u(vw)^i x$ as in the proof of Lemma 3 is labelled with a factor of any of the others; they are minimal with respect to each other. It is only the label of the acyclic path that subsumes the labels of further iterations.

4.3 *Final forbidden factors*

Suppose $\mathscr{A}$ is a DFA. A factor w is a final forbidden factor of $L(\mathscr{A})$ iff there is no path from q_0 to an accepting state in the transition graph of $\mathscr{A}$ that includes w as a suffix but there is some path from q_0 to an accepting state that includes w as a proper substring. (If there is no such accepting path, then w is a free forbidden factor.) If $\mathscr{A}$ is normalised then w is a final forbidden factor iff all paths labelled w from any state in Q end at a non-accepting state and there is some such path. This will be the case iff the path from Q in $\mathrm{PSG}(\mathscr{A})$ labelled w ends at a non-empty vertex that is disjoint with F.

Proposition 3. *No final forbidden factor of any stringset includes a free forbidden factor of that stringset as a substring.*

This is because no string includes a free forbidden factor anywhere, whereas final forbidden factors are forbidden only as suffixes; to be final but not free, there must be some string that includes the factor as a non-suffix.

In terms of the PSG free forbidden factors label paths from Q to $\emptyset$, so as long as we only consider paths that lead to non-empty vertices, we don't have check to see if the factor is subsumed by a free forbidden factor. Note, though, that a final forbidden factor may include another as a suffix. (It is irrelevant whether it includes a final forbidden factor as a non-suffix, since final forbidden factors are, by definition, only relevant as suffixes.)

Lemma 4. *If a path from Q to a non-empty vertex disjoint from F in $\mathrm{PSG}(\mathscr{A})$, with $L(\mathscr{A}) \in \mathrm{SL}$, includes a cycle other than a trivial cycle on Q, then there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to a non-empty vertex disjoint from F as a suffix.*

Theorem 2. *If $L(\mathscr{A}) \in SL$ then a string w is a final forbidden factor of $L(\mathscr{A}) \in SL$ iff it labels a path in* PSG($\mathscr{A}$) *from Q to a non-empty vertex disjoint from F. It is minimal if that path does not include any cycles other than cycles of singletons and w does not include the label of any other such path as a suffix.*

The proofs are essentially the same as the proof of Lemma 3 and Theorem 1.

4.4 *Algorithms for extracting forbidden local factors*

Theorem 1 guarantees that if we do a breadth-first bottom-up traversal of PSG($\mathscr{A}$) then we will discover each minimal forbidden factor before we discover any of its proper suffixes. Expanding the frontier of the search in discrete stages, every (reverse) path from $\emptyset$ to Q found in the k^{th} stage will be a minimal forbidden k-factor.

There may be more than one such path so we do need to avoid gathering more than one instance of the factor. In general, there will be open paths (not reaching Q) that are labelled with the same factor. Extended to Q, they would include the factor as a proper suffix. So we exclude these from the frontier for the next stage.

We structure the bottom-up traversal of PSG($\mathscr{A}$) as a top-down traversal of PSG$^{\text{R}}$($\mathscr{A}$), in which each of the edges of PSG($\mathscr{A}$) is reversed. For convenience (and convergence) we trim self-edges on $\emptyset$ and Q while reversing the graph. Since we are traversing bottom-up, we actually find w^{R} of each factor w, but we gather these in a list structure, inserting at the head, which reverses the factor again as we construct it.

For the purposes of the algorithm, a *Path* in an edge-labelled graph $\langle V, E\rangle$, as a computational structure, is a three-tuple $\langle v, S, w\rangle$, where $v \in V$ is the final vertex of the path, $S \subseteq V$ is the (unordered) set of vertices along the path and $w \in \Sigma^*$ is the sequence of labels of the edges in the path, in reverse order. A *Frontier* is a set of paths. Forbidden factors are gathered in stages, with Stage$_i$ expanding Frontier$_{i-1}$ to Frontier$_i$, gathering the set FF$_i$ of all minimal forbidden i-Factors in the process.

The initial frontier Frontier$_0$, when searching for free forbidden factors, includes just the trivial (0-length) path from $\emptyset$. When searching for final forbidden factors, Frontier$_0$ includes the trivial path from each vertex that is a subset of $Q - F$.

Theorem 1 guarantees that, if we eliminate paths labelled with a forbidden i-Factor from Frontier$_i$ the search will converge after finitely many iterations, k, with Frontier$_k$ empty. (Note it is an empty set of Paths, not a set including a path ending at $\emptyset$.) The set of minimal free forbidden factors will be the union of the sets of factors gathered at stages 2 through k, where $L(\mathscr{A}) \in \mathrm{SL}_k$. (Forbidden 1-factors are not included, since they are forbidden units.) The search for final forbidden factors will terminate after $k-1$ iterations, with the minimal k-final forbidden factors including the right-end marker.

The time complexity of these algorithms is $\Theta(\mathbf{card}(\Sigma)^k)$, which is optimal for algorithms that return sets of k-factors. For an arbitrary automaton that recognises an SL stringset, we know from the pair-graph algorithm that k is no more than $\mathbf{card}(Q)^2$. Thus the complexity is $O(\mathbf{card}(\Sigma)^{(\mathbf{card}(Q)^2)})$.

4.5 *Initial forbidden factors*

The initial forbidden factors of $L(\mathscr{A})$ are just the final forbidden factors of $L(\mathscr{A})^{\mathrm{R}}$. We identify these by constructing $\mathscr{A}^{\mathrm{R}}$ and applying the algorithm for identifying final forbidden factors. This adds a determinisation step prior to generating the PSG, so this ends up being doubly exponential in the size of the automaton, $O(\mathbf{card}(\Sigma)^{(2^{(\mathbf{card}(Q)^2)})})$. As a practical matter, this was not an issue in our application, so we did not pursue a direct algorithm, but it would be worthwhile to do so in future work.

4.6 *Forbidden words for SL stringsets*

If $L(\mathscr{A}) \in \mathrm{SL}_k$ and $\mathscr{A}$ is deterministic, then the words it forbids are just the labels of paths of length $k-2$ (to allow for the endmarkers) from the (single) initial state to a state in $Q-F$. These can be gathered by doing a bounded traversal of $\mathscr{A}$. The time complexity of this traversal is $\Theta(\mathbf{card}(\Sigma)^k)$, thus $O(\mathbf{card}(\Sigma)^{(\mathbf{card}(Q)^2)})$.

4.7 *Forbidden units*

If $\mathscr{A}$ is normalised (minimal and trim), the forbidden units of $L(\mathscr{A})$ are just the symbols of Σ that do not label any edge in δ. In PSG($\mathscr{A}$) these will label edges Q to $\emptyset$ and will be gathered in Stage$_1$ while gathering free forbidden factors. But these may not be the only forbidden units of interest. In many applications there will be an alphabet that includes

all symbols that occur in any of a collection of stringsets and the subset of that alphabet that is not included in the alphabet of the FSA will also be significant. This is the case in many linguistic applications, for example (as in "this lect forbids unstressed heavy syllables").

In those applications we need to include the difference between some default alphabet and the set of symbols that label edges in $\mathscr{A}$. Since we are building PSG($\mathscr{A}$) anyway, the simplest way of doing this is to just take the difference between the default alphabet and the labels of the out-edges from Q. If we union that with the labels of the subset of those edges that lead to $\emptyset$ we get the free forbidden 1-factors as well. We can avoid gathering the latter in both the set of free forbidden factors and the set of forbidden units by not including the forbidden factors gathered in Stage_1. (Or, in order to simplify the code, by removing them from the set of free forbidden factors.)

Both of these approaches involve constructing the PSG and are consequently exponential in the number of states in the automaton. Alternatively, if all that is needed is the set of forbidden units, the set of permitted units can be gathered from the set of transitions of the automaton in time $\Theta(\mathbf{card}(Q)^2 \cdot \mathbf{card}(\Sigma))$. The forbidden units are then the difference between the default alphabet (plus the alphabet of the automaton, if it is not known to be a subset of the default) and the set of permitted units.

5 APPROXIMATING REGULAR STRINGSETS IN SL

Every stringset can be fully defined by the conjunction of a set of SL constraints (possibly trivial: Σ^*, $\emptyset$ and Σ^+ are SL_1, SL_1 and SL_2, respectively) along with a set of properly non-SL constraints. In applications that are exploring constraints across a collection of stringsets – most linguistic applications for instance – these SL constraints are significant. If the stringset is Regular we can factor the constraints so that the non-SL constraints capture just the non-strictly-local aspects of the patterns.

The problem isn't finding factors that characterise the stringset; the problem is that there are too many of them. $\Sigma^* - L(\mathscr{A})$, augmented with left and right endmarkers, is a set of forbidden factors that characterises $L(\mathscr{A})$ exactly. It is, of course, in general infinite.

The algorithms for SL stringsets are still partially correct for non-SL stringsets. The problem is that if $L(\mathscr{A})$ is non-SL, then there will be

non-singleton cycles (in addition to those on $\emptyset$) and the traversal will not terminate. These non-singleton cycles actually localise the reason that the stringset is not SL. They capture circumstances under which the automaton fails to synchronise ever; they identify places in which SSC (Proposition 1) fails for $L(\mathscr{A})$.

As with the set of forbidden words, the set of labels of the paths in $\mathrm{PSG}(\mathscr{A})$ from Q to $\emptyset$ that include non-singleton cycles are all legitimate forbidden factors of $L(\mathscr{A})$, but again there are infinitely many of them. The stringset they define is what we would like to isolate as the non-SL fragment of $L(\mathscr{A})$.

It is tempting to try modifying the traversal so it follows only singleton cycles. But, unfortunately, if there are non-singleton cycles the chain of the proof of Lemma 3 may be infinite, so there is no guarantee of termination even when following only singleton cycles.

Even if this were not the case, in general forbidden factors that label paths that take a non-singleton cycle one or more times can be necessary constraints in defining the SL fragment of a stringset. The reason is that it may be the case that there are acyclic paths that eventually subsume, in the sense of being included as a substring (specifically, being a suffix: see Lemma 2), paths with further iterations of the cycle. The subsuming path may be longer than the cyclic one and it may take several iterations of the cycle to form the acyclic path as a suffix.

The issue is termination. How many times must a cycle be taken before it has yielded all of the additional forbidden factors that are not just instances of an unbounded pattern? Note that if the cyclic paths will ultimately be subsumed, there will be at least one path from Q that will eventually be a suffix of that path. Traversing the PSG bottom-up, these paths will share the same initial sequence of labels.

Lemma 5. *Let $\mathscr{A}$ be any DFA. If a path from Q to $\emptyset$ in $\mathrm{PSG}(\mathscr{A})$ includes a cycle other than a trivial cycle on Q or $\emptyset$, then either:*

- *there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to $\emptyset$ as a suffix*
- *or there is a finite bound on the number of times the cycle can be taken before all paths with labels that share the same suffix as this*

one, beginning at the first iteration of the cycle, are instances of a non-SL pattern.

This is simply because if there is no acyclic path with labels that share the same suffix, then the label of the path is an instance of a factor that contains a substring that can be iterated unboundedly many times.

In order to identify the factors that label cyclic paths from Q to $\emptyset$ which will eventually be subsumed by an acyclic path, i.e., that are not simply instances of an unbounded pattern, we modify the bottom-up algorithm to follow all paths in parallel.[8] Rather than a frontier that is a set of paths, the frontier becomes a set of equivalence classes of paths with each path in a class sharing the same sequence of labels. If a class includes a path that reaches Q, then all paths in the class are subsumed by that path; the class is removed from the frontier of the traversal. On the other hand, if a class includes only cyclic paths, then it will never be subsumed in this way, the paths in the class are instances of unbounded patterns and we drop them; hence this traversal can be limited to acyclic paths and is guaranteed to terminate in time bounded by the size of the PSG, i.e. $\Theta(2^{\mathbf{card}(Q)})$.

The same strategy suffices for initial and final factors with the initial frontier taken to be the single class of trivial paths from subsets of Q that are disjoint with F, all of which are labelled ε. Note that the naïve algorithm for initial forbidden factors is in this case triply exponential in the size of the automaton.

In gathering forbidden words of non-SL stringsets the bound k is not known from building the PSG, but it can be estimated by the maximum of the length of the longest free forbidden factor minus 2, the length of the longest forbidden final or initial factor minus 1 and the length of the longest acyclic path from a start state to a non-accepting state in the automaton. This bound can be found implicitly while gathering the forbidden words. Once all words shorter than the bounds provided by the free, final and initial forbidden factors have been collected, all cyclic paths can be trimmed.

Unlike the case of SL stringsets, there is no smaller bound on the size of the factors in the approximation. The time complexity of the

[8] The reversed PSG is non-deterministic and this is just the same strategy used in determinising an NFA using the powerset construction.

traversals of these algorithms is $O(\mathbf{card}(\Sigma)^{(2^{\mathbf{card}(Q)})})$. The number of forbidden factors of an SL approximation of a non-SL Regular stringset is similarly bounded above, although we leave open the question of whether there are DFAs that witness this complexity (i.e., whether it is a big-Ω bound, as well). But relative to this upper bound the algorithm is optimal, and if a smaller bound can be established the traversal will almost certainly be bounded in the same way. The following theorem shows that the approximation is optimal in the sense of producing a stringset of minimal size.

Theorem 3. *If $L(\mathscr{A})$ is non-SL, the free, final and initial forbidden factors gathered by the modified bottom-up traversal of* $\mathrm{PSG}(\mathscr{A})$*, when combined with the forbidden words up to the bound given above, define the minimal SL superset of $L(\mathscr{A})$ that does not include the effect of arbitrarily many instances of properly non-SL constraints.*

Proof. By Theorems 1 and 2, every path from Q to $\emptyset$ or a vertex disjoint from F is either a free forbidden or final forbidden factor of $L(\mathscr{A})$, respectively. The same is true for the initial forbidden factors and the forbidden words. Thus the approximation does not exclude any string that is not also excluded by $L(\mathscr{A})$.

Those paths that are trimmed in the traversal either include another such path as a suffix or are instances of an unbounded pattern, i.e., include iterations of non-singleton cycles that may be iterated arbitrarily many more times without being subsumed by an acyclic path. These instances of unbounded patterns are the (infinite) set of forbidden factors defined by a properly non-SL constraint. Since the constraints are all negative, these instances can only reduce the stringset defined by the constraints. Including all of them yields $L(\mathscr{A})$; including none of them yields the SL approximation that is minimal in the sense of the theorem. ■

5.1 *Residue automata*

As noted in the proof of Theorem 3, the approximation, if not exact, will overgenerate. We turn now to the problem of characterising the strictly non-SL constraints that it omits.

Most work on approximating stringsets with stringsets in a weaker complexity class has focused on approximating CFLs with Regular stringsets (Nederhof 2000 includes a good survey) or Tree-Adjoining

Stringsets (TALs) with CFLs (Schabes and Waters 1993; Rogers 1994). Whenever the class of stringsets that is being approximated includes CFLs, the (symmetric) difference between the approximation and the target will not, in general, be a decidable set. Consequently, there is little that can be determined about that difference.

We have the advantage that all of our stringsets are Regular, and so the difference is not only decidable but an automaton recognising it is effectively constructible. Moreover, in this case, we know that every string excluded by the approximation is necessarily excluded by the target. The approximation never undergenerates. To isolate the non-SL characteristics of the target, we construct an automaton that recognises exactly the set of strings that are overgenerated by the SL approximation.

Using well-known algorithms for combining automata, an automaton $\mathscr{A}_{\text{FF}}$ that recognises the set of strings licensed by the set of forbidden factors can be constructed. One starts with deterministic automata that recognise each of the given factors, complements them and then builds the automaton that recognises the intersection of those complements. It is then straightforward to construct $\mathscr{A}_{\text{res}}$, the residue automaton,[9] which recognises exactly $L(\mathscr{A}_{\text{FF}}) - L(\mathscr{A})$. This residue automaton captures exactly the non-SL aspects of $L(\mathscr{A})$.

6 STRICTLY PIECEWISE STRINGSETS

A stringset L is *Strictly k-Piecewise* ($L \in \text{SP}_k$) iff it is strictly determined by its k-subsequences.

Let $\text{G}^{<}_{L,k} \stackrel{\text{def}}{=} \text{F}^{<}_{\leq k}(L)$ be the set of piecewise factors that occur in L. Then the stringset generated by $\text{G}^{<}_{L,k}$ is:

$$L(\text{G}^{<}_{L,k}) = \left\{ w \in \Sigma^* \mid \text{F}^{<}_{\leq k}(w) \subseteq \text{G}^{<}_{L,k} \right\}.$$

If $L \in \text{SP}_k$ then $L(\text{G}^{<}_{L,k}) = L$.

[9] The term "residue" is motivated from the perspective of factoring constraints. These automata should not be confused with the *residual automata* of Denis *et al.* (2002), NFAs in which every state corresponds to the residual stringset w.r.t. some prefix. "Residual" in that context is justified from the perspective of factoring strings.

Let $\overline{\mathrm{G}^{<}_{L,k}} = \mathrm{F}^{<}_{\leq k}(\Sigma^*) - \mathrm{G}^{<}_{L,k}$, the set of forbidden piecewise factors of L. This is more natural in many applications, including many linguistic ones (as in "no syllable with primary stress occurs following another such syllable"). Given $\overline{\mathrm{G}^{<}_{L,k}}$, the stringset L can be described as the intersection of the complements of finitely many shuffle ideals:

$$L = \bigcap_{w \in \overline{\mathrm{G}^{<}_{L,k}}} \Big[\overline{\mathrm{SI}(w)}\Big].$$

A stringset is said to be SP if it is SP_k for any finite k. The following propositions characterise SP and SP_k.

Proposition 4 (Subsequence Closure)**.** *A stringset L is SP iff*

$$(\forall w \in L, v \in \Sigma^*)[v \sqsubseteq w \implies v \in L].$$

This is because if $v \sqsubseteq w$ and w is in L, then v is a permitted factor and thus cannot be excluded from the stringset.

Proposition 5. *An SP stringset L is SP_k iff*

$$\Big(\forall w \notin L\Big)\Big[\Big(\exists v \notin L\Big)\big[\,|v| \leq k \wedge v \sqsubseteq w\big]\Big].$$

In other words, if $L \in \mathrm{SP}_k$ and a string w is forbidden in L, then w contains a subsequence v of length less than or equal to k such that v is also forbidden in L.

The properties of Strictly Piecewise stringsets combine to allow efficient extraction of $\overline{\mathrm{G}^{<}_{L,k}}$ from the automaton representation of such a stringset.

Theorem 4. *If $\mathscr{A}$ is a DFA representing an SP stringset, then a factor w is a forbidden subsequence of $L(\mathscr{A})$ iff there is no path labelled w from the initial state, q_0, to an accepting state.*

Proof. Let $w \in \Sigma^*$ be a string. We first show that if there is no path labelled w from the initial state q_0 to an accepting state, then w is a forbidden subsequence of $L(\mathscr{A})$. Suppose the hypothesis; i.e., that for all accepting states q_f, there is no path $q_0 \overset{w}{\rightsquigarrow} q_f$. Then $w \notin L(\mathscr{A})$, and by the contrapositive of Proposition 4, for all strings $x \in \Sigma^*$, if $w \sqsubseteq x$, then $x \notin L(\mathscr{A})$. Then w is by definition a forbidden subsequence.

Next, we show that if w is a forbidden subsequence of $L(\mathscr{A})$, then there is no path labelled w from the initial state q_0 to an accepting

state q_f. Let w be a forbidden subsequence. Since $\mathscr{A}$ is deterministic, there is exactly one path from q_0 labelled w. Since w is a forbidden subsequence and necessarily a subsequence of itself, $w \notin L(\mathscr{A})$. So the final state of this path must not be accepting. In other words, there is no path labelled w from the initial state, q_0, to an accepting state. ■

A direct consequence of Theorem 4 is that, if $\mathscr{A}$ is a DFA that represents an SP stringset, then its forbidden factors can be extracted by a simple traversal of the graph, collecting the strings that do not end in an accepting state. One concern is that a graph may have cycles, but in the case of a cyclic path $q_0 \overset{w}{\rightsquigarrow} q_1 \overset{x}{\rightsquigarrow} q_1 \overset{y}{\rightsquigarrow} q_2$ there is also a path $q_0 \overset{w}{\rightsquigarrow} q_1 \overset{y}{\rightsquigarrow} q_2$ that avoids the cycle, and in this case wxy is a permitted subsequence iff wy is. So an acyclic traversal of a DFA of an SP stringset L is sufficient to extract $\overline{\mathrm{G}^{<}_{L,k}}$. This extraction algorithm is thus $\Theta(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)})$. Since the maximum length of a piecewise factor of the SP stringset recognized by $\mathscr{A}$ is $\mathbf{card}(Q)$, this is also the number of possible forbidden factors of that stringset and this algorithm is optimal for algorithms that return the set of subsequences. To reduce the result to a minimal set of forbidden factors is quadratic in the same measure, as each extracted factor must be tested for subsequence against each longer one. Given this minimal set of forbidden factors, the necessary factor width k is simply the size of the longest factor.

Note that this traversal will not necessarily yield the expected result on an arbitrary NFA, as an edge could be missing following one path but present following another with the same label.

7 APPROXIMATING REGULAR STRINGSETS IN SP

The minimal SP superset of any stringset L is just the closure of L under subsequence: any such superset must include L, and every SP superset must be closed under subsequence.

In Section 6 we describe an algorithm to extract subsequences from DFAs that represent SP stringsets. To use this to obtain an SP approximation of an arbitrary Regular stringset the DFA must first be closed under subsequence. We can achieve this closure by adding edges on ε in parallel to each existing edge of the automaton; Theorem 5 shows that the set of permitted subsequences of this generated stringset is exactly the same as that of the input.

Theorem 5. *If $\mathscr{A}$ is a DFA and $\mathscr{A}'$ is the automaton produced by adding ε-edges in parallel to each existing edge of $\mathscr{A}$ (applying the subsequence closure algorithm), then the set of permitted factors of $L(\mathscr{A})$ is the same as that of $L(\mathscr{A}')$.*

Proof. In order to show that these sets are equal, we will show that each is a subset of the other.

To show that the set of permitted subsequences of $L(\mathscr{A})$ is a subset of that of $L(\mathscr{A}')$, let $u \in \Sigma^*$ be a permitted subsequence of $L(\mathscr{A})$. Then there exists a word $v \in L(\mathscr{A})$ such that $u \sqsubseteq v$, and since $L(\mathscr{A}) \subseteq L(\mathscr{A}')$ by construction, it follows that $v \in L(\mathscr{A}')$. Thus u is a permitted subsequence in the latter. Since this holds for all such u, the set of permitted subsequences of $L(\mathscr{A})$ is a subset of that of $L(\mathscr{A}')$.

To show that the reverse holds as well, let $w \in \Sigma^*$ be a permitted subsequence of $L(\mathscr{A}')$. Then there exists some string $x \in L(\mathscr{A}')$ such that $w \sqsubseteq x$. Since $\mathscr{A}'$ was formed by allowing a computation to skip edges in $\mathscr{A}$, there must exist some string $y \in L(\mathscr{A})$ such that $x \sqsubseteq y$. By transitivity, it follows that $w \sqsubseteq y$ and thus w is a permitted subsequence of $L(\mathscr{A})$. Since this holds for all such w, the set of permitted subsequences of $L(\mathscr{A}')$ is a subset of that of $L(\mathscr{A})$.

Since each is a subset of the other, these sets are equal. ∎

Since $L(\mathscr{A}')$ is by construction closed under subsequence, it is Strictly Piecewise; Theorem 5 guarantees that its permitted subsequences are the same as those of $L(\mathscr{A})$. If the desired outcome is simply an SP approximation, this algorithm is good: this subsequence closure is the closest SP approximation that contains its input as a subset, and the NFA is produced in linear time in the number of node pairs, $\Theta(\mathbf{card}(Q \times Q))$. If all that is needed is the approximation, this NFA can be determinised and is of size $\Theta(2^{\mathbf{card}(Q)})$. The extraction algorithm from Section 6 can then be used, if desired, to extract the set of forbidden factors from this approximation. The time complexity of the whole process is $\Theta(\mathbf{card}(\Sigma)^{(2^{\mathbf{card}(Q)})})$.

On the other hand, a modified version of the algorithm is more efficient. Let $\mathscr{A}$ be a trim DFA, and create a map $ec : Q \to \{Q\}$ such that $ec(q)$ is the set of states reachable from q (the ε-closure of q). As finding $ec(q)$ is a graph traversal that touches each edge exactly once, it is $\Theta(\mathbf{card}(\Sigma \times Q))$. In the worst case, we find this for each state, so finding the ε-closure map is $\Theta(\mathbf{card}(\Sigma \times Q \times Q))$.

To compute the subsequence closure of $\mathscr{A}$ without explicitly including any edges on ε, we must create new edges and determine the set of initial and accepting states. For each edge $\langle \sigma, p, q \rangle$, create new edges $\{\langle \sigma, p, r \rangle \mid r \in ec(q)\}$. Since every state is reachable from the initial state, in the subsequence closure the set of initial states is simply the set of all states. Since $\mathscr{A}$ is trim, if there is any accepting state at all, then every state must be able to reach a final state, and thus similarly in the subsequence closure the set of accepting states is also the set of all states. Since the original automaton $\mathscr{A}$ is deterministic, there are at most $\mathbf{card}(\Sigma \times Q)$ edges, and since $\mathbf{card}(ec(q))$ is at most $\mathbf{card}(Q)$, assuming sublinear lookup of $ec(q)$, the computation of this subsequence closure is then $\Theta(\mathbf{card}(\Sigma \times Q \times Q))$.

Next, add a marked non-accepting sink state $\perp \notin Q$ and complete the automaton, adding edges to $\perp$ from each state q for each symbol not already labelling an out-edge of q. The time complexity of the completion is $\Theta(\mathbf{card}(\Sigma \times Q))$. Because $\mathscr{A}$ was trim, this sink will necessarily be the unique non-accepting sink. Then there will be at most $\mathbf{card}(\Sigma \times Q)$ edges out of each state.

From this point, we traverse the automaton nondeterministically (grouping paths with the same label) from the set of initial states (equal to Q), where the desired paths are those that end in the unique non-accepting sink state $\perp$ and at least one component path is acyclic. If all component paths are cyclic, then there is a strictly shorter forbidden subsequence that subsumes it, thus the paths are bounded in length by that of the longest possible acyclic path: $\mathbf{card}(Q)$. We alleviate the concern that in an NFA a factor may appear to be forbidden along one path but actually be permitted along another through the requirement that all component paths lead to $\perp$. Then since paths will be at most $\mathbf{card}(Q)$ edges long where each may be any of the $\Theta(\mathbf{card}(\Sigma \times Q))$ edges that start at the current state, we gather $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$ paths. The overall time complexity for this modified algorithm is then the sum of each step, dominated by the gathering of paths. In all, it is $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$.

Again, finding a minimal set requires a quadratic-time filter across the extracted factors, which can either be done during the extraction or as a post-processing step.

As the longest forbidden factor is of length $\Theta(\mathbf{card}(Q))$, there are $\Theta(\Sigma^{\mathbf{card}(Q)})$ possible minimal forbidden factors and this modified algo-

rithm for extracting piecewise factors from a non-SP stringset is still suboptimal, though it is still of lower asymptotic complexity than determinising and using the algorithm of Section 6.

If all that is needed is the Strictly Piecewise approximation, the NFA of the subsequence closure can be produced in $\Theta(\mathbf{card}(Q \times Q))$ time, or the DFA in $\Theta(2^{\mathbf{card}(Q)})$ time. If the set of forbidden factors is desired, the algorithm described in this section provides this in $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$ time. It should be noted that the Strictly Piecewise factors of Regular stringsets are derived from the approximation, in contrast to the Strictly Local factors and approximations where this relationship is inverted. This saves a considerable amount of time in practice.

We now demonstrate the application of this procedure by applying it to the stress pattern of Amele, described in StressTyp2 (Goedemans *et al.* 2015) as follows: "In words of all sizes, primary stress falls on the leftmost heavy syllable, else on the initial syllable." This is shown in Figure 2.

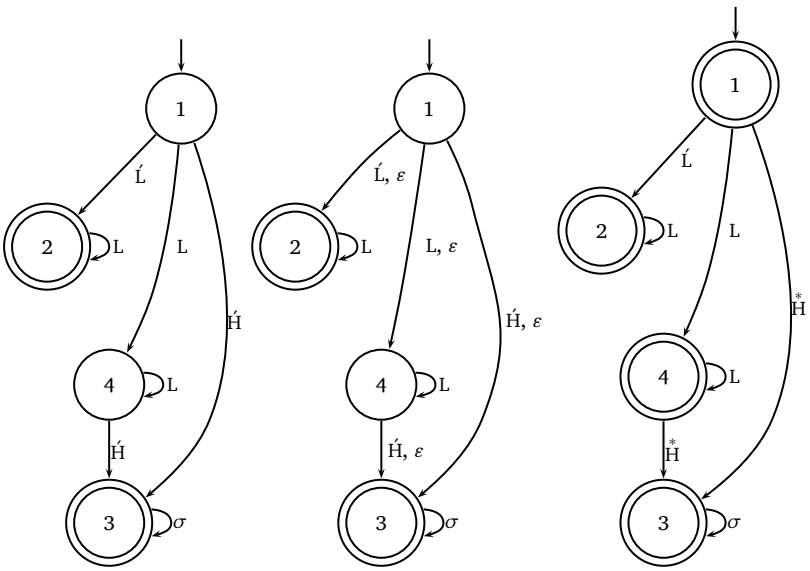

Figure 2: The application of the subsequence closure algorithm to Amele. At left, the normalised DFA representing Amele itself; in centre, the addition of ε-edges; and at right, the normalised result

Note that Amele is fully described by a set of forbidden subsequences and one additional constraint: that some syllable with primary stress must occur, which, following Hyman (2009), we will refer to as obligatoriness. The (non-trimmed) residue of the subsequence

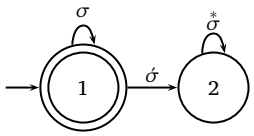

Figure 3:
The residue of Amele's Strictly Piecewise approximation

closure of Amele with Amele itself is shown in Figure 3. The complement of this residue (the *coresidue*) is exactly obligatoriness. While it does not always work out this cleanly, this certainly motivates further analysis of the residue in order to obtain additional constraints.

8 LOCAL AND PIECEWISE TESTABLE CONSTRAINTS

As Rogers *et al.* (2012) showed, three quarters of the lects in the StressTyp2 database are Strictly Local. They also identified a set of Strictly Piecewise constraints that enforce aspects of the remaining lects, although SL and SP constraints are not sufficient by themselves. Working analytically from the English glosses of the stress patterns, they derived a set of nine Locally Testable constraints that, together with the SL and SP constraints suffice to define 104 of the 106 patterns, the exceptions being Cyrenaican and Negev Bedouin Arabics, which are properly Regular.[10] Thus they established that, with those two exceptions, all of the stress patterns are, at most, conjunctions of Locally Testable and Strictly Piecewise (LT + SP) constraints.

Most prominent among these LT constraints is the requirement that primary stress falls on some syllable of every word, i.e. obligatoriness. Since SP stringsets are closed under substring and SL stringsets are closed under substitution of suffixes, this requirement cannot be enforced, in general, by any conjunction of SL and SP constraints. In the lects that are SL, primary stress is required to fall within a fixed distance of either the initial or final syllable which allows obligatoriness to be enforced by initial or final forbidden factors. But SP constraints are oblivious to the ends of words, which accounts for the fact that none of the lects are purely SP.

Note that the complement of obligatoriness – that no primary stress occurs – is SL_1, which puts it in the class of co-SL constraints.

[10] Regular but not Star-Free. See McNaughton and Papert (1971).

These are constraints that are disjunctions of positive literals, a significantly restricted subset of the full set of LT constraints. From a cognitive perspective, co-SL constraints are very nearly as simple as SL constraints. Paraphrasing, SL constraints correspond to "you can't say X", where the X is a fixed string of syllables, while co-SL constraints correspond to "you can't not say X". Note though that, in general, since co-SL constraints are disjunctive, X may be a set of alternatives.

In the case of obligatoriness, though, there is a single disjunct. All that is required is that primary stress is identifiable somewhere. Adding obligatoriness to the sets of forbidden local and piecewise factors we extract allows us to characterise another 18 lects: 98 of the 106 lects (92.5%) are SL + SP + obligatoriness and thus extremely simple in the cognitive resources they require.

This raises the question of whether we can capture the remaining lects by extracting their co-SL constraints. Unfortunately, this is not as simple as just complementing the stringset and extracting forbidden factors. While the original stringset may (in our case, will) enforce a positive literal constraint, it will be lost in the noise of all of the strings that fail to be in the original stringset for other reasons.

Proposition 6. *If L is a stringset that forbids piecewise or free local factors, then $\overline{L}$ cannot forbid piecewise or free, initial or final local factors. If L forbids only initial local factors, $\overline{L}$ cannot forbid piecewise, free or final local factors, although it may forbid initial local factors. The same is true, mutatis mutandis, of final local factors.*

To see why this is so, suppose L has an initial forbidden local factor v. It follows that for all $w \in \Sigma^*$, vw is not in L. Since this applies for all w, $\overline{L}$ has neither free nor final forbidden local factors, nor does it have any forbidden piecewise factors.

If v were instead a free forbidden local factor or a forbidden piecewise factor, these same conclusions hold. However, it is also true in that case that for all $w \in \Sigma^*$, wv is not in L. Since this applies for all w, $\overline{L}$ has no initial forbidden factors either.

If v were instead a final forbidden local factor, then L^{R} has an initial forbidden local factor (v^{R}). Thus $\overline{L^{\mathrm{R}}}$ has neither free nor final forbidden factors. As they are equivalent, this holds for $\overline{L}^{\mathrm{R}}$. Then $\overline{L}$ has neither free nor initial forbidden factors.

The situation is not hopeless, though. Having identified forbidden local and piecewise factors, we can look for co-SL or co-SP constraints in the residue set, the difference between our approximation and the original stringset. Since the forbidden factor constraints have been stripped from this, we can in principle and do in fact get useful co-SL constraints in this way. This is an extension of the methodology that is demonstrated in Section 7 which resulted in the isolation of obligatoriness. This allows us to formally characterise all of the lects that are SL + SP + co-SL *ab initio*, fully computationally.

For the non-strict, non-co-strict constraints, we fall back on the LT constraints identified in the prior work. We treat these as hypotheses and systematically test each subset of the constraints to see if it suffices to complete our approximation. In this way, we determined that five of the eight non-strict, Subregular constraints identified in that work turn out to be unnecessary. The remaining three are of the form $\neg X \vee \neg \acute{\text{H}} \ltimes$ where $X \in \{\text{H}, \grave{\text{H}}, \text{S}\}$.

Each of the two properly Regular stringsets was associated, in the prior work, with a specific Regular constraint. But since one of these constraints is just the conjunction of the other with a fourth LT constraint ($\neg \acute{\text{L}} \vee \rtimes \acute{\text{L}} \ltimes$, i.e., primary stress falls on a light syllable only in monosyllabic words), these can be unified with a single, simple and linguistically suggestive Regular constraint: that a light syllable with primary stress is immediately preceded by an uninterrupted sequence of an odd number of unstressed light syllables that is not itself immediately preceded by more unstressed light syllables. In the actual analysis from StressTyp2, this shows up as a requirement that unstressed syllables and those with secondary stress alternate, with the secondary stress syllables not surfacing.

9 RESULTS AND PROSPECTUS

We have designed and implemented algorithms that, given a finite-state automaton, compute a set of forbidden words, units, initial, free and final local factors that define an SL approximation of the stringset recognised by the FSA, along with a minimal DFA that recognises the residue set: the set of strings in the approximation that are not in the stringset recognised by the FSA. Similarly, we have designed and implemented an algorithm that computes a set of forbidden piecewise

factors that define an SP approximation. The intersection of these approximations is an SL + SP approximation. Then, working with the residue automaton, which recognises exactly the set of strings the approximation accepts that are not in the original stringset, we identify the forbidden factors in its complement. Those that are consistent with the original stringset are required factors of that stringset, that is, they are co-SL constraints. This gives us an SL + SP + co-SL approximation. If the FSA recognises a stringset that is in the union of these three types then the approximation is exact.

We have used these algorithms to implement a system that completely characterises any Regular stringset that is SL + SP + co-SL, fully computationally. We have applied this to the lects in the StressTyp2 database. SL + SP + co-SL constraints cover 98 (92.5%) of the 106 lects in the database that have corresponding FSAs.

For the remaining 6 Subregular lects, we have systematically tested the sufficiency of each of the subsets of the properly LT constraints that Rogers *et al.* (2012) employed in their characterisation. Of these, only obligatoriness and three other constraints were needed to fully characterise these lects. The only properly Regular constraint that is required in characterising the remaining two lects is a hidden alternation pattern that requires an odd number of syllables to occur in certain spans of the word. This is about as simple as non-trivial properly Regular constraints can get and it is reminiscent of the notion of metrical foot that plays an important role in much phonotactic analysis.

Our results sharpen the results of the prior work in the following additional ways. For the individual lects the maximum number of forbidden words is 20. Since the size of our default alphabet is 15 (five degrees of weight and three degrees of stress) and all lects have at least one weight and two levels of stress, the maximum number of forbidden units is 13. The maximum number of forbidden initial factors is 15. The maximum number of forbidden free and final factors is 386 and 117, respectively, but these are all due to Pirahã, an outlier. Without Pirahã they are 185 and 32, respectively. The maximum number of forbidden subsequences is 90, but this is due to Bhojpuri (per Shukla Tiwari), another outlier. Without this lect it is 18.

For the union factor types, there are 14 distinct forbidden units (only unstressed light syllables occur in every lect), 44 distinct for-

bidden words, 35 distinct initial forbidden factors, 904 distinct free forbidden factors and 230 distinct final forbidden factors. The maximum width of forbidden words, initial factors and free factors is 5. The maximum width of final forbidden factors is 6, due to a single lect (Içuã Tupi) which is also the only example of a properly SL_6 stringset, the other SL patterns all being SL_5 or less. There are 126 distinct forbidden subsequences with a maximum width of 4, but this again is due to Bhojpuri (per Shukla Tiwari). Without this lect, the maximum width is only 3.

That is still a lot of factors, too many to draw much insight from. But these are all in ground form, with each syllable and stress combination represented by a distinct alphabet symbol. In future work we plan to adapt the alphabet type to be tuples of features or perhaps non-reëntrant feature structures (adding full feature structures we will leave for others), which will provide opportunities to generalise across those features. We know, just from the phonology, that this will reduce the total number of exemplars significantly.

The algorithms we have presented here have asymptotic time complexity that is exponential in the size of the automaton if the stringset it recognises is Strictly Local or Strictly Piecewise. If it is not, they obtain an optimal Strictly Local approximation in time doubly exponential in the size of the automaton. (The naïve algorithm for finding initial forbidden factors, which suffices for our application, is doubly exponential.) The complexity of the algorithm for obtaining the optimum SP approximation is still just singly exponential. This relatively high degree of complexity is typical of algorithms in this domain, most of which are based on properties of the Syntactic Monoid (or Semigroup), which is exponential in the size of the automaton; the powerset graph is only marginally smaller than the Syntactic Monoid. With the exception of the naïve algorithm, they are all optimal for algorithms that return the sets of forbidden factors of the stringset.

In practice, the algorithms are quite efficient on moderate sized automata. The full methodology, running on our Haskell workbench, processes the entire set of 106 lects (≤ 33 states) in under one minute running on an AMD64 based PC with four cores at 3.7 GHz with 12GB of RAM, with a disproportionate fraction of that time spent processing the SL stringsets with large k. Thus the workbench, with only minimal

optimisation, has proven to be a useful interactive tool for exploring Regular sets of strings.

Nevertheless, the performance can certainly be improved significantly. The asymptotic bound is due to the potential size of the set of factors. This is not, however, the dominant factor in the practical performance. Rather it is the time it takes to generate a minimal DFA from the forbidden factors. This is only necessary for construction of the residue automata. If all that is required is the sets of factors it can be dispensed with. Moreover, it is an easy target for optimisation and is of the type of "embarrassingly parallel" algorithms that Haskell can parallelise extremely effectively. In essence, the performance will ultimately be bound only by the number of cores one has available.

ACKNOWLEDGEMENTS

The work reported here builds on the work of at least a dozen undergraduate students and alumni of Earlham College over the course of about ten years. We are greatly indebted to their willingness to think carefully about hard problems and to collaborate effectively both within the group and over time. We are also indebted to the collaboration of Jeff Heinz and his students in Linguistics and Cognitive Science at the University of Delaware, and to the detailed and extremely helpful suggestions of the anonymous reviewers.

REFERENCES

Pascal CARON (2000), Families of locally testable languages, *Theoretical Computer Science*, 242:361–376.

Jane CHANDLEE (2014), *Strictly local phonological processes*, Ph.D. thesis, University of Delaware.

François DENIS, Aurélien LEMAY, and Alain TERLUTTE (2002), Residual finite state automata, *Fundamenta Informaticae*, 51(4):339–368.

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

Margaret FERO, Dakotah LAMBERT, Sean WIBEL, and James ROGERS (2014), Abstract categories of phonotactic constraints, `https://apps.cur.org/ncur2018/archive/Display_NCUR.aspx?id=83628`, Retrieved 30 January 2018, presented at the National Conference on Undergraduate Research (NCUR'14).

Jie FU, Jeffrey HEINZ, and Herbert G. TANNER (2011), An algebraic characterization of strictly piecewise languages, in Mitsunori OGIHARA and Jun TARUI, editors, *Theory and applications of models of computation*, volume 6648 of *Lecture Notes in Computer Science*, pp. 252–263, Springer, Berlin.

R. W. N. GOEDEMANS, Jeffrey HEINZ, and Harry VAN DER HULST (2015), StressTyp2, `http://st2.ullet.net/`, retrieved 30 January 2018.

Jeffrey HEINZ (2010), Learning long-distance phonotactics, *Linguistic Inquiry*, 41(4):623–661.

Jeffrey HEINZ (2018), The computational nature of phonological generalizations, in Larry HYMAN and Frank PLANK, editors, *Phonological typology*, volume 23 of *Phonetics and Phonology*, chapter 5, pp. 126–195, Mouton De Gruyter, Berlin.

John E. HOPCROFT and Jeffrey D. ULLMAN (1979), *Introduction to automata theory, languages and computation*, Addison-Wesley, Boston, MA.

Larry M. HYMAN (2009), How (not) to do phonological typology: the case of pitch-accent, *Language Sciences*, 31(2–3):213–238.

Adam JARDINE (2016), *Locality and non-linear representations in tonal phonology*, Ph.D. thesis, University of Delaware.

M. LOTHAIRE, editor (1983), *Combinatorics on words*, Cambridge University Press, New York.

Robert MCNAUGHTON and Seymour PAPERT (1971), *Counter-free automata*, MIT Press, Cambridge, MA.

Mark-Jan NEDERHOF (2000), Practical experiments with regular approximation of context-free languages, *Computational Linguistics*, 26(1). `http://aclweb.org/anthology/J00-1003`.

Geofforey K. PULLUM and Barbara C. SCHOLZ (2001), On the distinction between model-theoretic and generative-enumerative syntactic frameworks, in Philippe DE GROOTE, Glyn MORRILL, and Christian RETORÉ, editors, *Logical Aspects of Computational Linguistics: 4th international conference*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pp. 17–43, Springer Verlag, Berlin.

James ROGERS (1994), Capturing CFLs with tree adjoining grammars, in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pp. 155–162, Association for Computational Linguistics, Las Cruces, NM.

James ROGERS (1996), A model-theoretic framework for theories of syntax, in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pp. 10–16, Association for Computational Linguistics, Santa Cruz, CA.

James ROGERS, Jeff HEINZ, Margaret FERO, Jeremy HURST, Dakotah LAMBERT, and Sean WIBEL (2012), Cognitive and sub-regular complexity, in Glyn MORRILL and Mark-Jan NEDERHOF, editors, *Formal Grammar 2012*, volume 8036 of *Lecture Notes in Computer Science*, pp. 90–108, Springer, Berlin.

James ROGERS, Jeffrey HEINZ, Gil BAILEY, Matt EDLEFSEN, Molly VISSCHER, David WELLCOME, and Sean WIBEL (2010), On languages piecewise testable in the strict sense, in Christian EBERT, Gerhard JÄGER, and Jens MICHAELIS, editors, *The mathematics of language: revised selected papers from the 10th and 11th biennial conference on the mathematics of language*, volume 6149 of *Lecture Notes in Artificial Intelligence*, pp. 255–265, FoLLI/Springer, Berlin.

Geoffrey SAMPSON (1975), *The form of language*, Weidenfeld and Nicolson, London.

Yves SCHABES and Richard C. WATERS (1993), Lexicalized context-free grammars, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pp. 121–129, Association for Computational Linguistics, Columbus, OH.

Imre SIMON (1975), Piecewise testable events, in Helmut BRAKHAGE, editor, *Automata theory and formal languages*, volume 33 of *Lecture Notes in Computer Science*, pp. 214–222, Springer Verlag, Berlin.

Wolfgang THOMAS (1982), Classifying regular events in symbolic logic, *Journal of Computer and Systems Sciences*, 25(3):360–376.