Formal Description of Syntax
ESSLLI'07

James Rogers
Earlham College
Indiana, USA
jrogers@cs.earlham.edu

1 Introduction

Syntax, in its most superficial aspect, is the study of patterns in linear sequences of events: sounds in a utterance, words on a page, etc. Patterns of this sort do not need to be very complicated before the combinatorial possibilities of their interactions become overwhelming. Formal Language Theory, the mathematical study of of patterns such as these, provides a means of abstracting away from the complexities of these interactions in order to reason clearly about their consequences. Formalization of assertions about the structure of utterances helps to make the theory that incorporates those assertions more explicit in that it reduces ambiguity, exposes unmotivated assumptions and supports a well-defined method of inference by which the consequences of the assertions can be explored. In Pullum, Rogers and Scholz (in preparation) these dimensions of explicitness are referred to as **univocality**, **overtness** and **projectability**.

But beyond the potential benefits of formalization in explicating a theory of syntax, the ways of reasoning about patterns that are employed in Formal Language Theory, because of the way the abstraction they employ tames combinatorial complexity, provide a means of thinking clearly about these components of a theory, even if, in the end, the theory is not to be fully formalized.

This course is an introduction to FLT for aspiring syntacticians. Its goal is to introduce these ways of reasoning in a way that directly serves the kinds of analysis one encounters in studying the syntax of human languages. It differs from traditional approaches to the material in that it does not take computational processes (formal grammars and automata) to be primary. Rather it focuses on mathematically well-defined descriptive mechanisms (formal logics) and brings the computational processes into play as a way of reasoning about the sets of structures definable by these means.

The study of definability by logical means is the domain of Model Theory and over the last thirty years or so Finite Model Theory, the model theory of finite structures, has proven to be a strikingly effective tool for reasoning about computational processes (an area known as Descriptive Complexity). We, in a sense, take the opposite perspective, starting with descriptions of sets of finite structures and employing the results of Finite Model Theory and Descriptive Complexity Theory to characterize the abstract properties of those sets.

We should be clear about the objects we are studying. While Syntax, as a branch of Linguistics, is a study of human languages, we will not be studying those languages themselves but, rather, modeling the physically realized form of utterances as sequences of symbols: strings over some finite alphabet. Thus we will be modeling fragments of languages as sets of strings. The idea is that the regularities of syntax will show up as patterns in the sets of strings and *v.v.*, at least for the fragment that the set represents. But it is important to be clear about the distinction between the ultimate object of study—the language itself—and the objects we use to model them—the sets of strings. From this perspective, the patterns in the sets of strings serve as a means of representing the regularities that the theory of syntax is based upon. Beyond that, we are not interested here in any particular theory of syntax or in particular sets of strings. Rather, our focus is the metatheory of this modeling process. We are interested in what we can understand about the sets we define by examining the methods we use to define them as mathematical processes in their own right.

In order to preserve the distinction between the sets of strings we are working with and the human languages they model, we will refer to these as **stringsets** rather than the more usual **formal languages**. This is convenient, as well, since the classes of descriptions that we will be studying are, themselves, formal languages. We will reserve the term **language**, in the sequel, for these logical languages.

We will start out with exceedingly simple, arguably nearly minimal, logical languages, languages that provide descriptive means so limited that no one would seriously advocate them as means for describing the syntax of human languages. But by starting at this nearly trivial level we will be able to introduce the tools and techniques of Formal Language Theory (and Finite Model Theory) in very simple forms. As we extend the descriptive power of our languages, the complexity of the techniques will increase but, by increasing the power incrementally we will be able to introduce this increased complexity incrementally as well.

One of the conclusions we will be able to draw from this approach is that patterns in strings do not have to get very complicated before one needs to introduce structure beyond the inherent structure of the strings in order to account for them. This is not at all unwelcome from a linguistic perspective, since Syntax is generally interested in analyzing strings in terms of such inferred structure.

The need for additional structure introduces a second theme of this course. Initially, working directly with strings, we will increase our descriptive power by varying the logical mechanisms we employ. But we will then move to increasing that power by increasing the complexity of the structures we reason about, employing the same range of logical mechanisms over classes of progressively more complex structures. One of the theorems we will encounter will show that, while the complexity of the definable sets of these more complicated structures will vary as we vary the logical mechanisms in much the same way that the complexity varies when defining sets of strings, the complexity of the patterns that show up in the stringsets these structures yield will remain constant. Once we introduce structure beyond the inherent structure of the strings, variations in logical mechanisms are not reflected in the stringsets, rather they are of "theory internal" significance, showing up as variations in the complexity of the regularities exhibited by the sets of structures we employ in analyzing the strings. Variations in the complexity of the stringsets themselves will be a consequence of variations in the complexity of the structures we use to analyze them.

This reader is not intended to stand alone, being little more than an enumeration of definitions, examples and theorems more or less in the order we will encounter them. Motivation, explanation and additional examples will be provided in the lectures.

1.1 Methodology

- Capture patterns in strings with logical formulae
  - Define in terms of relationships within string
- Define stringsets as set of models of those formulae
  - "Grammars" are sets of axioms
- Model Theory
  - Fundamental questions have to do with what we can determine about the nature of a stringset based on the logical machinery needed to define it.
- Descriptive Complexity
  - Characterizations of classes of definable sets in terms of algorithmic processes and *v.v.*
  - Structure of the definable sets and structure of class of definable sets
  - Limits of definability

2 Basic Concepts

2.1 Strings as sequences (An Inductive Definition)

An **Alphabet** is any non-empty finite set of symbols:

$$\text{e.g., } \Sigma = \{a, b, c\}.$$

Definition 1 (Σ^*: Strings over an alphabet Σ)
(An **Inductive Definition**.)
*Given any alphabet Σ the set of all **Strings over** Σ is the smallest set Σ^* such that:*

- *The empty sequence is a string over Σ:* $\varepsilon \in \Sigma^*$.
- *If v is a string over Σ, σ is a symbol in Σ and $w = v\sigma$, then w is a string over Σ:* $v \in \Sigma^*$, $\sigma \in \Sigma$ *and* $w = v\sigma$ *implies* $w \in \Sigma^*$.

2.2 A Recursive Definition

Definition 2 (Length of a string) (A **Recursive Definition**.)
For all $w \in \Sigma^*$:

$$|w| = \begin{cases} 0 & \text{if } w = \varepsilon, \\ |v| + 1 & \text{if } w = v\sigma. \end{cases}$$

2.3 Proof by Structural Induction

Lemma 3 *For any alphabet Σ and all $w \in \Sigma^*$, the length of Σ is finite:*

$$w \in \Sigma^* \Rightarrow |w| \in \mathbb{N}.$$

Proof(by **Structural Induction**) By Definition 1, either $w = \varepsilon$ or $w = v \cdot \sigma$ for some simpler string $v \in \Sigma^*$.

(Basis:) Suppose $w = \varepsilon$. Then, by Definition 2, $|w| = 0 \in \mathbb{N}$.

(Induction:) Suppose $w = v\sigma$ and that all strings in Σ^* that are strictly simpler (in the sense of Definition 1) than w have finite length. Then $|v| \in \mathbb{N}$ (by hypothesis) and $|w| = |v| + 1$ (by Definition 2) $\in \mathbb{N}$. ⊣

Definition 4 (Concatenation of strings) *For all $w, v \in \Sigma^*$:*

$$u \cdot w = \begin{cases} u & \text{if } w = \varepsilon, \\ (u \cdot v)\sigma & \text{if } w = v\sigma. \end{cases}$$

Lemma 5 (Identity for concatenation) *The empty string is both a left and right identity element for concatenation:*

$$w \cdot \varepsilon = w = \varepsilon \cdot w.$$

Proof Exercise. ⊣

Lemma 6 *Concatenation of strings is associative*

$$(u \cdot v) \cdot w = u \cdot (v \cdot w).$$

Proof Exercise. ⊣

2.4 Concatenation and iteration of stringsets

Definition 7

$$L_1 \cdot L_2 \overset{def}{=} \{w \cdot v \mid w \in L_1,\ v \in L_2\}$$

Definition 8 (Iteration of a Stringset) *If L is a stringset and $i \in \mathbb{N}$ then:*

$$L^i = \begin{cases} \{\varepsilon\} & \text{if } i = 0, \\ L^j \cdot L & \text{if } i = j + 1. \end{cases}$$

2.5 Kleene and positive closure

Definition 9 (Kleene Closure) *If L is a stringset its **Kleene closure** (or **iteration closure**) L^* is:*

$$L^* = \bigcup_{i \geq 0} [L^i].$$

Definition 10 *L^+ is the **positive closure** of L:*

$$L^+ = \bigcup_{i \geq 1} [L^i].$$

(Exercise) Show that, in general, $L^+ \neq L^* - L^0$.

2.6 Finite stringsets

Definition 11 (Fin**)** *The class of* Finite Stringsets *(*Fin*) over an alphabet* Σ *is the smallest set such that:*

- $\emptyset$ *is a finite stringset (i.e.,* $\emptyset \in \mathrm{Fin}$*)*
- $\{\varepsilon\}$ *is a finite stringset (*$\{\varepsilon\} \in \mathrm{Fin}$*)*
- *If* $\sigma \in \Sigma$ *then* $\{\sigma\}$ *is a finite stringset (*$\sigma \in \Sigma \Rightarrow \{\sigma\} \in \mathrm{Fin}$*)*
- *If* L_1 *and* L_2 *are finite stringsets (*$L_1, L_2 \in \mathrm{Fin}$*) then:*
  - $L_1 \cdot L_2$ *is a finite stringset (*$L_1 \cdot L_2 \in \mathrm{Fin}$*) and*
  - $L_1 \cup L_2$ *is a finite stringset (*$L_1 \cup L_2 \in \mathrm{Fin}$*)*

(Exercise) Prove that Fin is actually the set of all finite stringsets, i.e., if **card**$(L) = n \in \mathbb{N}$ then $L \in \mathrm{Fin}$ and *v.v.* (Start by proving that if $w \in \Sigma^*$ then $\{w\} \in \mathrm{Fin}$.)

2.7 Relational structures

Definition 12 (Relational Signature) *A* ***relational signature*** $\mathcal{R}$ *is a finite set of* ***predicate symbols*** *divided into a sequence of disjoint subsets* $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdots$. *If* $\rho \in \mathcal{R}_n$ *then the* ***arity*** *of* ρ *is* n.

Definition 13 (*n*-ary relation) *An* n***-ary relation*** *over sets* $S_1, S_2, \ldots, S_n$ *is a set of* n***-tuples****:*

$$R \subseteq S_1 \times S_2 \times \cdots \times S_n = \{\langle x_1, x_2, \ldots, x_n\rangle \mid x_i \in S_i\}.$$

$$S^n \overset{def}{=} \overbrace{S \times S \times \cdots \times S}^{n}$$

Definition 14 (Relational Model)
A ***relational model*** *over a relational signature* $\mathcal{R}$ *is a tuple* $\mathcal{A} = \langle A, \rho_1^{\mathcal{A}}, \rho_2^{\mathcal{A}}, \ldots\rangle$, *where* A *is the domain of* $\mathcal{A}$ *and there is a* $\rho_i^{\mathcal{A}}$ *for each* $\rho_i \in \mathcal{R}$ *and, if* $\rho_i \in \mathcal{R}_n$ *then* $\rho_i^{\mathcal{A}} \subseteq A^n$.

2.8 Strings as relational structures

Definition 15 (String Models) *A* ***(***$\triangleleft$***) String Model*** *(a* ***Successor String Model****) over an alphabet* Σ *is a tuple*

$$\mathcal{W} = \langle W, \triangleleft, P_\sigma\rangle_{\sigma \in \Sigma}$$

in which the domain W *is a finite set which is totally ordered by the transitive closure of* $\triangleleft$.
A ***(***$\triangleleft^+$***) String Model*** *(a* ***Precedence String Model****) over an alphabet* Σ *is a tuple*

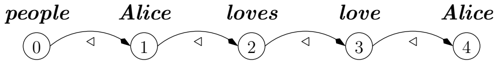
$$\mathcal{W} = \langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma \in \Sigma}$$

in which the domain W *is a finite set which is totally ordered by* $\triangleleft^+$ *and in which* $\triangleleft^+$ *is the transitive closure of* $\triangleleft$.

So the signature of a ($\triangleleft^+$) string model is

$$\{P_\sigma \mid \sigma \in \Sigma\}\,(= \mathcal{R}_1) \cup \{\triangleleft, \triangleleft^+\}\,(= \mathcal{R}_2).$$

2.9 Example



$$\begin{aligned}
\textit{people Alice loves love Alice} &= \langle W, \triangleleft, P_{\textbf{\textit{people}}}, P_{\textbf{\textit{Alice}}}, P_{\textbf{\textit{loves}}}, P_{\textbf{\textit{love}}} \rangle \\
W &= \{0, 1, 2, 3, 4\} \\
\triangleleft &= \{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,3\rangle, \langle 3,4\rangle\} \\
P_{\textbf{\textit{people}}} &= \{0\} \\
P_{\textbf{\textit{Alice}}} &= \{1, 4\} \\
P_{\textbf{\textit{loves}}} &= \{2\} \\
P_{\textbf{\textit{love}}} &= \{3\}
\end{aligned}$$

2.10 Isomorphism

Definition 16 (Isomorphism) *Two models $\mathcal{A}$ and $\mathcal{B}$ are isomorphic, $\mathcal{A} \cong \mathcal{B}$ (with respect to a relational signature $\mathcal{R}$), if there is a one-to-one and onto map (a bijection) associating points in the domain of one model with points in the domain of the other that respects the relations of the signature in the sense that a tuple of points are related by the interpretation of $\rho \in \mathcal{R}$ in one iff their images under the map are related by the interpretation of ρ in the other.*

Models that are isomorphic with respect to the signature $\mathcal{R}$ cannot be distinguished by any property that depends only on the relationships denoted by the predicates in $\mathcal{R}$.

2.11 Canonical string models

Observation 17 *All string models are isomorphic to some string model in which W is an **initial segment** of $\mathbb{N}$ (i.e., $W = \{0, 1, \ldots, n-1\}$) and $\triangleleft$ and $\triangleleft^+$ are the natural successor and less-than relations on $\mathbb{N}$. We will take this to be the **canonical** model of the string.*

Observation 18 *If $\mathcal{W}$ is a string model over an alphabet Σ and $\Sigma \subseteq \Gamma$, then $\mathcal{W}$ can be extended to a string model over Γ by adjoining empty ($\emptyset$) interpretations of the symbols in $\Gamma - \Sigma$.*

2.12 The empty string

Observation 19 *The empty string, as a relational model, has an empty domain ($W = \emptyset$) and, consequently, the interpretations of the predicates of its signature are all empty as well: $\mathcal{W} = \langle \emptyset, \emptyset, \ldots, \emptyset \rangle$. While there is a distinct model for the empty string for each distinct alphabet, these differ only in the number and names of the relations they contain. Since the interpretation of all predicates is determined, in the spirit of Observation 18 we can take the canonical model of the empty string to be simply $\langle \emptyset \rangle$, extending it for whatever signature is required.*

2.13

Definition 20 (Concatenation of String Models) *Given two strings*

$$w = \langle W, \triangleleft^w, \triangleleft^{+^w}, P^w_\sigma\rangle_{\sigma\in\Sigma} \qquad \textit{and} \qquad v = \langle V, \triangleleft^v, \triangleleft^{+^v}, P^v_\sigma\rangle_{\sigma\in\Sigma}$$

$$w \cdot v \overset{def}{=} \begin{cases} w & \textit{if } v = \varepsilon, \\ v & \textit{if } w = \varepsilon, \\ \langle \quad W \uplus V,\ \triangleleft^w \uplus \triangleleft^v \cup \{\langle \max^w, \min^v\rangle\}, & \\ \qquad \triangleleft^{+^w} \uplus \triangleleft^{+^v} \cup (W \times V),\ P^w_\sigma \uplus P^v_\sigma \quad \rangle_{\sigma\in\Sigma} & \\ \qquad\qquad \textit{otherwise.} & \end{cases}$$

where $\max^w$ *is the maximum point in* W *and* $\min^v$ *is the minimum point in* V*.*

2.14 Formal Problems

Definition 21 (Formal Problem, Decision Problem) *A **formal problem** is a precise definition of two classes of mathematical objects, the class of **instances** of the problem and the class of **solutions**, along with a function mapping each instance into (the set of) its solution(s). A problem is a **decision problem** iff its class of solutions is just* {true, false}.

2.15 Some formal problems

Definition 22 ((Fixed) Recognition Problem) *An instance of the **(Fixed) Recognition Problem**, for a specific stringset, is a string over the appropriate alphabet. The solution is ‘*true*’ if the string is in the set, ‘*false*’ otherwise.*

Definition 23 (Universal Recognition Problem) *An instance of the **Universal Recognition Problem**, for a class of formal descriptions of stringsets, is a pair consisting of a description in the class and a string over the appropriate alphabet. The solution is ‘*true*’ if the string is in the set defined by the description, ‘*false*’ otherwise.*

Definition 24 (Parsing Problem) *An instance of the **Parsing Problem** for a class of mathematical structures encoding the syntactic structure of strings is a string over the appropriate alphabet. The solution is either a structure in the class that encodes the syntax of that string or ‘*fail*’ if there is none.*

Variations on the parsing problem include requiring the solution to be the set of all structures that encode the structure of the given string, as well as Universal forms in which the instance is a description in a class of formal descriptions of sets of structures along with a string.

Definition 25 (Emptiness Problem) *An instance of the **Emptiness Problem** for a class of formal descriptions of stringsets is a description in that class. The solution is ‘*true*’ if there are no strings that satisfy the description, ‘*false*’ otherwise.*

Definition 26 (Finiteness Problem) *An instance of the **Finiteness Problem** for a class of formal descriptions of stringsets is a description in that class. The solution is ‘*true*’ if the set of strings that satisfy the the description is finite, ‘*false*’ otherwise.*

Definition 27 (Universality Problem) *An instance of the **Universality Problem** for a class of formal descriptions of stringsets is a description in that class. The solution is* ‘true’ *if every string over the appropriate alphabet satisfies the description,* ‘false’ *otherwise.*

2.16 Algorithms

Definition 28 (Algorithm) *An **algorithm** for a problem is a **definite procedure**, one that is defined in terms of a sequence of unambiguous mathematically precise steps, which, starting with any instance of the problem, is guaranteed to produce a (correct) solution for that instance after finitely many steps. If the procedure, given any instance, always produces some solution after finitely many steps we say that it is **terminating**. If any solution that the procedure arrives at is, in fact, one of the solutions of the instance it started with we say that it is **partially correct**. If it is both partially correct and terminating we say that it is **totally correct**. Properly, algorithms are required to be totally correct.*

2.17 Rec

We say that a problem is **computable** iff there is an algorithm that solves it. Computable decision problems are often said to be **decidable**. The stringsets for which the (fixed) recognition problem is computable are said to be **recursive**. The class of all recursive stringsets is denoted **Rec**.

Definition 29 (Rec) *A stringset is in the class* **Rec** *iff its (fixed) recognition problem is computable.*

3 Propositional Languages for Strings—Strictly Local Stringsets

3.1 k-factors

Definition 30 (k-factors) *Given a string w and a length k, the set of k-factors of w is:*

$$F_k(w) \overset{def}{=} \begin{cases} \{y \mid w = x \cdot y \cdot z,\ x, y, z \in \Sigma^*, |y| = k\} & \text{if } |w| > k, \\ \{w\} & \text{otherwise.} \end{cases}$$

The set of k-factors of a stringset L is the set of k-factors of the strings that it contains:

$$F_k(L) \overset{def}{=} \bigcup_{w \in L} [F_k(w)].$$

Definition 31 (Augmented string) *Suppose $\{\rtimes, \ltimes\} \notin \Sigma$. An **augmented string** over Σ is a string $w \in \Sigma^*$ with marked ends: $\rtimes w \ltimes$*

3.2 Strictly Local Descriptions

Definition 32 (Strictly k-Local Description) *A **strictly k-local description** is an arbitrary set of k-factors drawn from the alphabet Σ, possibly starting with '$\rtimes$' and/or ending with '$\ltimes$':*

$$\mathcal{G} \subseteq F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$$

A string model w satisfies such a description iff the augmented string $\rtimes \cdot w \cdot \ltimes$ includes only k-factors given in the description:

$$w \models \mathcal{G} \overset{def}{\Longleftrightarrow} F_k(\rtimes \cdot w \cdot \ltimes) \subseteq \mathcal{G}$$

$L(\mathcal{G})$ is the stringset defined by $\mathcal{G}$, the set of finite *strings which satisfy it:*

$$L(\mathcal{G}) \overset{def}{=} \{w \mid w \models \mathcal{G},\ w \textit{ finite}\}.$$

A stringset is Strictly k-Local (SL_k) iff it can be defined by a Strictly k-Local description. It is Strictly Local (SL) iff it is SL_k for some k.

3.3 Capabilities of Strictly 2-Local Descriptions

- Valency of verbs
  - Exclude *"likes $\ltimes$"*, *"slept Alice"*, etc.
- (Local) number agreement
  - Exclude *"Alice like"*, *"dogs likes"*, etc.
- Presence of an (initial) subject
  - Exclude *"$\rtimes$ like"*, *"$\rtimes$ slept"*, etc.
- (Local) selectional restrictions
  - Exclude *"biscuit likes"*, *"biscuit slept"*, etc.

3.4 Canonical SL stringsets

Iterated sequences of k distinct symbols are SL_k.
$\{(ab)^i \mid 0 \leq i\} \in \mathrm{SL}_2$, as witnessed by

$$\{\rtimes\ltimes,\ \rtimes a,\ ab,\ ba,\ b\ltimes\}$$

$\{(a_1 a_2 \cdots a_k)^i \mid 0 \leq i\} \in \mathrm{SL}_k$ ($a_i = a_j$ iff $i = j$), as witnessed by

$$\begin{array}{ll} \{ & \rtimes\ltimes,\ \rtimes a_1 \cdots a_{k-1},\ a_1 \cdots a_k,\ a_2 \cdots a_k a_1,\ \ldots, \\ & a_i a_{i+1} \cdots a_k a_1 \cdots a_{i-1},\ \ldots \\ & a_2 \cdots a_k \ltimes \\ & \} \end{array}$$

(Exercise) What stringsets would the class SL_1 include?

3.5 SL$_2$ example

Example 33

$$\begin{array}{ll}\mathcal{D}_{33} = \{ & \rtimes \textbf{\textit{Alice}},\ \rtimes \textbf{\textit{people}}, \\ & \textbf{\textit{Alice sleeps}},\ \textbf{\textit{people sleep}},\ \textbf{\textit{Alice loves}},\ \textbf{\textit{people love}}, \\ & \textbf{\textit{love Alice}},\ \textbf{\textit{love people}},\ \textbf{\textit{loves Alice}},\ \textbf{\textit{loves people}}, \\ & \textbf{\textit{sleep}} \ltimes,\ \textbf{\textit{sleeps}} \ltimes,\ \textbf{\textit{Alice}} \ltimes,\ \textbf{\textit{people}} \ltimes \qquad \} \end{array}$$

$\in L(\mathcal{D}_{33})$	$\notin L(\mathcal{D}_{33})$
Alice sleeps	*people sleep Alice*
people sleep	*Alice love people*
Alice loves people	...

But also licenses just *Alice* and *people*.

3.6 SL$_3$ example

Example 34

$$\begin{array}{ll}\mathcal{D}_{34} = \{ & \rtimes \textbf{\textit{Alice sleeps}},\ \rtimes \textbf{\textit{Alice loves}}, \\ & \rtimes \textbf{\textit{people sleep}},\ \rtimes \textbf{\textit{people love}}, \\ & \textbf{\textit{Alice loves Alice}},\ \textbf{\textit{Alice loves people}}, \\ & \textbf{\textit{people love Alice}},\ \textbf{\textit{people love people}}, \\ & \textbf{\textit{Alice sleeps}} \ltimes,\ \textbf{\textit{people sleep}} \ltimes, \\ & \textbf{\textit{loves Alice}} \ltimes,\ \textbf{\textit{loves people}} \ltimes, \\ & \textbf{\textit{love Alice}} \ltimes,\ \textbf{\textit{love people}} \ltimes \qquad \} \end{array}$$

Claim 35 *If $w \in L(\mathcal{D}_{34})$ then w includes a verb.*

But *people love Alice sleeps* $\in L(\mathcal{D}_{34})$.

3.7 SL$_4$ example

Example 36

$$\begin{array}{ll}\mathcal{D}_{36} = \{ & \rtimes \textbf{\textit{Alice sleeps}},\ \textbf{\textit{Alice sleeps}} \ltimes, \\ & \rtimes \textbf{\textit{Alice loves Alice}},\ \textbf{\textit{Alice loves Alice}} \ltimes, \\ & \rtimes \textbf{\textit{Alice loves people}},\ \textbf{\textit{Alice loves people}} \ltimes, \\ & \rtimes \textbf{\textit{people sleep}},\ \textbf{\textit{people sleep}} \ltimes, \\ & \rtimes \textbf{\textit{people love Alice}},\ \textbf{\textit{people love Alice}} \ltimes, \\ & \rtimes \textbf{\textit{people love people}},\ \textbf{\textit{people love people}} \ltimes \quad \} \end{array}$$

Claim 37 *$L(\mathcal{D}_{36})$ is finite.*

3.8 Finite stringsets and SL

Lemma 38 (Fin $\subsetneq$ SL) *Every finite set of strings is definable with an* SL$_{l+1}$ *description, where l is the maximum length of the strings in the set. Conversely, there* are SL$_2$ *stringsets which are not finite.*

Claim 39 *$L(\mathcal{D}_{34})$ is not finite.*

$$\{\textit{Alice}\ (\textit{loves Alice})^i \mid i \in \mathbb{N}\} \subseteq L(\mathcal{D}_{34})$$

3.9 Cognitive interpretation of SL

- Any cognitive mechanism that can distinguish member strings from non-members of an SL$_k$ stringset must be sensitive, at least, to the length k blocks of events that occur in the presentation of the string.

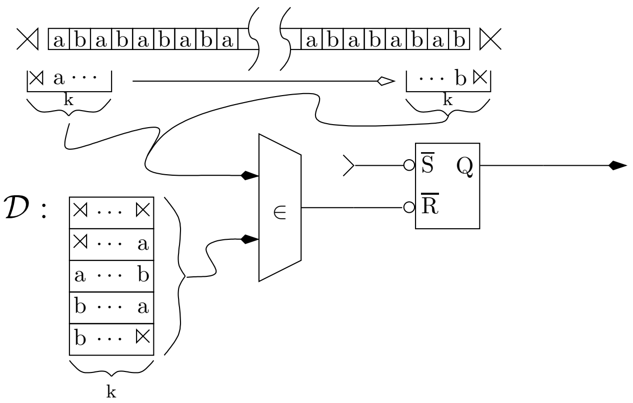
- If the strings are presented as sequences of events in time, then this corresponds to being sensitive, at each point in the string, to the immediately prior sequence of $k-1$ events.

- Any cognitive mechanism that is sensitive *only* to the length k blocks of events in the presentation of a string will be able to recognize *only* SL$_k$ stringsets.

3.10 Abstract recognition algorithms

Definition 40 (Automata) *An* ***automaton*** *is a formal presentation of an abstract algorithm for the (Fixed) Recognition Problem for a specific stringset. A* ***class of automata*** *is a collection of automata representing the same algorithm over a range of stringsets determined by parameters of the algorithm. The Universal Recognition Problem for a class of automata takes the values of those parameters and a string as input and asks if the string is accepted by the corresponding automaton.*

3.11 Automata for SL



Definition 41 (k-local Scanners) *A k-**local Scanner** $\mathcal{M}$ is a pair: $\langle \Sigma, T\rangle$, where $T \subseteq F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)$.*
A ***computation*** *of a k-local scanner is a sequence of one or more k-factors: $\langle x_1, x_2, \ldots, x_m\rangle$ where*

- *each $x_i \in T$,*

- *$x_1 = \rtimes\sigma_1 \cdots \sigma_{k-1}$,*

- *$x_m = \sigma_{n-(k-2)} \cdots \sigma_n \ltimes$ and*

- *for all $i < m$, $x_i = \sigma_i\sigma_{i+1} \cdots \sigma_{i+(k-1)}$ and $x_{i+1} = \sigma_{i+1} \cdots \sigma_{i+(k-1)}\sigma_{i+k}$.*

*A string w is **accepted** by a k-local scanner ($w \in L(\mathcal{M})$) iff the sequence of k-factors occurring in $\rtimes w \ltimes$, in order, is a computation of $\mathcal{M}$.*
*A stringset L is **recognized** by a k-local scanner $\mathcal{M}$ iff $L = L(\mathcal{M})$.*

Lemma 42 *A stringset L is SL_k iff there is a k-local scanner $\mathcal{M}_L$ which recognizes it.*

Proof $L \in \mathrm{SL}_k$ iff $L = (\mathcal{D}_L)$ for some strictly k-local description $\mathcal{D}_L \subseteq F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)$.
Let $\mathcal{M}_L = \langle \Sigma, \mathcal{D}_L \rangle$.
Then $w \in L(\mathcal{M}_L)$ iff $F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_L$ iff $w \in L(\mathcal{D}_L)$. ⊣

Theorem 43 (SL $\subseteq$ Rec.) *Both the fixed and universal recognition problems for* SL *are decidable.*

Lemma 42 establishes decidability of the fixed recognition problem. The fact that the universal recognition problem is decidable follows from the fact that the construction of the equivalent strictly k-local scanner from a SL_k description is **effective** (i.e., it is an algorithmic process).

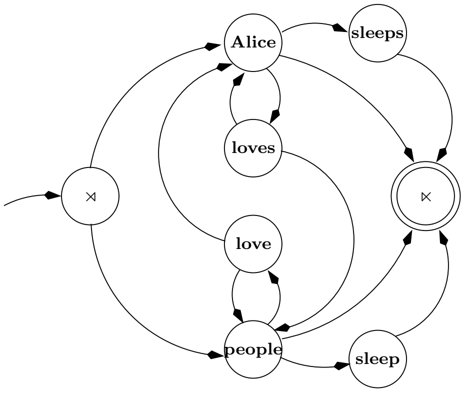
3.12 Myhill graphs

Definition 44 (Myhill Graphs) *A **Myhill Graph** over an alphabet Σ is a directed graph $\mathcal{G} = \langle V, E \rangle$, where:*

$$\begin{array}{lcll} V & = & \Sigma \cup \{\rtimes, \ltimes\} & \textit{The } \textbf{\textit{vertices}} \textit{ of the graph} \\ E & \subseteq & V \times V & \textit{The } \textbf{\textit{edges}} \textit{ of the graph} \end{array}$$

*For consistency with the graphs we will introduce later, we will mark the vertex $\rtimes$ as the **start state** with an "edge from nowhere" and will mark the vertex $\ltimes$ as the **final state** with a double circle.*

3.13 The Myhill graph corresponding to $\mathcal{D}_{33}$



Lemma 45 *A string w is accepted by a scanner for an SL_2 description $\mathcal{D}$ iff the augmented string $\rtimes w \ltimes$ is the sequence of vertices visited along some path from $\rtimes$ to $\ltimes$ in the Myhill graph corresponding to $\mathcal{D}$.*

Proof Each edge $\langle v_1, v_2\rangle$ corresponds to the 2-factor v_1v_2. There is a path $\langle\langle \rtimes, v_1\rangle, \langle v_1, v_2\rangle, \ldots, \langle v_n, \ltimes\rangle\rangle$ from $\rtimes$ to $\ltimes$ in the Myhill graph corresponding to $\mathcal{D}$ iff $\langle \rtimes v_1, v_1v_2, \ldots, v_n\ltimes\rangle$ is a computation of the scanner for $\mathcal{D}$, which is to say iff $v_1v_2\cdots v_n \in L(\mathcal{D})$. ⊣

3.14 Generalized Myhill Graphs

Definition 46 (Generalized Myhill Graphs) *A **k-Myhill graph** over an alphabet Σ is a directed, edge-labeled graph with a set of distinguished vertices $\mathcal{G} = \langle V, E, \ell, F\rangle$ where:*

$$\begin{array}{rcl} V & = & \{w \in \Sigma^* \mid |w| < k\} \\ E & \subseteq & V \times V \\ \ell & : & E \to \Sigma \\ F & \subseteq & V. \end{array}$$

and

$$\ell(\langle v_1, v_2\rangle) = \sigma \Rightarrow \begin{cases} v_2 = v_1\sigma, & |v_1| < (k-1) \\ v_1 = \sigma' w \textit{ and } v_2 = w\sigma, & \textit{otherwise.} \end{cases}$$

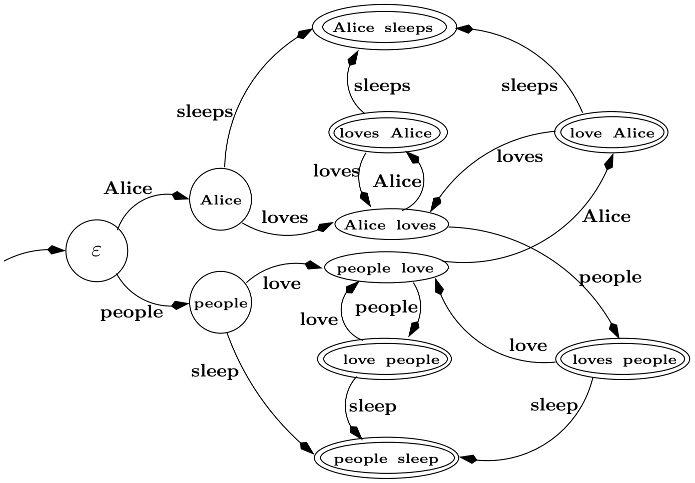
An SL_k description $\mathcal{D}$ corresponds to the k-Myhill graph over the same alphabet in which:

$$\begin{array}{rcl} \langle v_1, v_2\rangle \in E_{\mathcal{D}},\ \ell(\langle v_1, v_2\rangle) = \sigma & \overset{\text{def}}{\Longleftrightarrow} & \begin{cases} v_1\sigma \in \mathcal{D} & |v_1| = k-1, \text{ or} \\ \rtimes v_1\sigma w \in \mathcal{D} & \text{for some } w \in \Sigma^* \end{cases} \\ v \in F_{\mathcal{D}} & \overset{\text{def}}{\Longleftrightarrow} & v\ltimes \in \mathcal{D}. \end{array}$$

A k-Myhill graph $\mathcal{G}$ corresponds to the k-local description:

$$\begin{array}{lll} \mathcal{D}_{\mathcal{G}} \overset{\text{def}}{=} \{ & v_1\sigma & \text{if } \langle v_1, v_2\rangle \in E,\ \ell(\langle v_1, v_2\rangle) = \sigma \text{ and } |v_1| = k-1, \\ & v_1\ltimes & \text{if } v_1 \in F \text{ and } |v_1| = k-1, \\ & \rtimes v_1\sigma & \text{if } \langle v_1, v_2\rangle \in E,\ \ell(\langle v_1, v_2\rangle) = \sigma, |v_1| = k-2, \\ & & \quad \text{and there is a path from `}\varepsilon\text{' to } v_1, \\ & \rtimes v_1\ltimes & \text{if } v_1 \in F,\ |v_1| < k-1 \\ & & \quad \text{and there is a path from `}\varepsilon\text{' to } v_1 \qquad\qquad \} \end{array}$$

3.15 The 3-Myhill graph corresponding to $\mathcal{D}_{34}$.



Lemma 47 *A string w is accepted by a scanner for an* SL_k *description $\mathcal{D}$ iff w is the sequence of edge labels along some path from ε to a vertex in F in the k-Myhill graph corresponding to $\mathcal{D}$.*

Theorem 48 (Emptiness is decidable for SL **stringsets)** *There is an algorithm that, given any* SL *description $\mathcal{D}$, decides $L(\mathcal{D}) \overset{?}{=} \emptyset$*

Proof Given any SL_k description $\mathcal{D}$, construct the corresponding k-Myhill graph. Since $w \in L(\mathcal{D})$ iff there is a path labeled w through the graph from 'ε' to a vertex in F, $L(\mathcal{D}) = \emptyset$ iff there is no path from 'ε' to any vertex in F. The (non-)existence of such a path can be determined by **breadth-first search**. ⊣

Theorem 49 (Finiteness is decidable for SL **stringsets)** *There is an algorithm that, given any* SL *description, decides if $L(\mathcal{D})$ is finite.*

(Proof exercise.)

Theorem 50 (Universality is decidable for SL **stringsets)** *There is an algorithm that, given any* SL *description $\mathcal{D}$, decides $L(\mathcal{D}) \overset{?}{=} \Sigma^*$*

(Proof exercise. Think in terms of the description itself rather than the Myhill graph of its scanner.)

3.16 Abstract generation algorithms

Definition 51 (Formal Grammar) *A* ***(formal) grammar*** *is a formal presentation of an abstract algorithm for constructing strings in a specific stringset. In this context the fixed recognition problem asks whether it is possible to construct a given string using the grammar. A* ***class of grammars*** *is a collection of grammars representing the same algorithm over a range of stringsets determined by parameters of the algorithm. In this context the universal recognition problem asks, given specific values for the parameters and a string, whether that string can be constructed by the corresponding grammar.*

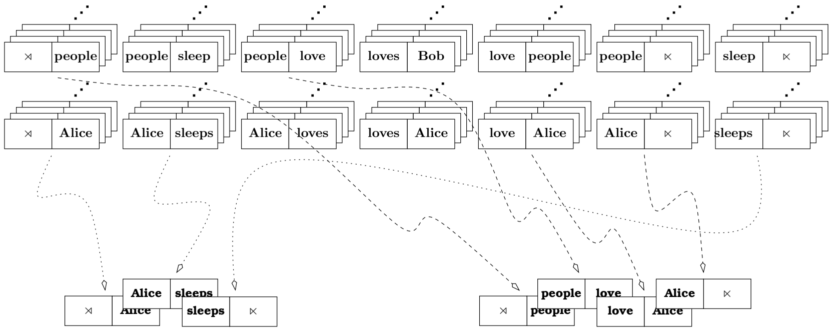
3.17 Recursively Enumerable (r.e.)

Definition 52 (r.e.) *A stringset L is in the class* **r.e.** *iff there is an algorithm which computes a function f mapping $\mathbb{N}$ to Σ^* for which $\{f(i) \mid i \in \mathbb{N}\} = L$ (i.e., the* ***range*** *of f is L).*

Lemma 53 $\mathbf{Rec} \subsetneq \mathbf{r.e.}$

A recursive enumerator is, in effect, an algorithm for searching through the set of strings in L. While if we happen to find a string w we can be certain that $w \in L$, if, on the other hand, we have not found w after searching for some finite time there is, in general, no way of knowing whether $w \notin L$ or we just haven't found it yet. Hence the search terminates iff $w \in L$. In fact, **Rec** is exactly that subclass of **r.e.** for which there is some algorithmic means of terminating the search.

3.18 Grammars for SL



$\rtimes$ *Alice sleeps* $\ltimes$ $\qquad$ $\rtimes$ *people love Alice* $\ltimes$

3.19

Definition 54 *A* ***strictly*** k***-local*** *grammar is a tuple $\mathcal{G} = \langle \Sigma, A, P \rangle$, where*

$$\begin{array}{rcl} A & \subseteq & \{\rtimes \cdot w \cdot \ltimes \mid w \in \Sigma^*,\ |w| < (k-1)\} \cup \\ & & \quad \{\rtimes \cdot w \mid w \in \Sigma^*,\ |w| = (k-1)\} \\ P & \subseteq & \{\langle w, \sigma \rangle \mid w \in \Sigma^*\ |w| = (k-1) \text{ and } \sigma \in \Sigma \cup \{\ltimes\}\ \} \end{array}$$

3.20 Example

$$\mathcal{G}_{34} = \langle \Sigma_{34}, A_{34}, P_{34} \rangle$$

$$\begin{array}{rcll} \Sigma_{34} & = & \{\textit{\textbf{Alice}}, \textit{\textbf{sleeps}}, \textit{\textbf{loves}}, \textit{\textbf{people}}, \textit{\textbf{sleep}}, \textit{\textbf{love}}\} & \\ A_{34} & = & \{ \quad \rtimes \textit{\textbf{Alice sleeps}},\ \rtimes \textit{\textbf{Alice loves}}, & \\ & & \quad\ \rtimes \textit{\textbf{people sleep}},\ \rtimes \textit{\textbf{people love}} & \} \\ P_{34} & = & \{ \quad \langle \textit{\textbf{Alice loves}}, \textit{\textbf{Alice}} \rangle,\ \langle \textit{\textbf{Alice loves}}, \textit{\textbf{people}} \rangle, & \\ & & \qquad \langle \textit{\textbf{people love}}, \textit{\textbf{Alice}} \rangle,\ \langle \textit{\textbf{people love}}, \textit{\textbf{people}} \rangle, & \\ & & \qquad \langle \textit{\textbf{Alice sleeps}}, \ltimes \rangle,\ \langle \textit{\textbf{people sleep}}, \ltimes \rangle, & \\ & & \qquad \langle \textit{\textbf{loves Alice}}, \ltimes \rangle,\ \langle \textit{\textbf{loves people}}, \ltimes \rangle, & \\ & & \qquad \langle \textit{\textbf{love Alice}}, \ltimes \rangle,\ \langle \textit{\textbf{love people}}, \ltimes \rangle & \} \end{array}$$

3.21 Derivations

Definition 55 (Derives relation) *If $\mathcal{G} = \langle \Sigma, A, P\rangle$ is a strictly k-local grammar, then w_1* ***directly derives*** *w_2 in $\mathcal{G}$:*

$$w_1 \underset{\mathcal{G}}{\Longrightarrow} w_2 \overset{def}{\Longleftrightarrow} w_1 = u \cdot v,\ w_2 = u \cdot v\sigma \text{ and } \langle v, \sigma\rangle \in P.$$

A ***derivation*** *of w_n from w_1 in $\mathcal{G}$ is a sequence of one or more strings:*

$$\langle w_1, w_2, \ldots, w_n\rangle \text{ where } w_i \underset{\mathcal{G}}{\Longrightarrow} w_{i+1},\ i < n.$$

A string w_1 ***derives*** *w_n in $\mathcal{G}$:*

$$w_1 \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} w_n \overset{def}{\Longleftrightarrow} \text{there is a derivation of } w_n \text{ from } w_1 \text{ in } \mathcal{G}.$$

3.22 Stringset generated by a strictly k-local grammar

Definition 56 *The stringset generated by a strictly k-local grammar $\mathcal{G} = \langle \Sigma, A, P\rangle$ is*

$$L(\mathcal{G}) \overset{def}{=} \{w \in \Sigma^* \mid w_1 \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} \rtimes w \ltimes,\ w_1 \in A\}.$$

Lemma 57 *If $\mathcal{D} \subseteq F_k(\rtimes \cdot \Sigma^* \ltimes)$ is a strictly k-local description and $\mathcal{G} \overset{def}{=} \langle \Sigma, A_{\mathcal{D}}, P_{\mathcal{D}}\rangle$ where*

$$\begin{array}{rcl} A_{\mathcal{D}} & = & ((\rtimes \cdot \Sigma^*) \cup (\rtimes \cdot \Sigma^* \ltimes)) \cap \mathcal{D} \\ P_{\mathcal{D}} & = & \{\langle v, \sigma\rangle \mid v\sigma \in (\mathcal{D} - A_{\mathcal{D}})\} \end{array}$$

then $L(\mathcal{D}) = L(\mathcal{G}_{\mathcal{D}})$.

3.23 Equivalence of strictly k-local description, scanners and grammars

Theorem 58 *The following are equivalent:*

- $L \in \mathrm{SL}$
- *$L = L(\mathcal{D})$ for a strictly k-local description $\mathcal{D}$*
- *$L = L(\mathcal{M})$ for a strictly k-local scanner $\mathcal{M}$*
- *$L = L(\mathcal{G})$ for a strictly k-local grammar $\mathcal{G}$*

3.24 Relationship between strictly k-local classes (I)

Lemma 59 $\mathrm{SL}_i \subseteq \mathrm{SL}_j$ *for all $i \leq j$.*

Proof To see that $\mathrm{SL}_i \subseteq \mathrm{SL}_{i+1}$ for all i, note that any strictly i-local description $\mathcal{D}_i$ can be extended to an equivalent strictly $i+1$-local description $\mathcal{D}_{i+1}$ in the following way:

$$\mathcal{D}_{i+1} \overset{\mathrm{def}}{=} \{\rtimes w \ltimes \mid \rtimes w \ltimes \in \mathcal{D}_i\} \cup \{\sigma_1\sigma_2 \cdots \sigma_{i+1} \mid \sigma_1\sigma_2 \cdots \sigma_i, \sigma_2 \cdots \sigma_{i+1} \in \mathcal{D}_i\}$$

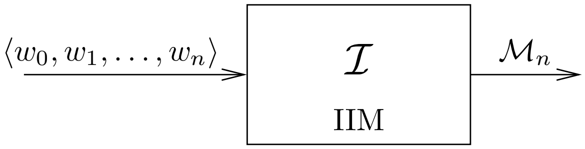
That $L(\mathcal{D}_{i+1}) = L(\mathcal{D}_i)$ can be verified by considering the derivations of the grammars for $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$: these are identical except that the derivation for $\mathcal{D}_{i+1}$ starts with the second string of the derivation for $\mathcal{D}_i$. (This is clearest, perhaps, if one considers the grammars in terms of tiles.)
The full lemma then follows by the fact that $\subseteq$ is reflexive and transitive. $\dashv$

3.25 Recognition via generation

Theorem 60 (SL $\subseteq$ **Rec (again))** *Both the fixed and universal recognition problem for* SL *are decidable.*

Proof Derivations in strictly k-local grammars grammars are *strictly length increasing*: if $w \underset{\mathcal{G}}{\Longrightarrow} v$ then $|v| > |w|$ (in fact, $|v| = |w|+1$). Thus, no derivation for a string of length l can take more than $l+1$ steps. (In fact the *only* derivation of the string, should it exist, will take exactly $l+1$ steps.) So if we search our derivations exhaustively in order of increasing length, we can stop searching when we see the first derivation of length greater than $l+1$. If we haven't found the string we are looking for by then, we never will.
The fact that the universal recognition problem is decidable, again, follows from the fact that the construction of the k-local grammar from an SL_k description is effective. $\dashv$

3.26 Identification in the limit



Definition 61 (Gold) *A class of stringsets is learnable in the limit from positive data if there is a computable function $\mathcal{I}$ mapping finite sequences of strings to automata for the class such that, if $\ell : \mathbb{N} \to \Sigma^*$ is an enumeration of L then there will be some $i \in \mathbb{N}$ such that, for all $n \geq i$,*

1. $\mathcal{I}(\langle \ell(0), \ell(1), \ldots, \ell(i), \ldots, \ell(n)\rangle) = \mathcal{I}(\langle \ell(0), \ell(1), \ldots, \ell(i)\rangle)$ *and*
2. $L(\mathcal{I}(\langle \ell(0), \ell(1), \ldots, \ell(n)\rangle)) = L$.

Theorem 62 *For all k, the class of* SL_k *stringsets is learnable in the limit.*

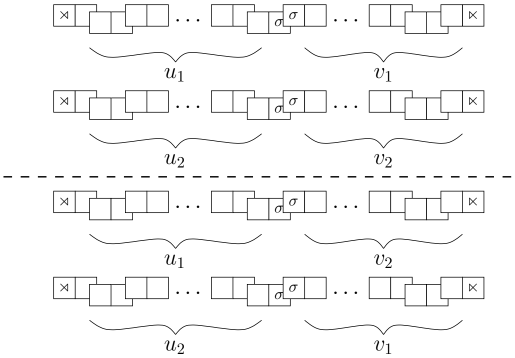
Proof Suppose $L = L(D_L)$ for some k-local description D_L. Without loss of generality, there are no useless k-factors in D_L, i.e., $D_L = \bigcup_{w \in L}[F_k(\rtimes w \ltimes)]$. Suppose ℓ is any enumeration of L. Let

$$\mathcal{I}(\langle \ell(0), \ell(1), \ldots, \ell(n)\rangle) = \mathcal{M}_n = \langle \Sigma, T_n\rangle, \text{ where } T_n \overset{\text{def}}{=} \bigcup_{0 \leq i \leq n} [F_k(\rtimes \ell(i) \ltimes)].$$

We claim that, since ℓ enumerates L, there will be some i for which $T_n = D_L$. Certainly, $T_n \subseteq D_L$ for all i. To see that there is some i for which $T_n \supseteq D_L$

as well, for all $n \geq i$, suppose $f \in D_L$. Then there is some $w \in L$ such that $f \in F_k(\rtimes w \ltimes)$. Since ℓ enumerates L there is some i such that $w = \ell(i)$. Then $f \in T_n$ for all $n \geq i$.
From that point on $\mathcal{M}_n = \mathcal{M}_i$ and $L(\mathcal{M}_n) = L(D_L) = L$. ⊣

3.27 Suffix Substitution Closure



Lemma 63 ((2-Local) Suffix Substitution Closure) *If L is a strictly 2-local stringset then for all strings u_1, v_1, u_2, and v_2 in Σ^* and all symbols σ in Σ:*

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

3.28 SSC characterizes SL

Theorem 64 (Suffix Substitution Closure) *A stringset L is Strictly Local iff there is some k such that whenever there is a string x of length $k-1$ and strings u_1, v_1, u_2, and v_2, such that*

$$\begin{array}{ccccccc} u_1 & \cdot & x & \cdot & v_1 & \in L \\ u_2 & \cdot & x & \cdot & v_2 & \in L \end{array}$$

then it will also be the case that

$$\begin{array}{ccccccc} u_1 & \cdot & x & \cdot & v_2 & \in L \end{array}$$

3.29 Proof of SSC ($\Leftarrow$)

Suppose that L is closed under k-local suffix substitution. Let

$$\mathcal{D}_L \overset{\text{def}}{=} \cup_{w\in L}[F_k(\rtimes w \ltimes)].$$

We claim that $L(\mathcal{D}_L) = L$.

($L \subseteq L(\mathcal{D}_L)$)

$$w \in L \Rightarrow F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_L \Rightarrow w \in L(\mathcal{D}_L)$$

($L(\mathcal{D}_L) \subseteq L$ ($\emptyset$ case))

$$L = \emptyset \Rightarrow \mathcal{D}_L = \emptyset \Rightarrow L(\mathcal{D}_L) = \emptyset.$$

Assume, for the remainder of the proof, that there is at least one string in L and, consequently (since $L \subseteq L(\mathcal{D}_L)$), in $L(\mathcal{D}_L)$.

3.30 Proof of SSC (non-$\emptyset$ case)

Suppose that $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in L(\mathcal{D}_L)$. Then there is a derivation in the corresponding generator which is of the form:

$$\langle \rtimes\sigma_1 \ldots \sigma_{k-1},\ \sigma_1 \ldots \sigma_{k-1}\sigma_k,\ \cdots\ ,\ \sigma_{n-k} \ldots \sigma_{n-2}\sigma_{n-1},\ \sigma_{n-(k-1)} \ldots \sigma_{n-1}\sigma_n \ltimes\rangle$$

The idea of the proof is to search through strings we know to be in L for those that agree with w on increasingly long prefixes. At stage i we will have some $w_i \in L$ which agrees with w in its first i positions. We will use the fact that the next k-factor of w, $\sigma_{i-(k-2)} \cdots \sigma_i\sigma_{i+1}$, is in $\mathcal{D}_L$ to show that there is some string z_i in L in which that k-factor occurs and then use suffix substitution closure to show that there is a string with the prefix from w_i and the suffix from z_i which agrees with w in its first $i+1$ positions.

3.31 Proof of SSC (construction)

$$\begin{array}{lll} n \le k-2 & \Rightarrow & \rtimes\sigma_1 \cdots \sigma_n \ltimes \in \mathcal{D}_L \\ & \Rightarrow & \rtimes\sigma_1 \cdots \sigma_n \ltimes \in F_k(\rtimes \cdot v \cdot \ltimes) \text{ for some } v \in L \Rightarrow v = w. \end{array}$$

Note that n could be 0, in which case v is ε.

(Stage 0) Otherwise $n \ge k-1$ and the k-factor $\rtimes\sigma_1 \cdots \sigma_{k-1} \in \mathcal{D}_L$. By construction of $\mathcal{D}_L$ there is some string $z \in L$ that starts with $\sigma_1 \cdots \sigma_{k-1}$. Choose any one of these for w_{k-1}.

(Invariants) For all $k-1 \le i \le n$:

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

(Exercise) Verify that these invariants are true for w_{k-1}.

(Stage $k \le i < n$)

$$\begin{array}{rclllll} w & = & \sigma_1\,\sigma_2\,\cdots\,\sigma_{i-(k-1)} & \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i & \sigma_{i+1}\,\cdots\,\sigma_n & \\ w_i & = & \sigma_1\,\sigma_2\,\cdots\,\sigma_{i-(k-1)} & \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i & v_i & \in L \\ z_i & = & \hfill x_i & \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i & \sigma_{i+1}\,y_i & \in L \\ w_{i+1} & = & \sigma_1\,\sigma_2\,\cdots\,\sigma_{i-(k-1)} & \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i & \sigma_{i+1}\,y_i & \in L \end{array}$$

$w_i \in L$ by Invariant 1.
Since $\sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i\sigma_{i+1} \in F_k(w) \subseteq \mathcal{D}_L$,
$\sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i\sigma_{i+1} \in z_i$, for some $z_i \in L$.
$z_i = x_i \cdot \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i\sigma_{i+1} \cdot y_i$.
Since L closed under substitution of suffixes that start with the same $(k-1)$-factor,

$$w_{i+1} = \sigma_1\,\sigma_2\,\cdots\,\sigma_{i-(k-1)} \cdot \sigma_{i-(k-2)}\,\cdots\,\sigma_{i-1}\,\sigma_i \cdot \sigma_{i+1}\,y_i \in L$$

(Stage n)
By the invariants: $w_n = u_n\sigma_{n-(k-2)} \cdots \sigma_n v_n$, where
$u_n = \sigma_1 \cdot \sigma_2 \cdots \sigma_{n-(k-1)}$ (possibly ε).
Since $w \in L(\mathcal{D}_L)$ and w ends with $\sigma_{n-(k-2)} \cdots \sigma_n$, the k-factor $\sigma_{n-(k-2)} \cdots \sigma_n\ltimes \in \mathcal{D}_L$ and there must be some

$$z_n = x_n \cdot \sigma_{n-(k-2)} \cdots \sigma_n \in L.$$

Since both $w_n = u_n\sigma_{n-(k-2)}\cdots\sigma_n v_n$ and $z_n = x_n \cdot \sigma_{n-(k-2)}\cdots\sigma_n \cdot \varepsilon$ are in L which is closed under substitution of suffixes that start with the same $(k-1)$-factor,

$$w_n = u_n\sigma_{n-(k-2)}\cdots\sigma_n \cdot \varepsilon \in L.$$

3.32 Non-SL Stringsets

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

To show that $L \notin \text{SL}$ (by contrapositive) it suffices to show

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

3.33 Adversary Arguments

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

∀—adversary's choice, ∃—your choice

- Your adversary, claiming that there is a k-local automaton that recognizes L, chooses k.
- You now choose two strings u_1xv_1 and u_2xv_2. Your choice should depend on the specific value of k your adversary chose (as well, of course, as on L).
- 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.

3.34 Example

Consider the stringset

$$L_{3.34} = \{wabv \mid w, v \in \{a,b\}^*\}$$

To show that this is *not* SL_k, suppose (for contradiction) that it was SL_k for some k. (Adversary chooses k.) Then it would exhibit the k-Suffix Substitution Property. Now both the strings a^kb and aba^k are in $L_{3.34}$ (our choice of strings) and these can be broken down as follows

$$\underbrace{a}_{u_1}\underbrace{a\cdots a}_{k-1}\underbrace{b}_{v_1} \text{ and } \underbrace{ab}_{u_2}\underbrace{a\cdots a}_{k-1}\underbrace{a}_{v_2}$$

By the Suffix Substitution Closure Property, then, $u_1 a^{k-1} v_2 = aa^{k-1}a$ would also be in $L_{3.34}$. But it is not. In this way, reasoning from the supposition that $L_{3.34} \in \mathrm{SL}_k$ we obtain a contradiction. Hence $L_{3.34} \notin \mathrm{SL}_k$ for any k.

Theorem 65 *No stringset modeling English is* SL_2

$$\begin{array}{rcccll}
\textit{I} & \cdot & \textit{absented} & \cdot & \textit{myself} & \in E \\
\textit{You} & \cdot & \textit{absented} & \cdot & \textit{yourself} & \in E \\
\hline
\textit{I} & \cdot & \textit{absented} & \cdot & \textit{yourself} & \notin E
\end{array}$$

Theorem 66 *No stringset modeling English is* SL

Example 67

$$\begin{array}{rcccll}
\textit{Which girls do they think} & \cdot & \overbrace{(\textit{that they think})}^{(k-1)/3} & \cdot & \textit{were responsible} & \in E \\
\textit{Which girl do they think} & \cdot & \overbrace{(\textit{that they think})}^{(k-1)/3} & \cdot & \textit{was responsible} & \in E \\
\hline
\textit{Which girls do they think} & \cdot & \overbrace{(\textit{that they think})}^{(k-1)/3} & \cdot & \textit{was responsible} & \notin E
\end{array}$$

3.35 Relationship between strictly k-local classes (II)

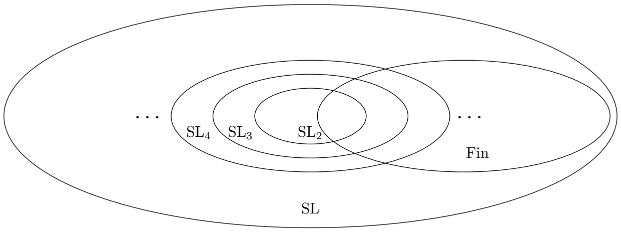
Lemma 68 $\mathrm{SL}_i \subsetneq \mathrm{SL}_j$ *for all* $i \leq j$.

Proof The inclusion is Lemma 59. On the other hand, it is easy to see that, for any given i, there are SL_{i+1} stringsets that are not SL_i, the singleton stringset $\{a^i\}$, for example. (Verify this.) ⊣

Theorem 69 (The SL hierarchy) *The classes of* SL_k *stringsets form a proper hierarchy:*

$$\mathrm{SL}_2 \subsetneq \mathrm{SL}_3 \subsetneq \cdots \mathrm{SL}_i \subsetneq \mathrm{SL}_{i+1} \subsetneq \cdots \subsetneq \mathrm{SL}.$$

3.36 Finite stringsets and SL



We have already seen that every finite stringset is SL_k for some k and that for all k, SL_k includes non-finite stringsets. Consequently, Fin $\subsetneq$ SL. On the other hand, it is again easy to see that, for any given k, there are finite stringsets that are not SL_k, the singleton stringset $\{a^k\}$, works here as well.

3.37 SL is not learnable

Theorem 70 *The class* SL *as a whole is not learnable in the limit from positive data.*

Proof
Let $\ell(i) \overset{\text{def}}{=} a^i$. Then ℓ is an enumeration of $L_{a^*} = \{a^i \mid 0 \leq i\} \in \text{SL}_2$. Suppose $\mathcal{I}$ is an IIM that learns L_{a^*} from the enumeration ℓ. Then there is some i for which $\mathcal{I}$ converges on an automaton for L_{a^*}. Let

$$\ell_i(j) = \begin{cases} a^j & j \leq i, \\ a^i & \text{otherwise.} \end{cases}$$

Then ℓ_i is an enumeration of $L_{a^{\leq i}} = \{a^j \mid 0 \leq j \leq i\}$, a finite set and, therefore, SL. But $\ell_i(j)$ and $\ell(j)$ agree for all $0 \leq j \leq i$ and, therefore, when fed the enumeration ℓ_i, $\mathcal{I}$ must converge on an automaton for L_{a^*}, an error.
Thus for any IIM that learns L_{a^*} there is some stringset $L_{a^{\leq i}}$ which it cannot learn. Since both stringsets are SL no IIM can learn all of SL. $\dashv$

3.38 Intersection of SL stringsets.

Lemma 71 *The class* SL_k*, for each* k*, is effectively closed under intersection, as is* SL *as a whole.*

Proof Suppose L_1 and L_2 are SL_k. Then each is defined by some strictly k-local description. Let $L_1 = L(\mathcal{D}_1)$ and $L_2 = L(\mathcal{D}_2)$.
Suppose, now, that some string w is in $L_1 \cap L_2$. Then

$$F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_1 \text{ and } F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_2.$$

Thus $F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_1 \cap \mathcal{D}_2$.
Conversely, suppose $F_k(\rtimes w \ltimes) \subseteq \mathcal{D}_1 \cap \mathcal{D}_2$. Then $F_k(\rtimes w \ltimes)$ must be a subset of each of $\mathcal{D}_1$ and $\mathcal{D}_2$ separately. Consequently, $w \in L(\mathcal{D}_1)$ and $w \in L(\mathcal{D}_2)$, which is to say w is in $L_1 \cap L_2$.
Thus

$$L_1, L_2 \in \text{SL}_k \Rightarrow L_1 \cap L_2 \in \text{SL}_k.$$

as witnessed by

$$L(\mathcal{D}_1 \cap \mathcal{D}_2) = L(\mathcal{D}_1) \cap L(\mathcal{D}_2).$$

Since SL_k is closed under intersection for each k, SL as a whole is as well. $\dashv$

3.39 Union of SL stringsets

Lemma 72 *The class of strictly local stringsets is* not *closed under union.*

Proof Let

$$L_1 = \{a^i b^j \mid i, j \geq 0\} \text{ and } L_2 = \{b^i a^j \mid i, j \geq 0\}$$

Both L_1 and L_2 are SL_2. (You should verify this by thinking about what the descriptions look like.)

We claim that $L_1 \cup L_2 \notin \mathrm{SL}_k$, for any k. To see this, suppose, by way of contradiction, that it was SL_k for some k. (Adversary chooses k.)
Then it would satisfy k-Suffix Substitution Closure. But $ab^{k-1} \in L_1 \cup L_2$, since it is in L_1, and $b^{k-1}a \in L_1 \cup L_2$, since it is in L_2, (our choice of strings) and, by Suffix Substitution Closure, this implies that $ab^{k-1}a \in L_1 \cup L_2$, which is not the case. Hence, $L_1 \cup L_2$ is not SL_k, for any k. ⊣

3.40 Complement of SL stringsets.

Definition 73 (Relative complement) *The* ***complement*** *of a set S_1* ***relative to*** *another S_2 is the set-theoretic difference between them:* $S_2 - S_1 \overset{def}{=} \{x \mid x \in S_2 \text{ and } x \notin S_1\}$.

Definition 74 (Complement of a stringset) *The* ***complement*** *of a stringset L over Σ is:* $\overline{L} \overset{def}{=} \Sigma^* - S$.

Lemma 75 *The class of strictly local stringsets is* not *closed under complement* .

Proof The fact that SL is closed under intersection but not under union implies that it is not closed under complement since, by DeMorgan's Theorem

$$L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}.$$

We know that the intersection of SL stringsets is also SL. If the complement of SL stringsets was also necessarily SL, then $\overline{\overline{L_1} \cap \overline{L_2}}$ would be SL contradicting the fact that there are SL stringsets whose union are not SL. ⊣

(Exercise) Let $L_{76} = \{a^i b^j \mid i, j \geq 0\}$. We have already checked that $L_{76} \in \mathrm{SL}_2$. Show, using Suffix Substitution Closure, that $\overline{L_{76}} \notin \mathrm{SL}_2$.

3.41 Concatenation of SL stringsets

Lemma 77 *The class of strictly local stringsets is* not *closed under concatenation.*

Proof Let $L_1 = \{wa \mid w \in \{a, b\}^*\}$ and $L_2 = \{bv \mid v \in \{a, b\}^*\}$. These are both SL_2 as witnessed by

$$\mathcal{D}_1 = \{\rtimes a, \rtimes b, aa, ab, ba, bb, a\ltimes\}$$

and

$$\mathcal{D}_2 = \{\rtimes b, aa, ab, ba, bb, a\ltimes, b\ltimes\}$$

But their concatenation is just the stringset of the example of Section 3.34:

$$L_{3.34} \overset{\text{def}}{=} \{wabv \mid w, v \in \{a, b\}^*\}$$

which we have shown to be non-SL. ⊣

3.42 Kleene closure of SL stringsets

Lemma 78 *The class of strictly local stringsets is* not *closed under Kleene closure.*

Proof $L_{aa} \overset{\text{def}}{=} \{aa\} \in \text{SL}_3$ (as witnessed by $\{\rtimes aa, aa\ltimes\}$) and also SL_k for any $k > 3$ (as witnessed by $\{\rtimes aa\ltimes\}$).
But $(L_{aa})^* = \{a^{2i} \mid i \geq 0\}$ which is not SL. $\dashv$

(Exercise) Show that $(L_{aa})^* \notin \text{SL}$.

3.43 Closure of SL_2 under Kleene closure

Lemma 79 *The class of strictly 2-local stringsets* is *closed under Kleene closure.*

Proof Exercise. You might think of this in terms of Myhill graphs. Show that, given a Myhill graph for an SL_2 stringset L, one can effectively extend it to one for L^+ and then to L^*. $\dashv$

4 Propositional Languages for Strings—Locally Testable Stringsets

Suppose that V is a transitive verb and that N is a noun phrase of length at least $k-1$ which can serve as either the subject or the object of V. We can then factor the string 'N V N' in both of the following ways:

$$\begin{array}{rcccll} \varepsilon & \cdot & \text{N} & \cdot & \text{V N} & \in E \\ \text{N V} & \cdot & \text{N} & \cdot & \varepsilon & \in E \\ \cline{1-5} \varepsilon & \cdot & \text{N} & \cdot & \varepsilon & \notin E \end{array}$$

Hence, in the presence of strings of arbitrary length which don't include a main verb and that can either start or end an expression, a strictly local description cannot enforce the presence of a main verb.
Strictly local descriptions can *forbid* the occurrence of a factor.
Strictly local descriptions cannot (in general) *require* the occurrence of a factor.
Complements of strictly local stringsets (co-SL $= \{\overline{L} \mid L \in \text{SL}\}$) *can* require the occurrence of a factor, but cannot forbid it.
Since SL_k closed under intersection, adding closure under complement gives closure under all Boolean operations.

4.1 k-Expressions

Definition 80 (k-Expressions over Strings) *The language of k-expressions (over strings) is the smallest set including:*

- *Atomic formulae: $f \in F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)$ is a k-expression.*
- *Disjunction: If φ_1 and φ_2 are k-expressions then $(\varphi_1 \vee \varphi_2)$ is a k-expression*
- *Negation: If φ_1 is a k-expression then $(\neg\varphi_1)$ is a k-expression.*

Definition 81 (Satisfaction of k-Expressions) *If w is a string and φ a k-expression, then*

$$w \models \varphi \overset{def}{\Longleftrightarrow} \begin{cases} \varphi = f \in F_k(\rtimes \cdot \Sigma^* \cdot \ltimes) \text{ and } f \in F_k(\rtimes \cdot w \cdot \ltimes), \\ \varphi = (\varphi_1 \vee \varphi_2) \text{ and } w \models \varphi_1 \text{ or } w \models \varphi_2, \\ \varphi = (\neg\varphi_1) \ (\textit{i.e., otherwise}) \text{ and } w \not\models \varphi. \end{cases}$$

4.2 Defined connectives

1. $(\varphi \wedge \psi) \triangleq (\neg((\neg\varphi) \vee (\neg\psi)))$ (**conjunction**),
2. $(\varphi \to \psi) \triangleq ((\neg\varphi) \vee \psi))$ (**implication**),
3. $(\varphi \leftrightarrow \psi) \triangleq ((\varphi \wedge \psi) \vee ((\neg\varphi) \wedge (\neg\psi)))$ (**bi-conditional**), ...

4.3 Capabilities of k-Expressions

$$\begin{array}{ll} & (\rtimes\,\text{my}) \\ \wedge & (\text{father}) \\ \wedge & (\text{loved the woman}\,\ltimes) \\ \wedge & (\,(\text{my father}) \vee (\text{my father's})\,) \\ \wedge & (\neg(\text{father father's})\,) \\ \wedge & (\neg(\text{father's loved})\,) \\ & \vdots \end{array}$$

$$\text{my}\,\overbrace{(\text{father's})}^{\geq 0}\,\text{father loved the woman}$$

4.4 Locally Testable stringsets

Definition 82 (Locally testable stringsets) *A stringset L is **Locally k-Testable** (LT_k) iff there is a k-expression φ such that:*

$$L = L(\varphi) \overset{def}{=} \{w \mid w \models \varphi,\ w \textit{ finite}\}$$

*A stringset L is **Locally Testable** (LT) iff it is Locally k-Testable for some k.*

Theorem 83 LT *recognition is decidable.*

Proof: Exercise

4.5 LT and SL

Lemma 84 ($\mathrm{SL}_k \subseteq \mathrm{LT}_k$) *A stringset L is Strictly k-Local iff it is definable by a k-expression in the form:*

$$\bigwedge_{f_i \notin \mathcal{G}} [\neg f_i].$$

where $\mathcal{G}$ is the strictly local description defining L.

Proof This is satisfied by all and only those strings in which no k-factor that is not in $\mathcal{G}$ occurs, that is, in which the k factors that *do* occur are limited to those in $\mathcal{G}$. Conversely, given a conjunction of negated k-factors, as in (84), we can convert it to an equivalent SL_k description by taking the complement (with respect to the set of all k-factors in $\rtimes \cdot \Sigma^* \cdot \ltimes$) of the set of k-factors it includes. $\dashv$

Lemma 85 *The class of Locally k-Testable stringsets is the closure of the class of Strictly k-Local stringsets under Boolean operations.*

Proof That every Boolean combination of Strictly k-Local stringsets is Locally k-Testable follows from Lemma 84 and closure of the class of k-expressions under the Boolean connectives. That every LT_k stringset is a Boolean combination of SL_k stringsets follows from the fact that every k-expression is a Boolean function of k-factors and that every k-factor f is logically equivalent to the negation of a k-expression in the form of (84): $\neg(\bigwedge[\neg f])$ (where the conjunction is over only the single negated factor $\neg f$). $\dashv$

Example 86 *Let*

$$\begin{array}{rcll}\mathcal{G}_{86} & = & \{ & \rtimes\textbf{\textit{which}},\ \textbf{\textit{which girls}},\ \textbf{\textit{girls do}},\ \textbf{\textit{which girl}},\ \textbf{\textit{girl do}}, \\ & & & \textbf{\textit{do they}},\ \textbf{\textit{they think}},\ \textbf{\textit{think that}},\ \textbf{\textit{that they}}, \\ & & & \textbf{\textit{think were}},\ \textbf{\textit{were responsible}}, \\ & & & \textbf{\textit{think was}},\ \textbf{\textit{was responsible}},\ \textbf{\textit{responsible}}\ltimes \qquad \}\end{array}$$

$\mathcal{G}_{86}$ is an SL_2 description that defines L_{67} except that it does not require number agreement. To enforce number agreement we just convert it to LT_2 form and reject co-occurrence of ***girls*** *and* ***was*** *or* ***girl*** *and* ***were***:

$$\varphi_{67} = \bigwedge_{f_i \notin \mathcal{G}_{86}} [\neg f_i] \ \wedge\ \neg(\textbf{\textit{girls}} \wedge \textbf{\textit{was}})\ \wedge\ \neg(\textbf{\textit{girl}} \wedge \textbf{\textit{were}})$$

Then $L_{67} = L(\varphi_{67})$.

Theorem 87 **($\mathrm{SL} \subsetneq \mathrm{LT}$)** *The class of strictly local stringsets is a proper subset of the class of locally testable stringsets.*

$L_{67} \in \mathrm{LT} - \mathrm{SL}$.

4.6 Character of the locally testable sets

Theorem 88 (Local Test Invariance) *A stringset L is Locally Testable iff*

there is some k such that, for all strings x and y:

if $\quad F_k(\rtimes \cdot x \cdot \ltimes) = F_k(\rtimes \cdot y \cdot \ltimes)$

then $\quad x \in L \Leftrightarrow y \in L.$

Again, this is a characterization. If there is some k for which a given stringset is the union of some subset of the classes of strings that are equivalent in this sense then we can form a k-expression that is satisfied by all and only the strings in those classes.
(Exercise) Suppose L exhibits k-test invariance. Show how to construct a k-expression that defines L. Show that your construction produces a finite k-expression even though L may be infinite. Show that a string w satisfies your k-expression iff $w \in L$.

4.7 LT equivalence

Theorem 89 *For all $x, y \in \Sigma^*$ let*

$$x \equiv_k y \overset{def}{\Longleftrightarrow} F_k(\rtimes \cdot x \cdot \ltimes) = F_k(\rtimes \cdot y \cdot \ltimes).$$

Let $[x]_k \overset{def}{=} \{y \mid y \equiv_k x\}$. Then

1. *$\equiv_k$ **partitions** Σ^*:*
   - $\Sigma^* = \bigcup_{x \in \Sigma^*}[[x]_k]$
   - *for all $x, y \in \Sigma^*$ either $[x]_k = [y]_k$ or $[x]_k \cap [y]_k = \emptyset$*
2. *$\{[x]_k \mid x \in \Sigma^*\}$ is finite.*
3. *If φ is a k-expression and $w \equiv_k v$ then $w \models \varphi \Leftrightarrow v \models \varphi$.*
4. *$L \in \mathrm{LT}$ iff $L = \bigcup[S]$ for some k and some $S \subseteq \{[x]_k \mid x \in \Sigma^*\}$.*

Observation 90 *There are only finitely many LT_k stringsets for any given Σ and k.*

4.8 Disjunctive Normal Form

Lemma 91 *Every k-expression is equivalent, in the sense of defining the same set of strings, to a k-expression which is disjunction of conjunctions of **literals:** k-factors and negated k-factors.*

Proof The set of strings that are LT_k-equivalent to w is definable as:

$$[w]_k = L(\bigwedge_{f \in F_k(\rtimes w \ltimes)}[f] \wedge \bigwedge_{f \notin F_k(\rtimes w \ltimes)}[\neg f])$$

Since there are only finitely many k-factors over a given alphabet Σ this is a finite conjunction of literals.
Since every LT stringset is a finite union of LT-equivalence classes, every LT stringset is definable as a finite disjunction of such formulae. ⊣

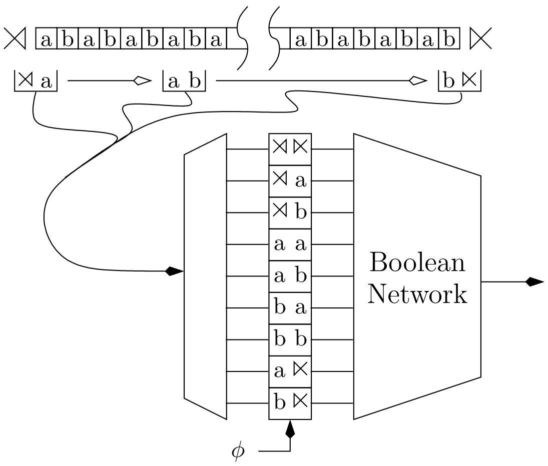
4.9 Cognitive interpretation of LT

- Any cognitive mechanism that can distinguish member strings from non-members of an LT_k stringset must be sensitive, at least, to the set of length k blocks of 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 length k blocks of events that occur at any prior point.

- Any cognitive mechanism that is sensitive *only* to the set of length k blocks of events in the presentation of a string will be able to recognize *only* LT_k stringsets.

4.10 LT Automata



4.11 Constructing LT_2 transition graphs

Given φ a 2-expression over Σ:
Vertices are states of the LT_2 automaton for φ:

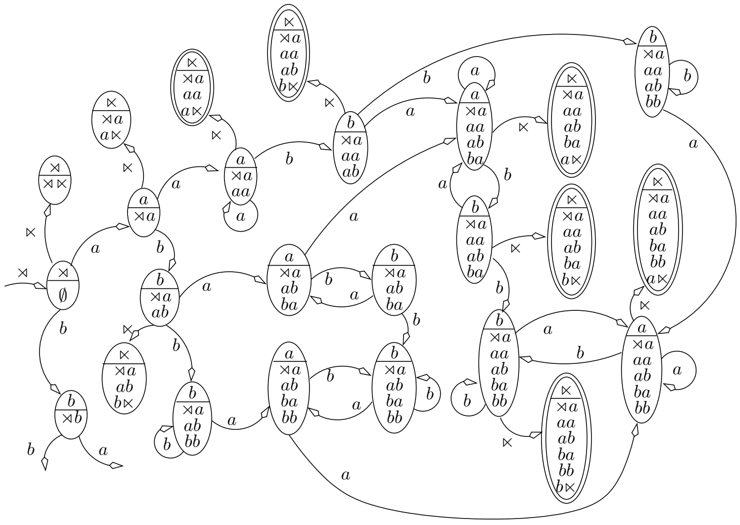
$$\langle \sigma_i, S_i \rangle \in (\Sigma \cup \{\rtimes, \ltimes\}) \times \mathcal{P}(F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)).$$

- Initial vertex: $\langle \rtimes, \emptyset \rangle$.

- For each vertex $\langle \sigma_i, S_i \rangle$ ($\sigma_i \neq \ltimes$), in turn, and each $\sigma \in \Sigma \cup \{\ltimes\}$,
  - if $\langle \sigma, S_i \cup \{\sigma_i \sigma\} \rangle$ is not yet in the vertex set, add it
  - in any case, add an edge labeled σ from $\langle \sigma_i, S_i \rangle$ to $\langle \sigma, S_i \cup \{\sigma_i \sigma\} \rangle$.

- For each vertex $\langle \ltimes, S_i \rangle$ if there is some w such that $w \models \varphi$ and $F_k(\rtimes w \ltimes) = S_i$, mark the vertex as accepting.

4.12 LT transition graphs



Theorem 92 *Emptiness of* LT *stringsets is decidable.*

Proof: exercise

Theorem 93 *Universality of* LT *stringsets is decidable.*

Proof: exercise

Theorem 94 *Finiteness of* LT *stringsets is decidable.*

Proof: exercise
(Exercise) Give a finite LT_2 stringset over $\{a, b\}$ which has no finite LT_2 superset.

4.13 Learnability of LT_k and LT

Theorem 95 *For all k, the class of LT_k stringsets is learnable in the limit.*

Proof: Exercise.

Theorem 96 *The class* LT*, as a whole, is not learnable in the limit from positive data.*

Since SL $\subseteq$ LT the counter-example witnessing that SL is not learnable works for LT as well.

4.14 Non-LT stringsets

One of the consequences of the characterization by local test invariance is that LT descriptions cannot distinguish between a single occurrence and multiple occurrences of main verbs.

Example 97

$$\text{my } \overbrace{\text{(father's)}}^{k-3} \text{ father resembled my } \overbrace{\text{(father's)}}^{k-3} \text{ father} \in E$$
$$\text{my } \overbrace{\text{(father's)}}^{k-3} \text{ father resembled my } \overbrace{\text{(father's)}}^{k-3} \text{ father resembled my } \overbrace{\text{(father's)}}^{k-3} \text{ father} \notin E$$

There is a pair of strings of this form for every $k \geq 3$ in which the symbol "father's" is iterated $k-3$ times. In each such pair the strings have exactly the same sets of k-factors. Hence, there can be no k-expression that correctly distinguishes each pair.

4.15 Relationship between locally k-testable classes

Theorem 98 (The LT hierarchy) *The classes of* LT_k *stringsets form a proper hierarchy:*

$$\mathrm{LT}_2 \subsetneq \mathrm{LT}_3 \subsetneq \cdots \mathrm{LT}_i \subsetneq \mathrm{LT}_{i+1} \subsetneq \cdots \subsetneq \mathrm{LT}.$$

That $\mathrm{LT}_i \subseteq \mathrm{LT}_{i+1}$ follows from the fact that every i-expression can be extended to an $(i+1)$-expression by extending the atomic formulae in the same way that one extends the k-factors of an SL_k definition to $k+1$ factors. That the inclusion is proper is witnessed, again, by the fact that $\{a^i\} \in \mathrm{LT}_{i+1} - \mathrm{LT}_i$.
(Exercise) Verify that $\{a^i\} \in \mathrm{LT}_{i+1} - \mathrm{LT}_i$.

4.16 Closure properties

The classes LT, as well as LT_k for each k, are closed under all Boolean operations by definition.

Lemma 99 (Non-closure under concatenation) *The class of locally testable stringsets is not closed under concatenation.*

Note that the prefixes that precede the main verb of the strings of Example 97 are definable by a k-expression, actually by a 2-expression, as are the suffixes that start at the main verb in the strings that have single main verbs. Since the main verb is permitted to occur only in the initial position of these suffixes, they can be distinguished from those containing more than one main verb by a k-expression. Thus, if the LT stringsets were closed under concatenation, we would be able to distinguish the first string (the concatenation of two LT stringsets) from the second. Hence, the example witnesses the fact that the LT stringsets are not, in general, closed under concatenation.

Lemma 100 (Non-closure under Kleene closure) *The class of locally testable stringsets is not closed under Kleene closure.*

The counterexample establishing non-closure of SL under Kleene closure ($(L_{aa})^*$ of Lemma 78) suffices to establish non-closure of LT under Kleene closure as well: all strings in the set $\{a^i \mid i \geq k\}$ have exactly the same set of k-factors and are, therefore, in the same class in the sense of k-test invariance. But some of these are of even length and should be included in $(L_{aa})^*$ and some odd and should not.

5 First-Order Languages for Strings

5.1 FO($\triangleleft^+$) (Strings)

$$\langle \mathcal{D}, \triangleleft, \triangleleft^+, P_\sigma \rangle_{\sigma \in \Sigma}$$

Variables ranging over positions in the strings: $\mathbb{X}_0 = \{x_0, x_1, \ldots\}$

Atomic formulae: $x \triangleleft y$, $x \triangleleft^+ y$, $x \approx y$, $P_\sigma(x)$, $x, y \in \mathbb{X}_0$

First-order Quantification: $(\exists x)[\varphi]$, $x \in \mathbb{X}_0$

5.2 Semantics of FO languages

Definition 101 (Free and Bound Variables) *A variable x occurring in a formula is* ***bound*** *iff it occurs within the scope (within the '$[\ldots]$') of a quantifier binding it: $(\exists x)$. A variable is* ***free*** *iff it is not bound.*

$\vec{x}$ denotes a sequence of variables $\langle x_0, x_1, \ldots, x_{n-1} \rangle$.
$\varphi(x_0, \ldots)$ denotes a formula with free variables among, but not necessarily including all of, $x_0, \ldots$.

Definition 102 (Assignments) *An* ***assignment*** *for a model $\mathcal{A}$ is a* ***partial function*** *(one that may be undefined for some elements of its domain) mapping variables in $\mathbb{X}_0$ to the domain of $\mathcal{A}$.*
If s is an assignment for $\mathcal{A}$ and a is in the domain of $\mathcal{A}$ then then $s[x \mapsto a]$ is the assignment that agrees with s on all variables except x to which it assigns a:

$$s[x \mapsto a](y) \overset{def}{=} \begin{cases} a & \textit{if } y = x, \\ s(y) & \textit{otherwise.} \end{cases}$$

Note that we do not require s to be undefined for x; it may be that $s[x \mapsto a]$ rebinds x to a.

Definition 103 (Satisfaction) *An assignment s* satisfies *a formula φ in a model $\mathcal{A}$ (denoted $\mathcal{A}, s \models \varphi$) iff one of the following holds:*

- *$\varphi =$ '$x \triangleleft y$', $s(x)$ and $s(y)$ are both defined and $s(y) = s(x) + 1$,*
- *$\varphi =$ '$x \triangleleft^+ y$', $s(x)$ and $s(y)$ are both defined and $s(x) < s(y)$,*
- *$\varphi =$ '$P_\sigma(x)$', $s(x)$ is defined and $s(x) \in P_\sigma$,*
- *$\varphi =$ '$x \approx y$', $s(x)$ and $s(y)$ are both defined and $s(x) = s(y)$,*
- *$\varphi =$ '$(\psi_1 \vee \psi_2)$' and either $\mathcal{A}, s \models \psi_1$ or $\mathcal{A}, s \models \psi_2$,*
- *$\varphi =$ '$(\neg\psi)$' and $\mathcal{A}, s \not\models \psi$, or*
- *$\varphi =$ '$(\exists x)[\psi]$' and, for some a in the domain of $\mathcal{A}$, $\mathcal{A}, s[x \mapsto a] \models \psi$.*

5.3 $\triangleleft$ is definable from $\triangleleft^+$

Let

$$x \blacktriangleleft y \overset{\text{def}}{=} (x \triangleleft^+ y \wedge \neg(\exists z)[x \triangleleft^+ z \wedge z \triangleleft^+ y])$$

Then, for all models $\mathcal{A}$ and assignments s

$$\mathcal{A}, s \models x \triangleleft y \Leftrightarrow \mathcal{A}, s \models x \blacktriangleleft y$$

5.4 Logical Sentences

Definition 104 (Sentences) *A (logical) sentence is a formula with no free variables.*
A model $\mathcal{A}$ satisfies a sentence (or not) independently of assignments:

$$\mathcal{A} \models \varphi$$

5.5 Models of Sentences

If $\mathcal{A} \models \varphi$ we say that the **model satisfies the sentence**, that the **sentence is true in the model** or that $\mathcal{A}$ is **a model of the sentence**.

Definition 105 (Models of a Set of Sentences) *The set of Σ-models which satisfy a given set of sentences Φ, the **models of** Φ, is denoted:*

$$\boldsymbol{Mod}(\Phi) \overset{\mathit{def}}{=} \{\mathcal{A} \mid \mathcal{A} \models \varphi, \textit{ for all } \varphi \in \Phi\}$$

We say that

$$\mathcal{A} \models \Phi \overset{\mathit{def}}{\Longleftrightarrow} \mathcal{A} \in \boldsymbol{Mod}(\Phi).$$

5.6 FO($\triangleleft^+$) definable stringsets

Definition 106 *$L \subseteq \Sigma^*$ is **FO($\triangleleft^+$) definable** iff there is a finite set of FO($\triangleleft^+$) sentences Φ (equivalently, a single sentence $\varphi(= \bigcap \Phi)$) such that $L = \boldsymbol{Mod}\,\Phi$.*

Theorem 107 *The fixed and universal recognition problems for the class of FO($\triangleleft^+$) definable stringsets are decidable.*

Proof: exercise.

5.7 Capabilities of FO Definitions over Strings

$\mathrm{NP}(x, y) \overset{\text{def}}{=}$ $\mathsf{my}(x) \wedge \mathsf{father}(y) \wedge x < y \wedge$
—NPs start with 'my' and end with 'father'...
$(\forall z)[(x < z \wedge z < y) \rightarrow \mathsf{father's}(z)]$
— ...with only 'father's' occurring in between

$(\exists x_1, x_2, x_3, x_5, x_5)[$

$\quad \mathrm{NP}(x_1, x_2) \wedge \mathsf{resembles}(x_3) \wedge \mathrm{NP}(x_4, x_5) \wedge x_2 < x_3 \wedge x_3 < x_4 \wedge$

—A sentence is an NP, 'resembles' and an NP, in order...

$\quad \neg(\exists y)[y < x_1 \vee (x_2 < y \wedge y < x_3) \vee (x_3 < y \wedge y < x_4) \vee \neg x_5 < y]$

$]$

— ...and nothing else

$$\text{my}\,\overbrace{(\text{father's})}^{\geq 0}\,\text{father resembled my}\,\overbrace{(\text{father's})}^{\geq 0}\,\text{father}$$

5.8 Logical Theories

Definition 108 (Theories of Models) *The theory of a Σ-model $\mathcal{A}$ (in a given language $L(\Sigma)$), denoted $\textbf{\textit{Th}}(\mathcal{A})$, is the set of all sentences of $L(\Sigma)$ that are satisfied by $\mathcal{A}$:*

$$\textbf{\textit{Th}}(\mathcal{A}) \overset{def}{=} \{\varphi \in L(\Sigma) \mid \mathcal{A} \models \varphi\}.$$

The theory of a set of models $\mathbb{A}$ is the set of all sentences satisfied by every member of $\mathbb{A}$:

$$\textbf{\textit{Th}}(\mathbb{A}) \overset{def}{=} \{\varphi \in L(\Sigma) \mid \mathcal{A} \models \varphi, \textit{ for all } \mathcal{A} \in \mathbb{A}\}.$$

Note that

$$\mathbf{Th}(\mathbb{A}) = \bigcap_{\mathcal{A} \in \mathbb{A}} [\mathbf{Th}(\mathcal{A})].$$

5.9 Validities, Logical Equivalence

A sentence of $L(\Sigma)$ is **valid** if it is satisfied in every Σ-model. The set of First-Order **validities** of a given language $L(\Sigma)$ is the FO theory of the set of all Σ-models.

Definition 109 (L-equivalent Models) *Two Σ-models $\mathcal{A}$ and $\mathcal{B}$ are **equivalent** with respect to a logical language L, denoted $\mathcal{A} \equiv_L \mathcal{B}$, iff $\textbf{\textit{Th}}(\mathcal{A}) = \textbf{\textit{Th}}(\mathcal{B})$.*

If L is First-Order, we say that $\mathcal{A}$ and $\mathcal{B}$ are **elementary equivalent**.

5.10 Consequences

Definition 110 (Logical Consequence) *Given two sentences φ and ψ in a language over Σ, we say that ψ is a **logical consequence** of φ (denoted $\varphi \models \psi$) if every Σ-model $\mathcal{A}$ which satisfies φ also satisfies ψ:*

$$\varphi \models \psi \quad \overset{def}{\Longleftrightarrow} \quad \mathcal{A} \models \varphi \Rightarrow \mathcal{A} \models \psi, \textit{ for all } \Sigma\textit{-models } \mathcal{A}.$$

A sentence φ is a logical consequence of a set of sentences Φ, all in $L(\Sigma)$, iff every Σ-model which satisfies all of the sentences in Φ also satisfies φ:

$$\Phi \models \varphi \quad \overset{def}{\Longleftrightarrow} \quad \mathcal{A} \models \Phi \Rightarrow \mathcal{A} \models \psi, \textit{ for all } \Sigma\textit{-models } \mathcal{A}.$$

We will denote the ***consequences*** *of a set of sentences* Φ *as:*

$$\boldsymbol{Cn}(\Phi) \overset{def}{=} \{\varphi \mid \Phi \models \varphi\}$$

Note that ψ will be in $\mathbf{Cn}(\Phi)$ iff whenever $\mathcal{A} \models \varphi$ for each $\varphi \in \Phi$ then $\mathcal{A} \models \psi$ as well and that, for each $\varphi \in \Phi$, it will be the case that ψ is in $\mathbf{Cn}(\varphi)$ iff whenever $\mathcal{A} \models \varphi$ then $\mathcal{A} \models \psi$. Hence, if $\varphi \in \Phi$ and ψ is in $\mathbf{Cn}(\varphi)$ then ψ will be in $\mathbf{Cn}(\Phi)$, that is:

$$\bigcup_{\varphi \in \Phi} [\mathbf{Cn}(\varphi)] \subseteq \mathbf{Cn}(\Phi).$$

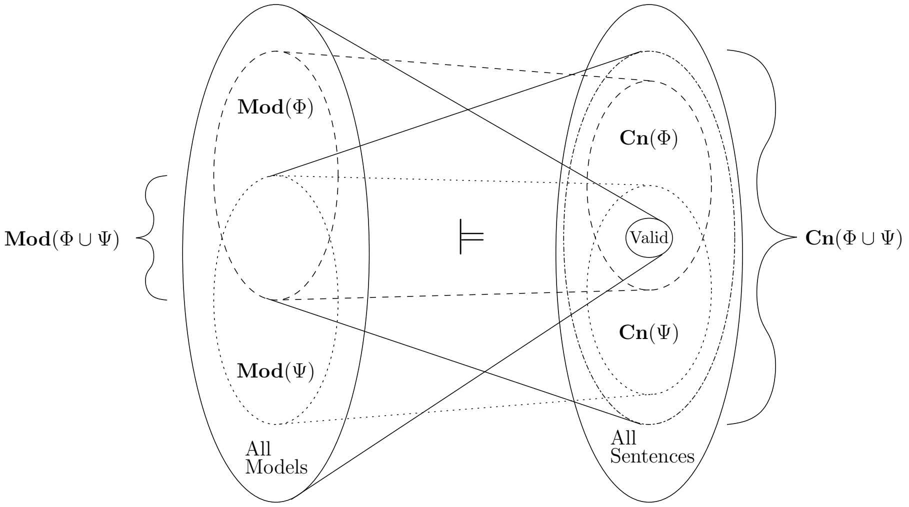
The consequences of the empty set of sentences, $\mathbf{Cn}(\emptyset)$, are those sentences that are satisfied by all models, since it is vacuously true that every sentence in $\emptyset$ is satisfied by all models. Thus $\mathbf{Cn}(\emptyset)$ in a language $L(\Sigma)$ is the theory of the set of all Σ-models, i.e., the validities of $L(\Sigma)$.

Definition 111 (Relative Consequence) *If* Φ *is a set of* $L(\Sigma)$ *sentences and* $\mathbb{C}$ *a class of* Σ*-models, then the* ***consequences of*** Φ ***relative to*** $\mathbb{C}$ *is*

$$\{\psi \mid \mathcal{A} \models \Phi \Rightarrow \mathcal{A} \models \psi, \quad \textit{for all}\ \mathcal{A} \in \mathbb{C}\}.$$

The class $\mathbb{C}$ is a meta-theoretic object; we put no restrictions whatsoever on how it might be defined. Most commonly, $\mathbb{C}$ will be the class of finite Σ-models. In general, the consequences of a set of sentences relative to the set of all finite models may be quite different from its consequences relative to the set all models. (In particular, note that any set of sentences that implies that the domain is infinite will be unsatisfiable relative to the set of finite models.)

5.11 Relationship between consequences and models



5.12 Logical Equivalence

Definition 112 (Logically Equivalent Sentences) *A pair of sentences* φ *and* ψ *are* ***logically equivalent*** *if they are consequences of each other, i.e.*

iff both $\varphi \models \psi$ and $\psi \models \varphi$. Similarly, a pair of sets of sentences Φ and Ψ are logically equivalent if both $\Phi \models \Psi$ and $\Psi \models \Phi$.

Φ and Ψ will be logically equivalent iff $\Psi \subseteq \mathbf{Cn}(\Phi)$ and $\Phi \subseteq \mathbf{Cn}(\Psi)$, i.e., iff $\mathbf{Cn}(\Phi) = \mathbf{Cn}(\Psi)$.

5.13 Theories as sets of sentences

Definition 113 (Formal Theory) *A **(formal) theory** is a set of sentences Φ which is closed under logical consequence:*

$$\Phi \models \psi \Rightarrow \psi \in \Phi.$$

*A theory Φ is **consistent** iff there is no sentence φ for which $\varphi, \neg\varphi \in \Phi$.*
*A theory Φ is **complete** iff for every sentence φ either $\varphi \in \Phi$ or $\neg\varphi \in \Phi$.*

Since no model satisfies both a sentence φ and its negation $\neg\varphi$ if any model satisfies a theory Φ then Φ is consistent. Conversely, if no model satisfies Φ then for all sentences φ it will be vacuously true that all models of Φ are also models of φ; the $\mathbf{Cn}(\Phi)$ will be the set of all sentences and will include, in particular, both φ and $\neg\varphi$ for every sentence φ. Hence, for our theories consistency coincides with satisfiability.
By the same reasoning, if a theory φ is inconsistent, i.e., if it contains any pair of sentences φ and $\neg\varphi$, then, because theories are closed under logical consequence, it must include *all* sentences ψ. Hence there is exactly one inconsistent theory: the set of all sentences in the language and we could define consistency of Φ by the existence of some sentence $\varphi \notin \Phi$.
The set of **valid** sentences of a language $L(\Sigma)$, since it is the set of consequences of $\emptyset$, is a subset of every theory. Moreover, it is non-empty, in fact, infinite: it includes such sentences as $(\forall x)[x \approx x]$ as well as all instances of the tautologies, e.g., $(\forall x)[P(x) \vee \neg P(x)]$. Thus no theory is empty; in fact every theory includes an infinite subset. Perhaps surprisingly, the set of validities also includes, for each φ and ψ related by consequence, an explicit statement of that relationship:

$$\text{If } \varphi \models \psi \text{ then `}\varphi \to \psi\text{'} \in \mathbf{Cn}(\emptyset).$$

This follows immediately from the definitions of → and of logical consequence.

5.14 Definable sets

Definition 114 (Definable Set of Models) *A set of Σ-models $\mathbb{A}$ is **definable** in a language $L(\Sigma)$ iff there is a finite set of sentences $\Phi \subseteq L_\Sigma$ such that $\mathbb{A} = \boldsymbol{Mod}(\Phi)$.*

Note that because we require our sets of axioms to be finite and because we interpret sets of sentences conjunctively a set of models is definable iff there is a single sentence $\psi = \bigwedge_{\varphi\in\Phi}[\varphi]$ such that $\mathbb{A} = \mathbf{Mod}(\psi)$.

$$\mathbf{Cn}(\Phi) = \mathbf{Th}(\mathbf{Mod}(\Phi)) = \mathbf{Th}(\mathbb{A}).$$

Definition 115 (Relative Definability) *A set of Σ-models $\mathbb{A}$ is **definable relative to a class** of Σ-models $\mathbb{C}$ in a language $L(\Sigma)$ iff there is a finite set of sentences $\Phi \subseteq L(\Sigma)$ such that $\mathbb{A} = \boldsymbol{Mod}(\Phi) \cap \mathbb{C}$.*

5.15 Character of FO-definable sets

Definition 116 (n-types) *Suppose $\mathcal{A}$ is a model with domain A.*
Let $a \in A$. The 1-type of a in $\mathcal{A}$ *is:*

$$\boldsymbol{tp}(\mathcal{A}, a) \overset{def}{=} \{\varphi(x_0) \mid \mathcal{A}, [x_0 \mapsto a] \models \varphi(x_0)\}.$$

Let $\langle a_0, \ldots, a_{n-1}\rangle \in A^n$. The n-type of $\langle a_0, \ldots, a_{n-1}\rangle$ in $\mathcal{A}$ *is:*

$$\boldsymbol{tp}(\mathcal{A}, \langle a_0, \ldots, a_{n-1}\rangle) \overset{def}{=} \\ \{\varphi(x_0, \ldots, x_{n-1}) \mid \mathcal{A}, [x_i \mapsto a_i] \models \varphi(x_0, \ldots, x_{n-1})\}.$$

*The set of n-**types realized in a model** $\mathcal{A}$ is the set of n-types of the n-tuples of its domain:*

$$S^n(\mathcal{A}) \overset{def}{=} \{\boldsymbol{tp}(\mathcal{A}, \vec{a}) \mid \vec{a} \in A^n\}.$$

Note that $\mathbf{tp}(\mathcal{A}, a)$ is just simplified notation for $\mathbf{tp}(\mathcal{A}, \langle a\rangle)$. Since types are just sets of formulae, they are either satisfied by an assignment in a given model or not. The tuples of points $\vec{b}$ which satisfy $\mathbf{tp}(\mathcal{A}, \vec{a})$ in $\mathcal{A}$ are just those for which $\mathbf{tp}(\mathcal{A}, \vec{b}) = \mathbf{tp}(\mathcal{A}, \vec{a})$. We will refer to the n-tuples from the domain of $\mathcal{A}$ which satisfy a given n-type as the tuples which **inhabit** the type in $\mathcal{A}$. The n-type of a tuple in $\mathcal{A}$ characterizes it up to logical indistinguishability. If $\mathbf{tp}(\mathcal{A}, a) = \mathbf{tp}(\mathcal{A}, b)$ then there is no First-Order formula that can distinguish the point $a \in A$ from the point $b \in A$. They are identical in terms of the properties they exhibit that we can specify in FO. This is the motivation for calling these *types*: the 1-types realized in a model categorize the members of its domain into classes of points that are indistinguishable wrt FO. Similarly, the n-types realized by a model categorize the n-tuples of points in the domain into classes of logically indistinguishable tuples.
We can generalize this across models. Note that $\mathcal{B}, [\vec{x} \mapsto \vec{b}] \models \mathbf{tp}(\mathcal{A}, \vec{a})$ iff the set of formulae satisfied by $[\vec{x} \mapsto \vec{b}]$ in $\mathcal{B}$ is exactly the same as the set satisfied by $[\vec{x} \mapsto \vec{a}]$ in $\mathcal{A}$, i.e., $\vec{b}$ has the same type in $\mathcal{B}$ as $\vec{a}$ has in $\mathcal{A}$:

$$\mathcal{B}, [\vec{x} \mapsto \vec{b}] \models \mathbf{tp}(\mathcal{A}, \vec{a}) \Leftrightarrow \mathbf{tp}(\mathcal{B}, \vec{b}) = \mathbf{tp}(\mathcal{A}, \vec{a}).$$

Note that $S^0(\mathcal{A})$ is the set of 0-types that are realized in $\mathcal{A}$. The 0-types contain formulae with no free variables, that is, they are sets of sentences. Since they depend only on the model, each model $\mathcal{A}$ realizes exactly one 0-type, the theory of $\mathcal{A}$:

$$\mathbf{tp}(\mathcal{A}, \langle\,\rangle) = \mathbf{Th}(\mathcal{A}).$$

$S^n(\mathcal{A})$ may well be infinite. Consider the model

$$\mathcal{A} = \langle \mathbb{N}, \mathrm{LT}\rangle,$$

where $\mathbb{N}$ is the set of natural numbers ($\{0, 1, 2, \ldots\}$) and $\mathrm{LT} = \{\langle i, j\rangle \mid i < j\}$. Suppose that $i \neq j$ for an arbitrary pair $i, j \in \mathbb{N}$. Without loss of generality, assume that $i < j$. For $j \geq 0$, let

$$\varphi_j(x_0) \overset{\mathrm{def}}{=} (\exists x_1, \ldots, x_j)[\bigwedge_{0 \leq l \neq m \leq j} [x_l \not\approx x_m] \wedge \bigwedge_{0 < l \leq j} [\mathrm{LT}(x_l, x_0)]\,]$$

The first conjunction says that there are j distinct points in the domain that are also distinct from the point assigned to x_0, the second says that they are all less than the point assigned to x_0. So if $\varphi_j(x_0)$ is true at some point, then the domain must include at least j points that are less than that point. Clearly then, $\varphi_j(x_0)$ will be satisfied in $\mathcal{A}$ by the assignment $[x_0 \mapsto j]$ but if $i < j$ it will not be satisfied by the assignment $[x_0 \mapsto i]$. In other words $\varphi_j(x_0) \in \mathbf{tp}(\mathcal{A}, j)$ but $\varphi_j(x_0) \notin \mathbf{tp}(\mathcal{A}, i)$ and the two types are distinct. Since there are infinitely many pairs $i < j \in \mathbb{N}$, there are infinitely many distinct types in $S^1(\mathcal{A})$. But even though all of these types are distinguishable by formulae in the language, no finite set of formulae will be able to distinguish all of them. Hence our actual definitions will not be able to resolve the n-tuples of our models as finely as the n-types do.

5.16 Quantifier Rank

Definition 117 (Quantifier Rank)

$$\boldsymbol{qr}(\varphi) \overset{def}{=} \begin{cases} 0 & \text{if } \varphi = \text{`}\sigma(\vec{x})\text{' or } \varphi = \text{`}x \approx y\text{'}, \\ \boldsymbol{qr}(\psi) & \text{if } \varphi = \text{`}(\neg\psi)\text{'}, \\ \max(\boldsymbol{qr}(\psi_1), \boldsymbol{qr}(\psi_2)) & \text{if } \varphi = \text{`}(\psi_1 \vee \psi_2)\text{'}, \\ \boldsymbol{qr}(\psi) + 1 & \text{if } \varphi = \text{`}(\exists x)[\psi]\text{'}. \end{cases}$$

5.17 (r, n)-types

Definition 118 ((r, n)-types) *Suppose $\mathcal{A}$ is a model with domain A, and $r \geq 0$.*
Let $a \in A$. The $(r, 1)$-type of a in $\mathcal{A}$ *is:*

$$\boldsymbol{tp}_r(\mathcal{A}, a) \overset{def}{=} \{\varphi(x_0) \mid \boldsymbol{qr}(\varphi(x_0)) = r \text{ and } \mathcal{A}, [x_0 \mapsto a] \models \varphi(x_0)\}.$$

Let $\langle a_0, \ldots, a_{n-1}\rangle \in A^n$. The (r, n)-type of $\langle a_0, \ldots, a_{n-1}\rangle$ in $\mathcal{A}$ *is:*

$$\begin{aligned} \boldsymbol{tp}_r(\mathcal{A}, \langle a_0, \ldots, a_{n-1}\rangle) &\overset{def}{=} \\ \{\varphi(x_0, \ldots, x_{n-1}) \quad | \quad & \boldsymbol{qr}(\varphi(x_0, \ldots, x_{n-1})) = r \text{ and} \\ & \mathcal{A}, [x_i \mapsto a_i] \models \varphi(x_0, \ldots, x_{n-1})\}. \end{aligned}$$

*The set of (r, n)-**types realized in a model** $\mathcal{A}$ is the set of (r, n)-types of the n-tuples of its domain:*

$$S^n_r(\mathcal{A}) \overset{def}{=} \{\boldsymbol{tp}_r(\mathcal{A}, \vec{a}) \mid \vec{a} \in A^n\}.$$

Since every formula of quantifier rank r can be converted to one of quantifier rank $r + 1$ via vacuous quantification, if two tuples $\vec{a}$ and $\vec{b}$ have the same (r, n)-type then they have the same (s, n)-type for all $s \leq r$:

$$\mathbf{tp}_r(\mathcal{A}, \vec{a}) = \mathbf{tp}_r(\mathcal{B}, \vec{b}) \;\Rightarrow\; \mathbf{tp}_{r-i}(\mathcal{A}, \vec{a}) = \mathbf{tp}_{r-i}(\mathcal{B}, \vec{b}),\, 0 \leq i \leq r$$

Again, $S^0_r(\mathcal{A})$ will contain a single type $\mathbf{tp}_r(\mathcal{A}, \langle\,\rangle)$, the set of sentences of quantifier rank r which are satisfied by $\mathcal{A}$. Let

$$\mathbf{Th}_r(\mathcal{A}) \overset{\text{def}}{=} \mathbf{tp}_r(\mathcal{A}, \langle\,\rangle).$$

We will often use $\mathbf{tp}^1_r(\mathcal{A})$ as simplified notation for $\mathbf{tp}_r(\mathcal{A}, \langle\,\rangle)$ and will extend the notation for logical equivalence with respect to a language to logical equivalence with respect to the fragment of that language with quantifier rank r:

$$\mathcal{A} \equiv_{1,r} \mathcal{B} \overset{\text{def}}{\Longleftrightarrow} \mathbf{tp}^1_r(\mathcal{A}) = \mathbf{tp}^1_r(\mathcal{B}).$$

Lemma 119

$$(\exists x_0)[\varphi(x_0)] \in \boldsymbol{tp}_r(\mathcal{A}, \langle\,\rangle) \Leftrightarrow \varphi(x_0) \in \boldsymbol{tp}_{r-1}(\mathcal{A}, a_0),\ \textit{for some } a_0 \in A$$

and, more generally,

$$\begin{aligned}&(\exists x_n)[\varphi(x_0, \ldots, x_n)] \in \boldsymbol{tp}_r(\mathcal{A}, \langle a_0, \ldots, a_{n-1}\rangle) \Leftrightarrow\\ &\quad \varphi(x_0 \ldots, x_n) \in \boldsymbol{tp}_{r-1}(\mathcal{A}, \langle a_0, \ldots, a_n\rangle),\ \textit{for some } \langle a_0, \ldots, a_n\rangle \in A^{n+1}\end{aligned}$$

This follows from the fact (by definition of $\models$) that:

$$\mathcal{A} \models (\exists x_0)[\varphi(x_0)] \Leftrightarrow \mathcal{A}, [x_0 \mapsto a_0] \models \varphi(x_0) \text{ for some } a_0 \in A$$

and, more generally,

$$\begin{aligned}&\mathcal{A}, s \models (\exists x_n)[\varphi(x_0, \ldots, x_n)] \Leftrightarrow\\ &\quad \mathcal{A}, s[x_n \mapsto a_n] \models \varphi(x_0 \ldots, x_n), \quad \text{for some } a_n \in A\end{aligned}$$

Lemma 120 $\boldsymbol{tp}_0(\mathcal{A}, \langle a_0, \ldots, a_{n-1}\rangle) = \boldsymbol{tp}_0(\mathcal{B}, \langle b_0, \ldots, b_{n-1}\rangle)$ *iff*

$$\begin{array}{rcll} a_i = a_j & \Leftrightarrow & b_i = b_j, & 0 \le i, j < n \\ \langle a_{i_1}, \ldots, a_{i_m}\rangle \in \rho^{\mathcal{A}} & \Leftrightarrow & \langle b_{i_1}, \ldots, b_{i_m}\rangle \in \rho^{\mathcal{B}}, & 0 \le i_1, \ldots, i_m \le n,\ \rho \in \mathcal{R}_m \end{array}$$

So the (r, n)-types that are realized by a model are determined by the $(r-1, n+1)$-types it realizes and *v.v.*. In particular, the (r, n)-types are determined by the $(0, n+r)$-types and the (unique) $(r, 0)$-type, the set of sentences of quantifier rank r in the theory of the model, is determined by the $(0, r)$-types it realizes, the sets of quantifier-free formulae that are satisfied by the r-tuples of points from the model's domain. And these, in turn, are determined by the atomic formulae satisfied by those r-tuples.
Consequently, two models $\mathcal{A}$ and $\mathcal{B}$ will satisfy the same set of sentences of quantifier rank r, i.e., $\mathbf{tp}^1_r(\mathcal{A}) = \mathbf{tp}^1_r(\mathcal{B})$, equivalently, $S^0_r(\mathcal{A}) = S^0_r(\mathcal{B})$, iff they realize the same $(0, r)$-types, $S^r_0(\mathcal{A}) = S^r_0(\mathcal{B})$, i.e., iff for every r-tuple of points in the domain of one of the models there is a corresponding r-tuple of points in the domain of the other that satisfies exactly the same set of quantifier free formulae, hence the same set of atomic formulae (each in their own model).

Lemma 121 *The number of logically distinct formulae of quantifier rank r with n free variables in any relational First-Order language L over a finite signature is finitely bounded.*

Corollary 122 *The number of distinct (r, n)-types realizable in any class of relational models is finite.*

Since there are only finitely many logically distinct formulae of quantifier rank r with n free variables, there is some finite sequence of formulae

$$\varphi_1(\vec{x}),\ \varphi_2(\vec{x}),\ \ldots,\ \varphi_m(\vec{x})$$

which enumerates the formulae of $\mathbf{tp}_r(\mathcal{A},\vec{a})$ in the sense that every formula in the type is logically equivalent to one of the $\varphi_i(\vec{x})$. We can characterize the entire type, then, with the (finite) disjunction of these:

$$\chi_r^{\mathcal{A},\vec{a}}(\vec{x}) = \bigvee_{0<i\leq m} [\varphi_i(\vec{x})].$$

Note that $\mathbf{qr}(\chi_r^{\mathcal{A},\vec{a}}(\vec{x})) = r$ and it has n free variables, hence $\chi_r^{\mathcal{A},\vec{a}}(\vec{x}) \in \mathbf{tp}_r(\mathcal{A},\vec{a})$.

Corollary 123 *For every model $\mathcal{A}$, every n-tuple $\vec{a}$ of points in the domain of $\mathcal{A}$, $n \geq 0$, and $r \geq 0$, there is a single FO formula $\chi_r^{\mathcal{A},\vec{a}}(\vec{x})$ which characterizes $\boldsymbol{tp}_r(\mathcal{A},\vec{a})$:*

$$\mathcal{B}, [\vec{x} \mapsto \vec{b}] \models \chi_r^{\mathcal{A},\vec{a}}(\vec{x}) \text{ iff } \boldsymbol{tp}_r(\mathcal{B},\vec{b}) = \boldsymbol{tp}_r(\mathcal{A},\vec{a})$$

Moreover $\chi_r^{\mathcal{A},\vec{a}}(\vec{x}) \in \boldsymbol{tp}_r(\mathcal{A},\vec{a})$

Hence, we can use $\mathbf{tp}_r(\mathcal{A},\vec{a})$ and $\chi_r^{\mathcal{A},\vec{a}}(\vec{x})$ more or less interchangeably. We will refer to $\chi_r^{\mathcal{A},\vec{a}}(\vec{x})$ as the **characteristic formula** of the type $\mathbf{tp}_r(\mathcal{A},\vec{a})$.

Theorem 124 *A property of models P, a subset of the set of all models over a relational signature Σ, is definable in $L^1(\Sigma)$ iff there is some $r \geq 0$ such that, for all Σ-models $\mathcal{A}$ and $\mathcal{B}$*

$$\mathcal{A} \equiv_{1,r} \mathcal{B} \quad \Rightarrow \quad \mathcal{A} \in P \Leftrightarrow \mathcal{B} \in P.$$

5.18 Concatenation and Types

Lemma 125

$$\begin{array}{ll} \textit{If} & \boldsymbol{tp}_r(\mathcal{A},\vec{a}) = \boldsymbol{tp}_r(\mathcal{B},\vec{b}) \text{ and } \boldsymbol{tp}_r(\mathcal{C},\vec{c}) = \boldsymbol{tp}_r(\mathcal{D},\vec{d}) \\ \textit{then} & \qquad \boldsymbol{tp}_r(\mathcal{A}\cdot\mathcal{C},\vec{a}\cdot\vec{c}) = \boldsymbol{tp}_r(\mathcal{B}\cdot\mathcal{D},\vec{b}\cdot\vec{d}). \end{array}$$

Corollary 126

$$\text{If } \mathcal{A} \equiv_{1,r} \mathcal{B} \text{ and } \mathcal{C} \equiv_{1,r} \mathcal{D} \text{ then } \mathcal{A}\cdot\mathcal{C} \equiv_{1,r} \mathcal{B}\cdot\mathcal{D}.$$

5.19 LT and FO

It should be clear that there are First-Order formulae that pick out the set of strings in which any given k-factor occurs. We need only to posit the existence of a contiguous block of k positions in the string at which the symbols are those given. Given the factor $f = a_0 \cdots a_{n-1}$ this would be:

$$\varphi_f \triangleq (\exists x_0, \ldots, x_{n-1})[\bigwedge_{0\leq i<(n-1)} [x_i \triangleleft x_{i+1}] \wedge \bigwedge_{0\leq i<n} [P_{a_i}(x_i)]\,].$$

We can capture the role of '$\rtimes$' and '$\ltimes$' as well by extending the formula with

$$\cdots \wedge \neg(\exists y)[y \triangleleft x_0] \quad \text{or} \quad \cdots \wedge \neg(\exists y)[x_{n-1} \triangleleft y], \quad \text{respectively.}$$

Since we have the same repertoire of logical connectives in both languages, we can convert any given k-expression into a First-Order formula that picks out exactly the set of strings that satisfy that expression. Hence the LT stringsets will all be First-Order definable.

5.20 Concatenation of FO($\triangleleft^+$)-definable stringsets

Theorem 127 *The class of FO-definable sets of $\triangleleft^+$-string models is closed under concatenation.*

Suppose L_1 and L_2 are both defined by First-Order formulae. Suppose further, without loss of generality, that the formulae only reference the beginning and end of the string via two variables $x_{\min}$ and $x_{\max}$, so the formulae are of the general form:

$$(\exists x_{\min}, x_{\max})[\neg(\exists y)[y \triangleleft x_{\min} \vee x_{\max} \triangleleft y] \wedge \varphi_i(x_{\min}, x_{\max})]$$

where the actual work of the definition is done by $\varphi_i(x_{\min}, x_{\max})$.
To convert this into a formula (with the same general form) that picks out all and only those strings that consist of a string from L_1 concatenated with one from L_2 we can combine these into

$$\begin{aligned}
&(\exists z_{\min}, z_{\max})[\\
&\qquad \neg(\exists y)[y \triangleleft z_{\min} \vee z_{\max} \triangleleft y] \wedge\\
&\qquad (\exists z_1, z_2)[z_{\min} \leq z_1 \wedge z_1 \triangleleft z_2 \wedge z_2 \leq z_{\max} \wedge \varphi_1(z_{\min}, z_1) \wedge \varphi_2(z_2, z_{\max})]\ ]
\end{aligned}$$

The second line of the formula is the $\varphi_{L_1 \cdot L_2}$. It posits the existence of adjacent positions z_1 and z_2 in the string with φ_1 holding from $z_{\min}$ through z_1, inclusive, and φ_2 holding from z_2 through $z_{\max}$. Thus, under the assumption that φ_1 and φ_2 pick out the strings in L_1 and L_2, respectively, the set of strings satisfying the formula will be exactly those that are formed from the concatenation of strings from L_1 and L_2.

5.21 Locally Testable with Order

Definition 128 (Locally Testable with Order (LTO)) *The language of **ordered k-expressions** is constructed in the same way as the language of k-expressions with the addition of the concatenation operator:*

- *if φ and ψ are ordered k-expressions, then $\varphi \bullet \psi$ is an ordered k-expression, with*
$$w \models \varphi \bullet \psi \overset{def}{\iff} w = w_1 \cdot w_2, \quad w_1 \models \varphi \text{ and } w_2 \models \psi.$$

*A stringset L is **Locally k-Testable with Order (**LTO$_k$**)** iff there is an ordered k-expression φ such that*

$$L = L(\varphi) = \{w \mid w \models \varphi,\ w \text{ finite}\}.$$

*A stringset is **Locally Testable with Order (**LTO**)** iff it is Locally k-Testable with Order for some k.*

5.22 FO($\triangleleft^+$) and LTO

Theorem 129 *A stringset is First-Order definable relative to the class of finite $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ models (in FO($\triangleleft^+$)) iff it is LTO.*

If $\varepsilon \in L(\varphi)$ then $L(\varphi) = L(\varphi \wedge (\exists x)[x \approx x]) \cup L((\forall x)[x \not\approx x])$.
Assume, then, $\varepsilon \notin L(\varphi)$.
Assume, also, that $\triangleleft$ does not occur, as it is FO definable from $\triangleleft^+$.

5.23 Basis: Quantifier rank 0

φ		$L(\varphi)$	2-expression
$\min \approx \min$,	$\max \approx \max$,	$\{w \in \Sigma^* \mid \lvert w\rvert \geq 1\}$	$\neg(\rtimes\ltimes)$
$\min \approx \max$,		$\{w \in \Sigma^* \mid \lvert w\rvert = 1\}$	$\bigvee_{\sigma,\gamma\in\Sigma}[\rtimes\sigma \bigwedge[\neg(\sigma\gamma)]\ ]$
$\min \triangleleft^+ \min$,	$\max \triangleleft^+ \max$,	$\emptyset$	$(\rtimes\ltimes) \wedge \neg(\rtimes\ltimes)$
$\min \triangleleft^+ \max$,		$\{w \in \Sigma^* \mid \lvert w\rvert \geq 2\}$	$\bigvee_{\sigma,\gamma\in\Sigma}[\sigma\gamma]$
$P_\sigma(\min)$,		$\{\sigma\} \cdot \Sigma^*$	$\rtimes\sigma$
$P_\sigma(\max)$		$\Sigma^* \cdot \{\sigma\}$	$\sigma\ltimes$

5.24 Induction step

$\varphi = (\exists x)[\psi(x)]$, where $\mathbf{qr}(\varphi)$ is $k+1$.

$$w \models (\exists x)[\psi(x)] \Leftrightarrow w, [x \mapsto p] \models \psi(x)$$

$$\underbrace{\sigma_0 \cdots \sigma_p}_{w_l} \cdot \underbrace{\sigma_{p+1} \cdots \sigma_{n-1}}_{w_r}$$

Let

$$S_\varphi \stackrel{\text{def}}{=} \{\langle \chi_k^{w_l,\langle p\rangle}(x), \chi_k^{w_r,\langle\,\rangle}\rangle \mid p = \max^{w_l} \text{ and } w_l \cdot w_r, [x \mapsto p] \models \psi(x)\}$$

$$S'_\varphi \stackrel{\text{def}}{=} \{\langle \chi_k^{w_l,\langle p\rangle}(x)[\max^{w_l}/x], \chi_k^{w_r,\langle\,\rangle}\rangle \mid \langle \chi_k^{w_l,\langle p\rangle}(x), \chi_k^{w_r,\langle\,\rangle}\rangle \in S_\varphi\}$$

$$L(\varphi) = \bigcup_{\langle\varphi_l,\varphi_r\rangle\in S'_\varphi} [L(\varphi_l) \cdot L(\varphi_r)]$$

Corollary 130 *Every FO($\triangleleft^+$) definable stringset is the union of a finite set of concatenations of* SL$_2$ *stringsets of the form:*

$$\begin{array}{l} \{w \in \Sigma^* \mid |w| \geq 1\} \\ \{\sigma\} \\ \emptyset \\ \{\varepsilon\} \end{array}$$

$$\begin{array}{l} \{w \in \Sigma^* \mid |w| = 1\} = \bigcup_{\sigma\in\Sigma}[\{\sigma\}] \\ \{w \in \Sigma^* \mid |w| \geq 2\} = \{w \in \Sigma^* \mid |w| = 1\} \cdot \{w \in \Sigma^* \mid |w| \geq 1\} \\ \{\sigma \cdot w \mid w \in \Sigma^*\} = \{\sigma\} \cdot \{w \in \Sigma^* \mid |w| \geq 1\} \\ \{w \cdot \sigma \mid w \in \Sigma^*\} = \{w \in \Sigma^* \mid |w| \geq 1\} \cdot \{\sigma\} \end{array}$$

Proof The construction of the proof captures **Mod** φ recursively as a finite union of concatenations of pairs stringsets which may be, themselves, finite unions of concatenations of pairs of stringsets, etc., with the base cases being of the six forms given in the middle column. These can all be constructed using concatenation an union from the first three forms given in the corollary. The fourth form $\{\varepsilon\}$ is needed only if $\varepsilon \in L$. If Since concatenation distributes over union, this resolves into a finite union of concatenations of stringsets of the form given. ⊣

Theorem 131 *Emptiness of FO($\triangleleft^+$) definable stringsets is decidable*

Proof Suppose L is FO($\triangleleft^+$) definable. From Corollary 130 L is equal to a finite union of concatenations of SL_2 stringsets. Thus L will be empty iff every one of these concatenations is empty. A concatenation of SL_2 stringsets will be empty iff any one of the individual stringsets is empty. Thus emptiness of FO($\triangleleft^+$) stringsets reduces to emptiness of SL_2 stringsets which we know to be decidable. $\dashv$

Since the SL_2 sets are not arbitrary but are just one of the four forms given in the corollary, we don't actually need to use the emptiness algorithm for SL_2 definitions. The concatenation will be empty iff one of the concatenated sets is $\emptyset$.

Theorem 132 *Finiteness of FO($\triangleleft^+$) definable stringsets is decidable.*

Proof: exercise

Theorem 133 *Universality of FO($\triangleleft^+$) definable stringsets is decidable.*

Proof: exercise

5.25 Cognitive interpretation of FO($\triangleleft^+$)

- Any cognitive mechanism that can distinguish member strings from nonmembers of an FO($\triangleleft^+$) stringset must be sensitive, at least, to the sets of length k blocks of events, for some fixed k, that occur in the presentation of the string when it is factored into segments, up to some fixed number, on the basis of those sets with distinct criteria applying to each segment..
- (More on the interpretation of FO($\triangleleft^+$) shortly.)
- Any cognitive mechanism that is sensitive *only* to the sets of length k blocks of events in the presentation of a string once it has been factored in this way will be able to recognize *only* LTO stringsets.

5.26 FO($\triangleleft^+$) is not learnable

Theorem 134 *FO($\triangleleft^+$) is not learnable in the limit from positive data.*

Since LT $\subseteq$ FO($\triangleleft^+$).

Theorem 135 LTO_k *is not learnable in the limit from positive data for any* $k \geq 2$.

Proof: (exercise)

5.27 Star-Free Sets

Definition 136 (Star-Free Set) *The class of **Star-Free Sets** (*SF*) is the smallest class of stringsets satisfying:*

- $\emptyset \in \mathrm{SF}$, $\{\varepsilon\} \in \mathrm{SF}$, *and* $\{\sigma\} \in \mathrm{SF}$ *for each* $\sigma \in \Sigma$.
- *If* $L_1, L_2 \in \mathrm{SF}$ *then:* $L_1 \cdot L_2 \in \mathrm{SF}$,
  $L_1 \cup L_2 \in \mathrm{SF}$,
  $\overline{L_1} \in \mathrm{SF}$.

Theorem 137 (McNaughton and Papert) *A set of strings is Locally Testable with Order (*LTO*) iff it is Star-Free.*

Corollary 138 (McNaughton and Papert) *A set of strings is First-order definable relative to the class of finite $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ models iff it is Star-Free.*

SF $\Rightarrow$ LTO:
Each base case is SL$_2$. LTO is closed under union, concatenation and complement.
LTO $\Rightarrow$ FO($\triangleleft^+$): Theorem 129
FO($\triangleleft^+$) $\Rightarrow$ SF:
$\{w \in \Sigma^* \mid |w| \geq 1\} = \overline{\{\varepsilon\}}$

5.28 Non-counting stringsets

Theorem 139 (McNaughton and Papert) *A stringset L is Star-Free iff it is **non-counting**, that is, iff there exists some $n > 0$ such that, for all strings u, v, w over Σ,*

> *if uv^nw occurs in L*
>
> *then $uv^{n+i}w$, for all $i \geq 1$, occurs in L as well.*

Corollary 140 (McNaughton and Papert) *A set of strings is First-Order definable relative to the class of finite $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ models (in FO($\triangleleft^+$)) iff it is non-counting.*

5.29 A non-counting stringset

$$\begin{array}{ll} \text{my }\overbrace{\text{father's}}\,\overbrace{\text{father's}}\text{ father resembled my father} & \in L \\ \hline \text{my }\underbrace{\text{father's}}\,\underbrace{\text{father's}}\;\underbrace{\text{(father's)}}_{\geq 1}\text{ father resembled my father} & \in L \end{array}$$

5.30 A non-FO($\triangleleft^+$) definable stringset

People left

People left by people left
People whom people left left

People left by people left by people left
People whom people whom people left left left

$\vdots$

$$\overbrace{\text{People whom people whom}\ldots\text{people whom people}}^{n\,\text{`people}'s}\,\overbrace{\text{left}\ldots\text{left left}}^{n\,\text{`left}'s}$$

$$\overbrace{\text{People people people}\ldots}^{n}\,\overbrace{\text{left left left}}^{n}$$

A more abstract (and formally tighter) example is the set of strings which are of even length. For concreteness, we can consider those just over the singleton alphabet $\{a\}$. For any n this set will include the string $a^n \cdot a^n \cdot \varepsilon$ but not $a^n \cdot a^{n+1} \cdot \varepsilon$. Hence, the set is not non-counting and, by Corollary 140 not FO($\triangleleft^+$) definable. Similarly, Even-B, the set of strings over $\{A, B\}$ in which the number of 'B's which occur is even is not FO($\triangleleft^+$) definable.

5.31 FO($\triangleleft$) definable stringsets

$\langle \mathcal{D}, \triangleleft, P_\sigma \rangle_{\sigma \in \Sigma}$

First-order Quantification (over positions in the strings)

FO($\triangleleft$)$\subseteq$FO($\triangleleft^+$).

Theorem 141 *The fixed and universal recognition problems for FO($\triangleleft$) definable stringsets are decidable.*

Theorem 142 *Emptiness, finiteness and universality of FO($\triangleleft$) definable stringsets are decidable.*

All because FO($\triangleleft$)$\subseteq$FO($\triangleleft^+$).

Example 143

$$\begin{array}{l}(\exists x_{0,0} \ldots, x_{0,k-1}, \ \ldots, x_{t-1,0}, \ldots, x_{t-1,k-1})[\\ \qquad \varphi_f(x_{0,0}, \ldots, x_{0,k-1}) \wedge \cdots \wedge \varphi_f(x_{t-1,0}, \ldots, x_{t-1,k-1}) \wedge\\ \qquad \text{—}f \textit{ occurs at each of the } [x_{i,0}, \ldots, x_{i,k-1}]\\ \qquad \bigwedge_{0 \leq i \neq j < t}[x_{i,0} \not\approx x_{j,0}]\\ \qquad \textit{—Each occurrence starts at a different position}\\ \quad]\end{array}$$

picks out the set of strings in which f occurs at least t times. The negation of (143) picks out the set of strings in which f occurs fewer than t times. Putting these together, we can build sentences that pick out the strings in which the number of occurrences of a given k-factor f falls in any fixed range $[n \ldots m]$ or any range greater than some fixed threshold $[n \ldots)$ or any finite combination of these.

5.32 Locally Threshold Testable stringsets

Definition 144 (Locally Threshold Testable) *A set L is* ***Locally Threshold Testable*** *(LTT) iff there is some k and t such that, for all $w, v \in \Sigma^*$:*

if for all $f \in F_k(\rtimes \cdot w \cdot \ltimes) \cup F_k(\rtimes \cdot v \cdot \ltimes)$ either $|w|_f = |v|_f$ or both $|w|_f \geq t$ and $|v|_f \geq t$,

then $w \in L \iff v \in L$.

Theorem 145 (Thomas) *A set of strings is First-order definable relative to the class of finite $\langle \mathcal{D}, \triangleleft, P_\sigma \rangle_{\sigma \in \Sigma}$ models (in FO($\triangleleft$)) iff it is Locally Threshold Testable.*

Example 146 *The stringset of Example 97 is* LTT *(as well as non-counting, hence* LTO*) but not, as we saw,* LT*. To see that it is* FO($\triangleleft$) *definable, note that one can restrict the strings to exactly one occurrence of* 'resembles' *with:*

$$(\exists x)[\mathsf{resembles}(x) \wedge (\forall y)[\mathsf{resembles}(y) \rightarrow y \approx x]].$$

Theorem 147 *FO($\triangleleft$)$\subsetneq$FO($\triangleleft^+$).*

Proof An example of a stringset that is LTO but not LTT is B-before-C, the set of all strings over $\{a, b, c\}$ in which b and c occur exactly once each with the b occurring before the c:

$$\{a^i ba^j ca^k \mid 0 \leq i, j, k\}.$$

As we just saw, restricting the strings to those in which b and c each occur exactly once is within the capability of FO($\triangleleft$). It is the requirement that the b precede the c that requires $\triangleleft^+$. To see this, consider the pair of strings:

$$a^k ba^k ca^k \qquad \text{and} \qquad a^k ca^k ba^k$$

These have exactly the same number of occurrences of each of their k-factors but the first is in the intended stringset while the second is not. Hence, regardless of the value of the threshold, this stringset is not LTT. To see that it is LTO note that it is the concatenation of three stringsets each of which are LT:

$$\{a^i b \mid 0 \leq i\} \cdot \{a^j c \mid 0 \leq j\} \cdot \{a^k \mid 0 \leq k\}$$

⊣

(Exercise) Show that $\triangleleft^+$ is not definable from $\triangleleft$.

5.33 Cognitive interpretation of FO($\triangleleft$)

- Any cognitive mechanism that can distinguish member strings from nonmembers of an FO($\triangleleft$) stringset must be sensitive, at least, to the multiplicity of the length k blocks of events, for some fixed k, that occur in the presentation of the string, distinguishing multiplicities only up to some fixed threshold t.

- If the strings are presented as sequences of events in time, then this corresponds to being able count up to some fixed threshold.

- Any cognitive mechanism that is sensitive *only* to the multiplicity, up to some fixed threshold, of the length k blocks of events in the presentation of a string will be able to recognize *only* FO($\triangleleft$) stringsets.

5.34 Cognitive interpretation of FO($\triangleleft^+$) (reprise)

- Any cognitive mechanism that can distinguish member strings from nonmembers of an FO($\triangleleft^+$) stringset, when the strings are presented as sequences of events in time, must be sensitive, at least, to the multiplicity of events, counting up to some fixed threshold with the counters being reset some fixed number of times based on those multiplicities.

5.35 FO($\triangleleft$) is not learnable

Theorem 148 *FO($\triangleleft$) is not learnable*

Since it extends LT.

Theorem 149 $\mathrm{LTT}_{k,t}$ *is not learnable if either* k *or* t *is not fixed.*

Since one can define L_{a^*} and, using either two $(i+1)$-factors or $i-1$ occurrences of a 2-factor, one can define $L_{a^{\leq i}}$ for each i.

6 Monadic Second-Order Languages for Strings

6.1 MSO (Strings)

$$\langle \mathcal{D}, \triangleleft, \triangleleft^+, P_\sigma \rangle_{\sigma\in\Sigma}$$

Variables ranging over positions in the strings: $\mathbb{X}_0 = \{x_0, x_1, \ldots\}$

Variables ranging over sets of positions in the strings: $\mathbb{X}_1 = \{X_0, X_1, \ldots\}$

Atomic formulae: $x \triangleleft y$, $x \triangleleft^+ y$, $x \approx y$, $P_\sigma(x)$, $X \approx Y$, $X(x)$, $\quad x, y \in \mathbb{X}_0,\ X \in \mathbb{X}_1$

First-order Quantification: $(\exists x)[\varphi],\ x \in \mathbb{X}_0$

Second-order Quantification: $(\exists X)[\varphi],\ X \in \mathbb{X}_1$

6.2 Semantics of MSO languages

Definition 150 (MSO Assignments) *An* ***MSO assignment*** *for a model* $\mathcal{A}$ *is a partial function mapping variables in* $\mathbb{X}_0$ *to the domain of* $\mathcal{A}$ *and variables in* $\mathbb{X}_1$ *to subsets of the domain of* $\mathcal{A}$.
If s *is an assignment for* $\mathcal{A}$ *and* S *is a subset of the the domain of* $\mathcal{A}$ *then then* $s[X \mapsto S]$ *is the assignment that agrees with* s *on all FO and MSO variables except* X *to which it assigns* S*:*

$$s[X \mapsto S](Y) \overset{def}{=} \begin{cases} S & \text{if } Y = X, \\ s(Y) & \text{otherwise.} \end{cases}$$

Definition 151 (Satisfaction) $\mathcal{A}, s \models \varphi$ *iff one of the following holds:*

- $\varphi =$ '$x \triangleleft y$', $s(x)$ *and* $s(y)$ *are both defined and* $s(y) = s(x) + 1$,
- $\varphi =$ '$x \triangleleft^+ y$', $s(x)$ *and* $s(y)$ *are both defined and* $s(x) < s(y)$,
- $\varphi =$ '$x \approx y$', $s(x)$ *and* $s(y)$ *are both defined and* $s(x) = s(y)$,
- $\varphi =$ '$P_\sigma(x)$', $s(x)$ *is defined and* $s(x) \in P_\sigma$,
- $\varphi =$ '$X \approx Y$', $s(X)$ *and* $s(Y)$ *are both defined and* $s(X) = s(Y)$,
- $\varphi =$ '$X(x)$', $s(X)$ *and* $s(x)$ *are both defined and* $s(x) \in s(X)$,
- $\varphi =$ '$(\psi_1 \vee \psi_2)$' *and either* $\mathcal{A}, s \models \psi_1$ *or* $\mathcal{A}, s \models \psi_2$,
- $\varphi =$ '$(\neg\psi)$' *and* $\mathcal{A}, s \not\models \psi$,
- $\varphi =$ '$(\exists x)[\psi]$' *and, for some* a *in the domain of* $\mathcal{A}$, $\mathcal{A}, s[x \mapsto a] \models \psi$, *or*
- $\varphi =$ '$(\exists X)[\psi]$' *and, for some subset* S *of the domain of* $\mathcal{A}$, $\mathcal{A}, s[X \mapsto S] \models \psi$.

6.3 Recognition for MSO definable stringsets

Theorem 152 *The fixed and universal recognition problems for MSO definable sets is decidable.*

Recognition = satisfaction. Satisfaction is decidable.

6.4 B-before-C is MSO definable

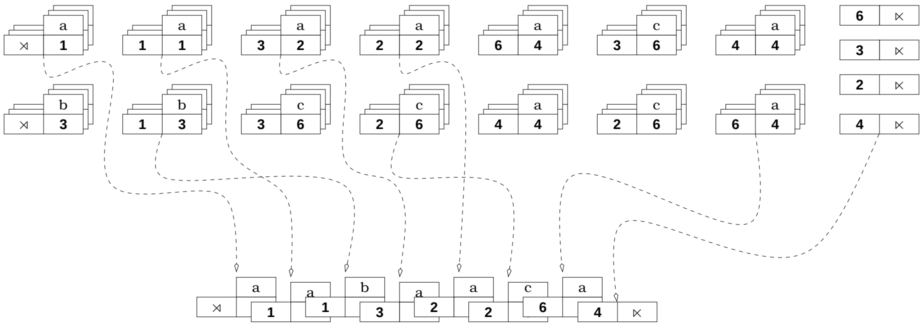
$$\begin{array}{ll}
(\exists X_0)[& (\forall x)[X_0(x) \leftrightarrow ((\forall y)[\neg y \lhd x] \vee (\exists y)[X_0(y) \wedge A(y) \wedge y \lhd x])\ ]\ ] \wedge \\
 & \text{—}X_0 \text{ contains the minimum point and} \\
 & \qquad \text{everything up to the first non-`}a\text{' which follows it} \\
(\exists X_1)[& (\exists x)[B(x) \wedge (\forall y)[B(y) \rightarrow y \approx x] \wedge X_1(x) \wedge (\forall y)[y \lhd x \rightarrow \neg X_1(y)]\ ] \\
 & \text{—There is exactly one `}b\text{' and it is the minimum point in } X_1 \\
 & (\exists x)[C(x) \wedge (\forall y)[C(y) \rightarrow y \approx x] \wedge X_1(x) \wedge (\forall y)[x \lhd y \rightarrow \neg X_1(y)]\ ] \\
 & \text{—There is exactly one `}c\text{' and it is the maximum point in } X_1 \\
 & (\forall x)[\ \ X_1(x) \leftrightarrow (\quad B(x) \vee C(x) \vee \\
 & \qquad\qquad\qquad \text{—}X_1 \text{ contains the `}b\text{' and the `}c\text{'}\ldots \\
 & \qquad\qquad\qquad (\exists y)[X_1(y) \wedge (A(y) \vee B(y)) \wedge y \lhd x] \vee \\
 & \qquad\qquad\qquad \text{—}\ldots\text{and the `}a\text{'s following the `}b\text{'}\ldots \\
 & \qquad\qquad\qquad (\exists y)[X_1(y) \wedge (A(y) \vee C(y)) \wedge x \lhd y] \\
 & \qquad\qquad\qquad \text{—}\ldots\text{and the `}a\text{'s preceding the `}c\text{'} \\
 & \quad]\ \wedge \\
(\exists X_2)[& (\forall x)[X_0(x) \leftrightarrow ((\forall y)[\neg x \lhd y] \vee (\exists y)[X_0(y) \wedge A(y) \wedge x \lhd y])\ ]\ ] \\
 & \text{—}X_2 \text{ contains the maximum point and} \\
 & \qquad \text{everything up to the first non-`}a\text{' which precedes it)}
\end{array}$$

(Exercise) Show that Even-B is MSO definable.

6.5 Satisfying assignments for X_0, X_1 and X_2

a	a	b	a	a	c	a
					X_2	X_2
		X_1	X_1	X_1	X_1	
X_0	X_0	X_0				

6.6 Generating sets of satisfying assignments



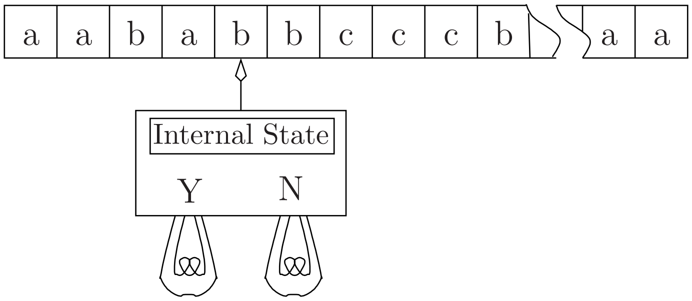
$\emptyset = 0 (= \rtimes)$, $\{X_0\} = 1$, $\{X_1\} = 2$, $\{X_0, X_1\} = 3$,
$\{X_2\} = 4, \{X_0, X_2\} = 5, \{X_1, X_2\} = 6, \{X_0, X_1, X_2\} = 7$

6.7 Finite-state automata

Definition 153 (Nondeterministic Finite-state Automaton) *A* ***Nondeterministic Finite-state Automaton*** *(NFA) is a 5-tuple $\langle Q, \Sigma, q_0, \delta, F \rangle$ where:*

- *Q is a finite set of **states**,*
- *Σ is the **input alphabet**,*
- *$q_0 \in Q$ is the designated **initial state**,*
- *$\delta \subseteq Q \times \Sigma \times Q$ is the **transition relation** and*
- *$F \subseteq Q$ is the set of **accepting states** (or "final" states).*

6.8



6.9 Recognizable stringsets

Definition 154 *A **computation**[1] of a FSA $\mathcal{M} = \langle Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, q_0^{\mathcal{M}}, \delta^{\mathcal{M}}, F^{\mathcal{M}}\rangle$ on a string $w \in \Sigma^*$ from a state $q \in Q^{\mathcal{M}}$ is a sequence of symbol/state pairs: $C = \langle \sigma_1, q_1\rangle\langle \sigma_2, q_2\rangle \cdots \langle \sigma_n, q_n\rangle$, in which:*

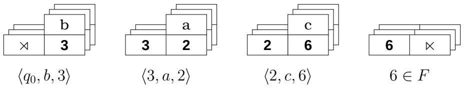
- *$\langle q, \sigma_1, q_1\rangle \in \delta^{\mathcal{M}}$ and*
- *$\langle q_i, \sigma_{i+1}, q_{i+1}\rangle \in \delta^{\mathcal{M}}$ for all $i < n$,*
- *$w = \pi_\ell(C)$*

*If, in addition, $q = q_0^{\mathcal{M}}$ and $q_n \in F^{\mathcal{M}}$, then the computation is **accepting**.*

Definition 155 *A string $w \in \Sigma^*$ is **accepted** by an FSA $\mathcal{M} = \langle Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, q_0^{\mathcal{M}}, \delta^{\mathcal{M}}, F^{\mathcal{M}}\rangle$ iff there is a accepting computation of $\mathcal{M}$ on w.*

Definition 156 *The language **recognized** by an FSA $\mathcal{M} = \langle Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, q_0^{\mathcal{M}}, \delta^{\mathcal{M}}, F^{\mathcal{M}}\rangle$ is the set of strings in Σ^* which are accepted by $\mathcal{M}$.*

6.10 Automata and tiling systems



[1] It is more common to define computations as a sequence of **Instantaneous Descriptions** which are pairs of a string and a state, with the string representing the portion of w which remains to be scanned. This formally equivalent to our notion of computation but less convenient for our purposes.

Computations of an FSA are just strings in $(\Sigma \times Q)^*$. The right projection of a computation is a **run**, the sequence of states the automaton visits in processing w (the left projection). Note that the transition between $\langle \sigma_i, q_i\rangle$ and $\langle \sigma_{i+1}, q_{i+1}\rangle$ depends only on q_i, σ_{i+1} and q_{i+1}. Let $L_\mathcal{M}$ be the set of computations of $\mathcal{M}$. $L_\mathcal{M}$ satisfies 2-Suffix Substitution Closure: $w_\mathcal{M} \cdot \langle q, \sigma\rangle \cdot y_\mathcal{M} \in L_\mathcal{M}$ and $v_\mathcal{M} \cdot \langle q, \sigma\rangle \cdot z_\mathcal{M}$ implies $w_\mathcal{M} \cdot \langle q, \sigma\rangle \cdot z_\mathcal{M} \in L_\mathcal{M}$. Consequently, the set of computations of an FSA is SL_2.
We can capture the set of computations of an FSA $\mathcal{M} = \langle Q^\mathcal{M}, \Sigma^\mathcal{M}, q_0^\mathcal{M}, \delta^\mathcal{M}, F^\mathcal{M}\rangle$ with an SL_2-style generator by interpreting $\delta^\mathcal{M}$ as a set of tiles, adding tiles of the form $\langle \rtimes, \sigma, q\rangle$ for each triple of the form $\langle q_0, \sigma, q\rangle$ and tiles of the form $\langle q, \ltimes\rangle$ for each $q \in F$. Then C is an accepting computation of $\mathcal{M}$ iff $\rtimes \cdot C \cdot \ltimes$ can be constructed using the tiles.

Theorem 157 (Chomsky Schützenberger) *A set of strings is recognizable iff it is a projection of a Strictly 2-Local set.*

6.11 Deterministic Finite-state Automata

Definition 158 (Deterministic Finite-state Automaton) *A **Deterministic Finite-state Automaton** (DFA) is an NFA in which the transition relation functional in the sense that for each $q_i \in Q$ and $\sigma \in \Sigma$ there is a* exactly one *$q_j \in Q$ such that $\langle q_i, \sigma, q_j\rangle \in \delta$.*

Theorem 159 *Every FSA is equivalent, in the sense of recognizing the same language, to a DFA.*

6.12 Powerset construction

Suppose $\mathcal{M} = \langle Q^\mathcal{M}, \Sigma^\mathcal{M}, q_0^\mathcal{M}, \delta^\mathcal{M}, F^\mathcal{M}\rangle$. Let $\widehat{\mathcal{M}} \overset{\text{def}}{=} \langle \widehat{Q}, \Sigma^\mathcal{M}, \hat{q}_0, \hat{\delta}, \widehat{F}\rangle$ where:

- $\widehat{Q} \overset{\text{def}}{=} \mathcal{P}(Q)$,
- $\hat{q}_0 \overset{\text{def}}{=} \{q_0\}$,
- $\hat{\delta} \overset{\text{def}}{=} \{q_j \mid \hat{q}_i \in \widehat{Q},\ q_i \in \hat{q}_i,\ \langle q_i, \sigma, q_j\rangle \in \delta^\mathcal{M}\}$,
- $\widehat{F} \overset{\text{def}}{=} \{\hat{q} \in \widehat{Q} \mid \hat{q} \cap F^\mathcal{M} \neq \emptyset\}$.

Claim 160

1. *$\widehat{\mathcal{M}}$ is deterministic.*
2. *Let*
$$\delta^*_\mathcal{M}(q, w) \overset{\textit{def}}{=} \begin{cases} \{q\} & \textit{if } w = \varepsilon, \\ \{q_i \mid \textit{There is a computation of } \mathcal{M} \textit{ on } w \\ \qquad\qquad \textit{from } q \textit{ ending in state } q_i\ \} \\ \qquad \textit{otherwise.} \end{cases}$$
and
$$\hat{\delta}^*(\hat{q}, w) \overset{\textit{def}}{=} \begin{cases} \hat{q} & \textit{if } w = \varepsilon, \\ \hat{q}_i & \textit{such that the computation of } \widehat{\mathcal{M}} \textit{ on } w \\ & \qquad\qquad \textit{from } \hat{q} \textit{ ends in state } \hat{q}_i \\ & \textit{otherwise.} \end{cases}$$
Then $\hat{\delta}^*(\hat{q}_0, w) = \delta^*_\mathcal{M}(q_0, w)$.

6.13 Closure properties

Lemma 161 *The class of recognizable stringsets is closed under Boolean operations.*

Construction for union and intersection:
Let $\widehat{Q} = Q^1 \times Q^2$.
Choose $\widehat{F}$ such that either (union) or both (intersection) components are in F^1, F^2.
(Exercise) Give a construction for converting a DFA for a stringset L into one for $\overline{L}$. Does this work for non-deterministic FSAs?

6.14 Projection and cylindrification

Projection: $\Sigma^1 \to \Sigma^2$, typically many-to-one.
Cylindrification: inverse projection

Lemma 162 *The class of recognizable stringsets is closed under projection and cylindrification.*

Apply map to tuples in δ.
E.g.:
If $a, b \mapsto c$ then $\langle q_i, a, q_j\rangle, \langle q_i, b.q_j\rangle \mapsto \langle q_i, c, q_j\rangle$,
If $c \mapsto a, b$ then $\langle q_i, c, q_j\rangle \mapsto \langle q_i, a, q_j\rangle, \langle q_i, b.q_j\rangle$.

6.15 Character of recognizable sets

Definition 163 (Nerode Equivalence) *Two strings w and v are* ***Nerode Equivalent*** *with respect to a stringset L over Σ (denoted $w \equiv_L v$) iff for all strings u over Σ, $wu \in L \Leftrightarrow vu \in L$.*

Theorem 164 (Myhill-Nerode) *: A stringset L is recognizable iff $\equiv_L$ partitions the set of all strings over Σ into finitely many equivalence classes.*

Proof ($\Rightarrow$)
L recognizable $\Rightarrow L = L(\mathcal{M})$ for some FSA $\mathcal{M}$. Wlog, by Theorem 159, $\mathcal{M}$ is deterministic. Let

$$w \equiv_{\mathcal{M}} v \Leftrightarrow \delta^*(q_0^{\mathcal{M}}, w) = \delta^*(q_0^{\mathcal{M}}, v).$$

Then $w \equiv_{\mathcal{M}}$ **refines** $w \equiv_L v$, i.e., $w \equiv_{\mathcal{M}} v \Rightarrow w \equiv_L v$.
Thus $\{[w]_{\mathcal{M}} \mid w \in \Sigma^*\}$ finite implies $\{[w]_L \mid w \in \Sigma^*\}$ finite.

6.16 Proof of MyHill-Nerode ($\Leftarrow$)

Suppose $\equiv_L$ partitions Σ^* into finitely many equivalence classes. Let $\mathcal{M}_L = \langle Q, \Sigma, q_0, \delta, F\rangle$, where:

$$\begin{array}{rcl} Q &=& \Sigma^*/_{\equiv_L} (= \{[w]_L \mid w \in \Sigma^*\}) \\ \delta &=& \{\langle [w]_L, \sigma, [w\sigma]_L\rangle \mid w \in \Sigma^*,\ \sigma \in \Sigma\} \\ q_0 &=& [\varepsilon]_L \\ F &=& \{[w]_L \mid w \in L\} \end{array}$$

(Exercise) Show that $w \equiv_L v \Rightarrow w\sigma \equiv_L v\sigma$ for all $w, v \in \Sigma^*$ and $\sigma \in \Sigma$.

6.17 MSO Quantifier Rank

Definition 165 (MSO Quantifier Rank)

$$\boldsymbol{qr}(\varphi) \overset{def}{=} \begin{cases} 0 & \text{if } \varphi = \text{`}\sigma(\vec{x})\text{' or } \varphi = \text{`}x \approx y\text{'}, \\ \boldsymbol{qr}(\psi) & \text{if } \varphi = \text{`}(\neg\psi)\text{'}, \\ \max(\boldsymbol{qr}(\psi_1), \boldsymbol{qr}(\psi_2)) & \text{if } \varphi = \text{`}(\psi_1 \vee \psi_2)\text{'}, \\ \boldsymbol{qr}(\psi) + 1 & \text{if } \varphi = \text{`}(\exists x)[\psi]\text{'}, \\ \boldsymbol{qr}(\psi) + 1 & \text{if } \varphi = \text{`}(\exists X)[\psi]\text{'}. \end{cases}$$

6.18 MSO types

Definition 166 ((r, m, n)-types) *Suppose $\mathcal{A}$ is a model with domain A, and $r \geq 0$.*
Let $\langle A_0, \ldots, A_{m-1}\rangle$ be an m-tuple of subsets of A and $\langle a_0, \ldots, a_{n-1}\rangle$ an n-tuple of points from A. The (r, m, n)-type of $(\langle A_0, \ldots, A_{m-1}\rangle, \langle a_0, \ldots, a_{n-1}\rangle)$ in $\mathcal{A}$ *is:*

$$\begin{aligned} &\boldsymbol{tp}_r(\mathcal{A}, \langle A_0, \ldots, A_{m-1}\rangle, \langle a_0, \ldots, a_{n-1}\rangle) \overset{def}{=} \\ &\quad \{ \quad \varphi(X_0, \ldots, X_{m-1}, x_0, \ldots, x_{n-1}) \mid \\ &\qquad\quad \boldsymbol{qr}(\varphi(X_0, \ldots, X_{m-1}, x_0, \ldots, x_{n-1})) = r \text{ and} \\ &\qquad\quad \mathcal{A}, [X_i \mapsto A_i, x_i \mapsto a_i] \models \varphi(X_0, \ldots, X_{m-1}, x_0, \ldots, x_{n-1}) \quad \}. \end{aligned}$$

*The set of (r, m, n)-**types realized in a model** $\mathcal{A}$ is the set of (r, m, n)-types of the n-tuples of its domain:*

$$S_r^{m,n}(\mathcal{A}) \overset{def}{=} \{\boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a}) \mid \vec{A} \in \mathcal{P}(A)^m, \vec{a} \in A^n\}.$$

$$S_r^{0,0}(\mathcal{A}) = \{\mathbf{tp}_r(\mathcal{A}, \langle\,\rangle, \langle\,\rangle)\}, \qquad \mathbf{Th}_r^2(\mathcal{A}) \overset{\text{def}}{=} \mathbf{tp}_r^2(\mathcal{A}) \overset{\text{def}}{=} \mathbf{tp}_r(\mathcal{A}, \langle\,\rangle, \langle\,\rangle).$$

$$\mathcal{A} \equiv_{2,r} \mathcal{B} \overset{\text{def}}{\iff} \mathbf{tp}_r^2(\mathcal{A}) = \mathbf{tp}_r^2(\mathcal{B}).$$

Lemma 167 *The number of logically distinct formulae of quantifier rank r with m free MSO variables and n free FO variables in any relational monadic Second-Order language L over a finite signature is finitely bounded.*

Corollary 168 *The number of distinct (r, m, n)-types realizable in any class of relational models is finite.*

Corollary 169 *For every model $\mathcal{A}$, every m-tuple $\vec{A}$ of subsets of the domain of $\mathcal{A}$, $m \geq 0$, every n-tuple $\vec{a}$ of points in the domain of $\mathcal{A}$, $n \geq 0$, and every $r \geq 0$, there is a single MSO formula $\chi_r^{\mathcal{A},\vec{A},\vec{a}}(\vec{X}, \vec{x})$ which characterizes $\boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a})$:*

$$\mathcal{B}, [\vec{X} \mapsto \vec{B}, \vec{x} \mapsto \vec{b}] \models \chi_r^{\mathcal{A},\vec{A},\vec{a}}(\vec{X}, \vec{x}) \text{ iff } \boldsymbol{tp}_r(\mathcal{B}, \vec{B}, \vec{b}) = \boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a})$$

Moreover $\chi_r^{\mathcal{A},\vec{A},\vec{a}}(\vec{X}, \vec{x}) \in \boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a})$

Theorem 170 *A property of models P, a subset of the set of all models over a relational signature Σ, is definable in $L^2(\Sigma)$ iff there is some $r \geq 0$ such that, for all Σ-models $\mathcal{A}$ and $\mathcal{B}$*

$$\boldsymbol{tp}_r^2(\mathcal{A}) = \boldsymbol{tp}_r^2(\mathcal{B}) \quad \Rightarrow \quad \mathcal{A} \in P \Leftrightarrow \mathcal{B} \in P.$$

Corollary 171

$$\boldsymbol{tp}_r^2(w_1) = \boldsymbol{tp}_r^2(w_2) \quad \Rightarrow \quad w_1 \in L \Leftrightarrow w_2 \in L.$$

6.19 Concatenation and MSO types

Lemma 172

$$\begin{array}{ll} \textit{If} & \boldsymbol{tp}_r(\mathcal{A},\vec{A},\vec{a}) = \boldsymbol{tp}_r(\mathcal{B},\vec{B},\vec{b}) \ \textit{and} \ \boldsymbol{tp}_r(\mathcal{C},\vec{C},\vec{c}) = \boldsymbol{tp}_r(\mathcal{D},\vec{D},\vec{d}) \\ \textit{then} & \boldsymbol{tp}_r(\mathcal{A}\cdot\mathcal{C},\vec{A}\cdot\vec{C},\vec{a}\cdot\vec{c}) = \boldsymbol{tp}_r(\mathcal{B}\cdot\mathcal{D},\vec{B}\cdot\vec{D},\vec{b}\cdot\vec{d}). \end{array}$$

Corollary 173

$$\textit{If } \mathcal{A} \equiv_{2,r} \mathcal{B} \textit{ and } \mathcal{C} \equiv_{2,r} \mathcal{D} \textit{ then } \mathcal{A}\cdot\mathcal{C} \equiv_{2,r} \mathcal{B}\cdot\mathcal{D}.$$

6.20 Recognizability characterizes MSO-definability

Theorem 174 (Medvedev, Büchi, Elgot) *A set of strings is MSO-definable relative to the class of finite $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ models iff it is recognizable.*

Proof(Only if) Suppose that L is MSO definable. Then there is some MSO sentence φ such that, for all strings w over Σ, $w \in L$ iff $w \models \varphi$. Let $r = \mathbf{qr}(\varphi)$. By Corollary 173, for all w, v, u, if $w \equiv_{2,r} v$ then $w \cdot u \equiv_{2,r} v \cdot u$. Thus, if $w \equiv_{2,r} v$ then

$$w \cdot u \in L \Leftrightarrow w \cdot u \models \varphi \Leftrightarrow v \cdot u \models \varphi \Leftrightarrow v \cdot u \in L.$$

Hence, equivalence with respect to $\equiv_{2,r}$ implies Nerode Equivalence; the Nerode Equivalence classes will be unions of the $\equiv_{2,r}$ classes. By Corollary 168, there are but finitely many $(r, 0, 0)$-types, hence finitely many of these equivalence classes and finitely many Nerode Equivalence classes. It follows, by the Myhill-Nerode Theorem, that L is recognizable. ⊣

Proof(If) Suppose $L = L(\mathcal{M})$ for some FSA $\mathcal{M}$.
Let $L_{\mathcal{M}} \subset (Q \times \Sigma)^+$ be the set of accepting computations of $\mathcal{M}$. As the set of all computations of $\mathcal{M}$ is SL and this is just the subset of that set which end in accepting states, $L_{\mathcal{M}}$ is SL and, hence, FO definable.
Let $\varphi'_{\mathcal{M}}$ be a variation of the First-Order sentence defining $L_{\mathcal{M}}$ in which instead of using $Q \times \Sigma$ as the alphabet, we use $Q \cup \Sigma$, translating each atomic formula $\langle q, \sigma\rangle(x)$ to $(q(x) \wedge \sigma(x))$.
Treating Q as MSO variables:

$$\varphi_{\mathcal{A}} \overset{\text{def}}{=} (\exists Q)[\varphi'_{\mathcal{A}}(Q)]. \tag{175}$$

⊣

Corollary 176 *A stringset L over and alphabet Σ is MSO definable over $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ iff $\equiv_L$ partitions the set of all strings over Σ into finitely many equivalence classes.*

Corollary 177 *Every MSO sentence over $\langle W, \triangleleft, \triangleleft^+, P_\sigma\rangle_{\sigma\in\Sigma}$ is logically equivalent to an MSO sentence of the form:*

$$(\exists X_1, \ldots, X_{n-1})[\varphi(X_1, \ldots, X_{n-1})] \tag{178}$$

where $\varphi(X_1, \ldots, X_{n-1})$ uses only First-Order quantification.

6.21 Definability $\Rightarrow$ Recognizability

- Treat Σ as set variables. Assume no variables reused.
- Reduce to $x \triangleleft y$, $x \approx y$, $X \approx Y$, $X(y)$, $\exists x$ and $\exists X$.
- Reduce to: only set variables, $\exists X$, $X \subseteq Y$ and $X \triangleleft Y$ where:

$$\begin{aligned} \text{Empty}(X) &\equiv (\forall Y)[Y \subseteq X \rightarrow X \subseteq Y] \\ \text{Singleton}(X) &\equiv (\forall Y)[Y \subseteq X \rightarrow (\text{Empty}(Y) \vee X \subseteq Y)] \\ x \triangleleft y &\equiv \text{Singleton}(X) \wedge \text{Singleton}(Y) \wedge X \triangleleft Y \end{aligned}$$

(Exercise) Show how to reduce $x \approx y$, $X \approx Y$ and $X(y)$ to $X \subseteq Y$ and $X \triangleleft Y$

6.22 Accepting Atomic Formulae

E.g., assignments satisfying $X \triangleleft Y$ are in $L(\mathcal{M})$ for $\mathcal{M}$ where:

$$\begin{aligned} Q^{\mathcal{M}} &\overset{\text{def}}{=} \{0, 1, 2, 3\} \\ \Sigma^{\mathcal{M}} &\overset{\text{def}}{=} \mathcal{P}(\{X, Y\}) \\ q_0^{\mathcal{M}} &\overset{\text{def}}{=} \emptyset \\ \delta^{\mathcal{M}} &\overset{\text{def}}{=} \{ \quad \langle 0, \emptyset, 0 \rangle,\ \langle 0, \{X\}, 1 \rangle, \langle 1, \{Y\}, 2 \rangle, \langle 2, \emptyset, 2 \rangle \\ &\qquad\quad \langle q, \sigma, 3 \rangle \text{ for all other } q \text{ and } \sigma\ \} \\ F^{\mathcal{M}} &\overset{\text{def}}{=} \{2\}. \end{aligned}$$

	$\emptyset$	$\emptyset$	$\{X\}$	$\{Y\}$	
$\rtimes$	0	0	1	2	$\ltimes$

6.23 Extending to Arbitrary Formulae

- $\vee$, $\wedge$ — Union, intersection
- $\exists$ — projection

$$(\exists Y)[\varphi] : \{X\}, \{X, Y\} \mapsto \{X\}$$

	$\emptyset$	$\emptyset$	$\{X\}$	$\emptyset$	
$\rtimes$	0	0	1	2	$\ltimes$

  - introduces non-determinism
- $\neg$ — Determinization and complement
  - potential exponential blow-up

6.24 Decision problems for MSO

Lemma 179 *Emptiness of MSO-definable stringsets is decidable.*

Lemma 180 *Universality of MSO-definable stringsets is decidable.*

Lemma 181 *Finiteness of MSO-definable stringsets is decidable.*

6.25 Cognitive interpretation of MSO

- Any cognitive mechanism that can distinguish member strings from non-members of an MSO-definable stringset must be capable of classifying the events in the input into a finite set of abstract categories and are sensitive to the sequence of those categories.

- Subsumes *any* recognition mechanism in which the amount of information inferred or retained is limited by a fixed finite bound.

- Any cognitive mechanism that has a fixed finite bound on the amount of information inferred or retained in processing sequences of events will be able to recognize *only* MSO-definable stringsets.