Formal Description of Syntax
ESSLLI'07

James Rogers
Earlham College

Updated reader:
http://www.cs.earlham.edu/ jrogers/files/reader.pdf

Copies of slides:
http://www.cs.earlham.edu/ jrogers/files/handout.pdf

Slide 1

Slide 2

Formalization of Syntax

Slide 3

Modeling language

Utterances—sequences of symbols (strings)

Fragments of language—stringsets

Syntax—patterns in stringsets

Slide 4

Formalization

Explicit theories

- Univocality—unambiguous
- Overtness—no hidden assumptions
- Projectability—well-defined method of inference

Abstraction

- Taming complexity
- Clarity of reasoning

Useful (essential?) even if goal is not a fully formalized theory.

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

Slide 5

Slide 6

Basic Concepts

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)
Slide 7 (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* $w \in \Sigma^*$.

A Recursive Definition

Definition 2 (Length of a string) (A **Recursive Definition**.)
Slide 8 *For all $w \in \Sigma^*$:*

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

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
Slide 9 $w = v \cdot \sigma$ 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:*

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

Slide 10

Proof Exercise. ⊣

Lemma 6 *Concatenation of strings is associative*

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

Proof Exercise. ⊣

Concatenation and iteration of stringsets

Definition 7

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

Slide 11

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

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

Kleene and positive closure

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

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

Slide 12

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

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

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

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.,* $\emptyset \in$ Fin*)*
- $\{\varepsilon\}$ *is a finite stringset (*$\{\varepsilon\} \in$ Fin*)*

Slide 13

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

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

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****:*

Slide 14

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

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

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* $\rho_i^{\mathcal{A}} \subseteq A^n$*.*

Strings as relational structures

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

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

in which the domain W is a finite set which is totally ordered by the transitive closure of $\lhd$.

Slide 15

*A ($\lhd^+$) **String Model** (a **Precedence String Model**) over an alphabet Σ is a tuple*

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

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

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

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

Example

people Alice loves love Alice $= \langle W, \lhd, P_{\textbf{\textit{people}}}, P_{\textbf{\textit{Alice}}}, P_{\textbf{\textit{loves}}}, P_{\textbf{\textit{love}}} \rangle$

Slide 16

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

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.*

Slide 17

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 $\mathcal{R}$.

Canonical string models

Observation 17 *For all strings $W = \langle W, \lhd^{\mathcal{W}}, {\lhd^+}^{\mathcal{W}}, P_\sigma^{\mathcal{W}} \rangle_{\sigma \in \Sigma}$, if $|W| = n$ then $W \cong \langle \{0, 1, \ldots, n-1\}, \lhd', {\lhd^+}', P'_\sigma \rangle_{\sigma \in \Sigma}$*

Slide 18

Observation 18 *If $\Sigma \subseteq \Gamma$*

$$\begin{aligned} \mathcal{W} &= \langle W \lhd^{\mathcal{W}}, {\lhd^+}^{\mathcal{W}}, P_\sigma^{\mathcal{W}} \rangle_{\sigma \in \Sigma} \in \Sigma^* \\ \mathcal{W}' &= \langle W \lhd^{\mathcal{W}}, {\lhd^+}^{\mathcal{W}}, P_\sigma^{\mathcal{W}}, \emptyset, \ldots \rangle_{\sigma \in \Sigma} \in \Gamma^* \end{aligned}$$

The empty string

Observation 19

Slide 19

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

where

$$W = \lhd^{\mathcal{W}} = {\lhd^+}^{\mathcal{W}} = P_\sigma^{\mathcal{W}} = \emptyset$$

Canonical empty string: $\varepsilon = \langle \emptyset \rangle$

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

$$w = \langle W, \lhd^w, {\lhd^+}^w, P_\sigma^w \rangle_{\sigma \in \Sigma} \qquad \textit{and} \qquad v = \langle V, \lhd^v, {\lhd^+}^v, P_\sigma^v \rangle_{\sigma \in \Sigma}$$

Slide 20

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

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

Formal Problems

Slide 21

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* $\{\text{true}, \text{false}\}$.

Some formal problems

Slide 22

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.*

Slide 23

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.*

Slide 24

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.*

Algorithms

Slide 25

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.*

Rec

Slide 26

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.*

Slide 27

Propositional Languages for Strings—Strictly Local Stringsets

k-factors

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

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

Slide 28

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

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

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

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 '$\ltimes$':*

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

Slide 29

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.*

Capabilities of Strictly 2-Local Descriptions

- Valency of verbs
  - Exclude *"likes $\ltimes$"*, *"slept Alice"*, etc.
- (Local) number agreement

Slide 30

  - 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.

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

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

Slide 31

$\{(a_1a_2\cdots a_k)^i \mid 0 \leq i\} \in \mathrm{SL}_k$ ($a_i = a_j$ iff $i = j$), as witnessed by

$$\begin{array}{ll}\{ & \rtimes\ltimes,\ \rtimes a_1\cdots a_{k-1},\ a_1\cdots a_k,\ a_2\cdots a_ka_1,\ \ldots,\\ & a_ia_{i+1}\cdots a_ka_1\cdots a_{i-1},\ \ldots\\ & a_2\cdots a_k\ltimes\\ & \}\end{array}$$

(Exercise) What stringsets would the class SL_1 include?

SL_2 example

Example 33

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

Slide 32

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

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

SL_3 example

Slide 33

Example 34

$$\begin{aligned}\mathcal{D}_{34} = \{\quad & \rtimes \boldsymbol{Alice\ sleeps},\ \rtimes \boldsymbol{Alice\ loves},\\ & \rtimes \boldsymbol{people\ sleep},\ \rtimes \boldsymbol{people\ love},\\ & \boldsymbol{Alice\ loves\ Alice},\ \boldsymbol{Alice\ loves\ people},\\ & \boldsymbol{people\ love\ Alice},\ \boldsymbol{people\ love\ people},\\ & \boldsymbol{Alice\ sleeps} \ltimes,\ \boldsymbol{people\ sleep} \ltimes,\\ & \boldsymbol{loves\ Alice} \ltimes,\ \boldsymbol{loves\ people} \ltimes,\\ & \boldsymbol{love\ Alice} \ltimes,\ \boldsymbol{love\ people} \ltimes \qquad \}\end{aligned}$$

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

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

SL_4 example

Slide 34

Example 36

$$\begin{aligned}\mathcal{D}_{36} = \{\quad & \rtimes \boldsymbol{Alice\ sleeps},\ \boldsymbol{Alice\ sleeps} \ltimes,\\ & \rtimes \boldsymbol{Alice\ loves\ Alice},\ \boldsymbol{Alice\ loves\ Alice} \ltimes,\\ & \rtimes \boldsymbol{Alice\ loves\ people},\ \boldsymbol{Alice\ loves\ people} \ltimes,\\ & \rtimes \boldsymbol{people\ sleep},\ \boldsymbol{people\ sleep} \ltimes,\\ & \rtimes \boldsymbol{people\ love\ Alice},\ \boldsymbol{people\ love\ Alice} \ltimes,\\ & \rtimes \boldsymbol{people\ love\ people},\ \boldsymbol{people\ love\ people} \ltimes \quad \}\end{aligned}$$

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

Finite stringsets and SL

Slide 35

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.*

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

Cognitive interpretation of SL

Slide 36

- 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.

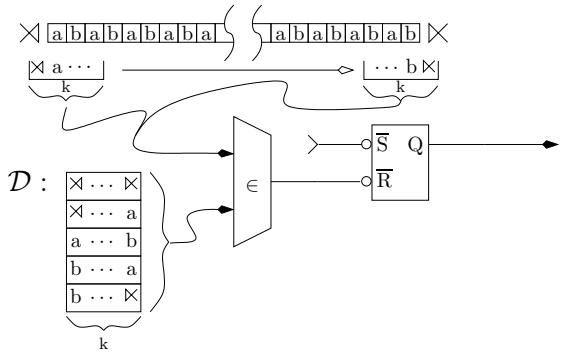
Abstract recognition algorithms

Slide 37

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.*

Automata for SL

Slide 38



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

Slide 39

- $x_m = \sigma_{n-(k-2)} \cdots \sigma_n \ltimes$ *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 $w \in L(\mathcal{D}_L)$. ⊣

Slide 40

Theorem 43 ($\mathrm{SL} \subseteq$ **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).

Myhill graphs

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

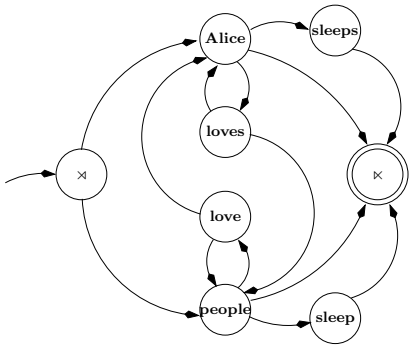
Slide 41

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

*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.*

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

Slide 42



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 $\mathcal{D}$.*

Slide 43

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 $v_1v_2\cdots v_n \in L(\mathcal{D})$. ⊣

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:*

Slide 44

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

and

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

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

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

$$v \in F_{\mathcal{D}} \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad v\ltimes \in \mathcal{D}.$$

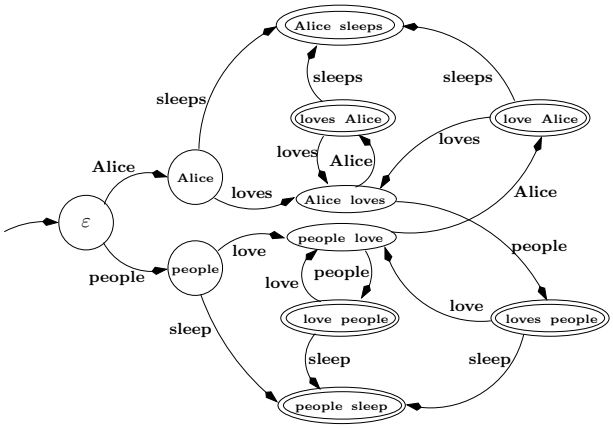
Slide 45

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

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

Slide 46

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



Slide 47

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 $L(\mathcal{D}) \stackrel{?}{=} \emptyset$*

Slide 48

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.)

Slide 49

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

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

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.*

Slide 50

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).*

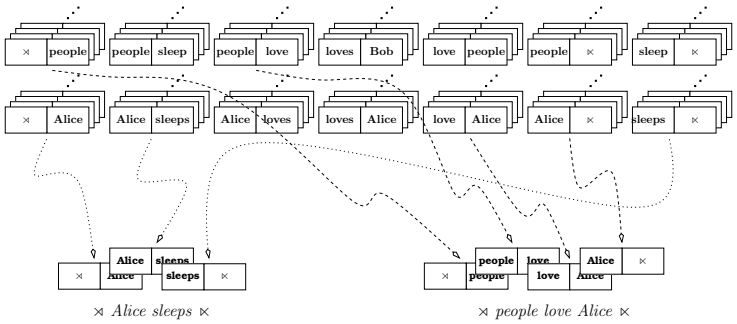
Lemma 53 **Rec** $\subsetneq$ **r.e.**

Slide 51

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.

Grammars for SL

Slide 52



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

Slide 53

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

Example

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

Slide 54

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

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 $\mathcal{G}$:*

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

Slide 55

*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 $\mathcal{G}$:*

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

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*

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

Slide 56

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*

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

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

Equivalence of strictly k-local description, scanners and grammars

Theorem 58 *The following are equivalent:*

Slide 57

- $L \in \mathrm{SL}$
- $L = L(\mathcal{D})$ *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* $\mathcal{G}$

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:

Slide 58

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

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. ⊣

Recognition via generation

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

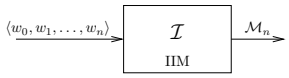
Proof Derivations in strictly k-local grammars grammars are *strictly length increasing*: if $w \underset{\mathcal{G}}{\Longrightarrow} v$ then $|v| > |w|$ (in fact, $|v| = |w| + 1$).

Slide 59

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. ⊣

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 $n \geq i$,*

Slide 60

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

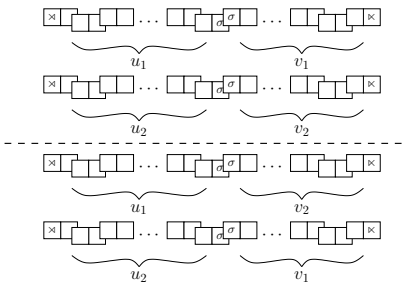
$$\mathcal{I}(\langle \ell(0), \ell(1), \ldots, \ell(n)\rangle) = \mathcal{M}_n = \langle \Sigma, T_n \rangle, \text{ where } T_n \stackrel{\text{def}}{=} \bigcup_{0 \leq i \leq n} [F_k(\rtimes \ell(i) \ltimes)].$$

Slide 61

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 $L(\mathcal{M}_n) = L(D_L) = L$. ⊣

Suffix Substitution Closure



Slide 62

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:*

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

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*

Slide 63

$$u_1 \cdot x \cdot v_1 \in L$$
$$u_2 \cdot x \cdot v_2 \in L$$

then it will also be the case that

$$u_1 \cdot x \cdot v_2 \in L$$

Proof of SSC ($\Leftarrow$)

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

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

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

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

Slide 64

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

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

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:

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

Slide 65

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.

Proof of SSC (construction)

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

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

Slide 66

(Stage 0) Otherwise $n \geq k-1$ 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 $k-1 \leq i \leq n$:

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 $k \leq i < n$)

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

Slide 67

$w_i \in L$ 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

Slide 68

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

Non-SL Stringsets

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

Slide 69

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

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

Adversary Arguments

$$
(\forall k)[\quad (\exists u_1, v_1, u_2, v_2 \in \Sigma^*, x \in \Sigma^{k-1})[
\quad u_1 x v_1 \in L \text{ and } u_2 x v_2 \in L \text{ and } u_1 x v_2 \notin L\ ]\quad]
$$

∀—adversary's choice, ∃—your choice

Slide 70

- Your adversary, claiming that there is a k-local automaton that recognizes L, chooses k.
- You now choose two strings $u_1 x v_1$ and $u_2 x v_2$. 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
  - $u_1 x v_1$ and $u_2 x v_2$ are both in L and
  - $u_1 x v_2$ is not in L.

Example

Consider the stringset

$$L_{71} = \{wabv \mid w, v \in \{a, b\}^*\}$$

Slide 71

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 $a^k b$ and aba^k are in L_{71} (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, $u_1 a^{k-1} v_2 = a a^{k-1} a$ would also be in L_{71}. But it is not. In this way, reasoning from the supposition that $L_{71} \in \mathrm{SL}_k$ we obtain a contradiction. Hence $L_{71} \notin \mathrm{SL}_k$ for any k.

Theorem 65 *No stringset modeling English is* SL_2

Slide 72

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

Theorem 66 *No stringset modeling English is* SL

Example 67

Slide 73

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

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.) ⊣

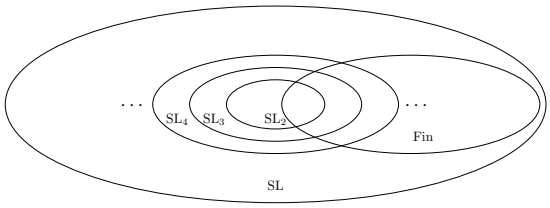
Slide 74

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

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

Finite stringsets and SL

Slide 75



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, $\mathrm{Fin} \subsetneq \mathrm{SL}$. 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.

SL is not learnable

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

Proof Let $L_{a^*} \stackrel{\text{def}}{=} \{a^i \mid 0 \leq i\}$. $L_{a^*} \in \mathrm{SL}_2$.
Let $\ell(i) \stackrel{\text{def}}{=} a^i$. Then $L_{a^*} = \{\ell(i) \mid i \in \mathbb{N}\}$.

Slide 76

Suppose $\mathcal{I}$ converges, at stage i, on $\mathcal{M}_i$ such that $L(\mathcal{M}_i) = L_{a^*}$.
Let

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

Then $\{\ell_i(j) \mid j \in \mathbb{N}\} = L_{a^{\leq i}} \stackrel{\text{def}}{=} \{a^j \mid 0 \leq j \leq i\}$.
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.
⊣

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

Slide 77

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

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. ⊣

Union of SL stringsets

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

Proof Let

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

Slide 78 Both L_1 and L_2 are SL_2.

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. ⊣

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:* $S_2 - S_1 \stackrel{def}{=} \{x \mid x \in S_2 \text{ and } x \notin S_1\}$.

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

Slide 79

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

Proof By DeMorgan's Theorem

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

Hence closure under intersection but not under union implies non-closure under complement. ⊣

Concatenation of SL stringsets

Lemma 76 *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

Slide 80

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

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 71:

$$L_{71} \stackrel{\text{def}}{=} \{wabv \mid w, v \in \{a,b\}^*\}$$

which we have shown to be non-SL. ⊣

Kleene closure of SL stringsets

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

Slide 81

Proof $L_{aa} \stackrel{\text{def}}{=} \{aa\} \in \text{SL}_3$ (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. ⊣

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

Closure of SL_2 under Kleene closure

Lemma 78 *The class of strictly 2-local stringsets* is *closed under Kleene closure.*

Slide 82

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^*. ⊣

Slide 83

Propositional Languages for Strings—Locally Testable Stringsets

Slide 84

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:

$$\begin{array}{rcl} \varepsilon \cdot \mathrm{N} \cdot \mathrm{V\,N} & \in E \\ \mathrm{N\,V} \cdot \mathrm{N} \cdot \varepsilon & \in E \\ \hline \varepsilon \cdot \mathrm{N} \cdot \varepsilon & \notin E \end{array}$$

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.

k-Expressions

Definition 79 (k-Expressions over Strings) *The language of k-expressions (over strings) is the smallest set including:*

Slide 85

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

Slide 86

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

Slide 87

Defined connectives

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

Slide 88

Capabilities of k-Expressions

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

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

Locally Testable stringsets

Definition 81 (Locally testable stringsets) *A stringset L is* ***Locally k-Testable*** *(*LT_k*) iff there is a k-expression φ such that:*

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

Slide 89

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

Theorem 82 LT *recognition is decidable.*

Proof: Exercise

LT and SL

Lemma 83 ($\mathrm{SL}_k \subseteq \mathrm{LT}_k$) *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.

Slide 90

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 (83), 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. ⊣

Lemma 84 *The class of Locally k-Testable stringsets is the closure of the class of Strictly k-Local stringsets under Boolean operations.*

Slide 91

Proof That every Boolean combination of Strictly k-Local stringsets is Locally k-Testable follows from Lemma 83 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 (83): $\neg(\bigwedge[\neg f])$ (where the conjunction is over only the single negated factor $\neg f$). ⊣

Example 85 *Let*

$$\begin{aligned}\mathcal{G}_{85} \;=\; \{\;& \rtimes\boldsymbol{which}, \boldsymbol{which\ girls}, \boldsymbol{girls\ do}, \boldsymbol{which\ girl}, \boldsymbol{girl\ do},\\ & \boldsymbol{do\ they}, \boldsymbol{they\ think}, \boldsymbol{think\ that}, \boldsymbol{that\ they},\\ & \boldsymbol{think\ were}, \boldsymbol{were\ responsible},\\ & \boldsymbol{think\ was}, \boldsymbol{was\ responsible}, \boldsymbol{responsible}\ltimes \qquad \}\end{aligned}$$

Slide 92

To enforce number agreement we just convert it to LT_2 *form and reject co-occurrence of* ***girls*** *and* ***was*** *or* ***girl*** *and* ***were****:*

$$\varphi_{67} = \bigwedge_{f_i \notin \mathcal{G}_{85}} [\neg f_i] \;\wedge\; \neg(\boldsymbol{girls} \wedge \boldsymbol{was}) \;\wedge\; \neg(\boldsymbol{girl} \wedge \boldsymbol{were})$$

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

Theorem 86 ($\mathrm{SL} \subsetneq \mathrm{LT}$) *The class of strictly local stringsets is a proper subset of the class of locally testable stringsets.*

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

Character of the locally testable sets

Theorem 87 (Local Test Invariance) *A stringset L is Locally Testable iff*

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

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

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

Slide 93

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 $w \in L$.

LT equivalence

Theorem 88 *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 $S \subseteq \{[x]_k \mid x \in \Sigma^*\}$.*

Slide 94

Observation 89 *There are only finitely many LT_k stringsets for any given Σ and k.*

Disjunctive Normal Form

Lemma 90 *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:

Slide 95

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

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. ⊣

Cognitive interpretation of LT

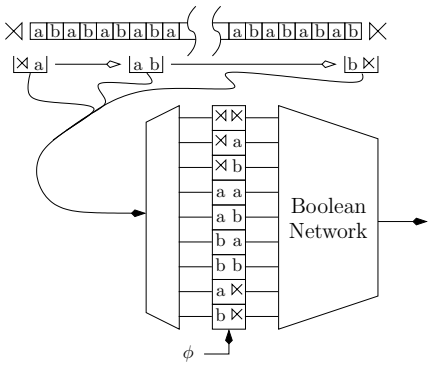
- Any cognitive mechanism that can distinguish member strings from non-members of an LT_k stringset must be sensitive, at least, to the set of length k blocks of events that occur in the presentation of the string.

Slide 96

- 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.

LT Automata

Slide 97



Constructing LT_2 transition graphs

Given φ a 2-expression over Σ:

Vertices are states of the LT_2 automaton for φ:

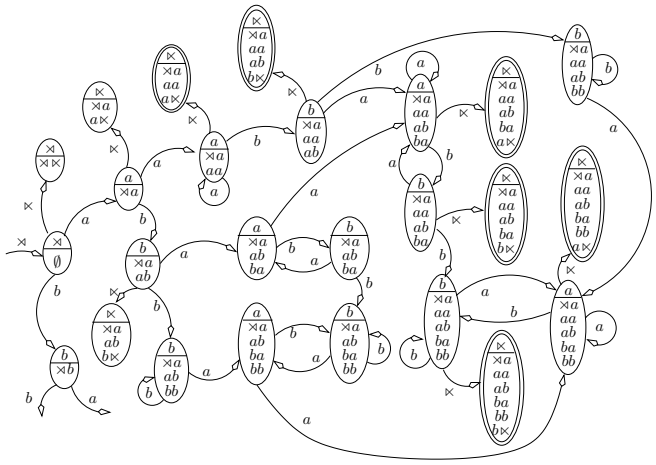
$$\langle \sigma_i, S_i \rangle \in (\Sigma \cup \{\rtimes, \ltimes\}) \times \mathcal{P}(F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)).$$

Slide 98

- 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.

LT **transition graphs**

Slide 99



Theorem 91 *Emptiness of* LT *stringsets is decidable.*

Proof: exercise

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

Proof: exercise

Slide 100

Theorem 93 *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.

Learnability of LT_k and LT

Theorem 94 *For all k, the class of LT_k stringsets is learnable in the limit.*

Slide 101

Proof: Exercise.

Theorem 95 *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.

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 96

Slide 102

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

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

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.

Relationship between locally k-testable classes

Theorem 97 (The LT hierarchy) *The classes of* LT_k *stringsets form a proper hierarchy:*

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

Slide 103

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 $\{a^i\} \in \mathrm{LT}_{i+1} - \mathrm{LT}_i$.

Closure properties

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

Slide 104

Lemma 98 (Non-closure under concatenation) *The class of locally testable stringsets is not closed under concatenation.*

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

Slide 105

First-Order Languages for Strings

FO($\lhd^+$) (Strings)

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

Slide 106

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

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

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

Semantics of FO languages

Slide 107

Definition 100 (Free and Bound Variables) *A variable x occurring in a formula is **bound** iff it occurs within the scope (within the '[...]') 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, $x_0, \ldots$.

Slide 108

Definition 101 (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:

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

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

Slide 109

Definition 102 (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:*

- $\varphi =$ *'$x \lhd y$', $s(x)$ and $s(y)$ are both defined and $s(y) = s(x) + 1$,*
- $\varphi =$ *'$x \lhd^+ y$', $s(x)$ 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 $\mathcal{A}$, $\mathcal{A}, s[x \mapsto a] \models \psi$.*

Slide 110

$\lhd$ is definable from $\lhd^+$

Let

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

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

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

Logical Sentences

Slide 111

Definition 103 (Sentences) *A (logical) sentence is a formula with no free variables.*

A model $\mathcal{A}$ satisfies a sentence (or not) independently of assignments:

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

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**.

Slide 112

Definition 104 (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:*

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

We say that

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

FO($\lhd^+$) definable stringsets

Definition 105 *$L \subseteq \Sigma^*$ is **FO($\lhd^+$) definable** iff there is a finite set of FO($\lhd^+$) sentences Φ (equivalently, a single sentence $\varphi(= \bigcap \Phi)$)*
Slide 113 *such that $L = \mathbf{Mod}\Phi$.*

Theorem 106 *The fixed and universal recognition problems for the class of FO($\lhd^+$) definable stringsets are decidable.*

Proof: exercise.

Capabilities of FO Definitions over Strings

$$\mathrm{NP}(x,y) \stackrel{\text{def}}{=} \mathsf{my}(x) \wedge \mathsf{father}(y) \wedge x < y \wedge$$

—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

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

Slide 114

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

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

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

]

— ...and nothing else

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

Logical Theories

Definition 107 (Theories of Models) *The theory of a Σ-model $\mathcal{A}$ (in a given language $L(\Sigma)$), denoted $\boldsymbol{Th}(\mathcal{A})$, is the set of all sentences of $L(\Sigma)$ that are satisfied by $\mathcal{A}$:*

$$\boldsymbol{Th}(\mathcal{A}) \stackrel{def}{=} \{\varphi \in L(\Sigma) \mid \mathcal{A} \models \varphi\}.$$

Slide 115 *The theory of a set of models $\mathbb{A}$ is the set of all sentences satisfied by every member of $\mathbb{A}$:*

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

Note that

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

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.

Slide 116 **Definition 108 (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 $\boldsymbol{Th}(\mathcal{A}) = \boldsymbol{Th}(\mathcal{B})$.*

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

Consequences

Definition 109 (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:*

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

Slide 117

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

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

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

$\mathbf{Cn}(\emptyset)$ = theory of the set of all Σ-models = validities of $L(\Sigma)$.

Definition 110 (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*

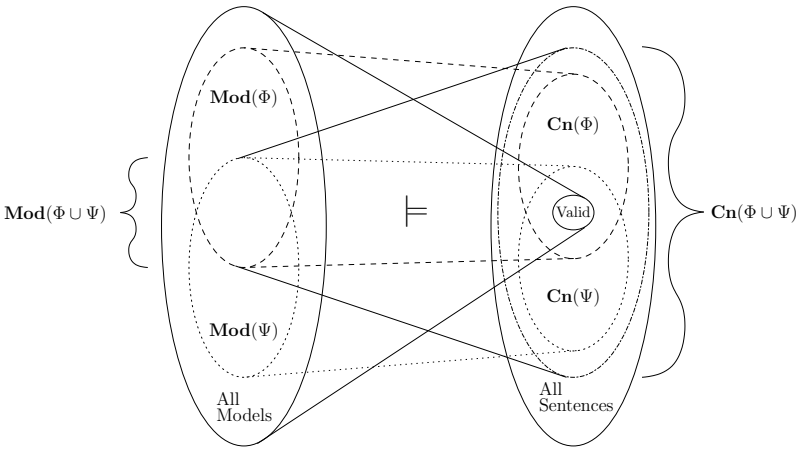
$$\{\psi \mid \mathcal{A} \models \Phi \Rightarrow \mathcal{A} \models \psi, \quad \text{for all } \mathcal{A} \in \mathbb{C}\}.$$

Slide 118

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.)

Relationship between consequences and models

Slide 119



Logical Equivalence

Slide 120

Definition 111 (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 $\mathbf{Cn}(\Phi) = \mathbf{Cn}(\Psi)$.

Theories as sets of sentences

Definition 112 (Formal Theory) *A **(formal) theory** is a set of sentences Φ which is closed under logical consequence:*

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

Slide 121

*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 $\neg\varphi \in \Phi$.*

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:

Slide 122

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

This follows immediately from the definitions of $\rightarrow$ and of logical consequence.

Definable sets

Definition 113 (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 $\mathbb{A} = \mathbf{Mod}(\psi)$.

Slide 123

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

Definition 114 (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 $\mathbb{A} = \boldsymbol{Mod}(\Phi) \cap \mathbb{C}$.*

Character of FO-definable sets

Definition 115 (n-types) *Suppose $\mathcal{A}$ is a model with domain A. Let $a \in A$. The* 1-type of a in $\mathcal{A}$ *is:*

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

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:*

Slide 124

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

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

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

Note that $\mathbf{tp}(\mathcal{A}, a)$ is just simplified notation for $\mathbf{tp}(\mathcal{A}, \langle a\rangle)$.

We can generalize this across models.
$\mathcal{B}, [\vec{x} \mapsto \vec{b}] \models \mathbf{tp}(\mathcal{A}, \vec{a})$ 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 $\mathcal{A}$:

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

Slide 125

$S^0(\mathcal{A})$ is the set of 0-types that are realized in $\mathcal{A}$.
The 0-types 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. Let $\mathcal{A} = \langle \mathbb{N}, \mathrm{LT} \rangle$, where
$\mathrm{LT} = \{\langle i, j \rangle \mid i < j\}$.

Suppose $i < j$. For $j \geq 0$, let

Slide 126

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

$\varphi_j(x_0) \in \mathbf{tp}(\mathcal{A}, j)$ but $\varphi_j(x_0) \notin \mathbf{tp}(\mathcal{A}, i)$
Therefore: infinitely many types are realized in $\mathcal{A}$.

Quantifier Rank

Definition 116 (Quantifier Rank)

Slide 127

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

(r, n)-types

Definition 117 ((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:*

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

Slide 128

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:*

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

*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:*

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

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 $s \leq r$:

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

Slide 129

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

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

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:

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

Lemma 118

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

and, more generally,

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

Slide 130

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,

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

Lemma 119 $\boldsymbol{tp}_0(\mathcal{A}, \langle a_0, \ldots, a_{n-1}\rangle) = \boldsymbol{tp}_0(\mathcal{B}, \langle b_0, \ldots, b_{n-1}\rangle)$ *iff*

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

Slide 131

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 120 *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 121 *The number of distinct (r, n)-types realizable in any class of relational models is finite.*

Slide 132

Corollary 122 *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 $\chi^{\mathcal{A},\vec{a}}_r(\vec{x})$ which characterizes $\boldsymbol{tp}_r(\mathcal{A}, \vec{a})$:*

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

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

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

Slide 133

Theorem 123 *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 $\mathcal{B}$*

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

Concatenation and Types

Slide 134

Lemma 124

If $\quad \boldsymbol{tp}_r(\mathcal{A}, \vec{a}) = \boldsymbol{tp}_r(\mathcal{B}, \vec{b})$ *and* $\boldsymbol{tp}_r(\mathcal{C}, \vec{c}) = \boldsymbol{tp}_r(\mathcal{D}, \vec{d})$

then $\quad \boldsymbol{tp}_r(\mathcal{A} \cdot \mathcal{C}, \vec{a} \cdot \vec{c}) = \boldsymbol{tp}_r(\mathcal{B} \cdot \mathcal{D}, \vec{b} \cdot \vec{d})$.

Corollary 125

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

LT and FO

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

$$w \models \varphi_f \Leftrightarrow w \models f \text{ as a } k\text{-expression.}$$

Slide 135

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

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

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.

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

Theorem 126 *The class of FO-definable sets of $\lhd^+$-string models is closed under concatenation.*

Suppose L_1 and L_2 are both defined by First-Order formulae of the general form:

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

Slide 136

where the actual work of the definition is done by $\varphi_i(x_{\min}, x_{\max})$.

$$\begin{array}{l}(\exists z_{\min}, z_{\max})[\\ \quad \neg(\exists y)[y \lhd z_{\min} \vee z_{\max} \lhd y] \wedge \\ \quad (\exists z_1, z_2)[z_{\min} \le z_1 \wedge z_1 \lhd z_2 \wedge z_2 \le z_{\max} \wedge \\ \qquad \varphi_1(z_{\min}, z_1) \wedge \varphi_2(z_2, z_{\max}) \qquad\qquad]\ ]\end{array}$$

Locally Testable with Order

Definition 127 (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*

Slide 137

$$w \models \varphi \bullet \psi \overset{def}{\iff} w = w_1 \cdot w_2, \quad w_1 \models \varphi \text{ and } w_2 \models \psi.$$

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.*

FO($\lhd^+$) and LTO

Theorem 128 *A stringset is First-Order definable relative to the class of finite $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ models (in $\mathrm{FO}(\lhd^+)$) iff it is* LTO.

Slide 138

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 $\lhd$ does not occur, as it is FO definable from $\lhd^+$.

Basis: Quantifier rank 0

Slide 139

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

Induction step

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

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

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

Slide 140

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 129 *Every FO($\lhd^+$) definable stringset is the union of a finite set of concatenations of* SL_2 *stringsets of the form:*

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

Slide 141

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

Theorem 130 *Emptiness of FO($\lhd^+$) definable stringsets is decidable*

Proof Suppose L is FO($\lhd^+$) definable. From Corollary 129 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($\lhd^+$) stringsets reduces to emptiness of SL_2 stringsets which we know to be decidable. ⊣

Slide 142

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 $\emptyset$.

Slide 143

Theorem 131 *Finiteness of FO($\lhd^+$) definable stringsets is decidable.*

Proof: exercise

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

Proof: exercise

Cognitive interpretation of FO($\lhd^+$)

Slide 144

- Any cognitive mechanism that can distinguish member strings from non-members of an FO($\lhd^+$) 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($\lhd^+$) 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.

FO($\lhd^+$) is not learnable

Theorem 133 *FO($\lhd^+$) is not learnable in the limit from positive data.*

Slide 145

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

Theorem 134 LTO_k *is not learnable in the limit from positive data for any* $k \geq 2$.

Proof: (exercise)

Star-Free Sets

Definition 135 (Star-Free Set) *The class of* ***Star-Free Sets*** *(*SF*) is the smallest class of stringsets satisfying:*

Slide 146

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

Theorem 136 (McNaughton and Papert) *A set of strings is Locally Testable with Order (*LTO*) iff it is Star-Free.*

Corollary 137 (McNaughton and Papert) *A set of strings is First-order definable relative to the class of finite* $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ *models iff it is Star-Free.*

Slide 147

SF $\Rightarrow$ LTO:
Each base case is SL_2. LTO is closed under union, concatenation and complement.

LTO $\Rightarrow$ FO($\lhd^+$): Theorem 128

FO($\lhd^+$) $\Rightarrow$ SF:
$\{w \in \Sigma^* \mid |w| \geq 1\} = \overline{\{\varepsilon\}}$

Non-counting stringsets

Theorem 138 (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*,*

Slide 148

if $uv^n w$ *occurs in L*

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

Corollary 139 (McNaughton and Papert) *A set of strings is First-Order definable relative to the class of finite* $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ *models (in FO(*$\lhd^+$*)) iff it is non-counting.*

A non-counting stringset

Slide 149

$$\frac{\text{my } \overbrace{\text{father's}\,\text{father's}} \text{ father resembled my father} \quad \in L}{\text{my } \underbrace{\text{father's}\,\text{father's}}\ \underbrace{(\text{father's})}_{\geq 1} \text{ father resembled my father} \quad \in L}$$

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

Slide 150

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

$$\vdots$$

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

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

Slide 151

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 139 not FO($\lhd^+$) 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($\lhd^+$) definable.

FO($\lhd$) definable stringsets

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

First-order Quantification (over positions in the strings)

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

Slide 152

Theorem 140 *The fixed and universal recognition problems for FO($\lhd$) definable stringsets are decidable.*

Theorem 141 *Emptiness, finiteness and universality of FO($\lhd$) definable stringsets are decidable.*

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

Example 142

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

Slide 153

picks out the set of strings in which f occurs at least t times. The negation of (142) 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.

Locally Threshold Testable stringsets

Definition 143 (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 $w \in L \iff v \in L$.

Slide 154

Theorem 144 (Thomas) *A set of strings is First-order definable relative to the class of finite $\langle \mathcal{D}, \lhd, P_\sigma \rangle_{\sigma \in \Sigma}$ models (in FO($\lhd$)) iff it is Locally Threshold Testable.*

Example 145 *The stringset of Example 96 is* LTT *(as well as non-counting, hence* LTO*) but not, as we saw,* LT*. To see that it is* FO($\lhd$) *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 146 *FO($\lhd$)$\subsetneq$FO($\lhd^+$).*

Slide 155 **Proof**

$$B\text{-before-}C \stackrel{\text{def}}{=} \{a^i b a^j c a^k \mid 0 \leq i, j, k\}$$

$$a^k b a^k c a^k \in B\text{-before-}C \qquad \text{and} \qquad a^k c a^k b a^k \notin B\text{-before-}C$$

but these have exactly the same number of occurrences of each of their k-factors. Hence, regardless of the value of the threshold, B-before-C is not LTT. To see that it is LTO note that it is the concatenation of three stringsets each of which are LT:
$\{a^i b \mid 0 \leq i\} \cdot \{a^j c \mid 0 \leq j\} \cdot \{a^k \mid 0 \leq k\}$ ⊣

Slide 156 **(Exercise)** Show that $\lhd^+$ is not definable from $\lhd$.

Cognitive interpretation of FO($\lhd$)

Slide 157

- Any cognitive mechanism that can distinguish member strings from non-members of an FO($\lhd$) 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($\lhd$) stringsets.

Cognitive interpretation of FO($\lhd^+$) (reprise)

Slide 158

- Any cognitive mechanism that can distinguish member strings from non-members of an FO($\lhd^+$) 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.

FO(◁) is not learnable

Theorem 147 *FO(◁) is not learnable*

Slide 159

Since it extends LT.

Theorem 148 $\mathrm{LTT}_{k,t}$ *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.

Slide 160

Monadic Second-Order Languages for Strings

MSO (Strings)

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

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

Slide 161

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

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

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

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

Semantics of MSO languages

Definition 149 (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 $\mathcal{A}$.*

Slide 162

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:

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

Definition 150 (Satisfaction) $\mathcal{A}, s \models \varphi$ *iff one of the following holds:*

- $\varphi =$ *'*$x \lhd y$*', $s(x)$ and $s(y)$ are both defined and $s(y) = s(x) + 1$,*
- $\varphi =$ *'*$x \lhd^+ y$*', $s(x)$ 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 $\mathcal{A}$, $\mathcal{A}, s[X \mapsto S] \models \psi$.*

Slide 163

Recognition for MSO definable stringsets

Slide 164

Theorem 151 *The fixed and universal recognition problems for MSO definable sets is decidable.*

Recognition = satisfaction. Satisfaction is decidable.

B-before-C is MSO definable

$(\exists X_0)[\quad (\forall x)[X_0(x) \leftrightarrow ((\forall y)[\neg y \lhd x] \vee (\exists y)[X_0(y) \wedge A(y) \wedge y \lhd x]) \,]\,] \wedge$

—X_0 contains the minimum point and

everything up to the first non-'a' which follows it

$(\exists X_1)[\quad (\exists x)[B(x) \wedge (\forall y)[B(y) \rightarrow y \approx x] \wedge X_1(x) \wedge (\forall y)[y \lhd x \rightarrow \neg X_1(y)] \,]$

—There is exactly one 'b' and it is the minimum point in X_1

$(\exists x)[C(x) \wedge (\forall y)[C(y) \rightarrow y \approx x] \wedge X_1(x) \wedge (\forall y)[x \lhd y \rightarrow \neg X_1(y)] \,]$

—There is exactly one 'c' and it is the maximum point in X_1

Slide 165

$(\forall x)[\;\; X_1(x) \leftrightarrow (\quad B(x) \vee C(x) \vee$

—X_1 contains the 'b' and the 'c'...

$(\exists y)[X_1(y) \wedge (A(y) \vee B(y)) \wedge y \lhd x] \vee$

—... and the 'a's following the 'b'...

$(\exists y)[X_1(y) \wedge (A(y) \vee C(y)) \wedge x \lhd y]$

—... and the 'a's preceding the 'c'

$\,] \wedge$

$(\exists X_2)[\quad (\forall x)[X_0(x) \leftrightarrow ((\forall y)[\neg x \lhd y] \vee (\exists y)[X_0(y) \wedge A(y) \wedge x \lhd y]) \,]\,]$

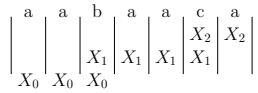
—X_2 contains the maximum point and

everything up to the first non-'a' which precedes it)

Slide 166 **(Exercise)** Show that Even-B is MSO definable.

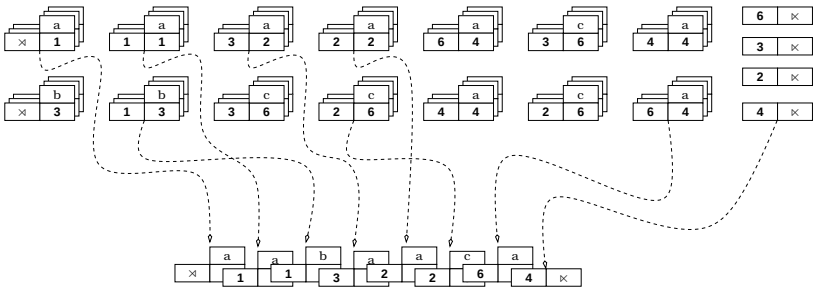
Satisfying assignments for X_0, X_1 and X_2

Slide 167



Generating sets of satisfying assignments

Slide 168



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

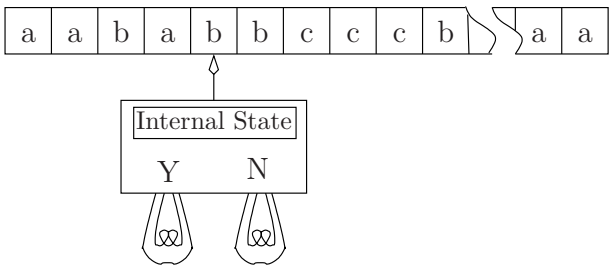
Finite-state automata

Definition 152 (Nondeterministic Finite-state Automaton)
*A **Nondeterministic Finite-state Automaton** (NFA) is a 5-tuple* $\langle Q, \Sigma, q_0, \delta, F \rangle$ *where:*

Slide 169

- 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).*

Slide 170



Recognizable stringsets

Definition 153 *A **computation** 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:*

Slide 171

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

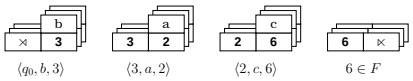
*If, in addition, $q = q_0^{\mathcal{M}}$ and $q_n \in F^{\mathcal{M}}$, then the computation is **accepting**.*

Definition 154 *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.*

Slide 172

Definition 155 *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 $\mathcal{M}$.*

Automata and tiling systems



Slide 173

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.

Theorem 156 (Chomsky Schützenberger) *A set of strings is recognizable iff it is a projection of a Strictly 2-Local set.*

Deterministic Finite-state Automata

Slide 174

Definition 157 (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 158 *Every FSA is equivalent, in the sense of recognizing the same language, to a DFA.*

Powerset construction

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

Slide 175

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

Claim 159

1. *$\widehat{\mathcal{M}}$ is deterministic.*
2. *Let*

$$\delta^*_{\mathcal{M}}(q, w) \stackrel{def}{=} \begin{cases} \{q\} & \text{if } w = \varepsilon, \\ \{q_i \mid \text{There is a computation of } \mathcal{M} \text{ on } w \\ \qquad\qquad \text{from } q \text{ ending in state } q_i\ \} \\ \text{otherwise.} \end{cases}$$

Slide 176

and

$$\hat{\delta}^*(\hat{q}, w) \stackrel{def}{=} \begin{cases} \hat{q} & \text{if } w = \varepsilon, \\ \hat{q}_i & \text{such that the computation of } \widehat{\mathcal{M}} \text{ on } w \\ & \qquad \text{from } \hat{q} \text{ ends in state } \hat{q}_i \\ & \text{otherwise.} \end{cases}$$

Then $\hat{\delta}^*(\hat{q}_0, w) = \delta^*_{\mathcal{M}}(q_0, w)$.

Closure properties

Lemma 160 *The class of recognizable stringsets is closed under Boolean operations.*

Slide 177

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?

Projection and cylindrification

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

Cylindrification: inverse projection

Slide 178

Lemma 161 *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 $\langle q_i, c, q_j\rangle \mapsto \langle q_i, a, q_j\rangle, \langle q_i, b.q_j\rangle$.

Character of recognizable sets

Definition 162 (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 163 (Myhill-Nerode) *: A stringset L is recognizable iff $\equiv_L$ partitions the set of all strings over Σ into finitely many equivalence classes.*

Slide 179

Proof ($\Rightarrow$)
L recognizable $\Rightarrow L = L(\mathcal{M})$ for some FSA $\mathcal{M}$. Wlog, by Theorem 158, $\mathcal{M}$ 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.

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:

Slide 180

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

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

MSO Quantifier Rank

Definition 164 (MSO Quantifier Rank)

Slide 181

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

MSO types

Definition 165 ((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:*

Slide 182

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

*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:*

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

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

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

Slide 183

Lemma 166 *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 167 *The number of distinct (r, m, n)-types realizable in any class of relational models is finite.*

Corollary 168 *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 $\chi_r^{\mathcal{A},\vec{A},\vec{a}}(\vec{X}, \vec{x}) \in \boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a})$

Slide 184

Theorem 169 *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 170

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

Concatenation and MSO types

Slide 185

Lemma 171

If $\boldsymbol{tp}_r(\mathcal{A}, \vec{A}, \vec{a}) = \boldsymbol{tp}_r(\mathcal{B}, \vec{B}, \vec{b})$ *and* $\boldsymbol{tp}_r(\mathcal{C}, \vec{C}, \vec{c}) = \boldsymbol{tp}_r(\mathcal{D}, \vec{D}, \vec{d})$
then $\boldsymbol{tp}_r(\mathcal{A} \cdot \mathcal{C}, \vec{A} \cdot \vec{C}, \vec{a} \cdot \vec{c}) = \boldsymbol{tp}_r(\mathcal{B} \cdot \mathcal{D}, \vec{B} \cdot \vec{D}, \vec{b} \cdot \vec{d})$.

Corollary 172

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

Recognizability characterizes MSO-definability

Slide 186

Theorem 173 (Medvedev, Büchi, Elgot) *A set of strings is MSO-definable relative to the class of finite* $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ *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 172, 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

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

Slide 187 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 167, 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
Slide 188 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:

$$\varphi_{\mathcal{A}} \stackrel{\text{def}}{=} (\exists Q)[\varphi'_{\mathcal{A}}(Q)]. \tag{174}$$

⊣

Corollary 175 *A stringset L over and alphabet Σ is MSO definable over $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ iff $\equiv_L$ partitions the set of all strings over Σ into finitely many equivalence classes.*

Slide 189

Corollary 176 *Every MSO sentence over $\langle W, \lhd, \lhd^+, P_\sigma \rangle_{\sigma \in \Sigma}$ is logically equivalent to an MSO sentence of the form:*

$$(\exists X_1, \ldots, X_{n-1})[\varphi(X_1, \ldots, X_{n-1})] \tag{177}$$

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

Definability $\Rightarrow$ Recognizability

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

Slide 190

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

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

Accepting Atomic Formulae

Slide 191

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

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

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

Extending to Arbitrary Formulae

Slide 192

- $\vee$, $\wedge$ — 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

Decision problems for MSO

Slide 193

Lemma 178 *Emptiness of MSO-definable stringsets is decidable.*

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

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

Cognitive interpretation of MSO

Slide 194

- 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.

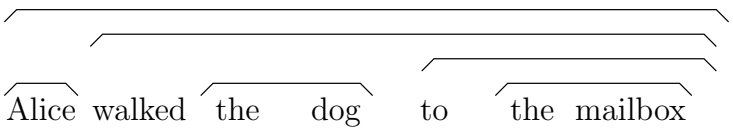
Slide 195

Trees

Slide 196

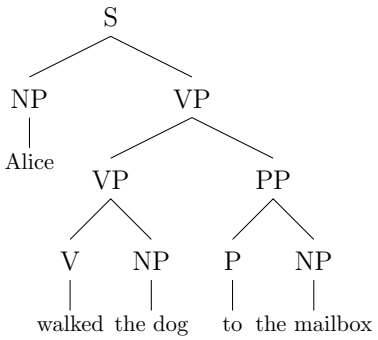
Extending the Language

- Weak insufficiency
- Descriptive insufficiency



Slide 197

Phrase Structure

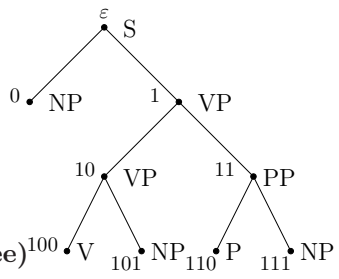


Slide 198

Σ-labeled Trees

Definition 181 (Tree-Domain)
$T \subset \{i \mid i < n\}^*$ *where*

- *For all* $w, v \in \{i \mid i < n\}^*$, $wv \in T \Rightarrow w \in T$
- *For all* $w \in \{i \mid i < n\}^*$, $j < i < n$, $wi \in T \Rightarrow wj \in T$



Definition 182 (Σ-labeled Tree)
$\mathcal{T} = \langle T, \tau \rangle$, *where*
T *is an* n*-branching tree-domain and* $\tau : T \to \Sigma$ *a labeling function.*

$\mathbb{T}_\Sigma$ is the set of all Σ-labeled trees

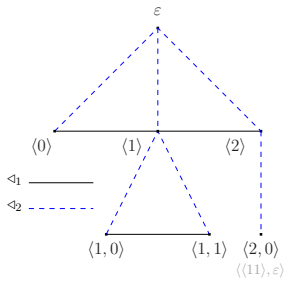
$\mathbb{T}_\Sigma^n$ is the set of all n-branching Σ-labeled trees

Slide 199

Tree Models

$\langle T, \lhd_1, \lhd_1^+, \lhd_2, \lhd_2^+, P_\sigma \rangle_{\sigma \in \Sigma}$

$T \subseteq$	—	Finite Tree domain
$\lhd_1$	—	Immediate left-of (global)
$\lhd_1^+$	—	Left-of (global)
$\lhd_2$	—	Immediate domination
$\lhd_2^+$	—	Proper domination
P_σ	—	Partition $\mathcal{D}$

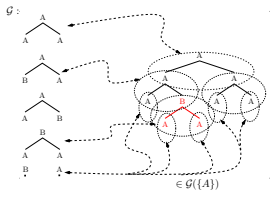


Σ-labeled Tree:

$$\mathcal{T} = \langle T, \tau \rangle,\ \tau : T \to \Sigma = \{x \mapsto \sigma \mid x \in P_\sigma\}$$

Slide 200

Local Tree Grammars



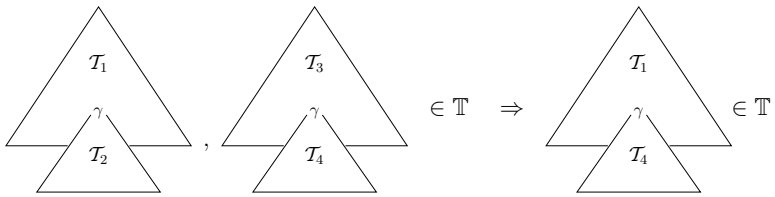
A *Local Tree Grammar* $\mathcal{G}$ over Σ is a finite set of local (height ≤ 1) Σ-labeled trees.

The set of Σ-labeled trees licensed by $\mathcal{G}$ relative to some set of start labels $S \subseteq \Sigma$ is: $\mathcal{G}(S) \stackrel{\text{def}}{=} \{\mathcal{T} \mid \text{LT}(\mathcal{T}) \subseteq \mathcal{G},\ \tau(\varepsilon) \in S\}$

$$\text{LTG} \leq \text{FO}(\lhd_2^+)$$

Slide 201

Subtree Substitution Closure

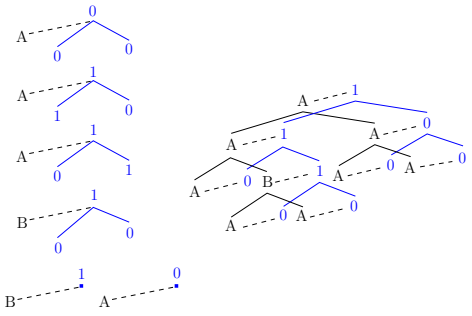


Theorem 183 *A set of labeled trees is Local iff it is closed under substitution of subtrees rooted at similarly labeled points.*

Slide 202

Tree Automata

A *Tree Automaton* over alphabet Σ and state set Q is a finite set $\mathcal{A} \subseteq \Sigma \times \mathrm{LT}(\mathbb{T}_Q)$.

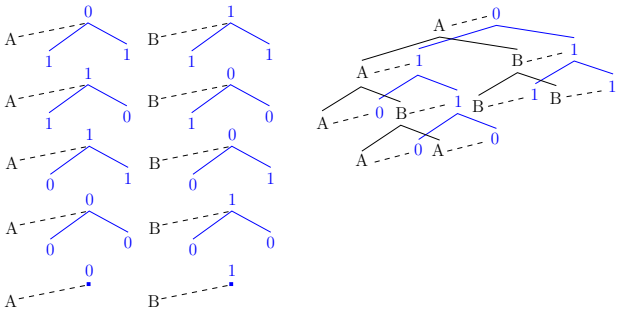


OneB: $\mathcal{A}(\{1\}) = \{\mathcal{T} \in \mathbb{T}_{\{A,B\}} \mid |\mathcal{T}|_B = 1\}$

$\mathrm{LTG} \lneq \mathrm{FO}(\lhd_2^+)$

Slide 203

Tree Automata



EvenB: $\mathcal{A}(\{0\}) = \{\mathcal{T} \in \mathbb{T}_{\{A,B\}} \mid |\mathcal{T}|_B \equiv 0 \pmod 2\}$

$$\text{LTG} \lneq \text{FO}(\lhd_2^+) \lneq \text{Reg (trees)}$$

Slide 204

A Myhill-Nerode Characterization

Theorem 184 *Suppose $\mathbb{T} \subseteq \mathbb{T}_\Sigma$. For all $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{T}_\Sigma$, let $\mathcal{T}_1 \equiv_{\mathbb{T}} \mathcal{T}_2$ iff, for every tree $\mathcal{T} \in \mathbb{T}_\Sigma$ and point s in the domain of $\mathcal{T}$, the result of substituting $\mathcal{T}_1$ at s in $\mathcal{T}$ is in $\mathbb{T}$ iff the result of substituting $\mathcal{T}_2$ is:*

$$\mathcal{T} \xleftarrow{s} \mathcal{T}_1 \in \mathbb{T} \iff \mathcal{T} \xleftarrow{s} \mathcal{T}_2 \in \mathbb{T}.$$

Then $\mathbb{T}$ is recognizable iff $\equiv_{\mathbb{T}}$ has finite index.

FO, MSO—Trees

Theorem 185 (Thatcher) *A set of Σ-labeled trees is recognizable iff it is a projection of a local set of trees.*

Theorem 186 (Thatcher and Wright, Doner) *A set of Σ-labeled trees is definable in MSO over trees iff it is recognizable.*

Slide 205

$$\mathrm{LTG} \lneq \mathrm{FO}(\lhd_2^+) \lneq \mathrm{MSO}(\lhd_2^+) = \mathrm{Reg} \qquad \text{(trees)}$$

Theorem 187 (Thatcher) *A set of strings L is the yield of a local set of trees (equivalently, is the yield of a recognizable set of trees) iff it is Context-Free.*

Corollary 188 *A set of strings L is the yield of a MSO (or FO) definable set of trees iff it is Context-Free.*

Parsing Model-Theoretic Grammars

Parsing string grammars

$$L(\varphi) = \{w \mid w \models \varphi\}$$

Parsing = satisfaction (model checking)

Slide 206

Parsing tree grammars

$$L(\varphi) = \{\mathrm{Yield}(\mathcal{T}) \mid \mathcal{T} \models \varphi\}$$

Let: $\psi_w \stackrel{\text{def}}{=}$ "yield of $\mathcal{T}$ is w".

Then: $\{\mathcal{T} \mid \mathcal{T} \models \psi_w \wedge \varphi\} =$ parse forest for w.

Recognition = satisfiability of $\psi_w \wedge \varphi$

FO—Trees

FO(+1): $\langle T, \lhd_1, \lhd_1^+, \lhd_2, P_\sigma \rangle_{\sigma\in\Sigma}$

Theorem 189 (Benedikt and Segoufin) *A regular set of trees is definable in FO(+1) over trees iff it is Locally Threshold Testable.*

Theorem 190 (Benedikt and Segoufin) *A regular set of trees is definable in FO(+1) over trees iff it is aperiodic.*

Slide 207

FO(mod):

$$\mathcal{T} \models (\exists^{r,q} x)[\varphi(x,\vec{y})] \stackrel{\text{def}}{\iff}$$

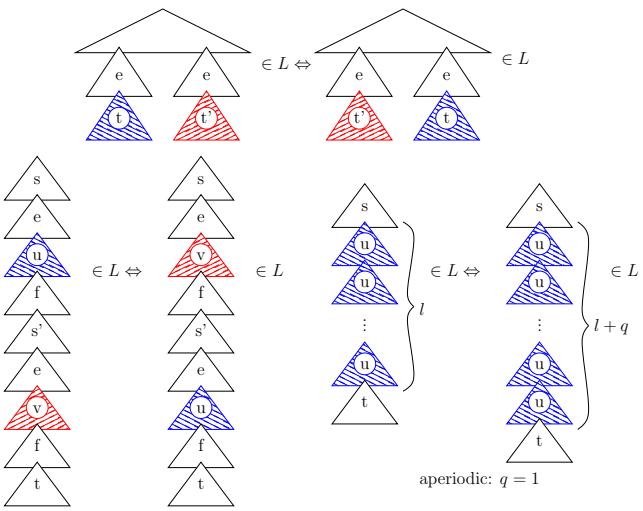
$$\mathbf{card}(\{a \mid \mathcal{T} \models \varphi(x,\vec{y})[x \mapsto a]\}) \equiv r \pmod q$$

Theorem 191 (Benedikt and Segoufin) *A regular set of trees is definable in FO(mod) over trees iff it is q-periodic.*

$$\text{LTG} \lneq \text{FO}(+) \lneq \text{FO}(\text{mod}) \lneq \text{FO}(<) \lneq \text{MSO} = \text{Reg.} \quad \text{over trees}$$

Aperiodic/q-periodic Regular Tree Languages

Slide 208



MSO and SF—trees

Theorem 192 (Thatcher and Wright, Doner)
MSO over trees = ∃MSO over trees.

Theorem 193 (Thomas)
MSO = "Anti-chain" MSO over trees without unary branching.
MSO = "Frontier" MSO over trees without unary branching.

Slide 209

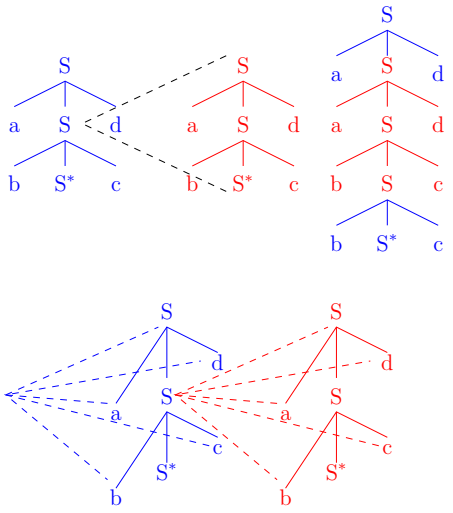
Theorem 194 (Thomas)
Every Regular tree language without unary branching is Star-Free.
Regular tree languages without unary branching are of uniformly bounded dot depth.

Without unary branching:

$$\mathrm{LTG} \lneq \mathrm{FO}(+1) \lneq \mathrm{FO}(\mathrm{mod}) \lneq \mathrm{FO}(<) \lneq \mathrm{SF} = \mathrm{MSO} = \mathrm{Reg}.$$

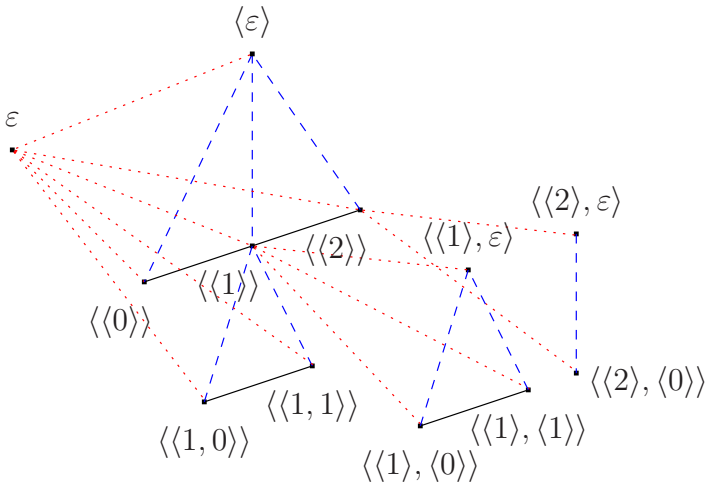
Beyond CFLs

Slide 210



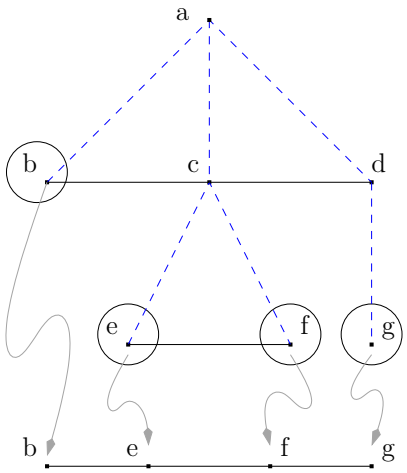
Slide 211

3-Dimensional Domains



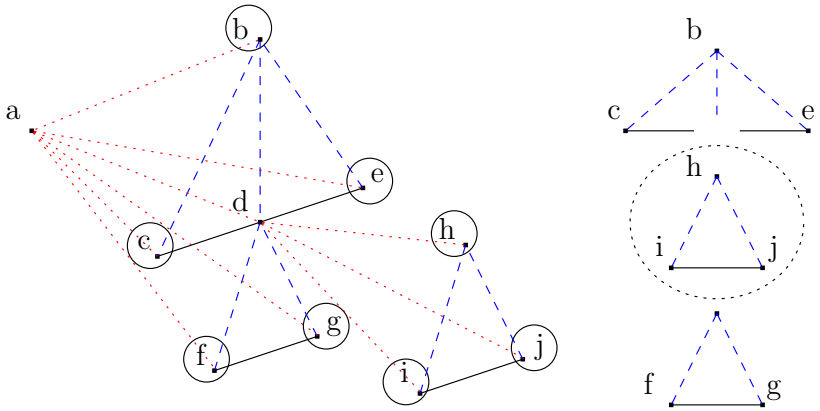
Slide 212

Yields of T2



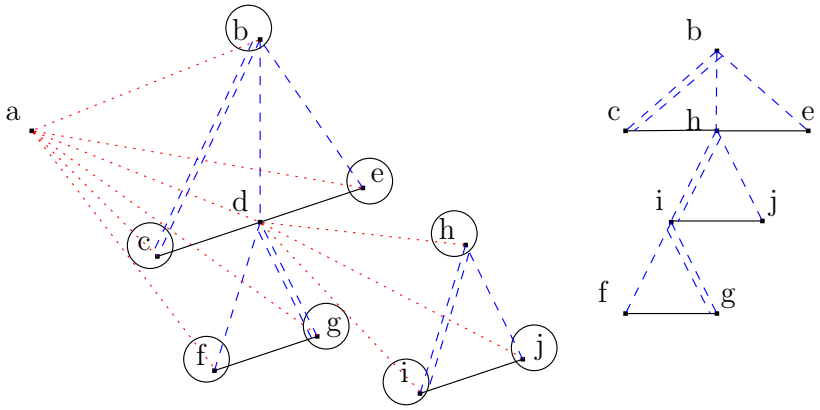
Slide 213

Yields of T3



Slide 214

Headed Structures



Σ-Labeled Headed T3

Slide 215

Definition 195 *A* Σ-Labeled Headed T3 *is a structure:*

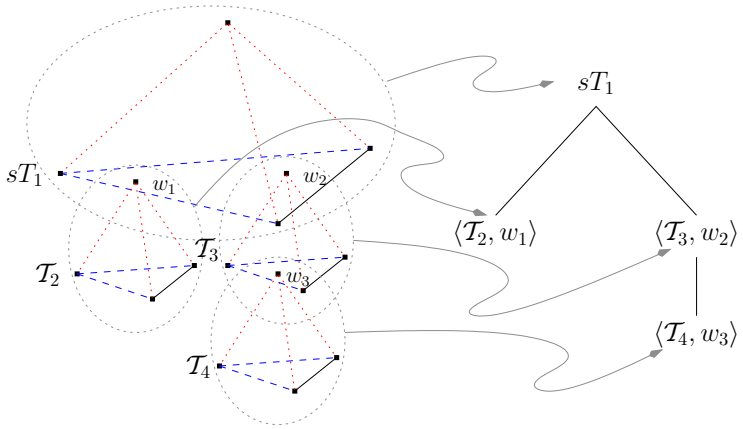
$$\mathcal{T} = \langle T, \lhd_i^+, R_i, H_i, P_\sigma \rangle_{1 \leq i \leq 3, \sigma \in \Sigma},$$

- P_σ—points labeled σ.
- R_i—roots of i-dimensional component structures.
- H_i—i-dimensional heads,
  - one on the principle spine of each $(i-1)$-dimensional component.
- $\lhd_i^+$—"inherited" proper domination

Theorem 196 *A set of* Σ*-labeled Headed T3 is MSO definable iff it is recognizable.*

Local Sets and Derivation Trees

Slide 216



Non-Strict TAGs and T3-Automata

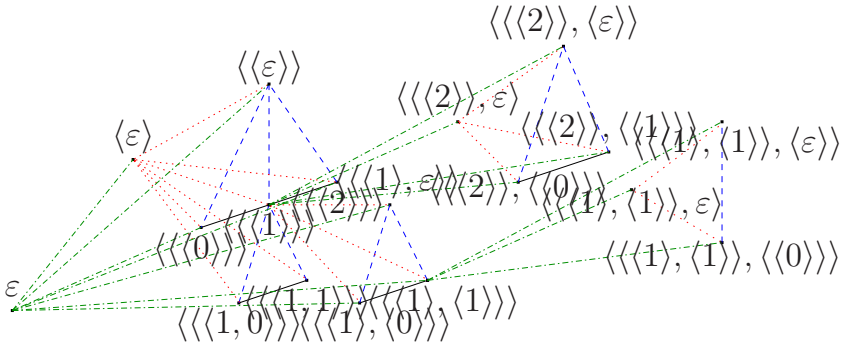
Slide 217

Theorem 197 *A set of Σ-labeled trees is the yield of a recognizable set of Σ-labeled T3 iff it is generated by a non-strict TAG with adjoining constraints.*

T3 Automata and Non-Strict TAGs with adjoining constraints are, in essence, just notational variants.

Higher-Dimensional Domains

Slide 218



Labeled Distinguished Grammars

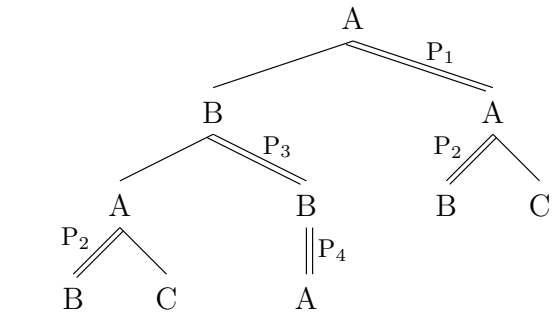
Slide 219

$P_1: \quad A \longrightarrow B\underline{A}$

$P_2: \quad A \longrightarrow \underline{B}C$

$P_3: \quad B \longrightarrow A\underline{B}$

$P_4: \quad B \longrightarrow \underline{A}$



$CW(T) = \{P_1P_2, P_3P_4, P_2\}$

The Control Language Hierarchy (Weir'92)

Slide 220

$$L(G,C) \stackrel{\text{def}}{=} \{\text{Yield}(\mathcal{T}) \mid \mathcal{T} \in T(G) \text{ and } \text{CW}(\mathcal{T}) \subseteq C\}$$

$\mathcal{C}_1$: CFL ($= L(G,C)$ for C Regular).

$\mathcal{C}_{i+1}$: $L(G,C)$ for $C \in \mathcal{C}_i$.

Theorem 198 *A string language is* $\text{Yield}^1_d(\mathbb{T})$ *for some* $\mathbb{T}$*, a recognizable set of Td,* $d \geq 2$*, iff it is in* $\mathcal{C}_{d-1}$*.*

Higher-Dimensional Grammars

Slide 221

Theorem 199 (Recognizable Sets and the CLH) *A string language is* $\mathrm{Yield}_d^1(\mathbb{T})$ *for some* $\mathbb{T}$*, a recognizable set of Td,* $d \geq 2$*, iff it is in* $\mathcal{C}_{d-1}$*.*

Theorem 200 *A set of* Σ*-labeled Headed Td is MSO definable iff it is recognizable.*

Corollary 201 *A string language is* $\mathrm{Yield}_d^1(\mathbb{T})$ *for some* $\mathbb{T}$*, a MSO definable set of Td,* $d \geq 2$*, iff it is in* $\mathcal{C}_{d-1}$*.*