Slide 1

Model Theoretic Phonology

James Rogers (Earlham)
Jeffrey Heinz (Delaware)

Course administration

- Slides with notes are posted on the ESSLLI WIKI: `http://esslli2014.info/wiki/topics-in-model-theoretic-phonology/` and `http://udel.edu/~heinz/esslli14/`
- Questions? Please ask us in class, outside of class, or by email.
  - `jrogers@cs.earlham.edu`
  - `heinz@udel.edu`

Slide 2

Model-Theoretic Phonology

- Models define structures and model theory allows one to study theories of these structures. What kind of statement can the theory make and what kind can't it make?
- Phonology is a linguistics subfield which studies the mental structures of speech sounds and the pronunciation of words. What kinds of statements do phonological theories need to make? What is the right theory of phonology?
- In this course, we study phonological words from a model-theoretic perspective.

Slide 3

What we cover in this course

Part 1 (Today). Foundations of formal language theory, model theory and phonology.

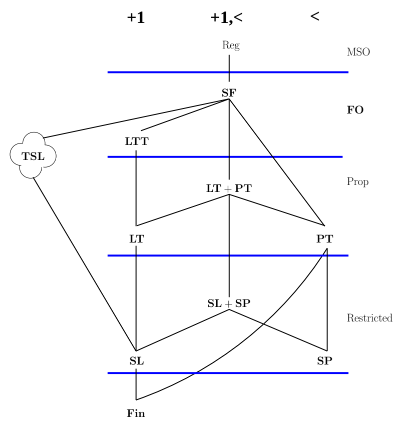
Part 2 Patterns of stress and accent, Strictly Local languages, and learnability.

Part 3 Language families defined with Successor under Propositional, First-Order and Monadic Second-Order logic.

Part 4 Harmony, Language families defined with Precedence under Propositional, First-order and Second-order logic.

Slide 4

What we cover in this course (in pictures)



Model theory allows up to map the space of stringsets along two dimension: the nature of the signature (the horizontal dimension) and the nature of the logic (the vertical dimension).

The lines are illustrate which classes of stringsets properly contain the others (and is closed under transitivity). So for instance the Locally Threshold Testable class properly contains the Locally Testable class, which properly contains the Finite class. This is equivalent to saying that any stringset definable with Propositional Logic with Successor word models is definable with First Order Logic with Successor word models, but not vice versa.

By the end of this course, this diagram will be familiar to you.

Fin	Finite
SL	Strictly Local
SP	Strictly Piecewise
LT	Locally Testable
PT	Piecewise Testable
LTT	Locally Threshold Testable
TSL	Tier-based Strictly Local
SF	Star Free
Reg	Regular

Slide 5

What we do *not* cover in this course

- Modal logic [PP02, Gra10]

Both of the above cited works apply modal logic in a model-theoretic setting to the study of phonology and phonological theory.

Modal logic very much complements the logics we cover here, and constitutes the subject matter of other courses here at ESSLLI.

Slide 6

Prerequisite knowledge

We will assume you have some knowledge of:

- Basic set theory and mathematical notion for functions

$$\cup,\ \cap,\ -,\ \times,\ \boldsymbol{\mathcal{P}}\ (i.e., powerset),\ f : A \to B$$

- Inductive Definitions
- Formal Language Theory
  - Regular Expressions
  - Grammars, such as Context-Free Grammars
  - Automata, such as Finite-State Automata
- Some familiarity with propositional and first-order logic.

With the preliminaries out of the way, let's get started!

Slide 7

Phonology

Three Aspects of Phonological Knowledge

1. *Phonotactic* knowledge
2. Knowledge of phonological *processes*
3. Knowledge of *contrast*

In this course, we will focus on (1) Phonotactics, and will not discuss (2) Processes or (3) contrast.

Slide 8

Phonotactic Knowledge - Knowledge of word well-formedness (1)

ptak thole hlad plast sram mgla vlas flitch dnom rtut

Halle, M. 1978. In *Linguistic Theory and Psychological Reality.* MIT Press.

Slide 9

Phonotactic Knowledge - Knowledge of word well-formedness (2)

possible English words	impossible English words
thole	ptak
plast	hlad
flitch	sram
	mgla
	vlas
	dnom
	rtut

Exercise 1 *How do English speakers know which of these words belong to different columns?*

They have knowledge they have learned, but it is untaught. What is the nature of this knowledge?

Slide 10

Phonotactics – Samala Version (1)

ʃtojonowonowaʃ

stojonowonowaʃ

stojonowonowas

ʃtojonowonowas

pisotonosikiwat

pisotonoʃikiwat

asanisotonosikiwasi

aʃanipisotonoʃikiwasi

Slide 11

Phonotactics – Samala Version (2)

possible Samala words	impossible Samala words
ʃtojonowonowaʃ	stojonowonowaʃ
stojonowonowas	ʃtojonowonowas
pisotonosikiwat	pisotonoʃikiwat
asanisotonoskiwasi	aʃanipisotonoʃikiwasi

Exercise 2 *How do Samala speakers know which of these words belong to different columns?*

Solution: Different types of sibilant sounds [ʃ,s] cannot co-occur in words.

By the way, *ʃtoyonowonowaʃ* means 'it stood upright' [App72]

Slide 12

Phonotactics – Language X

possible words of Language X	impossible words of Language X
ʃotkoʃ	sotkoʃ
ʃoʃkoʃ	ʃotkos
ʃosokoʃ	ʃoʃkos
soʃokos	soskoʃ
sokosos	
pitkol	
pisol	
piʃol	

Exercise 3 *How do speakers of Language X know which of these words belong to different columns?*

Solution: Sibilant sounds which begin and end words must agree (but not ones word medially).

Slide 13

Phonotactics – Language Y

possible words of Language Y	impossible words of Language Y
ʃotkoʃ	ʃoʃkoʃ
sotkoʃ	ʃoskoʃ
ʃotkos	soʃkos
pitkol	ʃoʃkos
soʃkostoʃ	soskoʃ
	soksos
	piskol
	piʃkol

Exercise 4 *How do speakers of Language Y know which of these words belong to different columns?*

Solution: Words must have an *even number* of sibilant sounds.

Slide 14

Typology

Attested Phonotactic Patterns

1. Words don't begin with [mgl]. (English)
2. Words don't contain both [ʃ] and [s]. (Samala)

Unattested Phonotactic Patterns

1. Words don't begin and end with disagreeing sibilants.
   (Language X = First/Last Harmony)
2. Words don't contain an even number of sibilants.
   (Language Y = Even-Sibilants)

Why are some logically possible patterns attested and others not?

Slide 15

Our Thesis

1. Phonology is constrained by computational complexity.
2. The model-theoretic perspective makes the levels of complexity clear.
3. The model-theoretic perspective helps make clear the cognitive functions at stake since the properties identified are *independent* of particular grammatical formalisms.

Wilhelm von Humboldt commented that in order to do typology, researchers need "an encyclopedia of categories" and "an encyclopedia of types." In this research program, the "encyclopedia of categories" is given by the model-theoretic analysis of formal languages and the "encyclopedia of types" comes from centuries of phonological analysis of natural languages.

Additionally, the model-theoretic perspective developed here can be extended to look at different kinds of structures, like trees [Rog94, Pul07, Gra13]. Working with strings provides a firm foundation upon which more complex linguistic structures can be studied.

So now let's turn to strings, languages, and grammars.

Slide 16

Strings and Stringsets

We assume a finite set of symbols, the alphabet Σ, and consider the monoid $(\Sigma, \cdot)$ where $\cdot$ is an associative, non-commutative operation called *concatenation* with λ as the identity element.

Thus,

$$\big(\forall u \in (\Sigma, \cdot)\big)\big[\lambda \cdot u = u \cdot \lambda = u\big]$$

Elements of $(\Sigma, \cdot)$ are defined inductively:

1. Base case: $\lambda \in (\Sigma, \cdot)$.
2. Inductive case: $u \in (\Sigma, \cdot) \wedge \sigma \in \Sigma \Rightarrow u \cdot \sigma \in (\Sigma, \cdot)$

We refer to elements of $(\Sigma, \cdot)$ as *strings*.

A *stringset* (=formal language) is a (possibly infinite) subset of $(\Sigma, \cdot)$.

The string λ itself is thus the unique string of length zero.

Slide 17

Concatenation and Kleene Star

We lift the definition of concatenation to stringsets. Following convention, we often leave out writing the operator $\cdot$ itself.

- If R and S are stringsets then $RS = \{uv \mid u \in R \wedge v \in S\}$.

Kleene star is another operation defined on stringsets.

- If S is a stringset then S^* is defined recursively:
  1. Base case: $\lambda \in S^*$.
  2. Recursive case: $w \in S^* \wedge v \in S \Rightarrow wv \in S^*$.

We observe $\Sigma^* = (\Sigma, \cdot)$, and so stringsets can also be said to be subsets of Σ^*.

Slide 18

Grammars and Languages

- Every grammar $\mathcal{G}$ we consider will be an object of finite size and will belong to a (possibly infinite) class of grammars $\mathbb{G}$.
- Grammars are associated to languages via a naming function.

$$\mathbb{L} : \mathbb{G} \to \mathcal{P}(\Sigma^*)$$

We give some examples with regular expressions.

Slide 19

Regular Expressions as Grammars

An RE is defined inductively as follows.

1. *The base cases:*
   - $\varnothing$ is an RE.
   - λ is an RE.
   - For all $\sigma \in \Sigma$, σ is an RE.
2. *The inductive cases:*
   - If R is an RE then so is (R^*).
   - If R and S are REs then so are $(R + S)$ and $(R \cdot S)$.
3. Nothing else is a regular expression.

Despite the choice of notation, the REs are just strings. As of yet they are 'meaningless' in the sense that they do not yet have any interpretation.

Slide 20

Regular Expressions - Stringsets

The naming function for REs $\mathbb{L}_{\mathrm{RE}}(\cdot)$ is inductively defined as follows:

1. *The base cases:*

$$\begin{aligned} \mathbb{L}_{\mathrm{RE}}(\varnothing) &\stackrel{\mathrm{def}}{=} \varnothing \\ \mathbb{L}_{\mathrm{RE}}(\lambda) &\stackrel{\mathrm{def}}{=} \{\lambda\} \\ \big(\forall \sigma \in \Sigma\big)\big[\mathbb{L}_{\mathrm{RE}}(\sigma) &\stackrel{\mathrm{def}}{=} \{\sigma\}\big] \end{aligned}$$

2. *The inductive cases:*

$$\begin{aligned} \mathbb{L}_{\mathrm{RE}}(R^*) &\stackrel{\mathrm{def}}{=} (\mathbb{L}_{\mathrm{RE}}(R))^* \\ \mathbb{L}_{\mathrm{RE}}(RS) &\stackrel{\mathrm{def}}{=} \mathbb{L}_{\mathrm{RE}}(R)\,\mathbb{L}_{\mathrm{RE}}(S) \\ \mathbb{L}_{\mathrm{RE}}(R+S) &\stackrel{\mathrm{def}}{=} \mathbb{L}_{\mathrm{RE}}(R) \cup \mathbb{L}_{\mathrm{RE}}(S) \end{aligned}$$

Definition 1 (Regular languages) *Stringsets definable with REs are the* regular languages *(**Reg**).*

The definition of REs gives the *syntax* of the objects in the class of grammars. The *semantics* is given by the definition of $\mathbb{L}_{\mathrm{RE}}$. We will follow this pattern throughout the course.

In the diagram, **Reg** stands at the top.

Slide 21

Generalized Regular Expressions — Grammars

GREs are REs extended with operators for intersection and complement

1. *Base cases*
   - If R is an RE then R is a GRE
2. *Inductive cases*
   - If R is a GRE then so is $(\overline{R})$.
   - If R and S are GREs then so is $(R \ \& \ S)$.
3. Nothing else is a generalized regular expression.

Slide 22

Generalized Regular Expressions — Stringsets

1. *The base cases:*

$$\big(\forall R \in RE\big)\big[\mathbb{L}_{\mathrm{GRE}}(R) \quad \stackrel{\mathrm{def}}{=} \quad \mathbb{L}_{\mathrm{RE}}(R)\big]$$

2. *The inductive cases:*

$$\begin{aligned} \mathbb{L}_{\mathrm{GRE}}(\overline{R}) \quad &\stackrel{\mathrm{def}}{=} \quad \Sigma^* - \mathbb{L}_{\mathrm{GRE}}(R) \\ \mathbb{L}_{\mathrm{GRE}}(R \ \& \ S) \quad &\stackrel{\mathrm{def}}{=} \quad \mathbb{L}_{\mathrm{GRE}}(R) \cap \mathbb{L}_{\mathrm{GRE}}(S) \end{aligned}$$

Lemma 1 (Equivalence of GREs and REs) *A stringset is definable with a GRE iff it is definable with an RE.*

The class of regular languages is closed under intersection and complement, hence GREs are syntactic sugar.

Note, however, that "syntactic sugar" does not mean "superfluous crutch". Generally expressions using '$\overline{}$' and '&' (i.e., negative and conjunctive constraints) may be *much* easier to write and comprehend (well, for most of us) than equivalent expressions written without them.

There are several conventions to note. For instance, $\cdot$, $+$, $\&$ are all associative so parentheses are often omitted. Often parentheses are omitted for * too, but it is understood to have precedence: So RS^* is always understood as $(R \cdot (S^*))$ and never as $(R \cdot S)^*$. We aren't going to dwell on this.

Slide 23

Star Free Expressions - Grammars and Stringsets

- A Star Free Expression is a GRE containing no 'And' (&) or Kleene star (*).

$$\boxed{\cdot,\ +,\ \overline{}}$$

- The language of an SFE is defined using the same naming function we used for defining the language of GREs.

Definition 2 (Star Free stringsets) *Stringsets definable with SFEs are the* Star Free *languages (**SF**).*

Theorem 1 (McNaughton and Papert 1971) ***SF*** $\subsetneq$ ***Reg***.

Closure under union and complement gives closure under intersection. Hence SFEs can be extended with & without extending the class of stringsets they define. Thus & is syntactic sugar for SFEs, and we will make use of & in SFEs.

That **SF** is subset of **Reg** is obvious from the definitions. That **Reg** is not a subset of **SF** is witnessed by Even-Sibilants. We will see a proof of this in a different form later.

Slide 24

Finite expressions - Grammars and languages

- A Finite Expression is an RE which contains no Kleene star.

$$\boxed{\cdot, \quad +}$$

- The language of a FE is defined using the same naming function we used for defining the language of REs.

Theorem 2 *The class of finite languages (**Fin**) are exactly those stringsets with finite cardinality. Every stringset definable with a FE is in **Fin**, and for every stringset in **Fin** there is a FE for it.*

Theorem 3 ***Fin*** $\subsetneq$ ***SF***.

Exercise 5

1. *For any finite expression E, $\mathbb{L}(E)$ has finite cardinality. Why?*
2. *Is **Fin** closed under intersection?*
3. *Is **Fin** closed under complement?*

Regarding Theorem 3, that **Fin** is a subset of **SF** is clear from the definitions. That it is a proper subset is witnessed by many examples, for instance $\mathbb{L}(\overline{\varnothing}) = \Sigma^*$ belongs to **SF** but not **Fin**.

In the diagram, **Fin** stands at the bottom.

Here is a summary.

Grammar	Operations	Language class
Generalized Regular expressions	$\cdot,\ +,\ ^*,\ \&,\ \overline{}$	**Reg**
Regular expressions	$\cdot,\ +,\ ^*$	**Reg**
Star Free expressions	$\cdot,\ +,\ \overline{}$	**SF**
Finite expressions	$\cdot,\ +$	**Fin**

Note that:

- **Reg** is the closure of **Fin** under concatenation, union and Kleene star.
- **SF** is the closure of **Fin** under concatenation, union and complement.

These expressions vary in which kinds of operators are permitted, which has consequences for the generative capacity. We can ask: which operators are necessary to describe human phonotactics? Model theory is a similar exercise, but exhibits a finer degree of control.

Slide 25

Word Models

We use the word 'word' synonymously with 'string.'

- A *model* of a word is a representation of it.
- A (Relational) Model contains two kinds of elements.

  A domain. This is a finite set of elements.

  Some relations over the domain elements.
- Guiding principles:
  1. Every word has some model.
  2. Different words must have different models.

Also, we are most interested in models which provide the minimum kind of information necessary to distinguish one word from another.

Note that relational models include only a domain and a finite number of relations, each of finite arity. In particular, there are no function symbols. We will accommodate (partial) n-ary functions (when necessary) as $(n+1)$-ary relations that are functional in their first n arguments, i.e., for each n-tuple of elements of the domain there is (at most) a single element of domain that extends it to an element of the relation.

Generally models are given in terms of their *signature*, which is a tuple containing the domain of the model and the relations.

$$\mathbb{M} = \langle \mathcal{D}, R_1, R_2, \ldots, R_n \rangle$$

Slide 26

Three Word models

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

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

$\mathcal{D}^{\mathbb{W}}$	—	Finite set of elements (positions)
$\triangleleft^{\mathbb{W}}$	—	immediate linear precedence on $\mathcal{D}$
$\triangleleft^{+\mathbb{W}}$	—	(arbitrary) linear precedence on $\mathcal{D}$
$P_\sigma^{\mathbb{W}}$	—	Subset of $\mathcal{D}$ at which σ occurs

Properly $\triangleleft$, etc., are symbols and $\triangleleft^{\mathbb{W}}$, etc., are sets, but usually there is no ambiguity and we will drop the superscript.

Three distinct models for words are shown here. The 'lower' two have less structure than the one on top. What is different between the three models is how they represent the *order* of symbols in words:

- $\triangleleft$ and $\triangleleft^+$ are binary relations. $\triangleleft$ represents the successor function on the domain, and $\triangleleft^+$ represents the less-than relation. Both linearly order the domain.
- The relations P_σ, one for each $\sigma \in \Sigma$, are unary relations over the domain, each picking out the subset of positions at which the symbol σ occurs. Normally the P_σ partition $\mathcal{D}$, but this is not actually necessary.

Slide 27

Example: $\mathbb{W}^{\triangleleft}$

Let $\Sigma = \{a, b\}$ and so $\mathbb{W}^{\triangleleft} = \langle D, \triangleleft, P_a, P_b \rangle$.

Consider the string *abbab*.

The model of *abbab* under the signature $\mathbb{W}^{\triangleleft}$ (denoted $\mathcal{M}^{\triangleleft}_{abbab}$) looks like this.

$$\mathcal{M}^{\triangleleft}_{abbab} = \left\langle \begin{array}{l} \{0, 1, 2, 3, 4\}, \\ \{(0, 1), (1, 2), (2, 3), (3, 4)\}, \\ \{0, 3\}, \\ \{1, 2, 4\} \end{array} \right\rangle$$

This says: There are five elements in the domain. Elements 0 and 1 stand in the (binary) successor relation. Elements 1 and 2 stand in the successor relation. . . Elements 0 stands in the (unary) relation P_a, as does element 3. Elements 1, 2, and 4 each stand in the unary relation P_b.

Exercise 6

1. *If we only considered signatures with a domain and no relations, could we distinguish different words?*
2. *If we left out the P_σ relations, could we distinguish different words?*
3. *If we left out the successor relation, could we distinguish different words?*

Slide 28

Example: $\mathbb{W}^{\triangleleft^+}$

Let $\Sigma = \{a, b\}$ and so $\mathbb{W}^{\triangleleft^+} = \langle D, \triangleleft^+, P_a, P_b \rangle$.
A model for *abbab* under the signature $\mathbb{W}^{\triangleleft^+}$ (denoted $\mathcal{M}^{\triangleleft^+}_{abbab}$) looks like this.

$$\mathcal{M}^{\triangleleft^+}_{abbab} = \left\langle \begin{array}{l} \{0,1,2,3,4\}, \\ \{(0,1),(0,2),(0,3),(0,4), \\ (1,2),(1,3),(1,4), \\ (2,3),(2,4),(3,4)\} \\ \{0,3\},\{1,2,4\} \end{array} \right\rangle$$

This says the same as before except the ordering is defined in terms the (arbitrary) linear precedence. Elements 0 and 1 stand in this relation. So do element 0 and 2. And elements 0 and 3. And so on.

How can we obtain models of strings? Here is a way for $\mathbb{W}^{\triangleleft}$. Consider any $w \in \Sigma^*$.

1. $\mathcal{D} \stackrel{\text{def}}{=} \{i \mid 0 \leq i < |w|\}$.

2. $\triangleleft \stackrel{\text{def}}{=} \{(i,j) \mid i \in \mathcal{D} \wedge j = i + 1\}$.

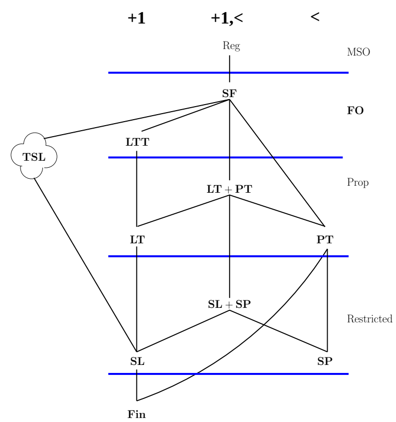
3. For all $\sigma \in \Sigma$, $P_\sigma \stackrel{\text{def}}{=} \{i \mid w_i = \sigma\}$.

(We let $|w|$ be the length of w and $|w|_i$ be the ith position in w. This notation can be defined more formally and recursively but we won't dwell on that.)

Exercise 7 *Write a way to obtain a model for strings with the signature* $\mathbb{W}^{\triangleleft^+}$. *(Hint: only part of 1 line needs to change.)*

Slide 29

Subregular Hierarchies



As we will see, we can describe four properly nested classes of languages with four different logics of increasing power when using the word models with successor and precedence:

$$\begin{array}{rllllllll} (+1): & \mathbf{SL} & — & \mathbf{LT} & — & \mathbf{LTT} & — & \mathbf{Reg} \\ (<): & \mathbf{SP} & — & \mathbf{PT} & — & \mathbf{SF} & — & \mathbf{Reg} \end{array}$$

Also we will see the following when looking at this way:

1. The English-style phonotactics is **SL**.
2. Samala Harmony is **SP**.
3. First-Last Harmony (Language X) is not **SL**, but is **LT**.
4. Even-Sibilants (Language Y) is not **LTT**, **PT** nor even **SF**, but is **Reg**.

Slide 30

Session 1 Summary

- Phonotactic knowledge can be described with stringsets. What *kinds* of stringsets are they?
- Generalized Regular Expressions, and restrictions thereof, can be used to define three classes of languages of decreasing generative capacity: **Reg**, **SF**, and **Fin**.
- Similarly, model theory allows us to study the nature of stringsets from two dimensions: the choice of signature and the power of the logic.
- One signature type uses the Successor relation to describe words.

Slide 31

Overview Session 2

Local Stringsets I

- Stress and accent patterns
- Strictly Local Stringsets
  - Grammar-theoretic definition
  - Automata-theoretic characterization
  - Abstract (set-theoretic) characterization
  - Model-theoretic characterization
- Language Identification in the Limit

Slide 32

What is stress and accent?

1. In many languages—but not all—certain syllables are more prominent than others. This prominence is referred to as *stress* and/or *accent*.
2. There are no universal phonetic correlates of stress, though common correlates involve pitch, duration, and loudness.
3. The presence of stress/accent is often detectable by its effects. In English, for example, unstressed vowels reduce (see notes).

Here are some examples of where stress falls in English words. Note how unstressed vowels often reduce to a schwa (from [Odd05, p. 89]).

mánətown	‘monotone’	mənátəniy	‘monotony’
tɛ́ləgræf	‘telegraph’	təlɛ́grəfiy	‘telegraphy’
ɛ́pəgræf	‘epigraph’	əpígrəfiy	‘epigraphy’
rɛ́lətɪv	‘relative’	rəléyšən	‘relation’
əkánəmiy	‘economy’	ɛ̀kənámɪk	‘economic’
díyfɛkt	‘defect (noun)’	dəfɛ́ktɪv	‘defective’
dɛ́məkræt	‘democrat’	dəmákrəsiy	‘democracy’
ítəliy	‘Italy’	ətǽlyən	‘Italian’
hámənɪm	‘homonym’	həmánəmiy	‘homonymy’
fənɛ́tɪks	‘phonetics’	fòwnətíšən	‘phonetician’
stətístɪks	‘statistics’	stæ̀təstíšən	‘statistician’
rəsíprəkl̩	‘reciprocal’	rɛ̀səprásətiy	‘reciprocity’
fənáləǰiy	‘phonology’	fòwnəláǰəkl̩	‘phonological’
láǰɪk	‘logic’	ləǰíšn̩	‘logician’
sínənɪm	‘synonym’	sənánəmiy	‘synonymy’
ərístəkræt	‘aristocrat’	ɛ̀rəstákrəsiy	‘aristocracy’

Slide 33

An Alphabet for Stress Patterns

Syllable Weight				Stress		
• L	=	Light		• σ	=	Unstressed Stress
• H	=	Heavy		• $\acute{\sigma}$	=	Primary Stress
• S	=	Super Heavy		• $\grave{\sigma}$	=	Secondary Stress
• σ	=	Arbitrary		• $\overset{+}{\sigma}$	=	Some Stress
				• $\overset{*}{\sigma}$	=	Arbitrary Stress

The entire alphabet is thus given by any combination of a primary glyph (Syllable Weight column) and a diactric, or absence thereof (the Stress column).

For instance, $\acute{H}$ is an alphabetic symbol, interpreted as a heavy syllable with primary stress. Similarly, σ indicates an unstressed, aribtrary syllable, and $\overset{*}{\sigma}$ indicates any syllable with any level of stress (including unstressed).

Slide 34

Stress in Pintupi [HH69]

a.	páɳa	‘earth’
b.	t^{j}úʈaya	‘many’
c.	máɭawàna	‘through from behind’
d.	púɭiŋkàlat^{j}u	‘we (sat) on the hill’
e.	t^{j}ámulìmpat^{j}ùŋku	‘our relation’
f.	ʈíɭiriŋ̀ulàmpat^{j}u	‘the fire for our benefit flared up’
g.	kúran^{j}ùlulìmpat^{j}ùɻa	‘the first one who is our relation’
h.	yúmaɻìŋkamàrat^{j}ùɻaka	‘because of mother-in-law’

Slide 35

Pintupi – Linguistic generalization

a. $\acute{\sigma}\,\sigma$

b. $\acute{\sigma}\,\sigma\,\sigma$

c. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma$

d. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\sigma$

e. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma$

f. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\sigma$

g. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma$

h. $\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\sigma$

- Primary stress falls on the first syllable and secondary stress on all nonfinal odd syllables.

An important difference between the generalization and the words in (a)-(h) is that the generalization describes an *infinite* set of words, whereas the (a)-(h) only describes eight.

Slide 36

Pintupi with expressions. Let $\Sigma = \{\acute{\sigma}, \grave{\sigma}, \sigma\}$.

- A generalized regular expression

$$\acute{\sigma}\left(\left((\sigma\,\grave{\sigma})^*\,\sigma(\sigma+\lambda)\right)+\lambda\right)$$

- A star free expression

1. Let $R = (\sigma\,\grave{\sigma})^*$.
2. Let

$$S = \lambda + \left(\begin{array}{cl} & \sigma\,\overline{\varnothing} \\ \& & \overline{\varnothing}\,\grave{\sigma} \\ \& & \overline{\overline{\varnothing}\,\acute{\sigma}\,\overline{\varnothing}} \\ \& & \overline{\overline{\varnothing}\,\grave{\sigma}\,\grave{\sigma}\,\overline{\varnothing}} \\ \& & \overline{\overline{\varnothing}\,\sigma\,\sigma\,\overline{\varnothing}} \end{array}\right)$$

3. Observe that $\mathbb{L}_{\mathrm{GRE}}(R) = \mathbb{L}_{\mathrm{GRE}}(S)$.

When we look at the definition of S, we can understand the star free expression in terms of its parts. These say "An admissible sequences is either λ or else it…

…	must begin with σ
and	must end with $\grave{\sigma}$
and	cannot contain any $\acute{\sigma}$
and	cannot contain any $\grave{\sigma}\,\grave{\sigma}$
and	cannot contain any $\sigma\,\sigma$."

Slide 37

Substrings (also called ***factors***)

1. For all $u, w \in \Sigma^*$, $u \precsim w$ ("u is a substring of w") $\stackrel{\text{def}}{=} (\exists x, y \in \Sigma^*)[xuy = w]$.
2. For all $w \in \Sigma^*$, $\mathrm{F}_k(w) \stackrel{\text{def}}{=} \{u \mid u \precsim w \wedge |u| = k\}$ if $k \leq |w|$ and $\{w\}$ otherwise.
3. For all $L \subseteq \Sigma^*$, $\mathrm{F}_k(L) \stackrel{\text{def}}{=} \bigcup_{w \in L} \mathrm{F}_k(w)$

Exercise 8 *Calculate the following.*

1. $F_2(aaa)$
2. $F_2(aaab)$
3. $F_{10}(aaab)$
4. $F_3(\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\grave{\sigma}\,\sigma\,\sigma)$

Slide 38

Strictly Local Stringsets

We introduce two special symbols marking word boundaries: $\rtimes, \ltimes \notin \Sigma$.

Definition 3 (Strictly Local stringsets) *A Strictly k-Local Grammar $\mathcal{G} = (\Sigma, T)$ where T is a subset of $F_k\big(\{\rtimes\}\Sigma^*\{\ltimes\}\big)$ and*

$$\mathbb{L}_{SL}\big((\Sigma, T)\big) \stackrel{def}{=} \{w \mid F_k(\rtimes w \ltimes) \subseteq T\}.$$

A stringset L is strictly k-local if there exists a strictly k-local $\mathcal{G}$ such that $\mathbb{L}_{SL}(\mathcal{G}) = L$. Such stringsets form the exactly the Strictly k-Local stringsets ($\boldsymbol{SL}_k$).

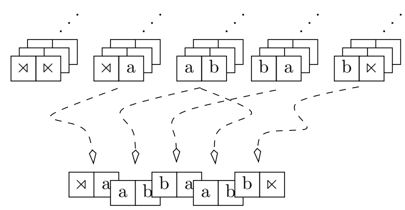
*A stringset is strictly local if there exists a k such that it is strictly k-local. Such stringsets form exactly the Strictly Local stringsets (**SL**).*

Exercise 9

1. *Show that, given an alphabet, Σ and a k, there are only finitely many Strictly k-local stringsets.*
2. *Show that **Fin** $\not\subseteq$ $\boldsymbol{SL}_k$ for any k.*
3. *Show that **Fin** $\subsetneq$ **SL**.*
4. *Show that there are infinitely many **SL** stringsets.*

Slide 39

Strictly Local stringsets as Tiling



- For $\mathcal{G} = (\Sigma, T)$, the factors in T can be thought of as a set of *tiles*. Placing matching tiles generates words.
- In the above diagram, the tiles are 2-factors and generate the word *abab*.

Slide 40

Modeling Pintupi with a Strictly Local stringset

Pintupi is Strictly 3-local.

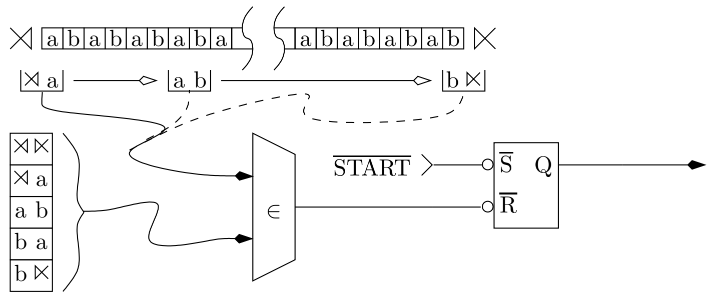
$$\mathcal{G} = \left\{ \begin{array}{cccc} \rtimes\,\acute{\sigma}\,\ltimes, & \acute{\sigma}\,\sigma\,\sigma, & \sigma\,\sigma\,\ltimes, & \grave{\sigma}\,\sigma\,\grave{\sigma}, \\ \rtimes\,\acute{\sigma}\,\sigma, & \acute{\sigma}\,\sigma\,\grave{\sigma}, & \sigma\,\grave{\sigma}\,\sigma, & \grave{\sigma}\,\sigma\,\sigma, \\ & \acute{\sigma}\,\sigma\,\ltimes, & & \grave{\sigma}\,\sigma\,\ltimes \end{array} \right\}$$

Exercise 10

1. *Generate some words with the above 3-factors.*
2. *Pintupi is not Strictly 2-local. Explain why not.*

Slide 41

SL stringsets - Scanners



- The tiling perspective naturally leads to a recognition strategy. Given a word, check the k-sized tiles in it one a time from left to right against the grammar. The diagram describes such a *scanner* for the case when $T = \{\rtimes\ltimes, \rtimes a, ab, ba, b\ltimes\}$.

Slide 42

SL stringsets - Abstract characterization

The theorem below establishes a set-based characterization of SL stringsets independent of any grammar, scanner, or automaton.

Theorem 4 (k-Local Suffix Substitution Closure) *For all* $L \subseteq \Sigma^*$, $L \in \boldsymbol{SL}$ *iff there exists* k *such that for all* $u_1,\ v_1,\ u_2,\ v_2,\ x \in \Sigma^*$ *it is the case that*

$$u_1xv_1, u_2xv_2 \in L \textit{ and } |x| = k-1 \Rightarrow u_1xv_2 \in L.$$

Exercise 11

1. *Show that the class of* $\boldsymbol{SL}_k$ *stringsets is* not *closed under*
   - *Union*
   - *Complement*
   - *If* $k > 2$, *Kleene star.*
2. *Is* $\boldsymbol{SL}$ *closed under any of these operations?*
3. *(For thought) Show that* $\boldsymbol{SL}_2$ is *closed under Kleene star.*

Slide 43

Using Theorem 4

- The theorem provides a law which simultaneously
  - provides a basis for *inference*
  - provides a method for establishing non-$\mathbf{SL}_k$ stringsets.

$$
\begin{array}{c|c|cc}
u_1 & \sigma_1 \cdots \sigma_{k-1} & v_1 & \in L \\
u_2 & \sigma_1 \cdots \sigma_{k-1} & v_2 & \in L \\
\hline
u_1 & \sigma_1 \cdots \sigma_{k-1} & v_2 & \in L
\end{array}
$$

Exercise 12 *Consider a Strictly* 2*-Local stringset* L *which contains the words aaa and aab. Using this theorem, explain what other words must be in* L*.*

Slide 44

Showing what is *not* SL$_k$.

Pintupi is not Strictly 2-local because we can find a counterexample.

$$\begin{array}{r|c|cc}
\acute{\sigma}\sigma & \sigma & & \in L \\
\acute{\sigma} & \sigma & \sigma & \in L \\
\hline
\acute{\sigma}\sigma & \sigma & \sigma & \notin L
\end{array}$$

Slide 45

Showing what is *not* SL.

Samala is not Strictly k-Local for any k.

$$
\begin{array}{c|c|cc}
s & o^k & s & \in L \\
\int & o^k & \int & \in L \\
\hline
s & o^k & \int & \notin L
\end{array}
$$

Exercise 13

1. *Using this theorem, explain why First/Last Harmony is not Strictly k-Local for any k.*
2. *Using this theorem, explain why Even-sibilants is not Strictly k-Local for any k.*

Slide 46

SL Hierarchy

Theorem 5 (SL-Hierarchy)

$$\boldsymbol{SL}_1 \subsetneq \boldsymbol{SL}_2 \subsetneq \boldsymbol{SL}_3 \subsetneq \cdots \subsetneq \boldsymbol{SL}_i \subsetneq \boldsymbol{SL}_{i+1} \subsetneq \cdots \subsetneq \boldsymbol{SL}$$

Every Finite stringset is $\mathbf{SL}_k$ for some k: $\mathbf{Fin} \subsetneq \mathbf{SL}$.

There is no k for which $\mathbf{SL}_k$ includes all Finite languages.

Slide 47

SL stringsets - Model Theoretic Characterization

$$\mathbb{W}^{\triangleleft} = \langle D, \triangleleft, P_\sigma \rangle_{\sigma \in \Sigma}$$

- Earlier we introduced the above model to describe words.
- Now we will introduce a logic based on a restricted form of propositional logic, along with a naming function, similar to what we did yesterday with regular expressions.

But first, to set the stage, we must discuss embeddings.

Slide 48

Embeddings

- An *injective homomorphism* between two models $\mathcal{M}_1$ and $\mathcal{M}_2$ with the same signature is a function h which maps every element in D_1, the domain of $\mathcal{M}_1$, to elements in D_2, the domain of $\mathcal{M}_2$, such that for all n-ary relations R and all n-tuples of elements of D_1,

$$R_1(x_1, \cdots x_n) \Leftrightarrow R_2(h(x_1), \cdots h(x_n)).$$

- Such homomorphisms are also called *embeddings*.
- If there exists an injective homomorphism from $\mathcal{M}_1$ to $\mathcal{M}_2$ we say that $\mathcal{M}_1$ can be embedded in $\mathcal{M}_2$, that $\mathcal{M}_1$ is a *submodel* of $\mathcal{M}_2$ ($\mathcal{M}_1 \precsim \mathcal{M}_2$) and $\mathcal{M}_2$ is an *extension* of $\mathcal{M}_1$.

We use the same symbol for "submodel" as we do for "substring", which we will justify in a moment.

Exercise 14

1. *Assume $\mathbb{W}^{\triangleleft}$. Is there an embedding from $\mathcal{M}_{ba}$ to $\mathcal{M}_{ccba}$? Explain.*
2. *Assume $\mathbb{W}^{\triangleleft}$. Is there an embedding from $\mathcal{M}_{ba}$ to $\mathcal{M}_{cabc}$? Explain.*

The following lemma is nearly immediate.

Lemma 2 *Consider any words $w, v \in \Sigma^*$. Then $\mathcal{M}_w$ can be embedded in $\mathcal{M}_v$ iff w is a substring of v:*

$$\mathcal{M}_w^{\triangleleft} \precsim \mathcal{M}_v^{\triangleleft} \Leftrightarrow w \precsim v.$$

Where the first '$\precsim$' is a relation between models and the second a relation between strings. Thus any confusion between the two types of relations is harmless.

Note that these are strong homomorphisms; a weak homomorphism requires only that $R_1(x_1, \cdots x_n) \Rightarrow R_2(h(x_1), \cdots h(x_n))$

Slide 49

Restricted Propositional Logic (RPL)

A *sentence* of RPL is defined inductively as follows.

1. *The base cases:*
   - For all $w \in \{\rtimes, \lambda\}\Sigma^*\{\ltimes, \lambda\}$, $(\neg w)$ is a sentence of RPL.
2. *The inductive case:*
   - If ϕ and ψ are sentences of RPL then so is $(\phi \wedge \psi)$.
3. Nothing else is a sentence of RPL.

Essentially, all sentences will have the form

$$(\neg w_0) \wedge (\neg w_1) \wedge \cdots \wedge (\neg w_n)$$

In other words sentences of the restricted propositional logic considered here are simply conjunctions of negations of atomic propositions (negative literals).

(We omit many parentheses because the semantics of the naming function (next slide) are such that $\wedge$ will be associative and commutative.)

This is not the only possible restricted propositional logic. We might limit it to disjunctions of positive literals, for example, which would allow definition of all and only the stringsets that are complements of stringsets definable with this RPL.

Slide 50

Restricted Propositional Logic - Stringsets

- To define the naming function, it is first necessary to say what it means for a word $w \in \Sigma^*$ to *model* ($\models$) a sentence ϕ in Restricted Propositional Logic.
- The idea is if $\mathcal{M}_w \models \phi$ then ϕ is *true* of w.
- Consider any $v \in \{\rtimes\}\Sigma^*\{\ltimes\}$.
  1. *The base cases:*
     - For all $w \in \{\rtimes, \lambda\}\Sigma^*\{\ltimes, \lambda\}$, $\mathcal{M}_v \models (\neg w) \Leftrightarrow \mathcal{M}_w \not\precsim \mathcal{M}_v$.
  2. *The inductive cases:*
     - For all ϕ, ψ in RPL, $v \models (\phi \wedge \psi) \Leftrightarrow v \models \phi$ and $v \models \psi$.
- Then

$$\mathbb{L}_{\mathrm{RPL}}(\phi) = \{w \mid \mathcal{M}_{\rtimes w \ltimes} \models \phi\}$$

The above definition is not signature-specific. (Although it does presume the presence of '$\rtimes$' and '$\ltimes$' in the alphabet, which will not always be the case.)

It follows that, under the $\mathbb{W}^{\triangleleft}$ signature, stringsets are defined as exactly those words which do *not* contain any of the atomic propositions as *substrings*.

Exercise 15

1. *Write a sentence of RPL that yields the Pintupi stress pattern.*
2. *How do the atomic elements of this sentence relate to the tiles (elements of T in the grammar-based definition) discussed earlier?*
3. *RPL differs from the traditional notion of propositional logic, in which the atomic formulae are* propositional variables *and a model is a* valuation*: an assignment of truth values to the propositional variables.*
   (a) *What, in RPL, corresponds to propositional variables?*
   (b) *What corresponds to a valuation?*

While word models have internal structure, in the propositional semantics it only contributes to the definition of $\precsim$. There is no way, in our propositional languages, to refer to the relations of the signature directly.

Two words are logically equivalent wrt RPL ($w \equiv_{RPL} v$) iff they share the same set of k-factors ($F_k(w) = F_k(v)$).

Slide 51

Cognitive complexity of SL

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

Slide 52

Identification in the limit from text [Gol67]

- A *positive presentation* of a language L is a total, surjective function $t_L : \mathbb{N} \to L$. It is also called a *text* for L and can be thought of as an infinite sequence of elements drawn from L such that every element of L occurs at least once. The initial portion of a text up to its ith element is denoted $t_L[i]$.
- Let $\mathbf{SEQ} \stackrel{\text{def}}{=} \{t_L[i] \mid L \subseteq \Sigma^* \text{ and } i \in \mathbb{N}\}$.
- For some class of grammars $\mathbb{G}$, a learner is a function $\phi : \mathbf{SEQ} \to \mathbb{G}$.
- Class $\mathbb{L}$ is *identifiable in the limit from positive data* if there exists a computable ϕ such that

$$(\forall L \in \mathbb{L})(\forall t_L)(\exists i \in \mathbb{N})(\forall j > i)(\exists \mathcal{G} \in \mathbb{G})$$
$$\left[\phi(t_L[j]) = \mathcal{G} \text{ and } \mathbb{L}(\mathcal{G}) = L\right]$$

According to the above definition, there is no text for the empty language. This is usaully accomodated by letting the codomain of t_L include an element '#' called 'pause' which means a moment when no information is forthcoming. Then there would be exactly one text for the empty language: $(\forall i \in \mathbb{N})[t_\varnothing(i) = \#]$.

The learning definition requires that for every language in the class, for every text for the language, that the learner *converge* to a single grammar and that this grammar be *correct* in the sense that it generates the target language exactly.

Surveys of different definitions of learning can be found in [OWS86, JORS99, LZZ08, ZZ08, Hei14].

Slide 53

Learning Fin

Theorem 6 (Gold 1967) ***Fin*** *is identifiable in the limit from positive data.*

- Consider grammars to be finite stringsets, and let $\mathbb{L}$ be the identity function. So $\mathbb{L}(\mathcal{G}) = \mathcal{G}$.
- Let $\mathsf{content}(t_L[i]) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid (\exists i)[t_L(i) = w]\}$.
- Then consider this learner:

$$\phi(t_L[i]) \stackrel{\text{def}}{=} \mathsf{content}(t_L[i])$$

Essentially, the learning algorithm just memorizes the words it has observed so far. Since these are finite languages, in any presentation, there will be a point when every word in the language has been seen. Thus the learner will have converged to a correct grammar for the language.

Slide 54

Non-Learnability of ANY 'superfinite' class

A class of languages is *superfinite* if it includes every finite language and at least one infinite language.

Theorem 7 (Gold 1967) *No superfinite class is identifiable in the limit from positive data.*

- Therefore, none of **SL**, **SF**, and **Reg** is learnable in this sense.
- Gold suggested three ways to proceed: consider non-superfinite classes, allow for *some negative* evidence, constrain the texts (t_L) learners are required to succeed on.

Two ways (at least) to prove this. Gold's original proof stands, but modern treatments are based on so-called 'locking' sequences [BB75, OWS86, JORS99]

- Show that if a learner can learn the infinite language on every text for it then there is a text for some finite language that the learner fails on.
- Show that if a learner identifies every finite language L then there is a text for the infinite language that the learner fails to identify the infinite language on.

Slide 55

Learning $\mathbf{SL}_k$

Theorem 8 (Garcia et al. 1993) *$\boldsymbol{SL}_k$ is identifiable in the limit for positive data.*

- Let $\mathbb{G}$ and $\mathbb{L}$ be given by the grammar-theoretic definition earlier.
- Consider this learner:

$$\phi(t_L[i]) \stackrel{\text{def}}{=} \mathrm{F}_k\Big(\mathsf{content}(t_L[i]\Big)$$

Essentially, this learner just remembers the k-factors of words it has observed. Since there are only finitely many such k-factors at some point in any text for a $\mathbf{SL}_k$ language, they will all be observed.

You may observe that this learner essentially applies a function to the content of the observed text and that this function returns grammatical information. The consequences of this observation were explored by [Hei10, KK10, HKK12].

Slide 56

Stress Typology

Heinz's Stress Pattern Database (ca. 2007)—109 patterns

9 are $\mathbf{SL}_2$	Abun West, Afrikans, ... Cambodian,... Maranungku
44 are $\mathbf{SL}_3$	Alawa, Arabic (Bani-Hassan),...
24 are $\mathbf{SL}_4$	Dutch,...
3 are $\mathbf{SL}_5$	Asheninca, Bhojpuri, Hindi (Fairbanks)
1 is $\mathbf{SL}_6$	Icua Tupi
28 are not **SL**	Amele, Bhojpuri (Shukla Tiwari), Arabic (Classical), Hindi (Kelkar), Yidin,...

72% are **SL**, all $k \leq 6$. 49% are $\mathbf{SL}_3$.

There is a polynomial time algorithm that, given a regular stringset (as a DFA) decides whether it is **SL** or not and, if it is, the minimum k for which it is $\mathbf{SL}_k$ [ELM+08].

Using this, a group of Earlham students has classified the patterns in [Hei07, Hei09] with respect to the **SL** hierarchy.

The results indicate that the majority of stress patterns are, in fact, quite simple and that the amount of context that is relevant is quite small.

Slide 57

Summary Session 2

- There are several natural definitions of **SL** and $\mathbf{SL}_k$ languages.
- $\mathbf{SL}_k$ is identifiable in the limit from positive data (but **SL** is not.
- Many phonotactic patterns and stress patterns are $\mathbf{SL}_k$ for small k (but not all are **SL**).

Slide 58

Overview Session 3

Local Stringsets II

- Some non-**SL** stress patterns
- Locally Testable Stringsets (Full Propositional(+1))
- Locally Threshold Stringsets (**FO**(+1))
- Regular Stringsets (MSO(+1))

Slide 59

Overview of Part 3.1:

Locally Testable Stringsets (LT)

- Some non-**SL** stress patterns
- Locally Testable Stringsets (Full Propositional(+1))
  - Model-theoretic characterization
  - Grammatical characterization
  - Automata-theoretic characterization
  - Abstract (set-theoretic) characterization
  - Cognitive complexity of **LT**.
- A non-**LT** stress pattern

Slide 60

Yidin [Dix77, HV87, Hei07]

- Primary stress on the leftmost heavy syllable, else the initial syllable
- Secondary stress iteratively on every second syllable in both directions from primary stress
- No light monosyllables

Yidin is an Australian language, first described in 1971. The description is somewhat controversial, since there were very few surviving informants. In any case, it is the patterns that concern us here, not the question of whether they are linguistically accurate.

Slide 61

Yidin

- Primary stress on the leftmost heavy syllable, else the initial syllable
  - First H gets primary stress (No-H-before-$\acute{H}$)
  - $\acute{L}$ only if initial (Nothing-before-$\acute{L}$)
  - $\acute{L}$ implies no H (No-H-with-$\acute{L}$)
- Secondary stress iteratively on every second syllable in both directions from primary stress
  - σ and $\overset{+}{\sigma}$ alternate (Alt)
- No light monosyllables
  - No $\acute{L}$ monosyllables (No-$\rtimes \acute{L} \ltimes$)
- At least one $\acute{\sigma}$ (Some-$\acute{\sigma}$) [Assumed]
- No more that one $\acute{\sigma}$ (At-Most-One-$\acute{\sigma}$) [Assumed]

We can extract a set of explicit constraints from the description.

These are not the only way of factoring the constraints and not fully independent. No-$\rtimes \acute{L} \ltimes$, for example, can be reduced to No $\acute{L} \ltimes$ in the presence of Nothing-before-$\acute{L}$. Which constraints are fundamental (which we refer to as primitive constraints) is a linguistic issue. Again, we are interested in these particular constraints, not in the issue of whether they are truly primitive.

We have factored the constraint that every word has exactly one syllable that gets primary stress, which is assumed in most cases, into two components: ≥ 1 (often called "obligatoriness") and ≤ 1 (often called "culmanitivity"). These two components not only have distinct formal complexity, they seem to be phonotactically independent [Hym09].

Exercise 16 *Which of these are **SL**? For those that are, what is k?*

Slide 62

Determining Complexity of Factored Stress Patterns

- We will factor patterns into the co-occurrence (conjunction, intersection) of primitive constraints.
- Our complexity classes form a proper hierarchy.
- Each of the classes is closed under intersection.
- Hence, the complexity of a compound constraint is no more than the maximal complexity of its primitive factors.

Slide 63

No-$\boldsymbol{H}$-with-$\boldsymbol{\acute{L}}$

$$\begin{array}{r|c|l} \rtimes \acute{L} & \overbrace{L\cdots L}^{k-1} & \ltimes \\ \rtimes \acute{H} & \overbrace{L\cdots L}^{k-1} & H \ltimes \\ \hline \star \rtimes \acute{L} & \overbrace{L\cdots L}^{k-1} & H \ltimes \end{array}$$

No-H-with-$\acute{L} \notin$ **SL**

Exercise 17

- *Show that Some-$\acute{\sigma}$ is not **SL**.*
- *How, then, can any stress pattern be **SL**?*

Because they are conjunctions only of negative literals, **SL** constraints can only *forbid* the occurrence of a factor, they cannot *require* an occurrence.

We could accommodate required factors by allowing positive literals, in which case we would have a conjunctive logic with the scope of negation limited to atomic formulae, but this gives a level of complexity that is not particularly interesting in itself. It is more useful to allow negation to have arbitrary scope, in which case we get a full Boolean logic, since disjunction can be reduced to conjunction and negation.

Slide 64

Full Propositional Logic for $\mathcal{W}^{\triangleleft}$ (Prop(+1)) —Syntax

k-Expressions

k-expressions are defined inductively as follows.

1. *The base cases:*
   - For all $w \in F_k(\{\rtimes\}\Sigma^*\{\ltimes\})$, w is a k-expression.
2. *The inductive cases:*
   - If ϕ is a k-expression then so is $(\neg\phi)$.
   - If ϕ and ψ are k-expressions then so is $(\phi \wedge \psi)$.
3. Nothing else is a k-expression.

Slide 65

Full Propositional Logic for $\mathcal{W}^{\triangleleft}$ (Prop(+1)) —Semantics

Consider any $v \in \{\rtimes\}\Sigma^*\{\ltimes\}$ and any k-expression ϕ:

1. *The base cases:*
   - If $\phi = w \in \{\rtimes, \lambda\}\Sigma^*\{\ltimes, \lambda\}$, $\mathcal{M}_v \models \phi \Leftrightarrow \mathcal{M}_w \precsim \mathcal{M}_v$.
2. *The recursive case:*
   - If $\phi = (\neg\psi)$ then $\mathcal{M}_v \models \phi \Leftrightarrow \mathcal{M}_v \not\models \psi$.
   - If $\phi = \psi_1 \vee \psi_2$ then $\mathcal{M}_v \models \phi \Leftrightarrow$ either $\mathcal{M}_v\psi_1$ or $\mathcal{M}_v\psi_2$

$$\mathbb{L}(\varphi) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \mathcal{M}_{\rtimes w \ltimes} \models \phi\}.$$

A stringset is *k-locally definable* iff it is $\mathbb{L}(\varphi)$ for some k-expression φ. It is *locally definable* iff it is k-locally definable for some k.

We can, of course, now use any Boolean-definable connectives, for example:

$$\begin{array}{rcl} \phi \to \psi & \equiv & \neg\phi \vee \psi \\ \phi \leftrightarrow \psi & \equiv & (\phi \to \psi) \wedge (\psi \to \phi) \\ \text{etc.} & & \end{array}$$

Implication ($\to$) is particularly useful in expressing linguistic constraints.

Slide 66

No-$\boldsymbol{H}$-with-$\boldsymbol{\acute{L}}$ and Some-$\boldsymbol{\acute{\sigma}}$ are Locally Definable

$$\text{Some-}\acute{\sigma} = \mathbb{L}(\acute{\sigma})$$

$$\text{No-}\acute{\sigma}\text{-with-}\acute{L} = \mathbb{L}(\acute{L} \rightarrow \neg H)$$

Exercise 18 *For each of these, what is k?*

Slide 67

k-Local Grammars

Definition 4 (k-Locally Testable Stringsets) *A k-Local Grammar is a pair $\mathcal{G} = \langle \Sigma, T \rangle$ where T is a subset of $\mathcal{P}(F_k(\{\rtimes\}\Sigma^*\{\ltimes\}))$.*

The stringset licensed by $\mathcal{G}$ is

$$\mathbb{L}_{LT}(\langle \Sigma, T \rangle) \stackrel{def}{=} \{w \mid F_k(w) \in T\}.$$

*A stringset L is k-local if there exists a k-local $\mathcal{G}$ such that $\mathbb{L}_{SL}(\mathcal{G}) = L$. Such stringsets form the exactly the k-Locally Testable stringsets (**LT_k**).*

*A stringset is Locally Testable if there exists a k such that it is k-local. Such stringsets form exactly the Locally Testable stringsets (**LT**).*

We can get grammars for $\mathbf{LT}_k$ stringsets by following the observation that, in the context of our propositional logics, words are, in essence, Boolean valuations of the atomic formulae, which are just the set of k-factors over the given alphabet.

So a word model just specifies which atomic formulae are to be interpreted as true (those that occur in the word) and which are false (those that do not).

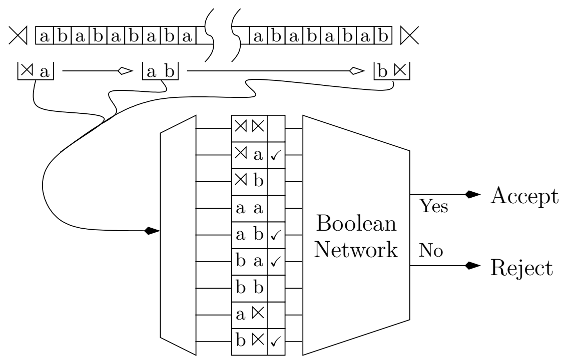
An $\mathbf{LT}_k$ grammar, then, just specifies which of these valuations (i.e., words) are acceptable.

It is immediate, then, that Local Grammars are equivalent in expressive power to k-expressions.

Exercise 19 *How does this definition differ from that of strictly k-local grammars?*

Slide 68

LT Automata



Membership in an $\mathbf{LT}_k$ stringset depends only on the set of k-Factors which occur in the string.

Recognizing an $\mathbf{LT}_k$ stringset requires only remembering which k-factors occur in the string.

Automata for **LT** are scanners that keep track of which factors occur in the word. So the internal table embodies the valuation represented by the word.

The k-expression is implemented in Boolean network

Slide 69

Character of Locally Testable sets

Theorem 9 (k-Test Invariance) *A stringset L is Locally Testable iff*

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

if $\rtimes \cdot x \cdot \ltimes$ and $\rtimes \cdot y \cdot \ltimes$ have exactly the same set of k-factors

then either both x and y are members of L or neither is.

Definition 5 (k-Local Equivalence)

$$w \equiv^L_k v \overset{def}{\Longleftrightarrow} F_k(\rtimes w \ltimes) = F_k(\rtimes v \ltimes).$$

It should be clear that **LT** definitions can't distinguish strings that have same k-factors. So, with respect to **LT** definitions, strings with the same set of k-factors are equivalent.

This equivalence categorizes the set of all strings into classes based on their set of k-factors. **LT** definitions can't break these classes—if one string in a class satisfies the definition then all strings in the class necessarily satisfy the definition as well.

In this way, a set of strings is **LT** iff it is the union of some $\mathbf{LT}_k$ equivalence classes, for some k.

Exercise 20 *Show that there are only finitely many $\boldsymbol{LT}_k$ stringsets.*

Slide 70

Using k-Local Equivalence

Inductive mode

Given some strings in an LT_k stringset, by considering the form of the strings that are in their equivalence classes of the given strings one can determine what other strings must be in the class.

Contradiction mode

To show that a stringset L is *not* $\mathbf{LT}_k$ it suffices to show any two strings $w \in L$ and $v \notin L$ which are in the same k-local equivalence class: $w \equiv_k^L v$.

To establish that a stringset is not **LT**, it suffices to show that such a counterexample exists for any k.

As with suffix-substitution closure, k-test invariance can be used inductively, to get a sense of the strings that must be included (at least) in an $\mathbf{LT}_k$ the stringset given knowledge of some of the strings it includes.

And, as with suffix-substitution closure, one can establish that a stringset is not **LT** by exhibiting a class of counterexamples parameterized by k.

Exercise 21

1. *Suppose that $L \in \boldsymbol{LT}_2$ and that both of the strings aaba and bb are in L.*
   - *Give the sets of k-factors of aaba and of bb.*
   - *Using that, describe what other strings must be included in L (at least).*
2. *Let L_{2a} be the set of strings over $\{a, b\}$ which include at least two 'a's. (In notation we would say $\{w \in \Sigma^* \mid |w|_a \geq 2\}$.) Show that L_{2a} is not* ***LT****.*

Slide 71

LT Hierarchy

Theorem 10 (LT-Hierarchy)

$$\boldsymbol{LT}_1 \subsetneq \boldsymbol{LT}_2 \subsetneq \boldsymbol{LT}_3 \subsetneq \cdots \subsetneq \boldsymbol{LT}_i \subsetneq \boldsymbol{LT}_{i+1} \subsetneq \cdots \subsetneq \boldsymbol{LT}$$

$$\mathbf{SL}_k \subsetneq \mathbf{LT}_k$$
$$\mathbf{LT}_k \subsetneq \mathbf{LT}_{k+1}$$
$$\mathbf{LT}_k \not\subseteq \mathbf{SL}_{k+1}$$
$$\mathbf{SL}_{k+1} \not\subseteq \mathbf{LT}k$$

$\mathbf{SL}_k$ and $\mathbf{LT}_k$ for parallel proper hierarchies. While for a given k, $\mathbf{SL}_k \subsetneq \mathbf{LT}_k$ (and consequently $\mathbf{SL}_k \subsetneq \mathbf{LT}_{k+i}$ for all $i \in \mathbb{N}$), all other relations between the hierarchies are incomparable.

Exercise 22 *Prove it (them).*

Slide 72

At-Most-One-$\acute{\sigma}$ is not LT

$$\rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \qquad \in L_{\mathrm{One}-\acute{\sigma}}$$

$$\rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \quad \notin L_{\mathrm{One}-\acute{\sigma}}$$

But

$$\rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes \quad \equiv^L_k \quad \rtimes \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \acute{\sigma} \overbrace{\sigma \cdots \sigma}^{k-1} \ltimes$$

At-Most-One-$\acute{\sigma}$ is not **LT** (hence not **SL**)

Slide 73

Cognitive interpretation of LT

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) $\mathbf{LT}_k$ language must be sensitive, at least, to the *set* of length k contiguous blocks of events that occur in the presentation of the string—both those that do occur and those that do not.
- 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 set of length k blocks of events that occurred at any prior point.
- Any cognitive mechanism that is sensitive *only* to the occurrence or non-occurrence of length k contiguous blocks of events in the presentation of a string will be able to recognize *only* $\mathbf{LT}_k$ languages.

Note that while negative judgments about **SL** constraints can be made as soon as an exception is encountered, in general judgments about properly **LT** constraints can't be made until entire string has been processed. In particular, there is no way to determine that some required factor does not occur until all of the factors of the word have been scanned.

Slide 74

Summary of Part 3.1

- We introduced the stress pattern of Yidin which will provide us with a framework for exploring the complexity of naturally occurring constraints.
- We factored that stress pattern into a set of primitive constraints.
- The overall complexity of the full pattern will be the supremum of the complexity of those primitive constraints.
- You established that Alt and Nothing-before-$\acute{L}$ are $\mathbf{SL}_2$, that (by itself) No-$\rtimes \acute{L} \ltimes$ is $\mathbf{SL}_3$ but that its conjunction with Nothing-before-$\acute{L}$ is just $\mathbf{SL}_2$.
- We established that No-H-with-$\acute{L}$ and Some-$\acute{\sigma}$ are not **SL**

Slide 75

Summary of Part 3.1 (cont.)

- We introduced k-expressions, the formulae of the full Propositional logic for $\mathcal{W}^{\triangleleft}$.
- We established that No-H-with-$\acute{L}$ and Some-$\acute{\sigma}$ *are* LT_1.
- We gave grammar- and automata-theoretic characterizations of **LT**.
- We gave an abstract characterization of **LT** in terms of Local Test Invariance and looked at how to use this to explore given **LT** stringsets and to show that a given stringset is not **LT**.
- We showed that At-most-one-$\acute{\sigma}$ is *not* **LT**.
- We gave a characterization of the cognitive complexity of **LT** constraints.

Slide 76

Overview of Part 3.2:

Locally Threshold Testable Stringsets (LTT)

- Model-theoretic characterization
- Abstract (set-theoretic) characterization
- Cognitive complexity of **LTT**.
- Some non-**LTT** stress pattern.

Slide 77

FO(+1)

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

First-order Quantification (over positions in the strings)

Syntax	Semantics		
$x \approx y$	$w, [x \mapsto i, y \mapsto j] \models x \approx y$	$\stackrel{\text{def}}{\iff}$	$j = i$
$x \triangleleft y$	$w, [x \mapsto i, y \mapsto j] \models x \triangleleft y$	$\stackrel{\text{def}}{\iff}$	$j = i + 1$
$P_\sigma(x)$	$w, [x \mapsto i] \models P_\sigma(x)$	$\stackrel{\text{def}}{\iff}$	$i \in P_\sigma$
$\varphi \wedge \psi$	$\vdots$		
$\neg\varphi$	$\vdots$		
$(\exists x)[\varphi(x)]$	$w, s \models (\exists x)[\varphi(x)]$	$\stackrel{\text{def}}{\iff}$	$w, s[x \mapsto i] \models \varphi(x)$ for some $i \in \mathcal{D}$

FO(+1)-Definable Stringsets: $\mathbb{L}(\varphi) \stackrel{\text{def}}{=} \{w \mid w \models \varphi\}$.

To be able to reason about multiple occurrences of the same symbol we will need to be able to talk about positions in the string. This is where the internal structure of the word models becomes essential.

FO(+1) is ordinary First-Order logic over the successor word models. The syntax of the logical formulae includes the predicate symbols for the successor relation ($\triangleleft$, we use this as an infix binary relation), and for each of the alphabet symbols (the P_σ). There are no constants in this language, so the only way to refer to positions is via first-order variables, i.e., variables which range over *individuals* of the domain. We assume an infinite supply of these.

The semantics of the logic is defined in terms of the *satisfaction* relation, a relation between models and logical formulae, which asserts that the formula is true in the model, i.e., that the property that the string has the property that the formula encodes. When there are free variables in the formula (those that are not in the scope of a quantifier) this is contingent on which positions are assigned to each of those variables. When we say

$$w, [x \mapsto i, y \mapsto j] \models \varphi(x, y)$$

we are asserting that the formula φ, in which x and y occur free, is true in the word w if x is bound to position i and y is bound to position j. By convention, if s is an assignment of positions to variables (a partial function from the set of variables to the domain of the structure), $s[x \mapsto i]$ denotes the assignment which is identical to s for all variables other than x and which binds x to i.

If there are no free variables in a formula, it expresses a (non-contingent) property of strings. Formulae without free variables are called *sentences.* A stringset is **FO**(+1) definable iff there is a **FO**(+1) sentences that is satisfied by all and only the strings in the set.

We also include the familiar Boolean connectives and the existential quantifier. By convention, we enclose the quantifier along with the variables it binds in ordinary parentheses and enclose the formula it scopes over in square brackets. So

$$(\exists x, y)[\varphi(x) \wedge \psi(y)]$$

is true in a model iff there is some assignment of positions in the domain of the model to the variables x in y which make the formulae φ (with x free) and ψ (with y free) true in that model.

Note that the universal quantifier $\forall$ (which asserts that all assignments to the variables make the matrix formula true in the model) is definable from $\exists$:

$$(\forall x)[\varphi(x)] \equiv \neg(\exists x)[\neg\varphi(x)].$$

Slide 78

Some FO(+1) Definable Constraints

$$\varphi_{\text{One-}\acute{\sigma}} = (\exists x)[\acute{\sigma}(x) \wedge (\forall y)[\acute{\sigma}(y) \rightarrow x \approx y]\ ]$$

Lemma 3 *Let f be any k-factor over $\{\rtimes, \ltimes\} \cup \Sigma$. There is a **FO**(+1) sentence occurs$_f$ which is satisfied by a string w iff f occurs as a substring of w.*

With the ability to distinguish distinct occurrences of a symbol we can assert that there is *exactly* on occurrence of primary stress in a word by asserting that there is *some* position in which primary stress occurs ($(\exists x)[\acute{\sigma}(x) \ldots$), and that there are no other positions in which primary stress occurs ($\wedge(\forall y)[\acute{\sigma}(y) \rightarrow x \approx y]\ ]$).

We no longer extend the alphabet with $\rtimes$ and $\ltimes$, as they are no longer necessary. We can assert that the position assigned to x is the initial position of the string with the formula:

$$\text{Initial}(x) \equiv \neg(\exists y)[y \triangleleft x]$$

We can define Final(x) similarly.

Exercise 23

1. *Write a **FO**(+1) sentence that is true of a string iff an unstressed syllable occurs somewhere in the string immediately before some syllable with secondary stress.*
2. *Prove Lemma 3. There are three (possibly four) cases to handle: when neither $\rtimes$ nor $\ltimes$ occur in the factor, when the factor starts with $\rtimes$ and when it ends with $\ltimes$. Depending on how you go about these, you may have to handle the case in which it both starts with $\rtimes$ and ends with $\ltimes$ separately.*
3. *Write an **FO**(+1) expression that asserts that the ante-penultimate (i.e., the syllable that precedes the syllable that precedes the final syllable) has no stress (neither primary nor secondary).*
4. *Write an **FO**(+1) expression that asserts that there are at least two distinct occurrences of light syllables in a word.*
5. *Argue that **FO**(+1) can express that there are at least, at most, or exactly n occurrences of a particular symbol for any natural number n.*

Slide 79

Character of the FO(+1) Definable Stringsets

Definition 6 (Locally Threshold Equivalent ($\equiv_{k,t}$)) *Two strings w and v are (k,t)-equivalent ($w \equiv_{k,t} v$) iff for all $f \in F_k(\rtimes \cdot w \cdot \ltimes) \cup F_k(\rtimes \cdot v \cdot \ltimes)$ either $|\rtimes \cdot w \cdot \ltimes|_f = |\rtimes \cdot v \cdot \ltimes|_f$ or both $|\rtimes \cdot w \cdot \ltimes|_f \geq t$ and $|\rtimes \cdot v \cdot \ltimes|_f \geq t$,*

Definition 7 (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 $w \equiv_{k,t} v$ then $w \in L \iff v \in L$.*

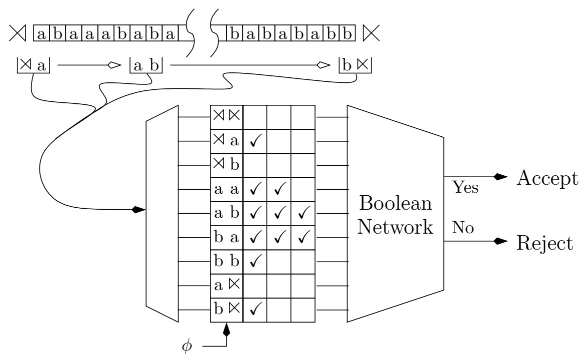
Theorem 11 (Thomas) *A set of strings is First-order definable over $\langle \mathcal{D}, \triangleleft, P_\sigma \rangle_{\sigma \in \Sigma}$ iff it is* Locally Threshold Testable.

$\mathbf{LT}_k = \mathbf{LTT}_{k,1}$, hence $\mathbf{LT} \subsetneq \mathbf{LTT}$

$\mathbf{LTT}_{k,t}$ stringsets categorize strings on the basis of (k,t)-equivalence; a stringset is $\mathbf{LTT}_{k,t}$ iff it is the union of some set of equivalence classes of Σ^* wrt $\equiv_{k,t}$.

Slide 80

LTT Automata



Membership in an **FO**(+1) definable stringset depends only on the multiplicity of the k-factors, up to some fixed finite threshold, which occur in the string.

Slide 81

Cognitive interpretation of FO(+1)

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) **FO**(+1) 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, (and, in particular, not to the order) of the length k blocks of events in the presentation of a string will be able to recognize *only* **FO**(+1) stringsets.

Slide 82

A non-FO(+1) Definable Constraint

No-H-before-$\acute{H}$

- Primary stress falls on the leftmost heavy syllable
- Yidin, Murik, Maori, Kashmiri, ...

$$\star\ H \ldots \acute{H}$$

$$\rtimes \overbrace{\grave{L}\,L \cdots \grave{L}\,L}^{2kt}\ \acute{H}\,H\ \overbrace{\grave{L}\,L \cdots \grave{L}\,L}^{2kt}\ \grave{H}\,H\ \overbrace{\grave{L}\,L \cdots \grave{L}\,L}^{2kt} \ltimes$$

$$\equiv^{L}_{k,t}$$

$$\star \rtimes \underbrace{\grave{L}\,L \cdots \grave{L}\,L}_{2kt}\ \grave{H}\,H\ \underbrace{\grave{L}\,L \cdots \grave{L}\,L}_{2kt}\ \acute{H}\,H\ \underbrace{\grave{L}\,L \cdots \grave{L}\,L}_{2kt} \ltimes$$

No-H-before-$\acute{H}$ requires the ability to reason about the order of occurrences of symbols without being explicit about adjacency. There are two ways of doing this. One is to move to a signature including $\triangleleft^+$, which we will do do in the next class.

The other is to extend k-expressions with concatenation. Both Some-H and Some-$\acute{H}$ are $\mathbf{LT}_1$ constraints, so No-H-before-$\acute{H}$ is just the complement of the concatenation of two **LT** stringsets. McNaughton and Papert [MP71] define **LTO** to be the closure of **LT** under concatenation and Boolean operations. They then show that **LTO** is equivalent to both **SF** and **FO**(<) (just two of at least three truly remarkable results in this book). We will return to this class of stringsets tomorrow.

Slide 83

Summary of Part 3.2

- We introduced the syntax and semantics of First-Order logic over $\mathcal{W}^{\triangleleft}$ known generally as FO(+1).
- We showed that No-More-than-One-$\acute{\sigma}$, and hence, One-$\acute{\sigma}$ is FO(+1) definable.
- We showed that the substring relation is FO(+1) definable.
- We gave Thomas's characterization of FO(+1) in terms of Local Threshold Testability and introduced the dual hierarchy of classes $\mathbf{LTT}_{k,t}$.
- We introduced **LTT** automata
- We characterized the cognitive complexity of **LTT** constraints.
- We showed that No-H-before-$\acute{H}$ is not **LTT**.

Slide 84

Overview of Part 3.3:

Regular Stringsets (Reg)

- MSO(+1)
- FSA as tiling systems
- Projections (Alphabetic Homomorphisms)
- Cognitive complexity of **Reg**.
- Yidin revisited

Slide 85

Monadic Second-Order Logic over Strings (MSO(+1))

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

First-order Quantification (positions)

Monadic Second-order Quantification (sets of positions)

Syntax	Semantics
$X(x)$	$w, s \models X(x) \overset{\text{def}}{\iff} s(x) \in s(X)$
$(\exists X)[\varphi(X)]$	$w, s \models (\exists X)[\varphi(X)] \overset{\text{def}}{\iff} w, s[X \mapsto S] \models \varphi(x)]$ for some $S \subseteq \mathcal{D}$

MSO(+1)-Definable Stringsets: $L(\varphi) \overset{\text{def}}{=} \{w \mid w \models \varphi\}$.

$\triangleleft^+$ is MSO-definable from $\triangleleft$, so there is no difference in terms of definability between **MSO**(+1) (for $\mathbb{W}^{\triangleleft}$ models) and **MSO**$(+1, <)$ (for $\mathbb{W}^{\triangleleft, \triangleleft^+}$ models).

Monadic Second-Order adds quantification over subsets of the domain. We use capital letters for set variables to distinguish them from individual variables (lower case). Again, there are no constants in this language so the only way to refer to specific sets is via these variables. We treat them as if they were monadic relation symbols: $X(x)$ asserts that the individual that is assigned to x is included in the set assigned to X.

To show that MSO$(+1, <) \equiv$ MSO(+1), it suffices to show that the $\triangleleft^+$ relation can be defined in MSO using only $\triangleleft$:

$$x \triangleleft^+ y \Leftrightarrow (\forall X)\Big[\big((\forall z_0, z_1)[(X(z_0) \wedge z_1 \triangleleft z_0) \to X(z_1)] \wedge X(y)\big) \to X(x)\Big]$$

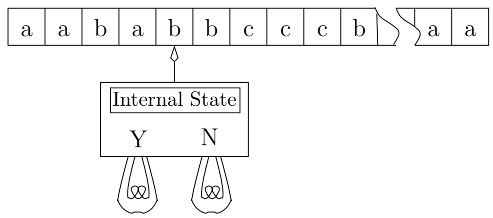
This says that every downward closed set (i.e., every set that includes the predecessors of all elements in the set) that includes y also includes x.

Exercise 24

- *Give an MSO$(+1, <)$ formula that is satisfied by all and only those strings that satisfy No-H-before-$\acute{H}$.*
- *Give an MSO(+1) formula (that does not use the MSO(+1) definition of $\triangleleft^+$) that does the same thing. (Hint, use an MSO variable to mark positions in the string. Then use $\exists X$ to erase the marks.)*

Slide 86

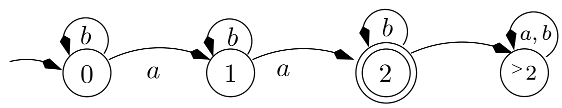
Finite State Automata



Finite State Automata can be thought of as scanners with a single symbol window and a state that stores arbitrary (but finitely bounded) information about the string that has been scanned so far in an internal state.

Slide 87

Finite State Automata (cont.)



$\langle 0, 1, a\rangle, \langle 0, 0, b\rangle, \langle 1, 2, a\rangle, \langle 1, 1, b\rangle, \langle 2, {}^{>}2, a\rangle, \langle 2, 2, b\rangle, \langle {}^{>}2, {}^{>}2, a\rangle, \langle {}^{>}2, {}^{>}2, b\rangle$

We can think of the FSA as a categorizer of strings; when it scans a string the state that it ends up in is the category of that string from the perspective of the FSA. The FSA places every string in Σ^* in at least one category. It is *deterministic* (a DFA) if it places each string in Σ^* in exactly one category; it is *non-deterministic* (an NFA) if it may place some strings in more than one. The information represented by a state is the set of properties of strings that are common to all of the strings that end up in that state.

When we say (on the previous slide) that the amount of information must be bounded, what we meant (precisely) is that there is a fixed finite bound on the number of categories, that is, the FSA has a fixed number of states. In particular, this means that the amount of information we are tracking can't depend on the length of the string.

When we say that it must be information about the string that has been scanned so far, we imply that it must be possible to keep track of that information as we scan the string one symbol at a time. What this means is that it must be possible to properly define a relation that tells how to update the state as the FSA scans a symbol. This is the *transition* relation of the FSA. It relates a pair of states with a symbol of the alphabet, e.g., $\langle q_i, q_j, \sigma\rangle$ which says that if the FSA is in state q_i and it is scanning the symbol σ it may go to state q_j. For a DFA, this relation is functional in the first and third component: for each q_i and σ there is exactly one q_j; if the DFA is in state q_i and is scanning σ it must go to state q_j.

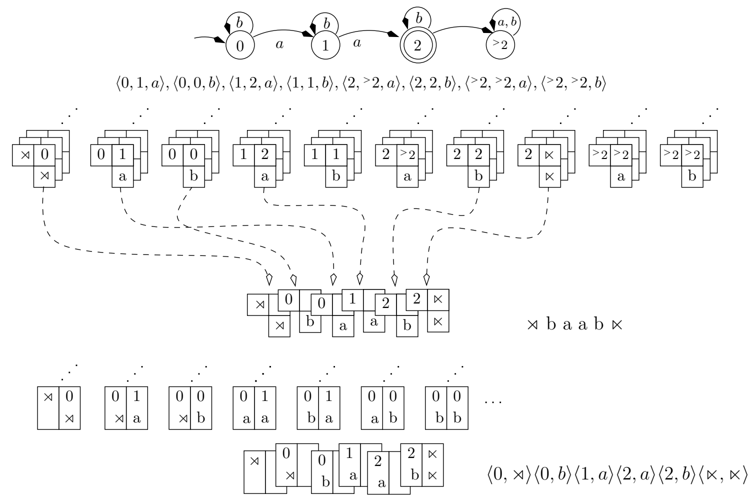
Some set of states are designated to be *accepting*, strings that are described by the information encoded in that state are strings that belong in the stringset the FSA defines. That stringset is the union of the sets of strings associated with those accepting states; we say that the FSA *recognizes* that set.

Exercise 25

- *Give a DFA that recognizes No-$\acute{H}$-before-$\acute{H}$.*
- *So No-$\acute{H}$-before-$\acute{H}$ is at most **Reg**. Show that it is actually only **SF**.*

Slide 88

FSA as Tiling Systems



Alternatively, we can interpret the triples of the transition relation as L-shaped tiles. The tiling is constrained by the states. This gives a tiling system that generates two strings in parallel: one a sequence of states and the other a sequence of symbols. The sequence of states is the sequence of states the FSA visits as it scans the sequence of symbols.

We can expand the tiles to square tiles by adding new tile types for each of the original tiles, a new type for each symbol of the alphabet in which the fourth corner has been filled in with that symbol.

We can think of these tiles as being pairs of pairs of a state and a symbol. This just gives us a new alphabet, one in which each "symbol" pairs a state and a symbol. The tiling, then, generates strings of these pairs.

With that perspective, the tiles are just an $\mathbf{SL}_2$ tiling system and the set of strings of pairs that it generates is just an $\mathbf{SL}_2$ stringset, one that happens to be strings of state/symbol pairs.

The key thing about this stringset is that, because of the way we constructed the generator out of the FSA tiling system, if we erase the state from each of the pairs in a string it generates, we are left with a string that is accepted by the FSA; if we do that for each of the strings in the $\mathbf{SL}_2$ stringset, we are left with the original stringset, which, of course, is a **Reg** stringset.

This is a remarkable connection between one of the weakest classes with one that, for our purposes, is the strongest.x

Slide 89

Projections of Stringsets

A *Projection* is an alphabetic homomorphism, a mapping of one alphabet into another: $h : \Gamma \to \Sigma$.

The image of a string under a projection is the result of applying that mapping to each symbol in the string in turn.

The image of a stringset under a projection is the set of images of the strings in the set.

Since the projection is functional, it can never gain information. The number of distinct symbols in the image of a string can never be more than the number of distinct symbols in the string itself.

In general projections may be many to one; they may lose information. We can think of them as striping away some of the distinctions that are made by the first alphabet.

Slide 90

Theorem 12 (Medvedev'64('56) [Med64]) *Every regular stringset is a projection (the image under an alphabetic homomorphism) of a strictly 2-local stringset.*

Let $\Gamma = Q \times \Sigma$ where Q is the set of states of an FSA. We've established that the set of strings over Γ which represent accepting runs of that automaton is $\mathbf{SL}_2$.

Let $h(\langle q, \sigma \rangle) = \sigma$. Then the image of the set of accepting runs under h is the set of strings that are accepted by the automaton.

Slide 91

Characterization of MSO$(+1)$

Definition 8 (Nerode equivalence)

$$w \equiv_L v \overset{def}{\Longleftrightarrow} (\forall u)[wu \in L \Leftrightarrow vu \in L].$$

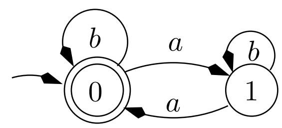
$$[w]_L \overset{def}{=} \{v \in \Sigma^* \mid w \equiv_L v\}$$

Theorem 13 *A stringset L is recognizable iff* ***card***$(\{[w]_L \mid w \in \Sigma^*\})$ *is finite. ($\equiv_L$ has finite index.)*

Nerode classes correspond to the minimal information that must be retained about a string in order to make a judgment about whether its continuations are members of the given stringset. As long as there are finitely many of these classes, these can be represented by a DFA.

Slide 92

MSO and Reg



$$
\begin{aligned}
(\exists X_0, X_1)[\quad & (\forall x,y)[(x \triangleleft y \wedge X_0(x) \wedge P_a(x)) \rightarrow X_1(y)] \wedge \\
& (\forall x,y)[(x \triangleleft y \wedge X_0(x) \wedge P_b(x)) \rightarrow X_0(y)] \wedge \\
& (\forall x,y)[(x \triangleleft y \wedge X_1(x) \wedge P_a(x)) \rightarrow X_0(y)] \wedge \\
& (\forall x,y)[(x \triangleleft y \wedge X_1(x) \wedge P_b(x)) \rightarrow X_1(y)] \wedge \\
& (\forall x)[\neg(\exists y)[y \triangleleft x] \rightarrow X_0(x)] \wedge \\
& (\forall x)[\neg(\exists y)[x \triangleleft y] \rightarrow X_0(x)] \qquad]
\end{aligned}
$$

MSO satisfaction is relative to the assignment of sets to MSO variables (as well as assignment of points to FO variables, but we can take these to be MSO variables with assignments restricted to be singleton sets).

Note that MSO variables pick out sets of points in same way that P_σ do.

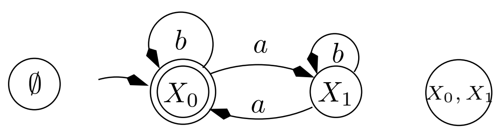
In order to capture a FSA with an MSO sentence, we can use these auxiliary labels to represent the state, as we did in capturing runs of the FSA in $\mathbf{SL}_2$. We require each position to be labeled with some state and Each transition of the DFA can then be captured with an MSO sentence, as can the requirements that the initial position is labeled with a start state and the final position with a final state. The conjunction of these defines a set of strings corresponding to the runs of the DFA.

We can then project away the extra labels by existentially binding them.

Slide 93

Automata for MSO

$$
\begin{array}{ll}
(\exists X_0, X_1)[& (\forall x,y)[(x \triangleleft y \wedge X_0(x) \wedge P_a(x)) \rightarrow X_1(y)] \wedge \\
 & (\forall x,y)[(x \triangleleft y \wedge X_0(x) \wedge P_b(x)) \rightarrow X_0(y)] \wedge \\
 & (\forall x,y)[(x \triangleleft y \wedge X_1(x) \wedge P_a(x)) \rightarrow X_0(y)] \wedge \\
 & (\forall x,y)[(x \triangleleft y \wedge X_1(x) \wedge P_b(x)) \rightarrow X_1(y)] \wedge \\
 & (\forall x)[\neg(\exists y)[y \triangleleft x] \rightarrow X_0(x)] \wedge \\
 & (\forall x)[\neg(\exists y)[x \triangleleft y] \rightarrow X_0(x)] \qquad\qquad\qquad]
\end{array}
$$



In building an automaton that recognizes the set of strings satisfying a given **MSO** sentence, the key requirement is, in essence, to invert the construction of the previous slide. Where we had used **MSO** variables to represent the states of the automaton, we will use the states of the automaton to encode the assignments of the **MSO** variables. Each state represents a subset of the free variables in the **MSO** formula. (WLOG we assume that all free variables are **MSO**). A string will end up in a given state iff the last position of the string is a member of each of the sets of positions assigned to the **MSO** variables encoded by the state.

The actual construction is done recursively on the structure of the formula. We start with automata for the atomic formulae and then construct automata for the compound formulae using these. For the most part, this involves standard automata construction techniques: union, determinization and complement, in particular. The construction for existential quantification is more complicated in that it involves a change in the alphabet—the number of free variables in the matrix of the formula is one more than that of the formula itself.

Slide 94

Cognitive Complexity of Reg

- Any cognitive mechanism that can distinguish member strings from non-members of a finite-state 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* finite-state stringsets.

This does not imply that such a mechanism actually requires unbounded resources. It could employ a mechanism that, in principle, requires unbounded storage which fails on sufficiently long or sufficiently complicated inputs.

Or would if it ever encountered such.

Slide 95

Yidin Reprise

- One-$\acute{\sigma}$ $\quad (\exists ! x)[\acute{\sigma}(x)]$ $\quad$ (**LTT**$_{1,2}$)
- No-H-before-$\acute{H}$ $\quad \neg(\exists x, y)[x \triangleleft^{+} y \wedge H(x) \wedge \acute{H}(y)]$ $\quad$ (**SF**)
- No-H-with-$\acute{L}$ $\quad \neg(H \wedge \acute{L})$ $\quad$ (**LT**$_1$)
- Nothing-before-$\acute{L}$ $\quad \neg\, \sigma\, \acute{L}$ $\quad$ (**SL**$_2$)
- Alt $\quad \neg\, \sigma\, \sigma \wedge \neg\, \acute{\sigma}\, \acute{\sigma} \wedge \neg\, \acute{\sigma}\, \grave{\sigma} \wedge \neg\, \grave{\sigma}\, \acute{\sigma} \wedge \neg\, \grave{\sigma}\, \grave{\sigma}$ $\quad$ (**SL**$_2$)
- No $\rtimes \acute{L} \ltimes$ $\quad \neg \rtimes \acute{L} \ltimes$ $\quad$ (**SL**$_3$)

Yidin is **SF**

Exercise 26 *The* ***FO***(+1) *formula establishes that No-H-before-$\acute{H}$ is* ***Reg****, not that it is* ***SF****. Show that it is* ***SF*** *(without using the Day 4 results).*

Slide 96

Summary of Part 3.3

- We introduced the syntax and semantics of Monadic Second-Order logic for $\mathbb{W}^{\triangleleft}$: MSO(+1)
- We introduced Finite State Automata, focusing on them as classifiers of strings. A stringset is **Reg** iff it is recognizable by an FSA.
- You showed that No-H-before-$\acute{H}$ is an MSO(+1) definable constraint. You also showed that it is **SF**, so we still don't have a good bound on its complexity.
- We introduced a tiling system for FSAs.
- We introduced projections of stringsets and used this, along with the tiling, to show that every **Reg** stringset is actually a projection of an $\mathbf{SL}_2$ stringset.

Slide 97

Summary of Part 3.3

- We have observed that MSO(+1) and **Reg** are equivalent.
- We gave Nerode's characterization of the **Reg** stringsets.
- We considered the cognitive complexity of **Reg** constraints.
- We showed that the complexity of No-H-before-$\acute{H}$ determines the overall complexity of the stress pattern of Yidin. Which is **SF** when viewed from the local perspective.

We have been busy little beavers.

Slide 98

Overview Session 4

- Harmony
- Subsequences
- Strictly Piecewise Languages/Restricted Propositional($<$)
- Piecewise Testable Languages/Propositional($<$)
- Star-Free Languages/FO($<$)
- Co-occurrence classes: Local+Piecewise/Propositional($+1, <$)

Slide 99

Long-Distance Dependencies

Samala (Chumash) sibilant harmony:

s does not occur in the same word as ʃ

[**ʃ**tojonowonowa**ʃ**] ‘it stood upright’ *[**ʃ**tojonowonowa**s**]

$$\overline{(\Sigma^* \cdot \mathrm{s} \cdot \Sigma^* \cdot \int \cdot \Sigma^*) + (\Sigma^* \cdot \int \cdot \Sigma^* \cdot \mathrm{s} \cdot \Sigma^*)}$$

Sarcee sibilant harmony:

s does not occur before ʃ

a. /si-tʃiz-aʔ/ → **ʃ**í**tʃ**íd**z**àʔ ‘my duck’

b. /na-s-ɣatʃ/ → nā**ʃ**ɣát**ʃ** ‘I killed them again’

c. cf. ⋆**s**í**tʃ**íd**z**àʔ

$$\overline{\Sigma^* \cdot \mathrm{s} \cdot \Sigma^* \cdot \int \cdot \Sigma^*}$$

Two kinds of sibilant harmony:

- Samala—symmetric
  - s does not occur *with* ʃ (either order).
- Sarcee—asymmetric
  - s does not occur *before* ʃ (but may come after).

Slide 100

Complexity of Sibilant Harmony

Symmetric sibilant harmony (Samala) is LT

$$\neg(\int \wedge \mathrm{s})$$

Asymmetric sibilant harmony (Sarcee) is not FO$(+1)$

$$\rtimes\, w \int w\, \mathrm{s}\, w \ltimes$$
$$\equiv^{L}_{k,t}$$
$$\star\ \rtimes w \int w\, \mathrm{s}\, w \int w \ltimes$$

Slide 101

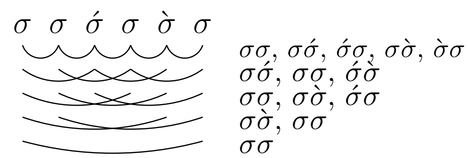
Precedence—Subsequences

Definition 9 (Subsequences)

$$v \sqsubseteq w \overset{def}{\Longleftrightarrow} v = \sigma_1 \cdots \sigma_n \textit{ and } w \in \Sigma^* \cdot \sigma_1 \cdot \Sigma^* \cdots \Sigma^* \cdot \sigma_n \cdot \Sigma^*$$

$$P_k(w) \overset{def}{=} \{v \in \Sigma^k \mid v \sqsubseteq w\}$$

$$P_{\leq k}(w) \overset{def}{=} \{v \in \Sigma^{\leq k} \mid v \sqsubseteq w\}$$



$$P_2(\sigma\,\sigma\,\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma) = \{\sigma\,\sigma, \sigma\,\acute{\sigma}, \sigma\,\grave{\sigma}, \acute{\sigma}\,\sigma, \acute{\sigma}\,\grave{\sigma}, \grave{\sigma}\,\sigma\}$$

$$P_{\leq 2}(\sigma\,\sigma\,\acute{\sigma}\,\sigma\,\grave{\sigma}\,\sigma) = \{\varepsilon, \sigma, \acute{\sigma}, \grave{\sigma}, \sigma\,\sigma, \sigma\,\acute{\sigma}, \sigma\,\grave{\sigma}, \acute{\sigma}\,\sigma, \acute{\sigma}\,\grave{\sigma}, \grave{\sigma}\,\sigma\}$$

Redo same sequence of classes but with arbitrary ($\triangleleft^+$) rather than immediate ($\triangleleft$) precedence.

Technical reasons: subsequences of length $\leq k$: $P_{\leq k}$

Slide 102

Word Models for Subsequences

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

Lemma 4 *If $\mathcal{M}_w^{\triangleleft^+}$ and $\mathcal{M}_v^{\triangleleft^+}$ are precedence models for the strings w and v, respectively, then*

$$\mathcal{M}_w^{\triangleleft^+} \precsim \mathcal{M}_v^{\triangleleft^+} \Leftrightarrow w \sqsubseteq v$$

To parallel the local side of the hierarchy completely, we could have used $\precsim$ for subsequence as well as substring (since they are both submodels). But since we will eventually want to talk about both relations at the same time we will distinguish them.

Slide 103

Restricted Propositional Logic (RPL)

A *sentence* of RPL is defined recursively as follows.

1. *The base cases:*
   - For all $w \in \Sigma^*$, $(\neg w)$ is a sentence of RPL.
2. *The inductive case:*
   - If ϕ and ψ are sentences of RPL then so is $(\phi \wedge \psi)$.
3. Nothing else is a sentence of RPL.

We repeat here the almost exactly the same definitions for syntax and semantics of RPL.

The only difference in the syntax is that we can no longer use the endmarkers $\{\rtimes, \ltimes\}$. This is because the ends of the strings are local phenomena and we want to restrict the languages on the piecewise side of the hierarchy to phenomena with arbitrary radius.

Slide 104

Restricted Propositional Logic - Stringsets

- Consider any $v \in \Sigma^*$.
  1. *The base cases:*
     - For all $w \in \Sigma^*$, $\mathcal{M}_v \models (\neg w) \Leftrightarrow \mathcal{M}_w \not\sqsubseteq \mathcal{M}_v$.
  2. *The inductive case:*
     - For all ϕ, ψ in RPL, $v \models (\phi \wedge \psi) \Leftrightarrow v \models \phi$ and $v \models \psi$.
- Then

$$\mathbb{L}_{\mathrm{RPL}}(\phi) = \{w \mid \mathcal{M}_w \models \phi\}$$

Slide 105

Strictly Piecewise Stringsets—SP [RHB+10]

Definition 10 (Strictly Piecewise Stringsets) *A stringset is Strictly Piecewise iff the $\mathcal{M}^{\triangleleft^+}$ models of its member strings is $\mathbb{L}_{RPL}(\phi)$ for some RPL sentence ϕ.*

Definition 11 (Strictly Piecewise Grammars) *A Strictly k-Piecewise Grammar $\mathcal{G} = \langle \Sigma, T \rangle$ where T is a subset of $\Sigma^{\leq k}$ and*

$$\mathbb{L}_k^{SP}\big(\langle \Sigma, T \rangle\big) \stackrel{def}{=} \{w \in \Sigma^* \mid P_{\leq k}(w) \subseteq T\}.$$

Membership in an $\mathbf{SP}_k$ stringset depends only on the individual ($\leq k$)-subsequences which do and do not occur in the string.

Again, the only distinction is the interpretation of the elements of T. Heinz [Hei07] defined an equivalent class as Precedence Languages.

Slide 106

Character of the Strictly k-Piecewise Sets [RHB+10]

Theorem 14 *A stringset L is Strictly k-Piecewise Testable iff it is closed under subsequence:*

$$w\sigma v \in L \Rightarrow wv \in L$$

Every naturally occurring stress pattern requires Primary Stress

$\Rightarrow$

No naturally occurring stress pattern is **SP**.

But **SP** can forbid multiple primary stress: $\neg\,\acute{\sigma}\,\acute{\sigma}$

Slide 107

Yidin constraints wrt SP

- One-$\acute{\sigma}$ is not **SP** — $\star\ \sigma\,\sigma\ \sqsubseteq\ \sigma\,\acute{\sigma}\,\sigma$
- No-H-before-$\acute{H}$ is $\mathbf{SP}_2$ — $\neg\, H\,\acute{H}$
- No-H-with-$\acute{L}$ is $\mathbf{SP}_2$ — $\neg\, H\,\acute{L} \wedge \neg\,\acute{L}\,H$
- Nothing-before-$\acute{L}$ is $\mathbf{SP}_2$ — $\neg\sigma\,\acute{L}$
- Alt is not **SP** — $\star\ \sigma\,\sigma\,\acute{\sigma}\ \sqsubseteq\ \sigma\,\grave{\sigma}\,\sigma\,\acute{\sigma}$
- No $\rtimes\,\acute{L}\,\ltimes$ is not **SP** — $\star\ \acute{L}\ \sqsubseteq \acute{L}\,L$

Slide 108

Cognitive interpretation of SP

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) $\mathbf{SP}_k$ stringset must be sensitive, at least, to the length k (not necessarily consecutive) sequences 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 up to $k - 1$ events distributed arbitrarily among the prior events.
- Any cognitive mechanism that is sensitive *only* to the length k sequences of events in the presentation of a string will be able to recognize *only* $\mathbf{SP}_k$ stringsets.

Slide 109

Full Propositional Logic for $\mathcal{W}^{\triangleleft^+}$ (Prop(<)) —Syntax

k-Piecewise-Expressions

k-Piecewise-expressions are defined inductively as follows.

1. *The base cases:*
   - For all $w \in \Sigma^{\leq k}$, w is a k-Piecewise-expression.
2. *The inductive cases:*
   - If ϕ is a k-Piecewise-expression then so is $(\neg\phi)$.
   - If ϕ and ψ are k-Piecewise-expressions then so is $(\phi \wedge \psi)$.
3. Nothing else is a k-Piecewise-expression.

Again, the only change in the syntax is the loss of the endmarkers…

Slide 110

Full Propositional Logic for $\mathcal{W}^{\triangleleft^+}$ (Prop(<)) —Semantics

Consider any $v \in \Sigma^*$ and any k-Piecewise-expression ϕ:

1. *The base cases:*
   - If $\phi = w \in \Sigma^{\leq k}$, $\mathcal{M}_v \models \phi \Leftrightarrow \mathcal{M}_w \precsim \mathcal{M}_v$.
2. *The recursive case:*
   - If $\phi = (\neg\psi)$ then $\mathcal{M}_v \models \phi \Leftrightarrow \mathcal{M}_v \not\models \psi$.
   - If $\phi = \psi_1 \vee \psi_2$ then $\mathcal{M}_v \models \phi \Leftrightarrow$ either $\mathcal{M}_v \psi_1$ or $\mathcal{M}_v \psi_2$

$$\mathbb{L}(\varphi) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \mathcal{M}_w \models \phi\}.$$

A stringset is *k-piecewise definable* iff it is $\mathbb{L}(\varphi)$ for some k-piecewise-expression φ. It is *piecewise definable* iff it is k-piecewise definable for some k.

…and the type of the models.
Imre Simon [Sim75] first introduced this class.

Slide 111

k-Piecewise Grammars

Definition 12 (k-Piecewise Testable Stringsets) *A k-Piecewise Grammar is a pair $\mathcal{G} = \langle \Sigma, T \rangle$ where T is a subset of $\mathcal{P}(\Sigma^{\leq k})$.*

The stringset licsensed by $\mathcal{G}$ is

$$\mathbb{L}_{PT}(\langle \Sigma, T \rangle) \stackrel{def}{=} \{w \mid P_{\leq k}(w) \in T\}.$$

A stringset L is k-piecewise if there exists a k-piecewise $\mathcal{G}$ such that $\mathbb{L}_{PT}(\mathcal{G}) = L$. Such stringsets form the exactly the k-Piecewise Testable stringsets ($\boldsymbol{PT}_k$).

A stringset is Piecewise Testable if there exists a k such that it is k-piecewise. Such stringsets form exactly the Locally Testable stringsets ($\boldsymbol{PT}$).

Slide 112

Character of Piecewise Testable sets

Theorem 15 (k-Subsequence Invariance) *A stringset L is Piecewise Testable iff*

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

if x and y have exactly the same set of $(\leq k)$-subsequences

then either both x and y are members of L or neither is.

$$w \equiv_k^P v \overset{\text{def}}{\Longleftrightarrow} P_{\leq k}(w) = P_{\leq k}(v).$$

Slide 113

Yidin constraints wrt SP

- One-$\acute{\sigma}$ is $\mathbf{PT}_2$ $\quad \acute{\sigma} \wedge \neg\, \acute{\sigma}\, \acute{\sigma}$
- No-H-before-$\acute{H}$ is $\mathbf{SP}_2$ $\quad \neg\, H\, \acute{H}$
- No-H-with-$\acute{L}$ is $\mathbf{SP}_2$ $\quad \neg\, H\, \acute{L} \wedge \neg\, \acute{L}\, H$
- Nothing-before-$\acute{L}$ is $\mathbf{SP}_2$ $\quad \neg \sigma\, \acute{L}$
- Alt is not **PT** $\quad \star\ \overbrace{\sigma\,\grave{\sigma} \cdots \sigma\,\grave{\sigma}}^{2k} \equiv^{P}_{k} \overbrace{\sigma\,\grave{\sigma} \cdots \sigma\,\grave{\sigma}}^{2k}\grave{\sigma}$
- No $\rtimes \acute{L} \ltimes$ is $\mathbf{PT}_2$ $\quad \acute{L} \rightarrow (\sigma\, \acute{L} \vee \acute{L}\, \sigma)$

Slide 114

Cognitive interpretation of PT

- Any cognitive mechanism that can distinguish member strings from non-members of a (properly) $\mathbf{PT}_k$ stringset must be sensitive, at least, to the set of length k subsequences of events that occur in the presentation of the string—both those that do occur and those that do not.
- 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 set of all length k subsequences of the sequence of prior events.
- Any cognitive mechanism that is sensitive *only* to the set of length k subsequences of events in the presentation of a string will be able to recognize *only* $\mathbf{PT}_k$ stringsets.

Slide 115

FO$(<)$

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

First-order Quantification (over positions in the strings)

Syntax	Semantics		
$x \approx y$	$w, [x \mapsto i, y \mapsto j] \models x \approx y$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$j = i$
$x \triangleleft^+ y$	$w, [x \mapsto i, y \mapsto j] \models x \triangleleft^+ y$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$i < j$
$P_\sigma(x)$	$w, [x \mapsto i] \models P_\sigma(x)$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$i \in P_\sigma$
$\varphi \wedge \psi$	$\vdots$		
$\neg\varphi$	$\vdots$		
$(\exists x)[\varphi(x)]$	$w, s \models (\exists x)[\varphi(x)]$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$w, s[x \mapsto i] \models \varphi(x)$ for some $i \in \mathcal{D}$

FO$(<)$-Definable Stringsets: $\mathbb{L}(\varphi) \stackrel{\text{def}}{=} \{w \mid w \models \varphi\}$.

Slide 116

FO(<) Definability

$\triangleleft$ is *FO*(<) definable

$$R_{\triangleleft}(x, y) \equiv x \triangleleft^{+} y \wedge (\forall z)[x \triangleleft^{+} z \rightarrow \neg z \triangleleft^{+} y]$$

Hence **FO**(+1) $\subsetneq$ **FO**(<). No-H-before-$\acute{H}$ witnesses that the inclusion is proper.

Alt is *FO*(<)

$$(\forall x, y)[R_{\triangleleft}(x, y) \rightarrow (\sigma(x) \leftrightarrow \overset{+}{\sigma}(y))]$$

Slide 117

Star Free Expressions - Grammars and Stringsets

- A Star Free Expression is a GRE containing no ‘And’ (&) or Kleene star (*).

$$\boxed{\cdot,\ +,\ \overline{}}$$

SF is the closure of **Fin** under concatenation, union and complement.

Slide 118

FO(<) and SF

To show that SF $\subseteq$ FO(<)

- **Fin** $\subsetneq$ **SL** $\subsetneq$ **FO**$(+1) \subsetneq$ **FO**$(<)$.
- **FO**$(<)$ is closed under disjunction by definition.
- Concatenation:
  If ϕ is a **FO** formula, let $\phi|_{\langle l,r\rangle}(l,r)$ be the relativization of ϕ to the interval $[l,r]$, where $\phi|_{\langle l,r\rangle}(l,r)$ is syntactically identical to ϕ except that each '$(\exists x)[\psi(x)]$' is replaced by '$(\exists x)[l \triangleleft^* x \wedge x \triangleleft^* r \wedge \psi(x)]$'
  Let $L_1 = \mathbb{L}(\phi_1)$ and $L_2 = \mathbb{L}(\phi_2)$. Then $L_1 \cdot L_2$ is $\mathbb{L}(\phi_{1\cdot 2})$ where

$$\phi_{1\cdot 2} \stackrel{\text{def}}{=} (\exists x_1, x_2, x_3)[\phi_1|_{\langle l,r\rangle}(x_1,x_2) \wedge \phi_2|_{\langle l,r\rangle}(x_2,x_3)]$$

Slide 119

FO(<) and SF

Theorem 16 (McNaughton & Papert [MP71]) *A set of strings is First-order definable over* $\mathbb{W}_{\triangleleft^+}$ *iff it is* Star-Free.

Slide 120

Yidin wrt Local and Piecewise Constraints

One-$\acute{\sigma}$	$\mathbf{LTT}_{1,2}$	$\mathbf{PT}_2$
Some-$\acute{\sigma}$	$\mathbf{LT}_1$	$\mathbf{PT}_1$
At-Most-One-$\acute{\sigma}$	$\mathbf{LTT}_{1,2}$	$\mathbf{SP}_2$
No-H-before-$\acute{H}$	$\boxed{\mathbf{SF}}$	$\mathbf{SP}_2$
No-H-with-$\acute{L}$	$\mathbf{LT}_1$	$\mathbf{SP}_2$
Nothing-before-$\acute{L}$	$\mathbf{SL}_2$	$\mathbf{SP}_2$
Alt	$\mathbf{SL}_2$	$\boxed{\mathbf{SF}}$
No $\rtimes \acute{L} \ltimes$	$\mathbf{SL}_3$	$\mathbf{PT}_2$

Yidin is **SF** with either local or piecewise constraints.

Slide 121

Yidin wrt Local and Piecewise Constraints

One-$\acute{\sigma}$	$\mathbf{LTT}_{1,2}$	$\mathbf{PT_2}$
Some-$\acute{\sigma}$	$\boxed{\mathbf{LT_1}}$	$\boxed{\mathbf{PT_1}}$
At-Most-One-$\acute{\sigma}$	$\mathbf{LTT}_{1,2}$	$\boxed{\mathbf{SP_2}}$
No-H-before-$\acute{H}$	**SF**	$\boxed{\mathbf{SP_2}}$
No-H-with-$\acute{L}$	$\mathbf{LT}_1$	$\boxed{\mathbf{SP_2}}$
Nothing-before-$\acute{L}$	$\boxed{\mathbf{SL_2}}$	$\boxed{\mathbf{SP_2}}$
Alt	$\boxed{\mathbf{SL_2}}$	**SF**
No $\rtimes \acute{L} \ltimes$	$\boxed{\mathbf{SL_3}}$	$\mathbf{PT}_2$

Yidin is co-occurence of **SL** and **PT** constraints or of **LT** and **SP** constraints

Slide 122

Stress Patterns wrt Local Constraints

- **SL** — 89 of 109 patterns
- **LT**
  None
- **LTT**
  Alawa, Bulgarian, Murik
- **SF**
  Amele, Arabic (Classical), Buriat, Cheremis (East), Cheremis (Meadow), Chuvash, Golin, Komi, Kuuku Yau, Lithuanian, Mam, Maori, K. Mongolian (Street), K. Mongolian (Stuart), K. Mongolian (Bosson), Nubian, Yidin
- **Reg**
  Arabic (Cairene), Arabic (Negev Bedouin), Arabic (Cyrenaican Bedouin)

Slide 123

Stress Patterns wrt Piecewise Constraints

- **SP**

  None

- **PT**

  Amele, Bulgarian, Chuvash, Golin, Lithuanian, Maori K. Mongolian (Street), Murik,

- **SF**

  Alawa, Arabic (Classical), Buriat, Cheremis (East), Cheremis (Meadow), Komi, Kuuku Lau, Mam, K. Mongolian (Bosson), K. Mongolian (Stuart), Nubian, Yidin

- **Reg**

  Arabic (Cairene), Arabic (Negev Bedouin), Arabic (Cyrenaican Bedouin)

Don't know where the **SL** patterns fall

Slide 124

Stress Patterns wrt Co-occurrence of Local and Piecewise Constraints

- **SL + SP** — 89 of 109 patterns
- **SL + PT** — Komi, Kuuku Lau, Yidin
- **LT + SP**

  Alawa Amele, Arabic (Classical), Bulgarian, Buriat, Cheremis (East), Cheremis (Meadow), Chuvash, Golin, Komi, Kuuku Lau, Lithuanian, Mam, Maori K. Mongolian (Bosson), K. Mongolian (Street), K. Mongolian (Stuart), Murik, Nubian, Yidin
- **SF** — None
- **Reg**

  Arabic (Cairene), Arabic (Negev Bedouin), Arabic (Cyrenaican Bedouin)

Those in **SL + PT** constraints are subset of those in **LT + SP** constraints.

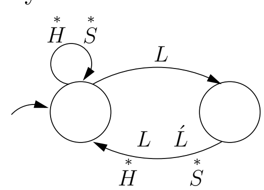
Slide 125

Arabic (Negev Bedouin)

- In sequences of light syllables, secondary stress falls on the even numbered syllables, counting from the left edge of the sequence.
- This pattern is used only for the sake of defining main stress. Secondary stress is absent on the surface.

Without reference to secondary stress

- Odd number of unstressed light syllables precedes a light syllable with primary stress



No $\acute{L}$ out of LH state

Slide 126

Arabic (Negev Bedouin) with explicit secondary stress

$$\varphi_{\text{Lalt}} = \neg\, L\, L \wedge \neg\, \grave{L}\, \grave{L} \wedge \neg\, \grave{L}\, \acute{L} \wedge \neg\, \acute{L}\, \grave{L} \wedge \neg\, \overset{*}{H}\, L \wedge \neg\, \overset{*}{S}\, L$$

If secondary stress is explicit, then Arabic (Negev Bedouin) is **LT**

Slide 127

Some Constraints

- Forbidden syllables ($\mathbf{SL}_1$, $\mathbf{SP}_1$)
  - No heavy syllables
- Required syllables ($\mathbf{LT}_1$, $\mathbf{PT}_1$)
  - Some primary stress
- Forbidden initial/final syllables ($\mathbf{SL}_2$, $\mathbf{SF}$)
  - Cannot start with unstressed light
  - Cannot start with unstressed heavy
  - Cannot end with stressed light
- Forbidden adjacent pairs ($\mathbf{SL}_2$, $\mathbf{SF}$)
  - No adjacent unstressed
  - No adhacent secondary stress
  - No heavy immediately following a stressed light

...

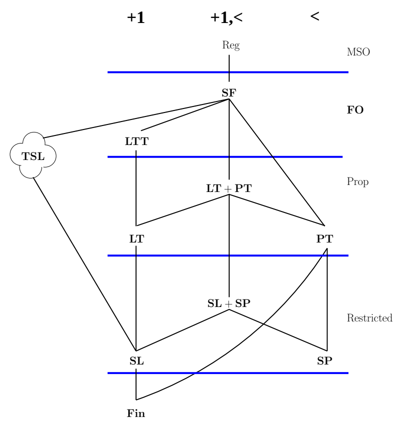
$L \wedge \neg \sigma\, L \Leftrightarrow \rtimes L$

Slide 128

Properly Regular Constraints

- Alternation (**Reg**)
  - Arabic (Negev Bedouin), . . .
  - This class of constraints accounts for all properly regular stress patterns (that are known to us).

What we covered in this course (in pictures)



Slide 129

Thanks for your excellent participation!

Apart from the notes Jim will send around, here are some references for further reading [MP71, RHB$^+$10, RHF$^+$13].

References

[App72] R.B. Applegate. *Ineseño Chumash Grammar*. PhD thesis, University of California, Berkeley, 1972.

[BB75] M. Blum and L. Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[Dix77] M. W. Dixon. *A Grammar of Yidin*. Cambridge University Press, 1977.

[ELM+08] Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. Deciding strictly local (SL) languages. In Jon Breitenbucher, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pages 66–73, 2008.

[Gol67] E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Gra10] Thomas Graf. Comparing incomparable frameworks: A model theoretic approach to phonology. *University of Pennsylvania Working Papers in Linguistics*, 16(2):Article 10, 2010. Available at: `http://repository.upenn.edu/pwpl/vol16/iss1/10`.

[Gra13] Thomas Graf. *Local and Transderivational Constraints in Syntax and Semantics*. PhD thesis, University of California, Los Angeles, 2013.

[Hal78] Morris Halle. Knowledge unlearned and untaught: What speakers know about the sounds of their language. In *Linguistic Theory and Psychological Reality*. The MIT Press, 1978.

[Hei07] Jeffrey Heinz. *The Inductive Learning of Phonotactic Patterns*. PhD thesis, University of California, Los Angeles, 2007.

[Hei09] Jeffrey Heinz. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351, 2009.

[Hei10] Jeffrey Heinz. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[Hei14] Jeffrey Heinz. Computational theories of learning and developmental psycholinguistics. In Jeffrey Lidz, William Synder, and Joe Pater, editors, *The Oxford Handbook of Developmental Linguistics*. Oxford University Press, 2014. To appear.

[HH69] Kenneth Hansen and L.E. Hansen. Pintupi phonology. *Oceanic Linguistics*, 8:153–170, 1969.

[HKK12] Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127, October 2012.

[HV87] Morris Halle and Jean-Roger Vergnaud. *An Essay on Stress*. The MIT Press, 1987.

[Hym09] Larry M. Hyman. How (not) to do phonological typology: the case of pitch-accent. *Language Sciences*, 31(2-3):213 – 238, 2009. Data and Theory: Papers in Phonology in Celebration of Charles W. Kisseberth.

[JORS99] Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. *Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change)*. The MIT Press, 2nd edition, 1999.

[KK10] Anna Kasprzik and Timo Kötzing. String extension learning using lattices. In Henning Fernau Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, volume 6031 of *Lecture Notes in Computer Science*, pages 380–391, Trier, Germany, 2010. Springer.

[LZZ08] Steffen Lange, Thomas Zeugmann, and Sandra Zilles. Learning indexed families of recursive languages from positive data: A survey. *Theoretical Computer Science*, 397:194–232, 2008.

[Med64] Yu. T. Medvedev. On the class of events representable in a finite automaton. In Edward F. Moore, editor, *Sequential Machines; Selected Papers*, pages 215–227. Addison-Wesley, 1964. Originally published in Russian in Avtomaty, 1956, 385–401.

[MP71] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

[Odd05] David Odden. *Introducing Phonology*. Cambridge University Press, 2005.

[OWS86] Daniel Osherson, Scott Weinstein, and Michael Stob. *Systems that Learn*. MIT Press, Cambridge, MA, 1986.

[PP02] Christopher Potts and Geoffrey Pullum. Model theory and the content of OT constraints. *Phonology*, 19:361–393, 2002.

[Pul07] Geoffrey K. Pullum. The evolution of model-theoretic frameworks in linguistics. In James Rogers and Stephan Kepser, editors, *Model-Theoretic Syntax at 10*, pages 1–10, Dublin, Ireland, 2007.

[RHB+10] James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artifical Intelligence*, pages 255–265. Springer, 2010.

[RHF+13] James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. Cognitive and sub-regular complexity. In Glyn Morrill and Mark-Jan Nederhof, editors, *Formal Grammar*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[Rog94] James Rogers. *Studies in the Logic of Trees with Applications to Grammatical Formalisms*. PhD thesis, University of Delaware, 1994. Published as Technical Report 95-04 by the Department of Computer and Information Sciences.

[Sim75] Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222. 1975.

[ZZ08] Thomas Zeugmann and Sandra Zilles. Learning recursive functions: A survey. *Theoretical Computer Science*, 397(1-3):4–56, 2008.