LING890—Topics in Computational Linguistics
Strictly Local Stringsets

Jim Rogers
jrogers@udel.edu

Spring '13

1 Some basic concepts

k-Factor
$\mathrm{F}_k(w)$

Definition 1 (k-Factors, $\mathbf{F}_k$) *Let $\boldsymbol{F}_k(w)$ denote the set of length k sequences of adjacent symbols that occur in w. If $|w| \leq k$ then $F_k(w)$ is just the (single) sequence of symbols in w.*

$$F_k(L) \stackrel{def}{=} \{F_k(w) \mid w \in L\}.$$

Similarly

$$F_{<k}(w) \stackrel{def}{=} \bigcup_{2<i<k} [F_i(w)]$$

etc.

Example 1

$$\begin{aligned} F_2(\rtimes abab \ltimes) &= \{\rtimes a, ab, ba, b \ltimes\} \\ F_3(\rtimes abab \ltimes) &= \{\rtimes ab, aba, bab, ab \ltimes\} \\ F_7(\rtimes abab \ltimes) &= \{\rtimes abab \ltimes\} \\ F_{\leq 3}(\rtimes abab \ltimes) &= F_2(\rtimes abab \ltimes) \cup \{\rtimes ab, aba, bab, ab \ltimes\}. \end{aligned}$$

1.1 Some terminology

grammar
generator

Grammar: Most generally, a grammar (or **generator**) is an abstract algorithm for constructing objects in some particular set, usually strings in a stringset or, later, trees in a treeset, etc. A grammar defines a set by **generating** it.

generation

grammar
formalism

Grammar Formalism: A grammar formalism is a general definition of a class of grammars. The formalism represents the common aspects of the class of grammars. A specific grammar in the class specializes the algorithm for a specific set of objects by specifying various parameters of the formalism. A grammar formalism defines a class of sets of objects: those which can be defined by a grammar within the formalism.

derivation **Derivation:** A derivation is a sequence of objects which represents the way in which a particular object can built by a particular grammar. A correct derivation will always end with an object in the class defined by the grammar, but the intermediate objects of the grammar are not necessarily of the same type.

automaton **Automaton:** An automaton (plural automata) is an abstract algorithm for checking whether an object is in a particular set. An automaton recognition defines a set by **recognizing** it.

class of automata **Class of automata:** A class of automata is a set of automata that share the same basic structure. The class is specified by giving a generalized algorithm. A specific automaton in the class specializes this for a specific set of objects by specifying various parameters. A class of automata defines a class of sets of objects, those that can be recognized by automata in the class.

computation **Computation:** A computation is a sequence of representations of the state of an automaton: the internal state, the input and whatever auxiliary information it may keep. These representations are known as **instantaneous descriptions** of the automaton. A computation may **terminate** (end) accepting the input or rejecting the input or it may continue indefinitely (the computation is infinite), in which case it is **non-terminating**.

instantaneous description

terminate

non-terminating

transition graph **Transition graph:** A transition graph for an automaton is a graphical (in the mathematical sense—circles and arrows) representation of all of the possible computations of the automaton.

abstract characterization **Abstract Characterization:** An abstract characterization is a property of a class of sets of objects that is based solely on the structure of the sets of objects in that class. Crucially, it does not depend explicitly on the way in which the set has been defined; it can be verified that a set does or does not exhibit the property without resorting to reasoning about specific mechanisms for defining it. For this reason, we (as in us) are particularly fond of abstract characterizations.

2 Strictly 2-Local Stringsets

2.1 Strictly 2-Local Grammars

Strictly 2-Local grammar **Definition 2 (Strictly 2-Local grammar)** *A $\mathcal{G}$ is a pair $\langle \Sigma, T \rangle$:*

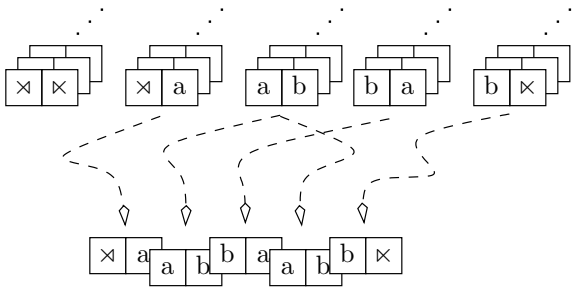

Figure 1: A strictly 2-local generator.

Σ —*an alphabet*
$T \subseteq F_2(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$ —*a set of 2-Factors over* $\Sigma \cup \{\rtimes, \ltimes\}$

Note that '$\rtimes$' may only occur as the first symbol of a factor in T and '$\ltimes$' may only occur as the last.

Definition 3 *The set of strings derived by a strictly 2-local grammar* $\mathcal{G}$ *is*

$$L(\mathcal{G}) \stackrel{def}{=} \{w \in \Sigma^* \mid F_2(\rtimes \cdot w \cdot \ltimes) \subseteq T\}.$$

augmented string

We will refer to $\rtimes \cdot w \cdot \ltimes$ as an **augmented string**.

SL_2

Definition 4 ($\mathbf{SL}_2$) *The class of* ***Strictly 2-Local stringsets*** *is the class of stringsets that can be generated by strictly 2-local grammars.*

$$SL_2 \stackrel{def}{=} \{L \mid (\exists \mathcal{G})[L = L(\mathcal{G})],\ \mathcal{G} \text{ strictly 2-local}\}$$

The choice to use strictly 2-local grammars (as opposed to one of the other ways of defining SL_2 stringsets) as our official definition of SL_2 is arbitrary.

derivation

Definition 5 *A* ***derivation*** *of a string* $w = \sigma_1\sigma_2 \cdots \sigma_{n-1}\sigma_n$ *by a* SL_2 *grammar* $\mathcal{G} = \langle \Sigma, T \rangle$ *is a sequence of 2-factors*

$$\langle \rtimes\sigma_1,\ \sigma_1\sigma_2,\ \ldots,\ \sigma_{n-1}\sigma_n,\ \sigma_n\ltimes \rangle$$

where each 2-factor is in T.

yield

The ***yield*** *an* SL_2 *derivation is the result of combining the factors in the derivation via a modified form of concatenation:*

$$\rtimes\sigma_1 \circ \sigma_1\sigma_2 \circ \ldots \circ \sigma_{n-1}\sigma_n \circ \sigma_n\ltimes \ = \ \rtimes\sigma_1\sigma_2 \cdots \sigma_{n-1}\sigma_n\ltimes$$

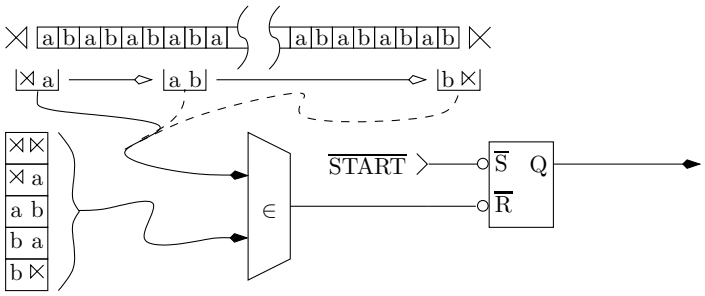

Figure 2: A strictly 2-local automaton.

where, for all $w \in \{\rtimes\} \cdot \Sigma^* \cup \Sigma^*$ *and* $\sigma_i, \sigma_j \in \Sigma \cup \{\ltimes\}$*:*

$$w\sigma_i \circ \sigma_i\sigma_j \stackrel{def}{=} w\sigma_i\sigma_j$$

tiling game

Derivations of strictly 2-local grammars can be viewed as a **tiling game** in which one is given a set of tiles of certain types (T here), with infinitely many tiles of each type, and rules for playing them:

- Start with any tile of the form $\rtimes\sigma \in T$.
- If the last (rightmost) tile is of the form $\sigma_1\sigma_2$ you may play any tile of the form $\sigma_2\sigma_3 \in T$ by overlapping the 'σ_2's.
- If you can play some tile(s), you must play one of them.
- A correct derivation ends with a tile of the form $\sigma\ltimes \in T$.

2.2 Strictly 2-Local Automata

Definition 6 (Strictly 2-Local Automata) *A* Strictly 2-Local Automaton $\mathcal{A}$ *is a pair* $\langle \Sigma, T \rangle$*, where:*

$$\begin{array}{lll} \Sigma & & \text{—}\textit{an alphabet} \\ T & \subseteq F_2(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\}) & \text{—}\textit{a set of 2-Factors over } \Sigma \cup \{\rtimes, \ltimes\} \end{array}$$

The automaton has a window of width two which it scans across the input (an augmented string), one symbol at a time, starting with the left end. It starts with the output set to TRUE. Each pair of symbols that shows up in

the window (including the first and last) is checked against a look-up table T. If, at any step of the computation, a pair is not in the table, the output is set to FALSE. If the output goes FALSE it stays FALSE until the automaton is started again. The automaton accepts iff it halts with the output TRUE.

scanner

Automata that work in this general way are called **scanners**.

Formally:

Instantaneous Description ID

Definition 7 *An **Instantaneous Description (ID)** of a strictly 2-local automaton $\mathcal{A} = \langle \Sigma, T \rangle$ is a pair:*

$$\langle \sigma, v \rangle \in (\{\rtimes\} \cup \Sigma) \times (\Sigma^* \cdot \{\ltimes\})$$

An ID $\langle \sigma_1, \sigma_2 u \rangle$ represents the state of the automaton when $\sigma_1\sigma_2$ is in the scanning window and u is the content of the input to the right of the window.

initial ID

*The **initial ID** of $\mathcal{A} = \langle \Sigma, T \rangle$ for a string $w \in \Sigma^*$ is $\langle \rtimes, w \ltimes \rangle$.*

accepting ID

*The **accepting ID** of $\mathcal{A} = \langle \Sigma, T \rangle$ is $\langle \ltimes, \varepsilon \rangle$.*

directly computes $\vdash_{\mathcal{A}}$

Definition 8 *The **directly computes** (in one step) relation of an SL_2 automaton $\mathcal{A} = \langle \Sigma, T \rangle$ is a relation $\vdash_{\mathcal{A}}$ between IDs:*

$$\langle \sigma_1, \sigma_2 u \rangle \vdash_{\mathcal{A}} \langle \sigma_2, u \rangle \overset{def}{\iff} \sigma_1\sigma_2 \in T.$$

computes $\vdash^*_{\mathcal{A}}$

*The **computes** (in zero or more steps) relation $\vdash^*_{\mathcal{A}}$ of an SL_2 automaton $\mathcal{A} = \langle \Sigma, T \rangle$ is the reflexive transitive closure of $\vdash_{\mathcal{A}}$:*

$$w \vdash^*_{\mathcal{A}} v \overset{def}{\iff} w = u \text{ or } w \vdash_{\mathcal{A}} u \text{ and } u \vdash^*_{\mathcal{A}} v \text{ for some ID } u.$$

computation on w

*A **computation of an SL_2 automaton on** w is a sequence of IDs starting with initial ID for w ($\langle \rtimes, w \ltimes \rangle$), in which the IDs are pairwise related by $\vdash_{\mathcal{A}}$.*

accepting computation

*An **accepting computation** of an SL_2 automaton $\mathcal{A}$ is one that ends with the accepting ID $\langle \ltimes, \varepsilon \rangle$.*

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \langle \rtimes, w \ltimes \rangle \vdash^*_{\mathcal{A}} \langle \ltimes, \varepsilon \rangle\}.$$

Theorem 1 (Equivalence of SL_2 grammars and automata) *A stringset L is $L(\mathcal{G})$ for some SL_2 grammar $\mathcal{G}$ iff it is $L(\mathcal{A})$ for some SL_2 grammar $\mathcal{A}$. Moreover, the equivalence is effective: there is an algorithm that, given $\mathcal{G}$ constructs $\mathcal{A}$ and* v.v..

Proof exercise. (Don't think too hard about the algorithm.)

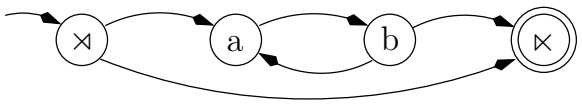

Figure 3: The Myhill graph of the automaton of Fig. 2

2.3 Myhill Graphs

Definition 9 (Myhill graph) *If $\mathcal{A} = \langle \Sigma, T \rangle$ is an SL_2 automaton, then the transition graph of $\mathcal{A}$ is a pair: $\langle V, E \rangle$ where $V = \Sigma \cup \{\rtimes, \ltimes\}$ and $E = T$.*
Myhill graph *This is known as the Myhill graph of $\mathcal{A}$.*

path **Lemma 1** *There is a **path** $\langle \rtimes, v_1, v_2, \ldots, v_n, \ltimes \rangle$ from $\rtimes$ to $\ltimes$ in the Myhill graph of $\mathcal{A}$ iff $w \in L(\mathcal{A})$.*

Proof in class.

2.4 Defining SL_2 Automata

Nubian, Dongolese
`http://phonology.cogsci.udel.edu/dbs/stress/language.php?id=266`.

Primary stress In words of all sizes, primary stress falls on the right-most non-final heavy syllable, else on the initial syllable.

Secondary stress In words of all sizes, secondary stress falls on all heavy syllables.

Factored:

1. No non-initial $\acute{L}$
2. No final $\acute{H}$
3. No $\acute{L}$ if there is any non-final $\overset{*}{H}$. (indeterminate stress)
4. No $\acute{\sigma}$ followed by any non-final $\overset{*}{H}$.
5. Some $\acute{\sigma}$
6. No more than one $\acute{\sigma}$
7. No H. (unstressed)

u_1	σ	v_1	$\in L$
u_2	σ	v_2	$\in L$
u_1	σ	v_2	$\in L$

Figure 4: 2-Suffix Substitution Closure

2.5 Abstract characterization of $\mathbf{SL_2}$

2-SSC

Lemma 2 (2-Suffix Substitution Closure (SSC)) *If $L \in SL_2$ then for all strings u_1, v_1, u_2, and v_2 in Σ^* and all symbols σ in Σ:*

$$u_1\sigma v_1 \in L \text{ and } u_2\sigma v_2 \in L \Rightarrow u_1\sigma v_2 \in L.$$

Proof is in two steps:

Lemma 3 *If $L \in SL_2$ then L satisfies 2-SSC.*

Proof in class.

Lemma 4 *If L satisfies 2-SSC then $L \in SL_2$.*

Proof: Suppose that L is closed under suffix substitution. Let

$$\begin{aligned} T_L &\stackrel{\text{def}}{=} \{\mathrm{F}_2(\rtimes w \ltimes) \mid w \in L\} \\ \mathcal{G}_L &\stackrel{\text{def}}{=} \langle \Sigma, T_L \rangle. \end{aligned}$$

We claim that $L(\mathcal{G}_L) = L$.
If $L = \emptyset$ then $T_L = \emptyset$ and $L(\mathcal{G}_L) = \emptyset$ as well.
Conversely, if $L \neq \emptyset$ then $T_L \supseteq \mathrm{F}_2(\rtimes w \ltimes)$ for at least one string w and, hence, $L(\mathcal{G}_L) \neq \emptyset$.
Assume, for the remainder of the proof, that there is at least one string in each of L and $L(\mathcal{G}_L)$.

The proof that $L \subseteq L(\mathcal{G}_L)$ is almost immediate.

The other direction is more interesting. Suppose that $w \in L(\mathcal{G}_L)$. Let n represent the length of w. Then there is a derivation of the corresponding generator which is of the form:

$$\langle \rtimes\sigma_1,\ \sigma_1\sigma_2,\ \ldots,\ \sigma_{n-1}\sigma_n,\ \sigma_n \ltimes \rangle, \text{ where } \sigma_1 \cdot \sigma_2 \cdots \sigma_n = w$$

Note that n could, potentially, be 0.

Construction in Stages

We show this, by construction, in **stages**.

	w	$=$	$\sigma_1\,\sigma_2\,\cdots\,\sigma_{i-1}$	σ_i	$\sigma_{i+1}\,\cdots\,\sigma_n$	$\in L(\mathcal{G}_L)$	
Stage 0:	w_0	$=$	$\sigma_1\,\cdots v_0$			$\in L$	Since $\rtimes\,\sigma_1 \in T_L$
Stage $i+1$:	w_i	$=$	$\sigma_1\,\sigma_2\,\cdots\,\sigma_{i-1}$	σ_i	v_i	$\in L$	By invariant, Stage i
	z_i	$=$	x_i	σ_i	$\sigma_{i+1}\,y_i$	$\in L$	Since $\sigma_i\sigma_{i+1} \in T_L$
	w_{i+1}	$=$	$\sigma_1\,\sigma_2\,\cdots\,\sigma_{i-1}$	σ_i	$\sigma_{i+1}\,y_i$	$\in L$	By SSC

Figure 5: The construction of the proof of Lemma 4

(Stage 0)

If $n = 0$, then $w = \varepsilon$ and the pair $\rtimes\ltimes \in T_L$ which implies that $\rtimes\ltimes \in \mathrm{F}_2(\rtimes \cdot v \cdot \ltimes)$ for some $v \in L$. There is only one possible such v, namely ε. Hence $\varepsilon \in L(\mathcal{G}_L) \Rightarrow \varepsilon \in L$ and we are done.

Otherwise $n > 0$, $\sigma_1 \in \Sigma$ and the pair $\rtimes\sigma_1 \in T_L$. By construction of T_L there is some string $z \in L$ that starts with σ_1. Choose any one of these for w_1.

Invariant

(Invariants)

For all $0 < i \leq n$:

1. $w_i \in L$.
2. $w_i = u_i\sigma_i v_i$ where $u_i = \sigma_1 \cdot \sigma_2 \cdots \sigma_{i-1}$ (possibly ε) and $v_i \in \Sigma^*$.

(Stage $0 < i < n$)

Since $\sigma_i\sigma_{i+1} \in T_L$ there is some string in L that includes the adjacent pair $\sigma_i\sigma_{i+1}$. Choose any one of these and call it z_i. It has the form $x_i\sigma_i\sigma_{i+1}y_i$, for some strings x_i and y_i in Σ^*. Since both $w_i = u_i\sigma_i v_i$ and $z_i = x_i\sigma_i\sigma_{i+1}y_i$ are in L and L is closed under substitution of suffixes, $u_i\sigma_i\sigma_{i+1}y_i$ is also in L (substituting $\sigma_{i+1}y_i$ for v_i in w_i). Let $w_{i+1} = u_i\sigma_i\sigma_{i+1}y_i$.

(Stage n)

By the invariants, $w_n = u_n\sigma_n v_n$, where $u_n = \sigma_1 \cdot \sigma_2 \cdots \sigma_{n-1}$ (possibly ε). Since the pair $\sigma_n\ltimes$ is in T_L, by construction of T_L there must be some string in L which ends with σ_n, i.e., that has the form $x_n\sigma_n$, equivalently, $x_n\sigma_n\varepsilon$. Since both $u_n\sigma_n v_n$ and $x_n\sigma_n\varepsilon$ are in L and L is closed under suffix substitution, $u_n\sigma_n\varepsilon$ is also in L. (Here v_2 of Lemma 2 is ε.) Thus $u_n\sigma_n\varepsilon = \sigma_1 \cdot \sigma_2 \cdots \sigma_n = w \in L$ and we are done.

We have shown, then, that $L = \emptyset$ iff $L(\mathcal{G}_L) = \emptyset$ as well, that if $w \in L$ then $w \in L(\mathcal{G}_L)$ and that if $w \in L(\mathcal{G}_L)$ then $w \in L$. Consequently, $L(\mathcal{G}_L) = L$

u_1	σ	v_1	$\in L$
u_2	σ	v_2	$\in L$
u_1	σ	v_2	$\notin L$

Figure 6: A counterexample to 2-Suffix Substitution Closure

and, given any stringset L that satisfies Suffix Substitution Closure we can effectively construct a strictly 2-local automaton that recognizes it.

⊣

2.6 Non-SL_2 stringsets

2.7 Closure properties of the class of SL_2 stringsets

Lemma 5 *The class of strictly 2-local stringsets is effectively closed under intersection.*

Proof in class.

Lemma 6 *The class of strictly 2-local stringsets is* not *closed under union.*

Proof in class.

Lemma 7 *The class of strictly 2-local stringsets is* not *closed under complement or relative complement.*

Proof exercise.

Lemma 8 *The class of strictly 2-local stringsets is* not *closed under concatenation.*

Proof exercise.

Lemma 9 *The class of strictly 2-local stringsets is closed under Kleene- and positive closure.*

N.B. This will not be the case for other classes of strictly local stringsets or for any other class of local stringsets.
Proof exercise.

Lemma 10 *The class of strictly 2-local stringsets is closed under reversal.*

Proof exercise.

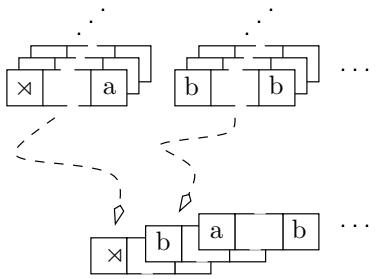

Figure 7: A strictly k-local generator.

Lemma 11 *The class of strictly 2-local stringsets (is/is not?) closed under alphabetic homomorphism.*

Proof (and polarity) exercise.

Lemma 12 *The class of strictly 2-local stringsets is not closed under general homomorphism.*

Proof exercise.

Lemma 13 *The class of strictly 2-local stringsets is not closed under substitution.*

Proof exercise.

3 Strictly k-Local Stringsets

Strictly k-Local grammar

Definition 10 (Strictly k-Local grammar) *A $\mathcal{G}$ is a pair $\langle \Sigma, T \rangle$:*

$$\begin{array}{lll} \Sigma & & \text{—an alphabet} \\ T & \subseteq F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\}) & \text{—a set of } k\text{-Factors over } \Sigma \cup \{\rtimes, \ltimes\} \end{array}$$

Definition 11 (Strictly k-Local automaton) *So is a SL_k automaton.*

Definition 12 (Derived set of an $\mathbf{SL}_k$ grammar) *The set of strings derived by a strictly k-local grammar $\mathcal{G}$ is*

$$L(\mathcal{G}) \stackrel{def}{=} \{w \in \Sigma^* \mid F_k(\rtimes \cdot w \cdot \ltimes) \subseteq T\}.$$

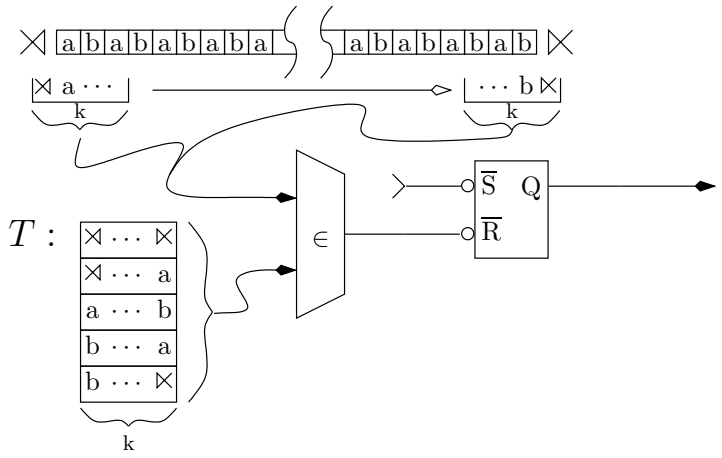

Figure 8: Strictly k-local automata.

Definition 13 (Derivation of an $\mathbf{SL}_k$ grammar) *A* derivation *of a SL_k is a sequence of k-factors from T, starting with a factor of form $\rtimes u$ and ending with a factor of form $v\ltimes$.*

The yield *of a derivation*

$$\langle \rtimes u_1,\, u_1\sigma,\, \ldots\, ,\sigma' v,\, v\ltimes \rangle$$

is

$$\rtimes u_1 \overset{(k-1)}{\circ} u_1\sigma \overset{(k-1)}{\circ} \cdots \overset{(k-1)}{\circ} \sigma' v \overset{(k-1)}{\circ} v\ltimes$$

where

$$w\sigma_1 \cdots \sigma_{k-1} \overset{(k-1)}{\circ} \sigma_1 \cdots \sigma_{k-1}\sigma_k \overset{def}{=} w\sigma_1 \cdots \sigma_{k-1}\sigma_k$$

and is undefined otherwise.

SL_k **Definition 14 ($\mathbf{SL}_k$)** *The class of **Strictly k-Local stringsets** is the class of stringsets that can be generated by strictly k-local grammars.*

$$SL_k \overset{def}{=} \{L \mid (\exists \mathcal{G})[L = L(\mathcal{G})],\ \mathcal{G} \text{ strictly } k\text{-local}\}$$

Definition 15 (Instantaneous Descriptions of $\mathbf{SL}_k$ Automata) *An* Instantaneous Description (ID) *of a strictly k-local automaton $\mathcal{A} = \langle \Sigma, T\rangle$ is a pair:*

$$\langle t_i, w_i\rangle \in F_{<k}(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\}) \times (\Sigma^* \cdot \{\ltimes\})$$

In the ID $\langle t_i, \sigma_i v_i \rangle$, $t_i\sigma_i$ is the sequence of symbols currently in the window and v_i is the suffix of the input that is on the tape to the right of the window.

Definition 16 ($\mathbf{SL}_k$ Computes Relation) *An ID $\langle \sigma_i \cdot \sigma_{i+1} \cdots \sigma_{i+k-1}, w_i \rangle$* Directly Computes *an ID $\langle \sigma_{i+1} \cdots \sigma_{i+k-1} \cdot \sigma_{i+k}, w_j \rangle$ in $\mathcal{A} = \langle \Sigma, T \rangle$:*

$$\langle \sigma_1\sigma_2 \cdots \sigma_{k-1}, \sigma_k w \rangle \vdash_{\mathcal{A}} \langle \sigma_2 \cdots \sigma_{k-1} \cdot \sigma_k, w \rangle,$$

iff, by definition, $\sigma_1 \cdot \sigma_2 \cdots \sigma_k \in T$.

The computes *(in zero or more steps) relation $\vdash^*_{\mathcal{A}}$ of an SL_k automaton $\mathcal{A} = \langle \Sigma, T \rangle$ is the reflexive transitive closure of $\vdash_{\mathcal{A}}$.*

Definition 17 ($\mathbf{SL}_k$ Computations) *The* computation *of an SL_k automaton $\mathcal{A} = \langle \Sigma, T \rangle$ on a string w is the maximal sequence of IDs in which each sequential pair of IDs is related by $\vdash_{\mathcal{A}}$ and which starts with the* initial configuration *of $\mathcal{A}$ on w: $\langle t_1, v_1 \rangle$ where $t_1 = \rtimes w_1$, $v_1 = w_2 \ltimes$, $w_1 \cdot w_2 = w$ and either $|\rtimes w_1| = k - 1$ or $v_1 = \ltimes$.*

A computation is accepting *iff its final ID is of the form $\langle w_i \ltimes, \varepsilon \rangle$.*

An automaton accepts a string w iff the computation of $\mathcal{A}$ on w is accepting.

Definition 18 *A stringset L is* recognized *by a strictly k-local automaton $\mathcal{A}$ iff $L = L(\mathcal{A})$ where*

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \langle \rtimes w_1, w_2 \ltimes \rangle \vdash^*_{\mathcal{A}} \langle v_i, \varepsilon \rangle\}.$$

3.1 k-Local Myhill Graphs

edge-labeled graph

Definition 19 (Edge-labeled graph) *An is a graph $\langle V, E \rangle$ where $E \subseteq V \times \Gamma \times V$ for some labeling alphabet Γ.*

k-local Myhill graph

Definition 20 (k-local Myhill graph) *A k-local Myhill graph is an edge-labeled graph where*

$$\begin{array}{rcl} V & = & (\{\rtimes\} \cdot \Sigma^{<k-1} \cup F_{k-1}(\Sigma^*) \cup \{\ltimes\} \\ E & \subseteq & V \times (\Sigma \cup \varepsilon) \times V \end{array}$$

where

$$\begin{array}{rcll} E & = & \{\langle \rtimes v_1, \sigma, v_2 \rangle \mid |\rtimes v_1| < k-1 \text{ and } v_2 = v_1\sigma\} \cup & \textit{(prefixes)} \\ & & \{\langle v_1, \sigma, v_2 \rangle \mid |v_1| = k-1,\ v_1 = \sigma_i w_i, \text{ and } v_2 = w_i\sigma\} \cup & \textit{(k-factors)} \\ & & \{\langle v_1, \varepsilon, \ltimes \rangle\} & \textit{(accepting state)} \end{array}$$

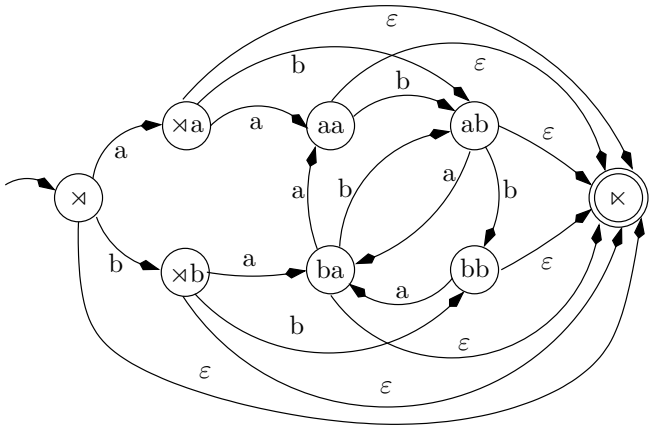

Figure 9: A 3-local Myhill graph.

u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_1	$\in L$
u_2	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\in L$
u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\in L$

Figure 10: k-Suffix Substitution Closure

Lemma 14 *If $\mathcal{A}$ is a strictly k-local automaton, then a string is in $L(\mathcal{A})$ iff it is the concatenation of the strings labeling the edges of a path from '$\rtimes$' to '$\ltimes$' in the k-local Myhill graph of $\mathcal{A}$.*

3.2 k-Local Suffix Substitution Closure

Lemma 15 (k-Local Suffix Substitution Closure) *If L is a strictly k-local stringset then for all strings u_1, v_1, u_2, and v_2 in Σ^* and all strings x in Σ^{k-1}:*

$$u_1 x v_1 \in L \text{ and } u_2 x v_2 \in L \Rightarrow u_1 x v_2 \in L.$$

Theorem 2 *A stringset is in SL_k iff it satisfies the k-Local Suffix Substitution Closure Property.*

3.3 $\mathbf{SL}_k$ *v.s* $\mathbf{SL}_l$, $k \neq l$

Theorem 3 (The SL Hierarchy)

$$SL_2 \subsetneq SL_3 \subsetneq \cdots SL_k \subsetneq SL_{k+1} \cdots, \; k \geq 2.$$

	$\overbrace{\qquad}^{k-1}$		
u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_1	$\in L$
u_2	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\in L$
u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\notin L$

Figure 11: A counterexample to k-Suffix Substitution Closure

and, in general,

$$SL_k \subsetneq SL_{k+i},\ k \geq 2,\ i \geq 1.$$

Proof exercise.

4 Strictly Local Stringsets

Definition 21 (Strictly Local Stringsets) *A stringset L is* strictly local *($L \in SL$) iff it is strictly k-local for some k.*

Theorem 4 ((General) Suffix Substitution Closure) *A stringset L is strictly local iff there is some k such that, for all strings u_1, v_1, u_2, and v_2 in Σ^* and all strings x in Σ^{k-1}:*

$$u_1xv_1 \in L \text{ and } u_2xv_2 \in L \Rightarrow u_1xv_2 \in L.$$

4.1 Non-SL Stringsets (Adversary Arguments)

k-SSC:

$$\begin{array}{l}(\forall L \subseteq \Sigma^*)[\quad L \in \text{SL} \Rightarrow \\ \qquad (\exists k)[\\ \qquad\qquad (\forall u_1, v_1, u_2, v_2 \in \Sigma^*, x \in \Sigma^{k-1}[\\ \qquad\qquad\quad u_1xv_1 \in L \wedge u_2xv_2 \in L \Rightarrow u_1xv_2 \in L\ ] \\ \qquad\qquad] \\ \qquad]\end{array}$$

Equivalently

$$\begin{array}{l}\neg(\exists L \subseteq \Sigma^*)[\quad L \in \text{SL} \wedge \\ \qquad (\forall k)[\\ \qquad\qquad (\exists u_1, v_1, u_2, v_2 \in \Sigma^*, x \in \Sigma^{k-1}[\\ \qquad\qquad\quad u_1xv_1 \in L \wedge u_2xv_2 \in L \wedge u_1xv_2 \notin L\ ] \\ \qquad\qquad] \\ \qquad]\end{array}$$

For any given k	$\overbrace{\quad\quad}^{k-1}$		
u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_1	$\in L$
u_2	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\in L$
u_1	$\sigma_1 \cdots \sigma_{k-1}$	v_2	$\notin L$

Figure 12: A counterexample to (general) Suffix Substitution Closure

$\exists$—your choice
$\forall$—your adversaries choice

- You choose L the stringset you intend to prove is not SL.
- Your adversary, claiming that there is a k-local automaton that recognizes it, chooses k. Presumably, their choice of k will depend on your choice of L. Not even *this* adversary is going to claim that *all* stringsets are SL.
- You now choose two strings u_1xv_1 and u_2xv_2. Again, your choice should depend on the specific value of k your adversary chose (as well, of course, as the L you chose to start with).
- You win iff the two strings you chose witness that the stringset does not satisfy the theorem, i.e., iff
  - u_1xv_1 and u_2xv_2 are both in L and
  - u_1xv_2 is not in L.

Proof is a *strategy* for winning game no matter what choices your adversary makes.

4.2 Non-SL stringsets

4.3 Operations on Strictly Local Stringsets

The only complication with these is the fact that you cannot assume anything about the 'k's. So to prove closure you have to show that if $L_1 \in \mathrm{SL}_k$ and $L_2 \in \mathrm{SL}_l$ then there is some m for which the result is SL_m. To prove non-closure you have to show that there is no such m.

Lemma 16 *The class of strictly local stringsets is effectively closed under intersection.*

Proof exercise.

Lemma 17 *The class of strictly local stringsets is* not *closed under complement or relative complement.*

Proof exercise.

Lemma 18 *The class of strictly stringsets is* not *closed under concatenation.*

Proof exercise.

Lemma 19 *The class of strictly local stringsets is not closed under Kleene or positive closure.*

Proof exercise.

Lemma 20 *The class of strictly stringsets is closed under reversal.*

Proof exercise.

Lemma 21 *The class of strictly stringsets (is/is not?) closed under alphabetic homomorphism.*

Proof (and polarity) exercise.

Lemma 22 *The class of strictly stringsets is not closed under general homomorphism.*

Proof exercise.

Lemma 23 *The class of strictly stringsets is not closed under substitution.*

Proof exercise.

4.4 Finite Languages and Strictly Local Languages

Theorem 5 *The class of finite languages is a proper subclass of SL.*

Lemma 24 *If $L \in Fin$ then there is some k for which $L \in SL_k$.*

$$L \in Fin \Rightarrow (\exists k)[L \in SL_k].$$

Proof exercise.

Lemma 25 *There are stringsets in SL which are not in Fin.*

$$SL - Fin \neq \emptyset.$$

Proof exercise.

Theorem 6 *There is no k for which the Fin is a subclass of SL_k.*

$$(\forall k)[Fin - SL_k \neq \emptyset].$$

Proof exercise.

5 Learning SL stringsets

range **Definition 22** *The **range** of a function $f : A \to B$, denoted $ran(f)$, is:*

$$ran(f) \stackrel{def}{=} \{f(x) \mid x \in A\}.$$

text **Definition 23** *A **text** for a stringset L is a function $\phi_L : \mathbb{N} \to (\Sigma^* \cup \{\#\})$ such that $ran(\phi_L) - \{\#\} = L$*

Informally, a text for L is a sequence of strings $\langle w_0, w_1, \ldots \rangle$ (where $w_i = \phi_L(i)$). Since ϕ_L is defined for all $i \in \mathbb{N}$ the sequence is infinite. The sequence is a presentation of the strings of L, one at a time, in any order, possibly with pauses (#).

Since $\text{ran}(\phi_L) - \{\#\} = L$, for each string $\sigma \in L$ there must be at least one $i \in \mathbb{N}$ for which $\phi_L(i) = \sigma$. (There may be many.) In other words, even though L may be infinite, each string in L must show up eventually, i.e., after finitely many strings have been presented. Note, though, that if L is infinite there is no $i \in \mathbb{N}$ for which all strings in L have been presented.

Since strings may appear many times in the text, the pause ('#') is only actually needed for the case in which $L = \emptyset$. We will generally ignore this case. We can take the set of all texts for L to be sequences of just strings from L plus the text for the empty stringset: $\langle \#, \#, \#, \ldots \rangle$. Note that this is the *only* possible text for $\emptyset$.

Note that a text (of this sort) presents only *positive* data. It says nothing about the strings that are not in the stringset. Absolutely nothing: it is impossible to determine at any point $i \in \mathbb{N}$ whether a string w is not in L or it just has not shown up yet.

learnable in the limit

Definition 24 *A class of stringsets $\mathbb{L}$ is **learnable in the limit** from positive data if there is an algorithm $\mathcal{I}$ which given any text $\langle w_0, w_1, \ldots \rangle$ for some $L \in \mathbb{L}$ outputs a sequence of grammars (or automata, etc.) $\langle \mathcal{I}(0), \mathcal{I}(1), \ldots \rangle$ such that:*

converges

1. *$\mathcal{I}$ **converges**: there is some $i \in \mathbb{N}$ such that $\mathcal{I}(j) = \mathcal{I}(i)$ for all $j \geq i$.*
2. *$\mathcal{I}$ is correct: Once $\mathcal{I}$ has converged to some grammar $\mathcal{I}(i)$, $L(\mathcal{I}(i)) = L$.*

Note that, even though $\mathcal{I}$ is required to converge, in general there will be no way to tell whether it has or not, just as there is no way to tell whether some string w will never show up in the text.

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

Lemma 26 *Any class of stringsets that includes Fin and at least one infinite stringset is not identifiable in the limit from positive data.*

superfinite

Such classes of stringsets are called **superfinite**. The converse is not true. There are classes of stringsets that are not superfinite that are, nevertheless, not identifiable in the limit from positive data.

Theorem 8 *For any fixed k, SL_k is identifiable in the limit from positive data.*

Theorem 9 *The class SL is not identifiable in the limit from positive data.*

6 Cognitive complexity of SL

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) SL_k stringset must be sensitive, at least, to the length k blocks of consecutive events that occur in the presentation of the string.
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to the immediately prior sequence of $k-1$ events.
- Any cognitive mechanism that is not sensitive to the length k blocks of consecutive events that occur in the presentation of the string will be unable to recognize some SL_k stringsets.

7 Exercises

7.1 Share these

Decide among yourselves who will do each, but make certain each one is done by someone. Also, make certain that each of you eventually does some of these. So, for example, those who did not do the optional exercises in the Foundations module ought to take on one or more of these.

If there are any you are feeling uncertain of, it would be a good idea to take one of them on. If you are feeling particularly confident about any, it would be a good idea to allow someone else to take it on and to offer to help if they get stuck.

1. Prove Theorem 1. (Pg. 5.)
   [**Hint:** To show that a string $w \in L(\mathcal{G}) \Leftrightarrow w \in L(\mathcal{A})$ it suffices to show the contrapositive: $w \notin L(\mathcal{G}) \Leftrightarrow w \notin L(\mathcal{A})$.]

2. Prove Lemmas 7 through 13. (Pp. 9–10.)
   [**Hint:** Some of these are immediate consequences of others. For Lemma 9, work from Myhill graphs.]

3. Prove Theorem 3. (Pg. 13.)
   [**Hint:** Work from SSC.]

4. Prove Lemmas 16 through 23. (Pp. 15–16.)

5. Prove Theorem 5. (Pg. 16.)
   [**Hint:** Prove the lemmas.]

6. Prove Theorem 6. (Pg. 17.)

7.2 Everyone

Again, you should discuss these among yourselves, figure out how they are going to work out and how the proof should go. But each of you should write up the answers on your own.

The same suggestions about roles that I made in the shared exercises apply here. If you are feeling uncertain, you should try working it out on your own. If you are particularly confident, you should let others chew on it awhile before showing the way.

1. Which of properties 6 and 7 of the Nubian example are SL. If either is, what is k? If either is not, prove it.

2. Which of the following primitive stress constraints are SL? For those that are, what is k? For those that are not, prove it.

   (a) No light syllable immediately precedes a heavy syllable with primary stress.

   (b) An initial light syllable may get primary stress only if the word is mono-syllabic.

   (c) The antepenultimate syllable is never heavy.

   (d) If the penult is heavy the antepenult never receives primary stress.

   (e) There is never more than one syllable with primary stress.

   [**Hint:** Forgive my bad impression of a phonologist.]

3. Which of the following stress patterns are SL. For those that are, what is k? For those that are not, prove it.

   (a) Cambodian

   (b) Cayuvava

   (c) Hopi

   (d) Icua Tupi

   (e) Maranungku

   (f) Murik

   [**Hint:** For our purposes `http://phonology.cogsci.udel.edu/dbs/stress/search.php` is authoritative. For those of you who have not studied enough Phonology yet to make sense of the constraints (and some of them may be inconsistent or incomplete unless one has the background) I suggest that you do what I do: use a phonologist as your informant.]