Extracting Forbidden Factors from Regular Stringsets

James Rogers
Dept. of Computer Science
Earlham College
Richmond, IN, USA
jrogers@cs.earlham.edu

Dakotah Lambert
Depts. of Mathematics and Computer Science
Earlham College
Richmond, IN, USA
djlambe11@earlham.edu

Abstract

The work presented here continues a program of completely characterizing the constraints on the distribution of stress in human languages that are documented in the StressTyp2 database with respect to the Local and Piecewise sub-regular hierarchies.

We introduce algorithms that, given a Finite-State Automaton, compute a set of forbidden words, units, initial factors, free factors and final factors that define a Strictly Local (SL) approximation of the stringset recognized by the FSA, along with a minimal DFA that recognizes the residue set: the set of strings in the approximation that are not in the stringset recognized by the FSA. If the FSA recognizes an SL stringset, then the approximation is exact (otherwise it overgenerates).

We have applied these tools to the 106 lects that have associated DFAs in the StressTyp2 database, a wide-coverage corpus of stress patterns that are attested in human languages. The results include a large number of strictly local constraints that have not been included in prior work categorizing these patterns with respect to the Local and Piecewise Sub-Regular hierarchies of Rogers et al. (2012), although, of course, they do not contradict the central result of that work, which establishes an upper bound on their complexity that includes strictly local constraints.

1 Introduction

A stringset L is Strictly k-Local if and only if (iff) it is completely determined by its k-factors: the substrings of length at most k that occur in strings $\rtimes \cdot w \cdot \ltimes$ for $w \in L$. (The '$\rtimes$' and '$\ltimes$' are endmarkers.) That is to say, L contains all and only the strings that are generated by the substring relation from that set of k-factors. The class of stringsets that are Strictly k-local for some k is known as SL. This is at the bottom of the local side of a collection of classes of stringsets, all strict subclasses of the class of Regular stringsets, which are hierarchically related and are characterized by finite sets of either substrings (the Local Hierarchy) or subsequences (the Piecewise Hierarchy) or by combinations of the two. In Rogers et al. (2012) we argue that these hierarchies provide a robust notion of cognitive complexity for constraints on strings.

The long-term project of our group is to characterize 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), we 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, we derived a set of "primitive" constraints sufficient to define all of the patterns by co-occurrence and classified them into abstract categories (Fero et al., 2014). Most of these constraints were, in fact, SL, and it turned out that all of the patterns could be defined by co-occurrence of constraints at the bottom two levels of the hierarchies. This is significant, since at these levels it is possible to determine whether a string satisfies a constraint solely on the basis of the information that is explicitly contained in the string, without inferring any additional structure. Recent work by Heinz and his co-workers (Heinz, forthcoming; Heinz, 2010; Chandlee, 2014; Jardine, 2016) suggests that much of phonology may be characterizable by correspondingly simple sets of structures or functions.

The work on primitive constraints, however,

Proceedings of the 15th Meeting on the Mathematics of Language, pages 36–46,
London, UK, July 13–14, 2017.

did not include any of the factors from the SL stringsets because the algorithm for determining if a given Finite State Automaton (FSA) recognizes 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.

In Section 2 we introduce our notation and basic formal definitions. In Section 3 we formally define Strictly Local stringsets and discuss their formal properties. In Section 4 we distinguish five types of forbidden factors—factors in the complement of the set of factors that generate the stringset. In Section 5 we develop our algorithms for extracting those factors given a Finite State Automaton. In Section 7 we extend these algorithms in a way that allows them to be used to partition non-SL stringsets in a way that provides a set of SL constraints that approximates it (to varying degrees of closeness) and an automaton that captures the non-SL aspects of the stringset. We close with thoughts about where these results lead.

2 Formal Preliminaries

A finite state automaton (FSA) is an edge-labeled 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 labeled σ from q_1 to q_2, I is the set of initial states, and F is the set of accepting states. Let $\mathcal{A} = \langle \Sigma, Q, \delta, I, F \rangle$.

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

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

This is an accepting path on w if q_n is in F, else it is a non-accepting path.

The automaton $\mathcal{A}$ 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. We also consider trim functional automata to be deterministic, where $\mathcal{A}$ is *trim* iff for all states $q \in Q$ there is some accepting path from q.

An automaton is *minimal* iff it is deterministic and no two states are Nerode-equivalent[1]. Further, it is *normalized* iff it is both minimal and trim.

Given a string w, the factors of w are those v that are substrings of w (notation: $v \preccurlyeq w$). If k is the length of v (notation: $|v| = k$) then v is a k-factor of w.

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

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

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

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

This is because if $\mathcal{A}$ 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 Strictly Local Stringsets

L is *Strictly k-Local* ($L \in \mathrm{SL}_k$) iff it is completely characterized by its k-factors. Let Σ be the alphabet of L and define $\mathrm{F}_k(\Sigma) = \{v \in \Sigma^* \mid |v| = k\}$ and $\mathrm{F}_{\leq k}(\Sigma) = \bigcup_{1 \leq i \leq k} [\mathrm{F}_i(\Sigma)]$. For any string $w \in \Sigma^*$, the k-factors of w are

$$\mathrm{F}_k(w) = \begin{cases} \{w\} \quad \text{if } |w| <= k, \\ \{v \in \mathrm{F}_k(\Sigma) \mid \\ \qquad w = w_1 v w_2,\ w_1, w_2 \in \Sigma^*\} \\ \qquad \text{otherwise.} \end{cases}$$

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

Let $G \subseteq \mathrm{F}_{\leq k}(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$ be the set of permitted factors in L. Then the stringset generated by G is

$$L(G) = \{w \in \Sigma^* \mid \mathrm{F}_{\leq k}(\rtimes \cdot w \cdot \ltimes) \subseteq G\} .$$

Since Σ is assumed to be finite, $\mathrm{F}_{\leq k}(\Sigma)$ is also finite, and an SL_k language can equivalently be defined in terms of its forbidden factors: $\overline{G} =$

[1] 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

$\mathrm{F}_{\leq k}(\Sigma) - G$. 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 characterizes SL_k.

Proposition 1 (Suffix Substitution Closure) (SSC)

$L \in SL_k$ *iff*
$(\forall x \in F_{k-1}(\Sigma))[$ *if* $w_1 = u_1 \cdot x \cdot v_1 \in L$ *and* $w_2 = u_2 \cdot x \cdot v_2 \in L$ *then* $u_1 \cdot x \cdot v_2 \in L$ $]$.

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

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

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

are Nerode Equivalent. If $\mathcal{A}$ 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 synchronize after at most $k-1$ steps.

This is the basis of the algorithm used by Edlefsen et al. (2008)[2] to determine if a given $\mathcal{A}$ recognizes an SL stringset and, if it does, to find the parameter k.

Proposition 2 *Suppose $\mathcal{A}$ is a normalized DFA. Then $L(\mathcal{A}) \in SL_k$ iff every path from Q in PSG($\mathcal{A}$) that is of length $k-1$ leads to a singleton vertex. 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, $L(\mathcal{A})$ 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 PSG($\mathcal{A}$) 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 PSG($\mathcal{A}$) 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 $\Theta(2^{\mathbf{card}(Q)})$.

The following is an immediate consequence of this proposition.

[2]The pair-graph algorithm was first published in Caron (2000).

Lemma 2 *If if $\mathcal{A}$ is a normalized DFA and $L(\mathcal{A}) \in SL_k$ then all cycles in PSG($\mathcal{A}$) are cycles of singletons.*

4 Classes of Forbidden Factors

Factors may or may not include either a left-end marker at the beginning or a right-end marker at the end. 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 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 '$\rtimes$', then it can only occur at the left end of the word; this is an *initial forbidden factor*. If the last symbol is '$\ltimes$', 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 length $k-1$ prefixes and suffixes, but not, in general length k prefixes and suffixes.[3] Finally, if a factor contains both end-markers it is a *forbidden word*, where the word it forbids is actually of length $k-2$.

5 Forbidden Factors of SL Stringsets

5.1 Free Forbidden Factors

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

So the set of all labels of paths Q to $\emptyset$ in PSG($\mathcal{A}$) are free forbidden factors of $L(\mathcal{A})$, moreover, that set includes all free forbidden factors of $L(\mathcal{A})$. Since in general PSG($\mathcal{A}$) may include cycles and even in the case that $L(\mathcal{A})$ is SL it may include cycles of singleton vertices, in general this

[3]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.

set of paths will be infinite. (In fact, since PSG($\mathcal{A}$) 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 labeled 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 strictly shorter. Thus none of those paths are minimal free forbidden factors. Note, also, that if $L(\mathcal{A}) \in$ 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 $L(\mathcal{A})$ is SL then there is some bound such that all cyclic paths from Q to $\emptyset$ in PSG($\mathcal{A}$) with length greater than that bound will be labeled 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 $L(\mathcal{A})$ is just the set of labels from paths from Q to $\emptyset$ in PSG($\mathcal{A}$) 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 PSG($\mathcal{A}$).

Lemma 3 *If v and w label* acyclic *paths from Q to $\emptyset$ in PSG($\mathcal{A}$) and $v \preccurlyeq w$, then $w = uv$ for some $u \in \Sigma^*$.*

Proof: $v \preccurlyeq w$ implies that $w = uvx$ for some $u, x \in \Sigma^*$. Since $Q \overset{v}{\rightsquigarrow} \emptyset$ and all vertices $S \subseteq Q$, 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 $\emptyset$, contradicting the assumption of acyclicity. ⊣

Lemma 4 *If a path from Q to $\emptyset$ in PSG($\mathcal{A}$), with $L(\mathcal{A}) \in$ SL 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 $L(\mathcal{A})$ 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:

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

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. ⊣

Theorem 1 *If $L(\mathcal{A}) \in$ SL then a string w is a free forbidden factor of $L(\mathcal{A}) \in$ SL iff it labels a path in PSG($\mathcal{A}$) 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 $L(\mathcal{A}) \in$ 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 labeled $u(vw)^i x$ as in the proof of Lemma 4 is labeled 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.

5.2 Final Forbidden Factors

Suppose $\mathcal{A}$ is a DFA. A factor w is a final forbidden factor of $L(\mathcal{A})$ iff there is no path from q_0 to an accepting state in the transition graph of $\mathcal{A}$ 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 no there is no such accepting path, then w is a free forbidden factor.) If $\mathcal{A}$ is normalized then w is a final forbidden factor iff all paths labeled 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 PSG($\mathcal{A}$) labeled w ends at a non-empty vertex that is disjoint with F.

Lemma 5 *Suppose $\mathcal{A}$ is a DFA. No final forbidden factor of $L(\mathcal{A})$ includes a free forbidden factor of $L(\mathcal{A})$ as a substring.*

This is because if v is a free forbidden factor of $L(\mathcal{A})$ then the path from Q in PSG($\mathcal{A}$) leads to $\emptyset$ and, hence, the path labeled v from any vertex of PSG($\mathcal{A}$) leads to $\emptyset$ as well.

Note that a final forbidden factor may include another as a suffix. (It is irrelevant whether it includes an final forbidden factor as a non-suffix, since final forbidden factors are, by definition, only relevant as suffixes.)

Theorem 2 *If a path from Q to a non-empty vertex disjoint from F in PSG($\mathcal{A}$), with $L(\mathcal{A}) \in$ 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.*

The proof is essentially the same as the proof of Lemma 4.

5.3 Initial Forbidden Factors

Suppose $\mathcal{A}$ is a DFA. A string w is an initial forbidden factor of $L(\mathcal{A})$ iff it is w^{R} (w reversed) for some w, a final forbidden factor of $L(\mathcal{A}^{\mathrm{R}})$, where $\mathcal{A}^{\mathrm{R}}$ is the DFA that recognizes the reversal of $L(\mathcal{A})$.

5.4 Forbidden Words

Suppose $\mathcal{A}$ is a DFA and $L(\mathcal{A}) \in \mathrm{SL}_k$. Then w is a forbidden word of $L(\mathcal{A})$ iff it labels a path of length less than or equal to k that leads from q_0 to a state in $Q - F$.

6 Algorithms

Theorem 1 guarantees that if we do a breadth-first bottom-up traversal of PSG($\mathcal{A}$) 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 labeled 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($\mathcal{A}$) as a top-down traversal of PSG$^{\mathrm{R}}(\mathcal{A})$, in which each of the edges of PSG($\mathcal{A}$) 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-labeled graph $\langle V, E\rangle$ as a computational structure, is a 3-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$_0$ for finding free forbidden factors includes just the trivial (0-length) path from $\emptyset$. For finding 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 labeled 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 $L(\mathcal{A}) \in \mathrm{SL}_k$. (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.

Pseudo-code for the algorithms is given in Figures 1 and 2.

6.1 Forbidden Words for SL Stringsets

If $L(\mathcal{A}) \in \mathrm{SL}_k$ and $\mathcal{A}$ 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 $\mathcal{A}$.

6.2 Forbidden Units

If $\mathcal{A}$ is normalized (minimal and trim), the forbidden units of $L(\mathcal{A})$ are just the symbols of Σ that do not label any edge in δ. In PSG($\mathcal{A}$) 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 most 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 $\mathcal{A}$. Since we are building PSG($\mathcal{A}$) 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.)

7 Forbidden Factors of non-SL Stringsets

Every non-SL stringset can be fully defined by the conjunction of a set of SL constraints (possibly trivial: Σ^*, $\emptyset$ and Σ^+ are 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. We would like to be able to factor the constraints so that the non-SL constraints capture, to the extent possible, just the non-strictly-local aspects of the patterns.

The problem isn't finding factors that characterize the stringset, the problem is that there are too many of them. $\Sigma^* - L(\mathcal{A})$, augmented with left and right endmarkers, is a set of forbidden factors that characterizes $L(\mathcal{A})$ exactly. It is, of course, in general infinite and necessarily so if $L(\mathcal{A})$ is not SL.

The algorithms for SL stringsets are still partially correct for non-SL stringsets. The problem is that if $L(\mathcal{A})$ 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 localize the reason that the stringset is not SL. They capture circumstances under which the automaton fails to synchronize ever; they identify places in which SSC (Proposition 1) fails for $L(\mathcal{A})$.

As with the set of forbidden words, the set of labels of the paths in PSG($\mathcal{A}$) that include non-singleton cycles are all legitimate forbidden factors of $L(\mathcal{A})$, 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 $L(\mathcal{A})$.

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 4 may be infinite, so there is no guarantee of termination even when following only singleton cycles.

Another approach would be to modify $\mathcal{A}$, working backward from PSG($\mathcal{A}$), in a way that would eliminate the non-singleton cycles. We have not really pursued this idea, but our sense is that it is likely to fail for the same reason as simply not following non-singleton cycles fails.

In any case, we are looking for a set of forbidden factors that approximates $L(\mathcal{A})$. Since none of our algorithms introduces constraints that are not manifest in the automaton, the approximation will overgenerate. The issue is how close do we need it to be.

7.1 SL Approximations

First of all, as we noted above, Σ^* is an SL approximation of every stringset over Σ. But it's a particularly licentious one. Another possibility is to only gather the forbidden factors that label non-cyclic paths in PSG($\mathcal{A}$). This will miss many forbidden factors that may well be significant—all those factors labeling paths with singleton cycles that would have eventually been subsumed if there were no non-singleton cycles. On the other hand, it gives the smallest set of forbidden factors that comprise a reasonable approximation of $L(\mathcal{A})$.

Another way of bounding the traversal is to note that no acyclic path from Q to $\emptyset$ in PSG($\mathcal{A}$) can be $2^{\mathbf{card}(Q)} - 2$ or longer. But the set of factors gathered by a traversal with this bound, although arguably the largest justifiable set of forbidden factors, is almost certainly unreasonably large.

SL approximations that are too large are misleading both in terms of the apparent complexity of the SL aspects of the constraints and in terms of the their non-SL aspects, which will appear to need to include many exceptions in order to account for the strings excluded by the SL approximation. When the SL approximation overesti-

mates, the non-SL residue undergeneralizes.

In some applications, there may be a theoretically justified bound on how long the relevant factors are, that is, on how many times a cycle should be followed in the traversal. As we noted in the introduction, all of the SL stress patterns in StressTyp2 are SL_k for $k \leq 6$. Thus one may well be justified in limiting the SL fragment to factors of length no more than six. Even assuming the bound is well-justified, this is still likely to generate too close an approximation. Forbidden factors that should properly be captured by the non-SL constraints, that involve non-singleton cycles that are not needed to terminate the traversal of singleton cycles, will be included. If the goal is to explore the nature of the constraints across a collection of stringsets these will likely be misleading, particularly since half of the patterns in StressTyp2 are SL_3 (or less, SL is an inclusive hierarchy in k).

It is straightforward to modify the algorithms given above for either of these approaches. Cycles can be completely excluded by modifying the definition of $\mathrm{Extensions}$. Limits on the size of the factor are just depth limits on the traversal. It is also straightforward to combine these, only following singleton cycles and only doing it up to a depth limit. To bound the search for forbidden words we first compute the sets of forbidden initial, free and final factors and then bound the depth to $\max(|\mathrm{frFF}| - 2, |\mathrm{inFF}| - 1, |\mathrm{fiFF}| - 1)$, where $|\mathrm{frFF}|$, $|\mathrm{inFF}|$, $|\mathrm{fiFF}|$ are the maximum width of the free, initial and final factors, respectively.

As our goal in developing these algorithms is to provide tools that phonologists can use productively in exploring systems of phonotactic constraints the third approach to bounding the traversal seems most useful, although we have currently only implemented the acyclic path approach.

7.2 Residue Automata

When the algorithms are run on automata that recognize non-SL stringsets the result is a set of forbidden factors for the approximated stringset. We are just as interested in the characteristics of the stringset that these forbidden factors miss. 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 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 recognizing 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 recognizes exactly the set of strings that are overgenerated by the SL approximation.

Using well-known algorithms for combining automata, it is straightforward to construct an automaton $\mathcal{A}_{\mathrm{FF}}$ that recognizes the set of strings licensed by the set of forbidden factors. One starts with deterministic automata that recognize each of the given factors, complements them and then builds the automaton that recognizes the intersection of those complements. It is then straightforward to construct $\mathcal{A}_{\mathrm{res}}$, the residue automaton[4] which recognizes exactly $L(\mathcal{A}_{\mathrm{FF}}) - L(\mathcal{A})$. This residue automaton captures exactly the non-SL aspects of $L(\mathcal{A})$, up to the degree to which the forbidden factors approximate the strictly SL aspects of $L(\mathcal{A})$.

8 Results and Prospectus

We have designed and implemented algorithms that, given a Finite-State Automaton, compute a set of forbidden words, units, initial factors, free factors and final factors that define an SL approximation of the stringset recognized by the FSA, along with a minimal DFA that recognizes the residue set: the set of strings in the approximation that are not in the stringset recognized by the FSA. If the FSA recognizes a stringset that is SL, then the approximation is exact.

As we explain in Section 7.1, the closeness of the approximation is a parameter that may be varied depending on the application. As we have implemented it, we obtain the smallest set of factors that is arguably a reasonable approximation.

[4]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 wrt some prefix. "Residual" in that context is justified from the perspective of factoring strings.

We have also implemented an algorithm that collects the union of the forbidden factors of each type from a collection of these results, although we don't present it here, the algorithm being obvious.

We have applied these tools to the 106 lects that have associated DFAs in the StressTyp2 database. 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 some lects have only 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.

For the union factor types, there are 14 distinct forbidden units (only unstressed light syllables occur in every lect), 44 distinct forbidden words, 35 distinct forbidden initial factors, 904 distinct forbidden free factors and 230 distinct forbidden final factors. The maximum width of forbidden words, initial factors and free factors is 5. The maximum width of forbidden final factor 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_4 or less.

That is still a lot of factors, too many to draw much insight from. But these are all in ground form, with each syllable type represented by a distinct alphabet symbol. In future work we plan to adapt the alphabet type to be tuples of features or perhaps non-re-entrant feature structures (adding full feature structures we will leave for others), which will provide opportunities to generalize 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, are asymptotically exponential-time in the size of the automaton, but that is actually optimal for algorithms that construct sets of ground factors: the worst case size of the set of factors of the stringset of an automaton with $\mathbf{card}(Q)$ states is $\Omega(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)})$. Nevertheless these algorithms are actually quite effective in practice. We have incorporated them into a Haskell workbench for manipulating automata with a particular focus on logical descriptions of sub-regular constraints. With only minimal optimization the algorithm computes the forbidden factors and the residue automaton for all 106 lects in our corpus in less than an hour, which is practical as it stands, but can be improved significantly. The asymptotic bound is due to the potential size of the power-set graph as well as the potential size of the set of factors. These are 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 an easy target for optimization; the intersection step, a critical path in the construction, can be done in time logarithmic in the number of factors, for example. There are many other easy opportunities for optimization and Haskell provides a particularly powerful platform form implementing them.

Acknowledgments

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 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, The 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*, pages 66–73.

Margaret Fero, Dakotah Lambert, Sean Wibel, and James Rogers. 2014. Abstract categories of phonotactic constraints. `https://ncurdb.cur.org/ncur2017/archive/`

Display_NCUR.aspx?id=83628, Retrieved 19 May 2017, April. Presented at the National Conference on Undergraduate Research (NCUR'14).

R. W. Goedemans, Jeffrey Heinz, and Harry van der Hulst. 2015. http://st2.ullet.net\slashfiles\slashfiles\slashst2-v1-archive-0415.tar.gz, April. Retrieved 24 Jun 2015.

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

Jeffrey Heinz. forthcoming. The computational nature of phonological generalizations. In Larry Hyman and Frans Plank, editors, *Phonological Typology*, Phonetics and Phonology. Mouton. Final version submitted November 2015. Expected publication in 2017.

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

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

Mark-Jan Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics, Volume 26, Number 1, March 2000*.

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*, pages 90–108. Springer.

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*, pages 155—162, Las Cruces, NM. Association for Computational Linguistics.

Yves Schabes and Richard C. Waters. 1993. Lexicalized context-free grammars. In *31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 121–129, Columbus, OH. Association for Computational Linguistics.

FREEFFS

Given: $PSG^{\,\mathrm{R}} = \langle V, E\rangle$
Where: $V \subseteq \mathcal{P}(Q),\ E \subseteq \Sigma \times \mathcal{P}(Q) \times \mathcal{P}(Q)$
▷ The reversed powerset graph of a DFA $\mathcal{A} = \langle \Sigma, \delta, I, F\rangle$

1 **Let:**
2 $\mathit{Front}_{\,0} = \{\langle \emptyset, \{\emptyset\}, \varepsilon\rangle\}$
▷ The initial frontier,
3 $\mathit{Goals} = \{Q\}$
4 $\mathit{Extensions}(\langle v, S, w\rangle) = \{\langle v', S \cup \{v\}, \sigma \cdot w\rangle \mid \langle \sigma, v, v'\rangle \in E$ and either $v' \notin S$ or v is singleton$\}$
▷ Outedges that are acyclic if not from a singleton
5 **while** $\mathit{Front}_{\,i} \neq \emptyset$
Construct:
6 $\mathit{FrFF} = \bigcup[\mathit{FF}_{\,1}, \mathit{FF}_{\,2}, \ldots]$,
▷ the set of free forbidden factors of $L(\mathcal{A})$
7 in stages: $\textsc{Stage}_1, \textsc{Stage}_2, \ldots$
▷ as given in Figure 2

FINALFFS

Given: $PSG^{\,\mathrm{R}} = \langle V, E\rangle$
▷ as in FREEFFS

1 **Let:**
2 $\mathit{Front}_{\,0} = \{\langle S, \{\emptyset\}, \varepsilon\rangle \mid S \subseteq Q - F\}$
▷ The set of trivial paths from vertices disjoint with F
3 $\mathit{Goals} = \{Q\}$
4 $\mathit{Extensions}(\langle v, S, w\rangle)$
▷ as in FREEFFS
5 **while** $\mathrm{Front}_i \neq \emptyset$
Construct:
6 $\mathit{FiFF} = \bigcup[\mathit{FF}_{\,1}, \mathit{FF}_{\,2}, \ldots]$,
▷ the set of final forbidden factors of $L(\mathcal{A})$
7 in stages: $\textsc{Stage}_1, \textsc{Stage}_2, \ldots$
▷ as given in Figure 2

Figure 1: Main procedures for Free and Final forbidden factors.

STAGE$_i$

Given:

$Front_{\,i-1} \subseteq \{\langle v, S, w\rangle \mid v \in V, S \subseteq V, w \in \Sigma^*\}$

▷ The frontier of the search, a set of $Path$

Where: $v \in V$ is the final vertex,

$S \subseteq V$ is the (unordered) set of vertices in the path,

$w \in \Sigma^*$ is the sequence of labels of the edges in the path, in reverse order

$Goals \subseteq V$ is the set of goal vertices

$Extensions$ is a function taking a $Path$ to its qualified extensions

Construct: $Front_{\,i}$, $FF_{\,i}$

1 **ForEach** $Path \in Front_{\,i-1}$

2 **ForEach** $\langle v', S \cup \{v\}, \sigma \cdot w\rangle \in$ Extensions(Path)

3 **if** $\sigma \cdot w \notin FF_i$ ▷ $\sigma \cdot w$ has not already been found to be an i-FF

4 **then if** $v' \in$ Goals ▷ $\sigma \cdot w$ is an i-FF

then

5 Front$_i \leftarrow$ Front$_i - \{\langle -, -, \sigma \cdot w\rangle \in$ Front$_i\}$

▷ Remove any paths labeled with this factor from Front$_i$

6 $FF_{\,i} \leftarrow FF_{\,i} \cup \{\sigma \cdot w\}$

▷ Add $\sigma \cdot w$ to FF_i

else

7 Front$_i \leftarrow$ Front$_i \cup \{\langle v', S \cup \{v\}, \sigma \cdot w\rangle\}$

▷ Add extension to Front$_i$

End of STAGE$_i$.

Figure 2: Gathering Forbidden Factors