The computational nature of phonological generalizations

Jeffrey Heinz

Linguistics Department

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Stony Brook University

University of Pennsylvania

November 16, 2017

Conclusion

The computational nature of phonology matters because:

1. It provides well-studied methods for relating extensional and intensional descriptions of generalizations.
2. It provides a mathematical foundation for comparing representation and logical power.
3. It often directly leads to psychological models of representation, memory and processing.
4. These models specify what learners must attend to, and thus explains the kinds of phonological generalizations that can be learned.
5. It makes typological predictions and provides explanations for the phonological generalizations we do and do not observe.

Today

I will argue that the computational nature of phonological generalizations is

1. not only “regular”, but also
2. “less than” regular in a particularly “local” way

Topics covered second half

1. Constraints on local structures (Rogers)
2. Constraints on long-distance segmental structures (Heinz)
3. Constraints on tonal patterns of well-formedness (Jardine)
4. Segmental transformations based on local structures (Chandlee)
5. Syllabification is also a local computation (Strother-Garcia)

Part I

What is phonology?

The fundamental insight

The fundamental insight in the 20th century which shaped the development of generative phonology is the following.

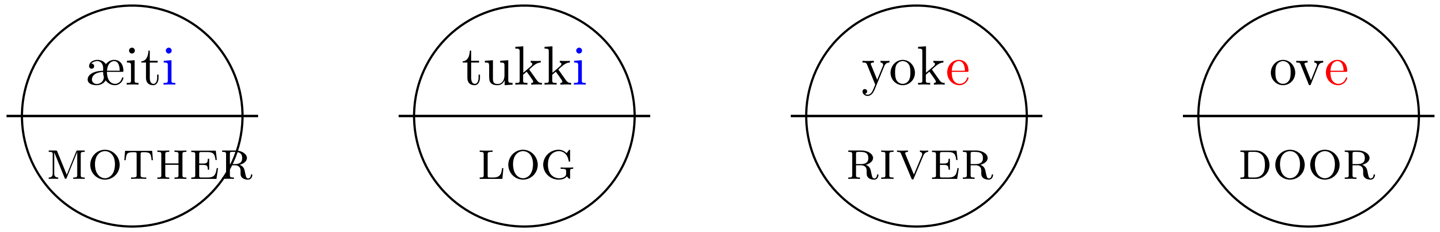
The **best** explanation of the systematic variation in the pronunciation of morphemes is to posit a single underlying mental representation of the phonetic form of each morpheme and to derive its pronounced variants with context-sensitive transformations.

(Kenstowicz and Kisseberth 1979, chap 6; Odden 2014, chap 5)

Example from Finnish

Nominative Singular	Partitive Singular	
aamu	aamua	‘morning’
kello	kelloa	‘clock’
kylmæ	kylmææ	‘cold’
kømpelø	kømpeløæ	‘clumsy’
æiti	æitiæ	‘mother’
tukki	tukkia	‘log’
yoki	yokea	‘river’
ovi	ovea	‘door’

Mental Lexicon



Word-final /e/ raising

1. e $\longrightarrow$ [+high] / __ #
2. *e# >> IDENT(HIGH)

If your theory asserts that . . .

There exist underlying representations of morphemes which are transformed to surface representations. . .

Then there are three important questions:

1. **What is the nature** of the abstract, underlying, lexical representations?
2. **What is the nature** of the concrete, surface representations?
3. **What is the nature** of the transformation from underlying forms to surface forms?

Theories of Phonology. . .

- disagree on the answers to these questions, but *they agree on the questions being asked.*

Phonological constraints and transformations are infinite objects

Extensions of grammars in phonology are infinite objects in the same way that perfect circles represent infinitely many points.

Word-final /e/ raising

1. *e#
2. *e# >> IDENT(HIGH)
3. $\mathrm{e} \longrightarrow [+\mathrm{high}] \;/\; \_\_\, \#$

Nothing precludes these grammars from operating on words of *any* length.

(ove,ovi), (yoke,yoki), (tukki,tukki),
(kello,kello),. . . (manilabanile,manilabanili), . . .

Grammars as functions

function	Notes	
$f : \Sigma^* \to \{0, 1\}$	Binary classification	(well-formedness)
$f : \Sigma^* \to \mathbb{N}$	Maps strings to numbers	(well-formedness)
$f : \Sigma^* \to [0, 1]$	Maps strings to real values	(well-formedness)
$f : \Sigma^* \to \Delta^*$	Maps strings to strings	(transformation)
$f : \Sigma^* \to \wp(\Delta^*)$	Maps strings to sets of strings	(transformation)

Truisms about grammars

1. Different grammars may generate the same constraints and transformations.

   Such grammars are *extensionally equivalent.*

2. Grammars are finite, *intensional* descriptions of their (possibly infinite) *extensions.*

3. Transformations may have properties *largely independent* of their grammars.

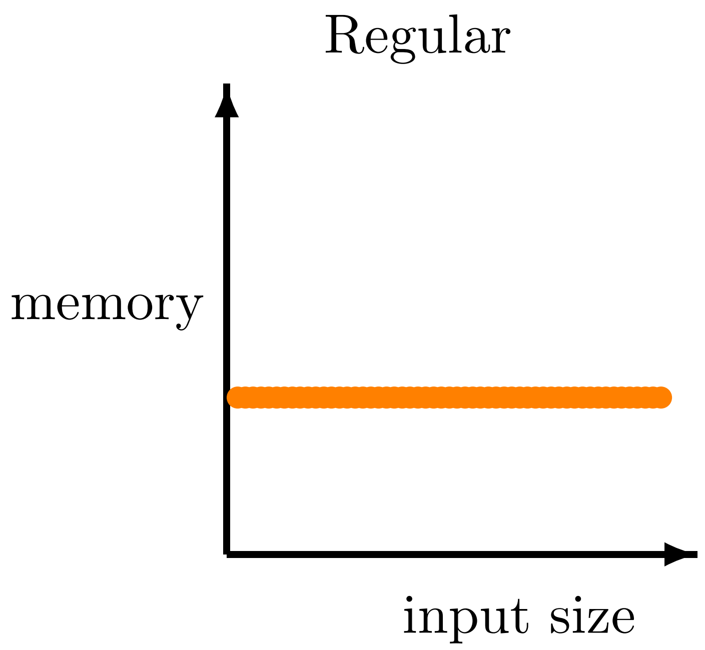
   - regular sets and functions (Kleene 1956, Elgot and Mezei 1956, Scott and Rabin 1959)
   - output-driven maps (Tesar 2014)
   - strict locality (Rogers and Pullum 2011, Chandlee 2014)

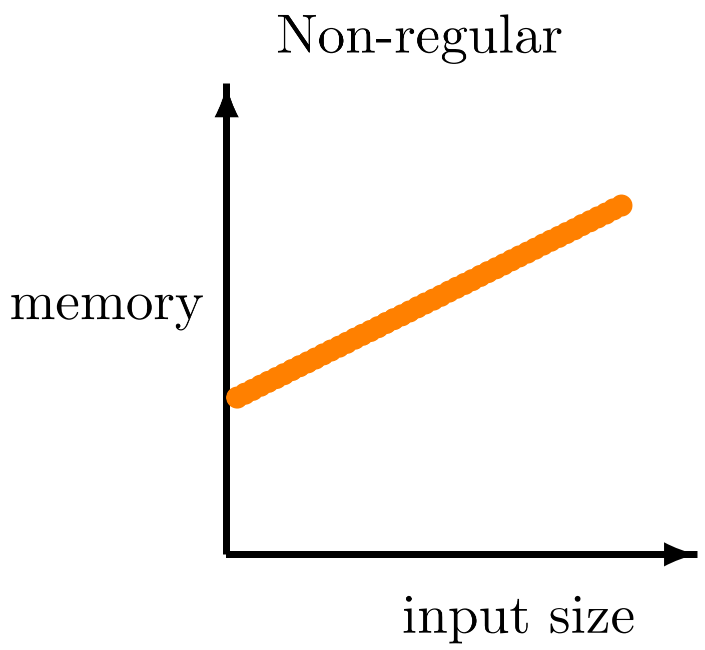
Part II

Phonological Generalizations are Regular

What “Regular” means

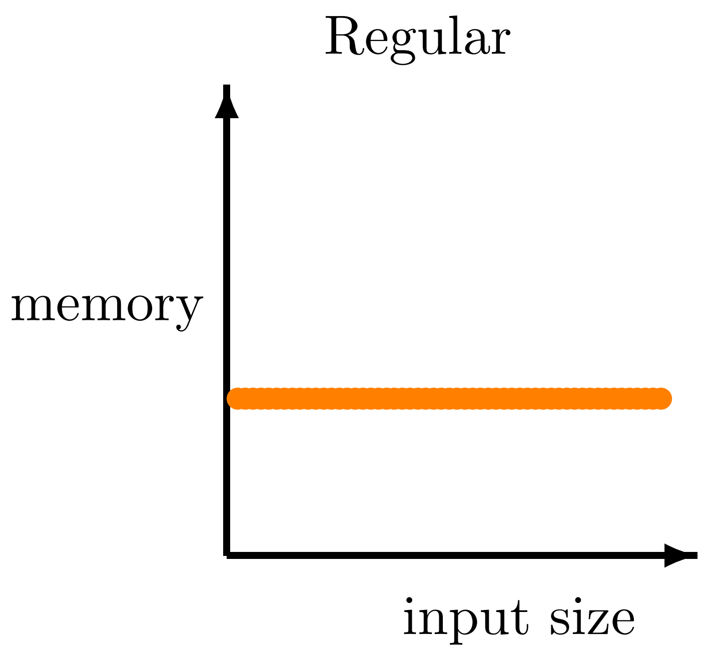
A set or function is regular provided **the memory required for the computation is bounded by a constant, *regardless of the size of the input.***

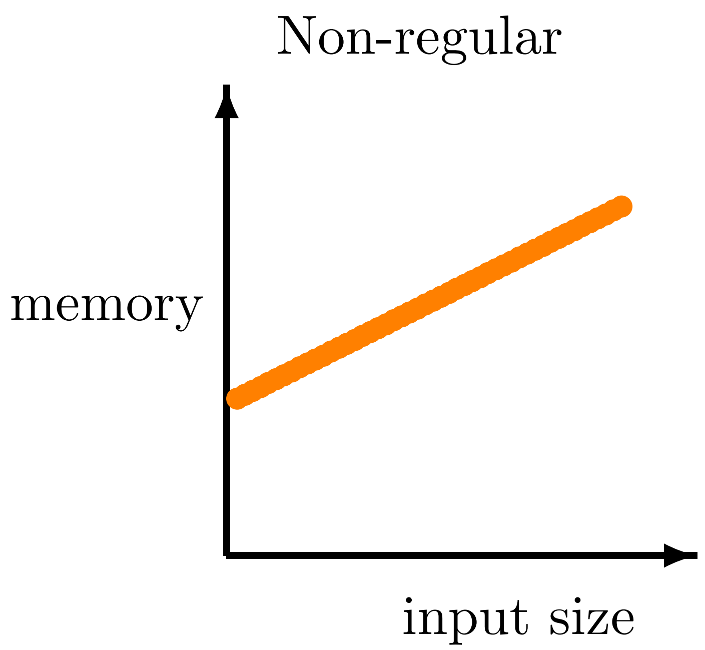




Some computations important to grammar

- For given constraint C and any representation w:
  - Does w violate C? How many times?
- For given grammar G and any underlying representation w:
  - What surface representation(s) does G transform w to? With what probabilities?





Regular grammars for sets and transformations

1. Regular expressions
2. Finite-state automata
3. Logic with models of strings

Example: Vowel Harmony

Progressive

Vowels agree in backness with the first vowel in the underlying representation.

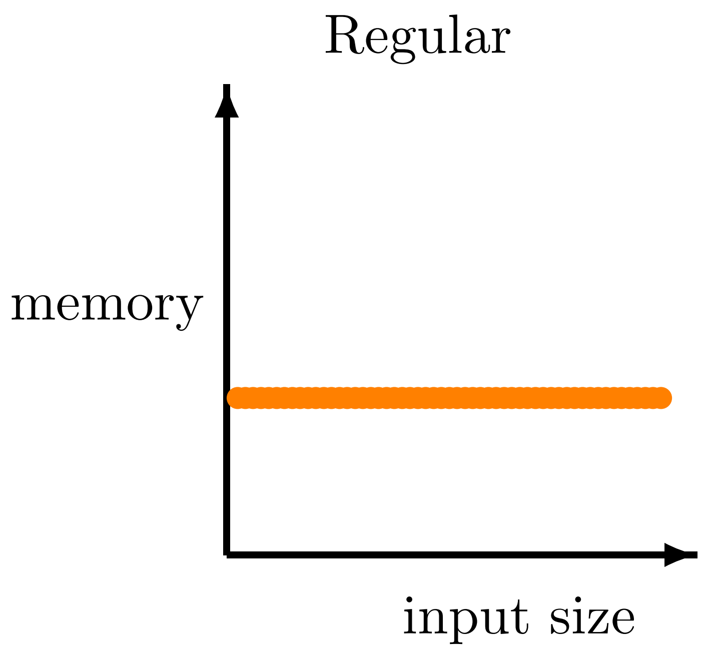
Majority Rules

Vowels agree in backness with the majority of vowels in the underlying representation.

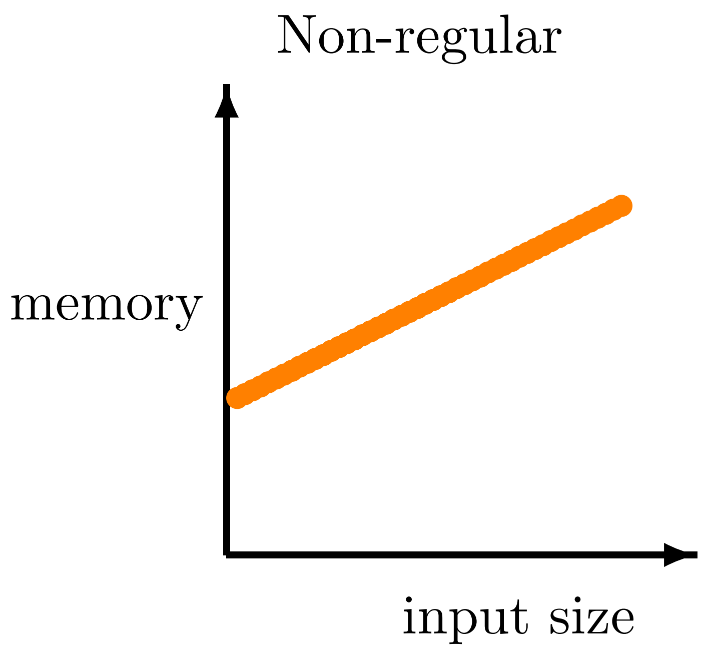
UR	Progressive	Majority Rules
/nokelu/	nokolu	nokolu
/nokeli/	nokolu	nikeli
/pidugo/	pidige	pudugo
/pidugomemi/	pidigememi	pidigememi

(Bakovic 2000, Finley 2008, 2011, Heinz and Lai 2013)

Progressive and Majority Rules Harmony



Progressive



Majority Rules

Some Perspective

Typological: Majority Rules is unattested. (Bakovic 2000)

Psychological: Human subjects fail to learn Majority Rules in artificial grammar learning experiments, unlike progressive harmony. (Finley 2008, 2011)

Computational: Majority Rules is not regular. (Riggle 2004, Heinz and Lai 2013)

Optimality Theory

1. There exists a CON and ranking over it which generates Majority Rules: AGREE(BACK)>>IDENTIO[BACK].
2. Changing CON may resolve this, but this solution misses the forest for the trees.

Phonological generalizations are regular

Evidence supporting the hypothesis that phonological generalizations are finite-state originate with Johnson (1972) and Kaplan and Kay (1994), who showed how to translate any phonological grammar defined by an ordered sequence of SPE-style rewrite rules into a finite-state automaton.

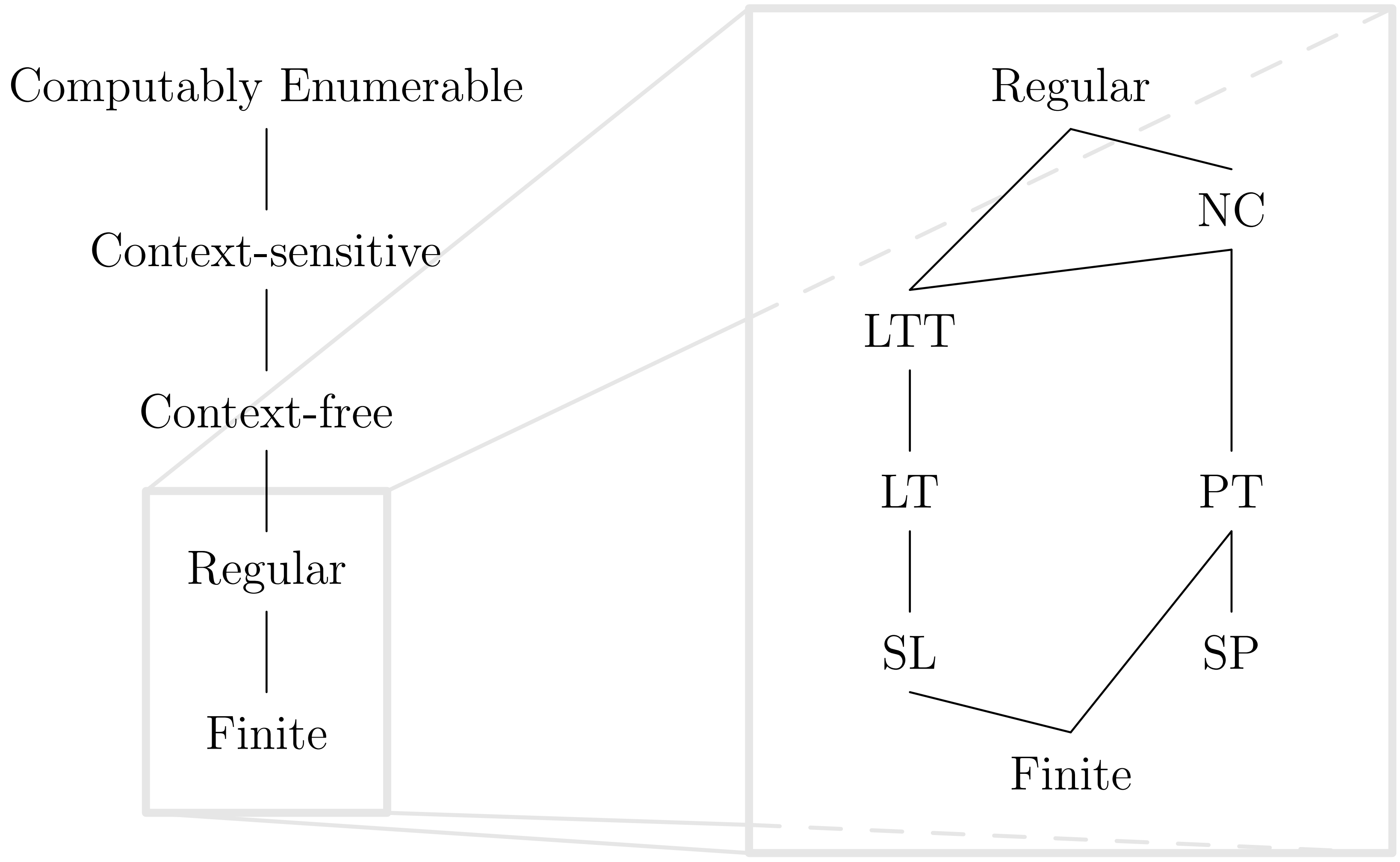
Consequently:

1. Constraints on well-formed surface and underlying representations are regular (since the image and pre-image of finite-state functions are finite-state). (Rabin and Scott 1959)
2. Since virtually any phonological grammar can be expressed as an ordered sequence of SPE-style rewrite rules, this means "being regular" is a property of the functions that *any* phonological grammar defines.

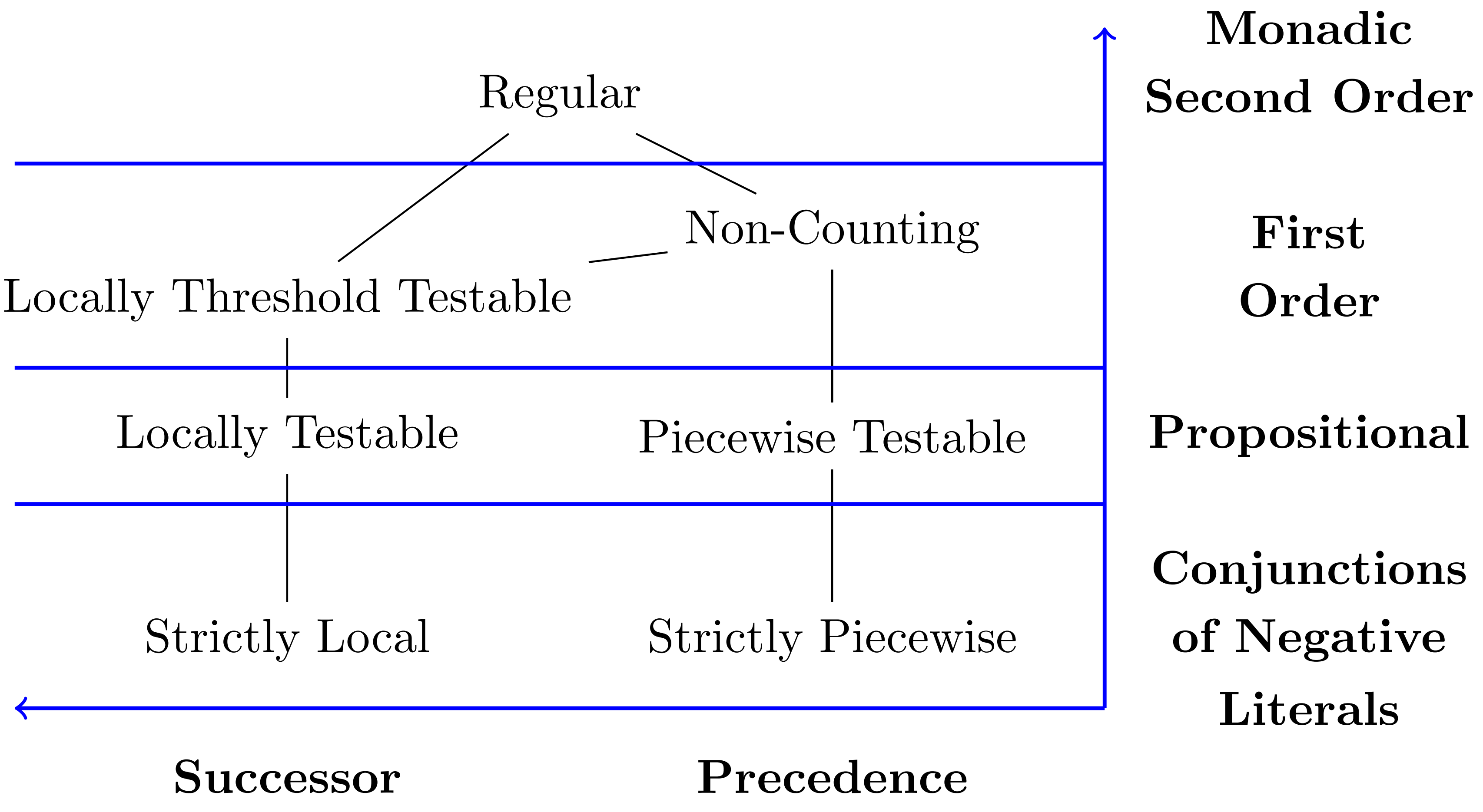
Part III

Phonological Constraints are “less than” Regular

The Chomsky Hierarchy

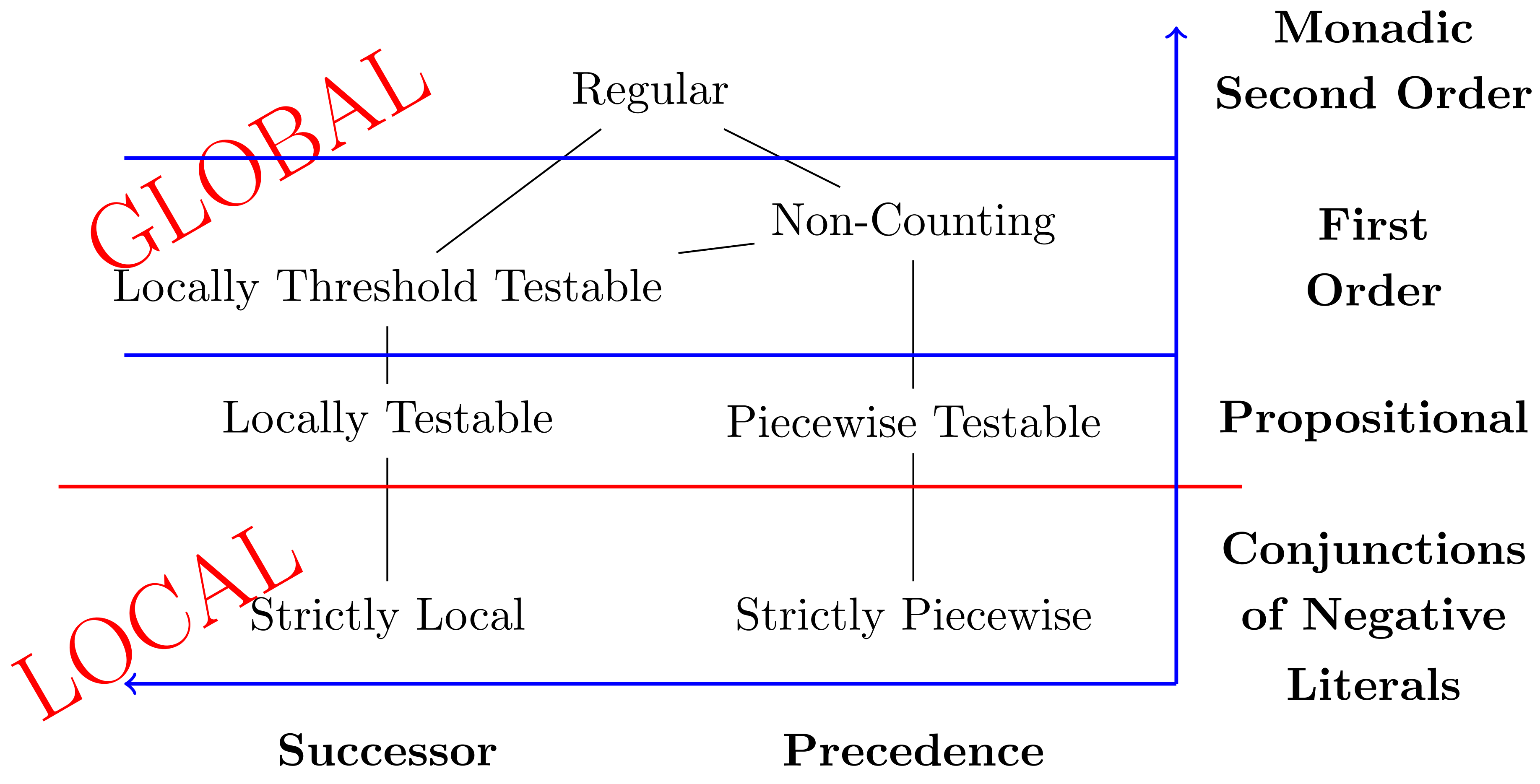


Subregular Hierarchies of Stringsets



(McNaughton and Papert 1971, Heinz 2010, Rogers and Pullum 2011, Rogers et al. 2013)

Subregular Hierarchies of Stringsets



(McNaughton and Papert 1971, Heinz 2010, Rogers and Pullum 2011, Rogers et al. 2013)

Representing words with successor

hypothetical [sriʃ]

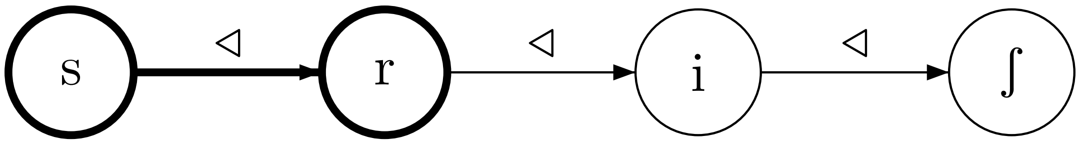
$$\begin{tikzcd} \text{s} \arrow[r, "\triangleleft"] & \text{r} \arrow[r, "\triangleleft"] & \text{i} \arrow[r, "\triangleleft"] & \text{ʃ} \end{tikzcd}$$

- The information about order is given by the successor ($\triangleleft$) relation.

Sub-structures

When words are represented with successor, **sub-structures are sub-strings** of a certain size.

- So [sr] is a sub-structure of [sriʃ]



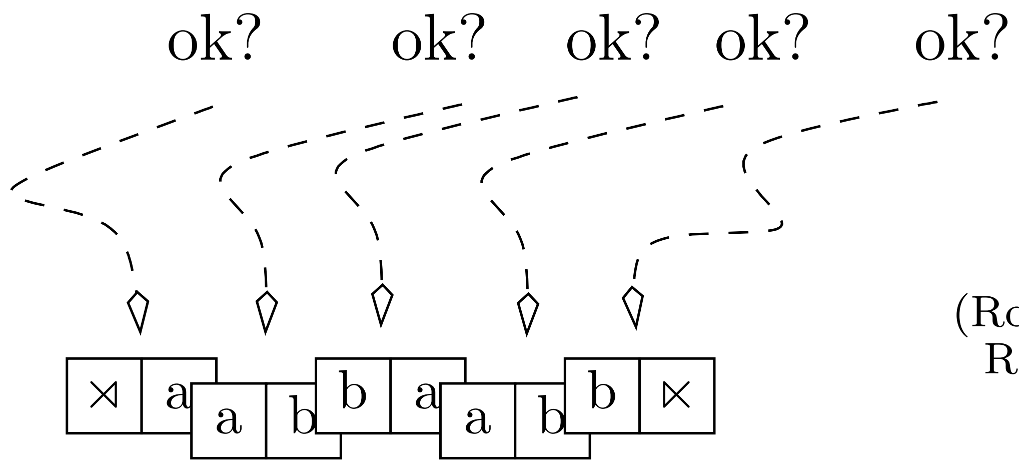
Strictly Local constraints for strings

When words are represented with successor, **sub-structures are sub-strings** of a certain size.

- **Strictly Local** constraints are ones describable with a finite list of *forbidden substrings*.

$$\neg s_1 \wedge \neg s_2 \ldots \wedge \neg s_n \quad (\triangleleft)$$

- For string $\rtimes abab\ltimes$, if we fix a diameter of 2, we have to check these substrings.

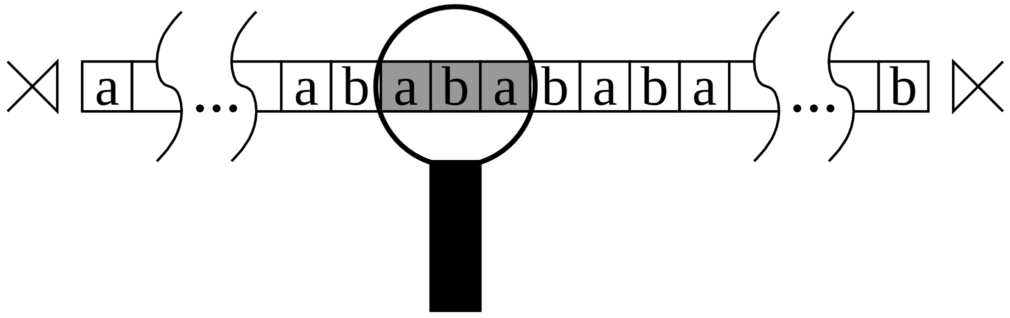


(Rogers and Pullum 2011, Rogers et al. 2013)

Strictly Local constraints for strings

When words are represented with successor, **sub-structures are sub-strings** of a certain size.

- We can imagine examining each of the substructures, checking to see if it is forbidden or not.
- The whole structure is well-formed only if each sub-structure is.

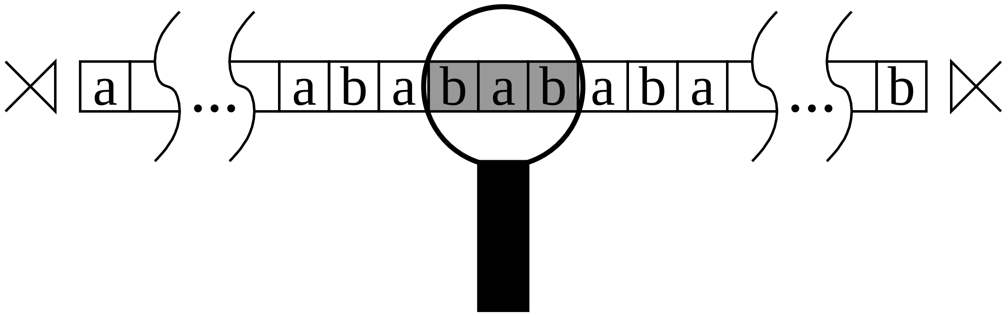


(Rogers and Pullum 2011, Rogers et al. 2013)

Strictly Local constraints for strings

When words are represented with successor, **sub-structures are sub-strings** of a certain size.

- We can imagine examining each of the substructures, checking to see if it is forbidden or not.
- The whole structure is well-formed only if each sub-structure is.

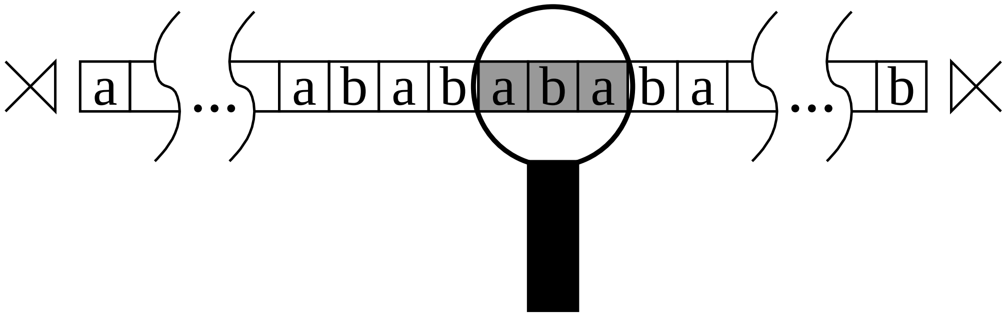


(Rogers and Pullum 2011, Rogers et al. 2013)

Strictly Local constraints for strings

When words are represented with successor, **sub-structures are sub-strings** of a certain size.

- We can imagine examining each of the substructures, checking to see if it is forbidden or not.
- The whole structure is well-formed only if each sub-structure is.



(Rogers and Pullum 2011, Rogers et al. 2013)

Examples of Strictly Local constraints

- *ab
- *NT

Example of Non-Strictly Local constraints

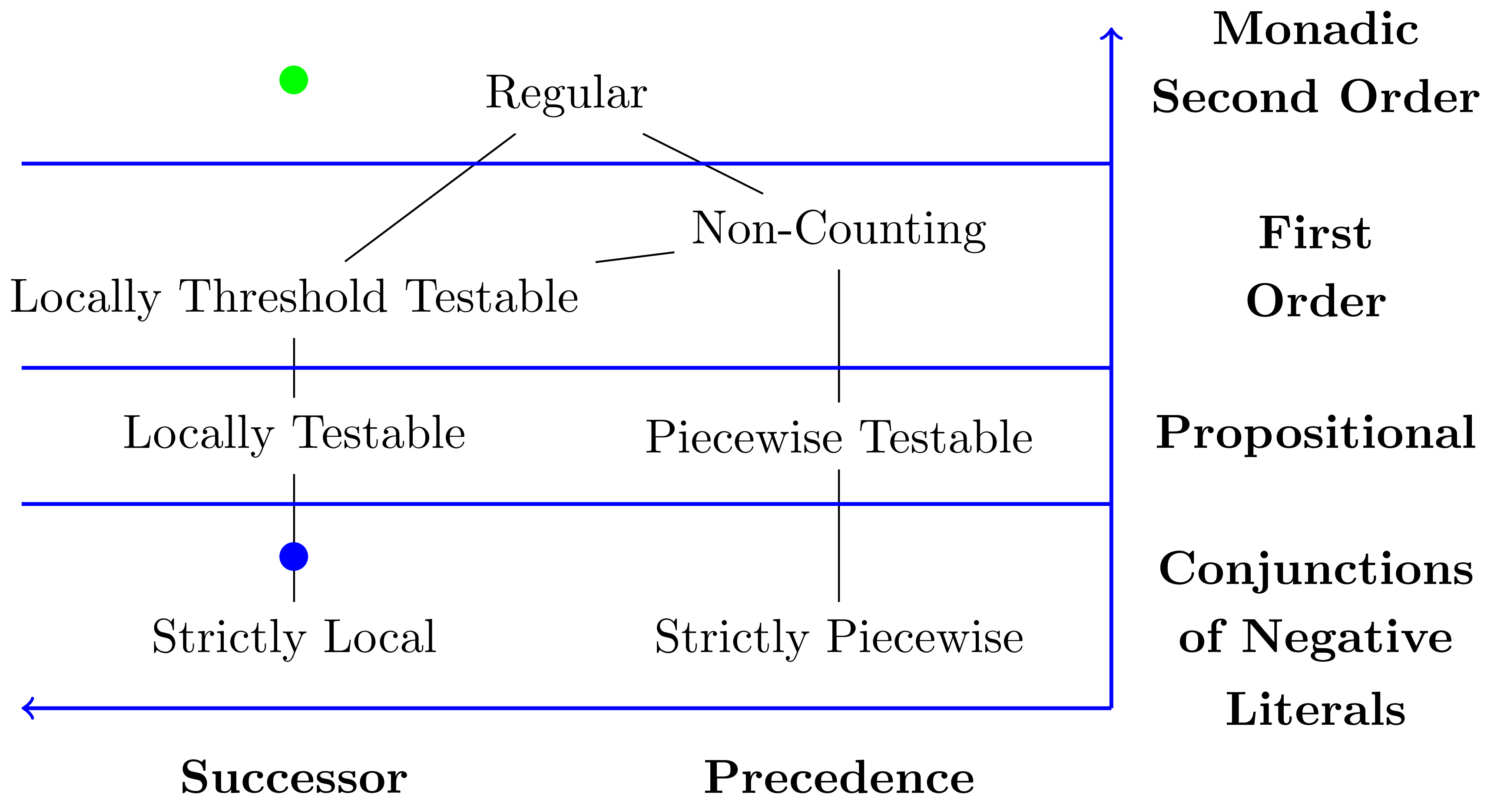
- *EVEN-Nasal
- *3-NT (so 2 NT structures OK, but not 3)
- *s. . . ʃ (Hansson 2001, Rose and Walker 2004, Hansson 2010, inter alia)

Sarcee (Cook 1978, 1984)

a.	/si-tʃiz-aʔ/	→	**ʃ**í**tʃ**íd**z**àʔ	'my duck'
			***s**í**tʃ**íd**z**àʔ	
b.	/na-s-ɣatʃ/	→	nā**ʃ**ɣát**ʃ**	'I killed them again'

- In Sarcee words, [−anterior] sibilants like [ʃ] may not follow [+anterior] sibilants like [s]. This constraint is called *s. . . ʃ.

Subregular Hierarchies of Stringsets



● *ab

● *s. . . ʃ

(Heinz 2010, Rogers et al. 2010)

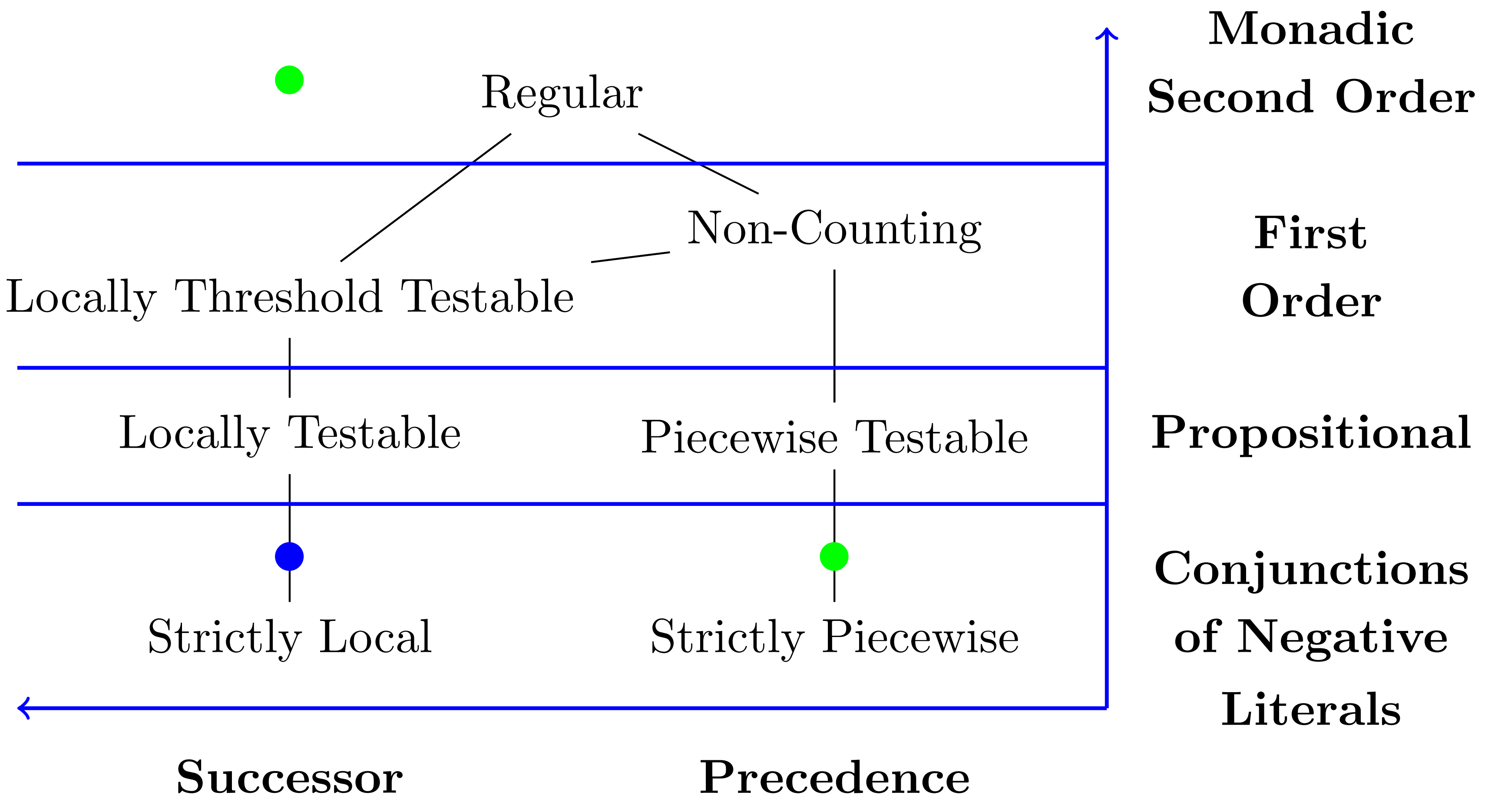
The “MSO with successor” Theory

Is this a good theory of possible constraints in phonology?

NO! Because. . .

1. Typologically, it overgenerates.
   (a) *EVEN-Nasal
   (b) *3-NT (so 2 NT structures OK, but not 3)
2. There are no feasible algorithms for learning the whole class of regular stringsets (Gold 1967, inter alia).

Subregular Hierarchies of Stringsets



● *ab

● *s. . . ʃ

(Heinz 2010, Rogers et al. 2010)

Representing Order in Sequences

hypothetical [sriʃ]

1. Successor

$$\begin{tikzcd} s \arrow[r, "\triangleleft"] & r \arrow[r, "\triangleleft"] & i \arrow[r, "\triangleleft"] & ʃ \end{tikzcd}$$

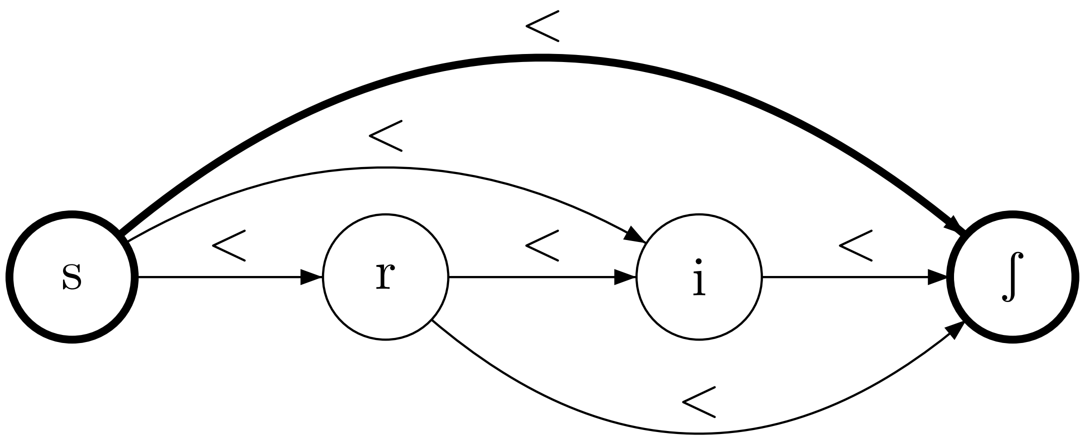
2. Precedence

$$\begin{tikzcd} s \arrow[r, "<"] \arrow[rr, bend left, "<"] \arrow[rrr, bend left=50, "<"] & r \arrow[r, "<"] \arrow[rr, bend right, "<"'] & i \arrow[r, "<"] & ʃ \end{tikzcd}$$

Strictly Piecewise Constraints

When words are represented with precedence, **sub-structures are sub-sequences** of a certain size.

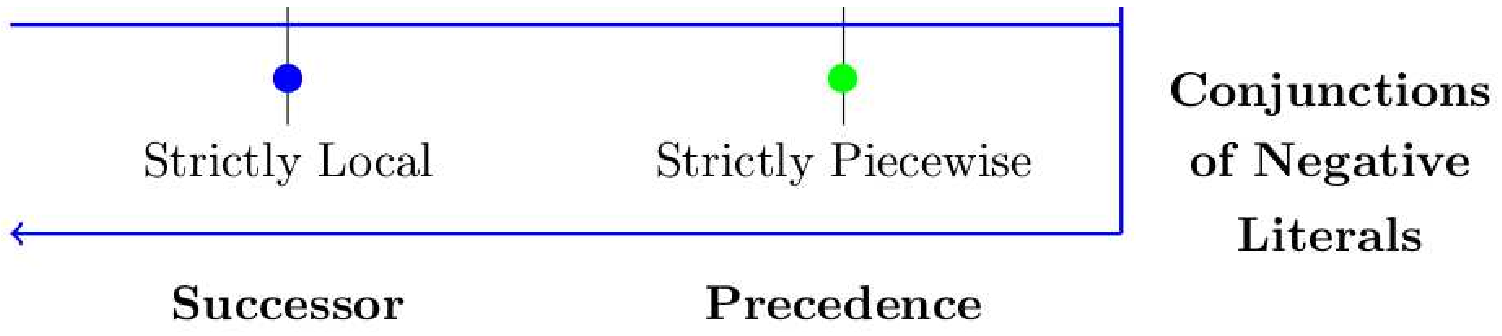
- So $\boxed{\text{s} < \text{ʃ}}$ is a sub-structure of [sriʃ].



- Strictly Piecewise constraints are ones describable with a finite list of *forbidden sub-structures* (with words represented using the precedence relation).

$$\neg s_1 \wedge \neg s_2 \ldots \wedge \neg s_n \quad (<)$$

The CNL with Successor and Precedence Theory



● *ab

● *ʃ. . . s

Is this a better theory of possible constraints in phonology?

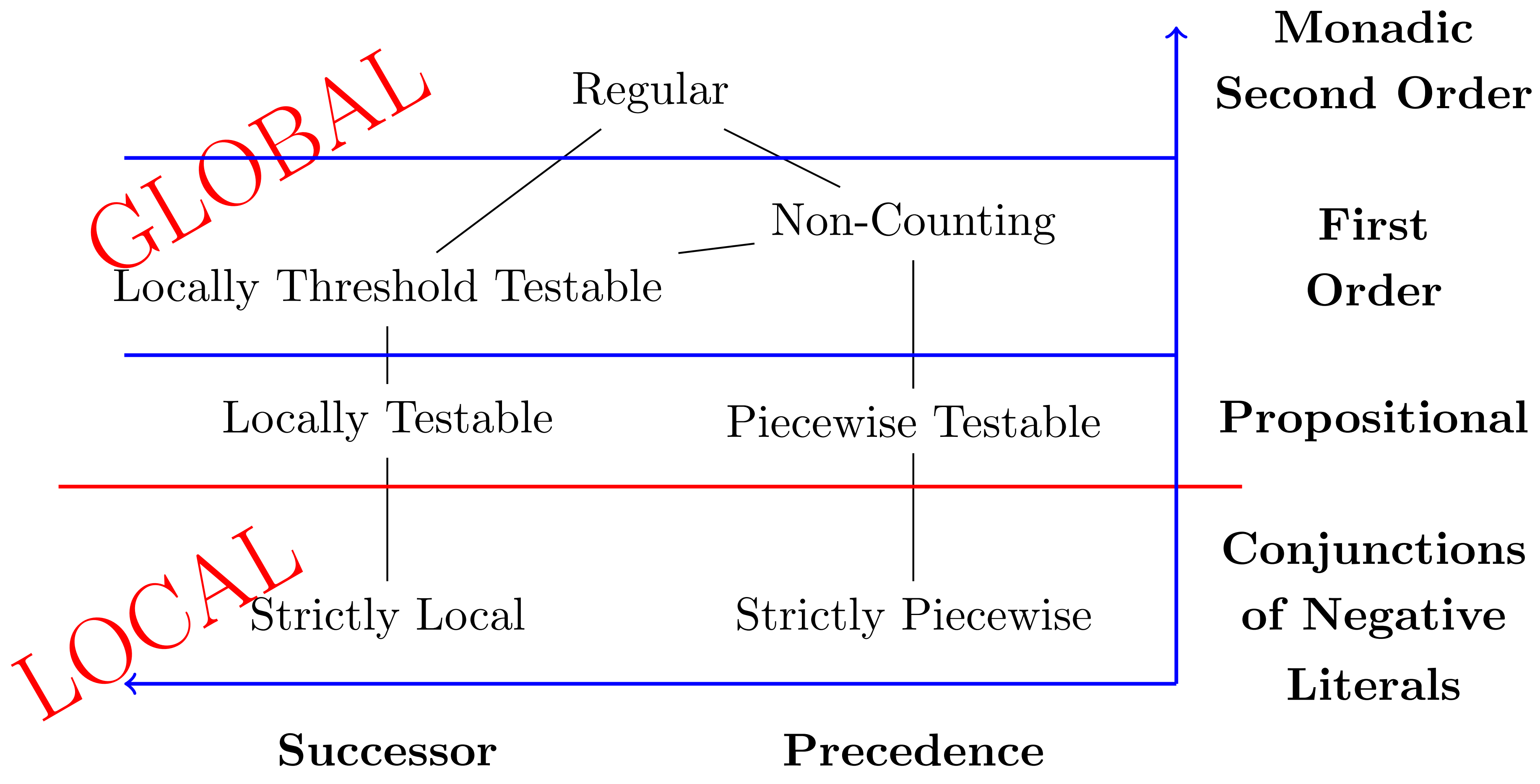
1. Typologically, it is better than “MSO with successor.”
   (a) Admits phonotactic constraints which arguably drive long-distance harmony patterns
   (b) Provably excludes constraints like *EVEN-Sibilants and “*3-NT”.
2. Both Strictly Local and Strictly Piecewise constraints are feasibly learnable (Garcia et al. 1991, Heinz 2010)
3. Human subjects learn SP patterns—but not similar LT ones—in lab experiments (Lai 2015).

Morals of this Story

1. Precedence is the transitive closure of successor.
2. Providing the power of transitive closure (MSO-definabilty) yields power to do lots of other things (so expands the typology undesirably)
3. Putting precedence directly into the representation allows a restricted expansion of the typology in a more desirable way.
4. The restriction also brings learnability benefits.

The subregular hierarchies demonstrate a firm mathematical foundation upon which the interplay between representation and computation in linguistic theory can be studied.

Subregular Hierarchies of Stringsets



(McNaughton and Papert 1971, Heinz 2010, Rogers and Pullum 2011, Rogers et al. 2013)

Autosegmental Representations

1. Jardine (2016, 2017) shows these ideas to autosegmental representations (ASRs), where the sub-structures are now sub-graphs of the autosegmental structure.
2. He argues this approach captures the typology better than Zoll's 2003 approach in OT and earlier derivational approaches.
3. He shows that his grammars can be learned from strings (not ASRs!) because ASRs are fundamentally stringlike (Jardine and Heinz 2015).

Some well-studied patterns of tonal association

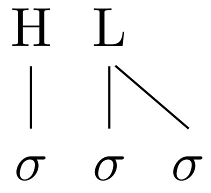
1. Position-specific contours (Mende, Hausa, Northern Karanga)
2. Position-specific plateaus (Mende, Hausa, Northern Karanga)
3. Melody constraints (Mende)
4. Quality-dependent plateaus (Kukuya)

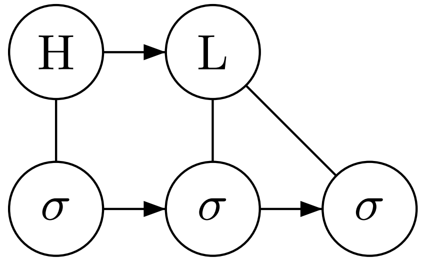
Tone and Autosegmental Representations (Jardine 2016, 2017)

- Autosegmental representations are **graphs** (Goldsmith 1976, Coleman and Local 1991)

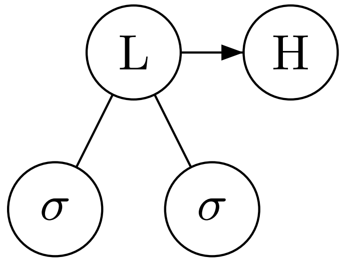
félàmà HLL 'junction'

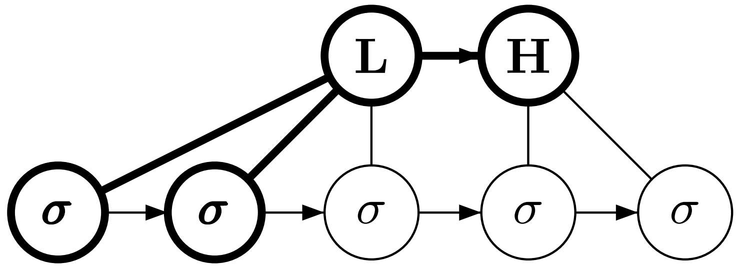
(Mende)



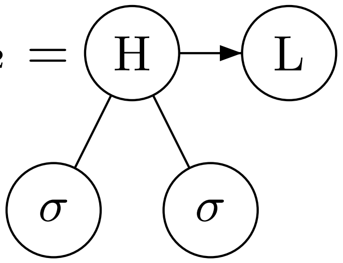


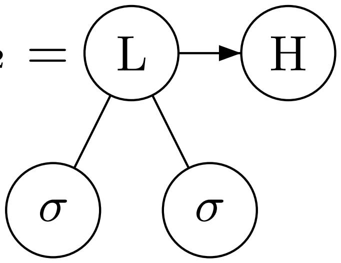
- A **sub-structure** is a finite, connected piece of a graph.

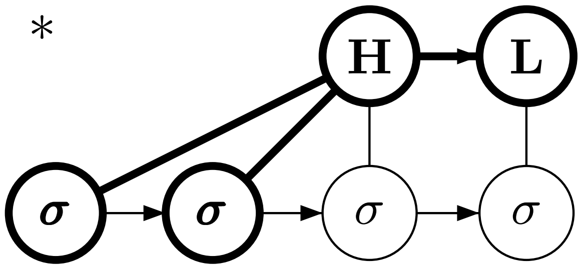


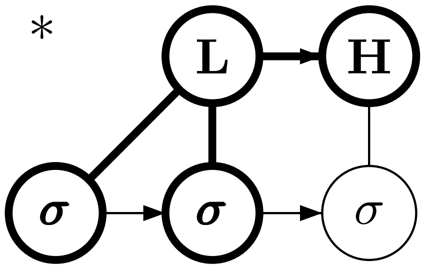


Case study: Mende — Plateaus

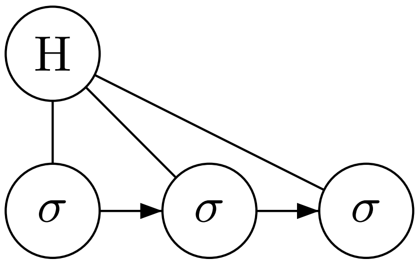
$\phi_{NF\text{-}H^2} =$ 

$\phi_{NF\text{-}L^2} =$ 

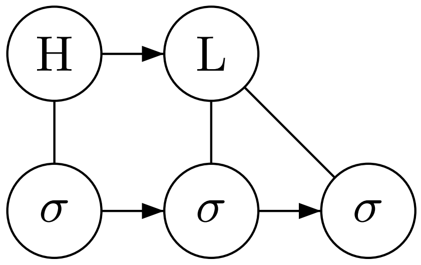
* 

* 

háwámá
HHH
‘waist’



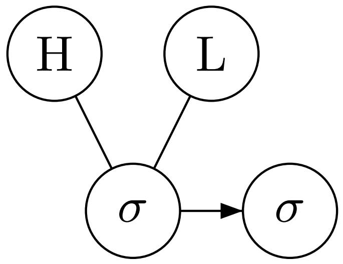
félàmà
HLL
‘junction’

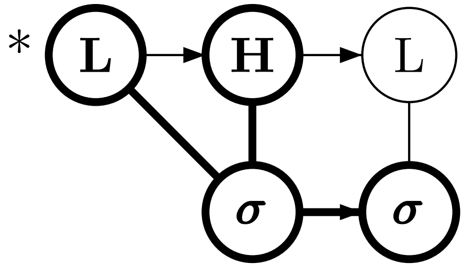


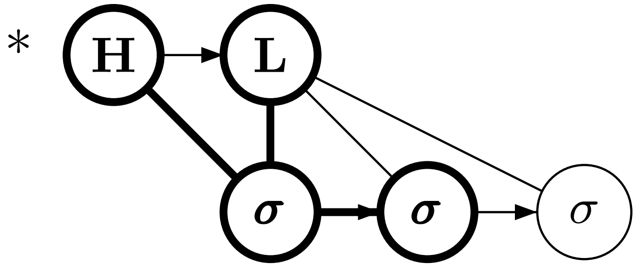
- Kukuya will use $\phi_{NF\text{-}H^2}$ but not $\phi_{NF\text{-}L^2}$

(Jardine 2016, 2017)

Case study: Mende — Contours

$\phi_{NF\text{-}Cont} =$ 





- c.f. (Zhang 2000)

(Jardine 2016, 2017)

Case study: Mende — Melody Constraint

$$\phi_{HLH} = \begin{tikzcd} \text{H} \arrow[r] & \text{L} \arrow[r] & \text{H} \end{tikzcd}$$

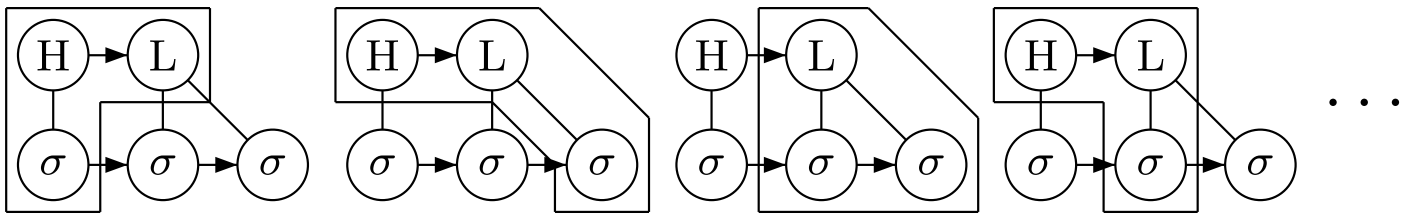
$$\begin{tikzcd} {*}\ \text{L} \arrow[r] \arrow[d, dash] & \mathbf{H} \arrow[r] \arrow[d, dash] & \mathbf{L} \arrow[r] \arrow[d, dash] & \mathbf{H} \arrow[d, dash] \\ \sigma \arrow[r] & \sigma \arrow[r] & \sigma \arrow[r] & \sigma \end{tikzcd}$$

(Jardine 2016, 2017)

Case study: Mende — Summary

$$\neg\phi_{HLH} \wedge \neg\phi_{NF\text{-}Cont} \wedge \neg\phi_{NF\text{-}H^2} \wedge \neg\phi_{NF\text{-}L^2}$$

Evaluation procedure now 'crawls' through graph



(Jardine 2016, 2017)

Part IV

Phonological Transformations are “less than” Regular

Input Strictly Local Transformations

1. Chandlee (2014 et seq.) extends these ideas to segmental transformations.
2. She argues this approach better captures the typology than OT and earlier derivational approaches.
3. She establishes theoretical, efficient learning results from (UR,SR) pairs.

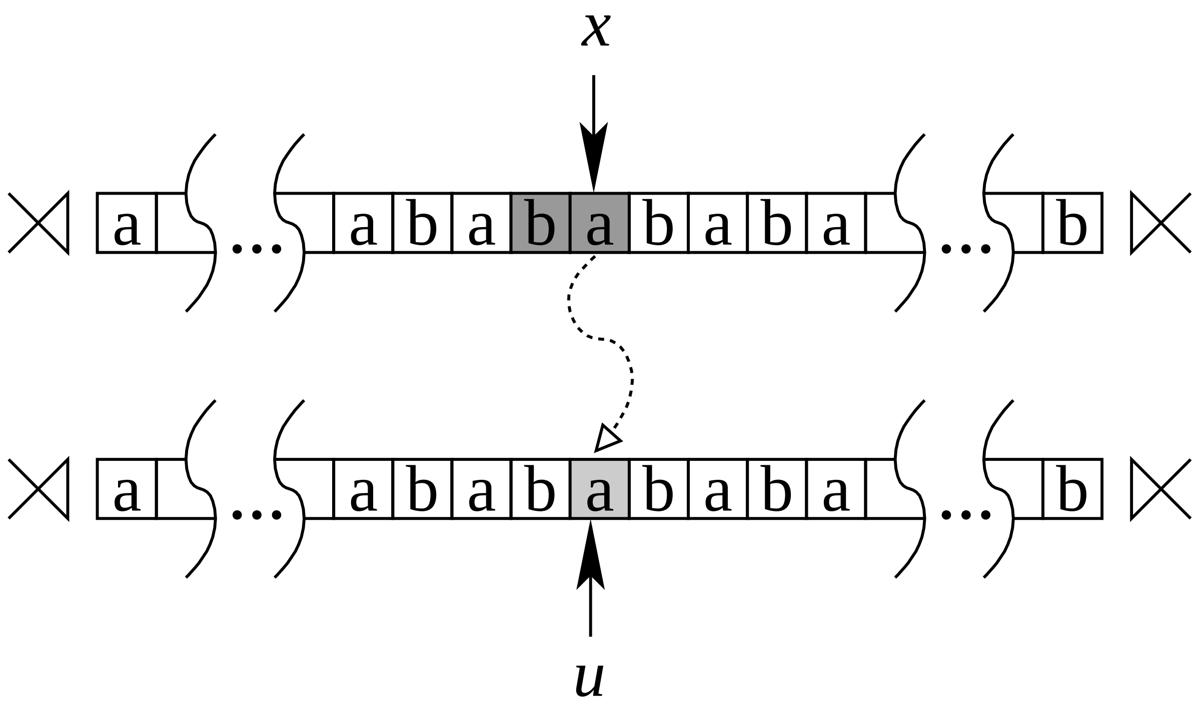
ISL defined

$$\boxed{x_0\ x_1\ \ldots\ x_n \mapsto u_0\ u_1\ \ldots\ u_n}$$

where

1. Each x_i is a single symbol and each u_i is a *string*.
2. There exists a $k \in \mathbb{N}$ such that for all input symbols x_i its output string u_i depends only on x_i and the $k-1$ elements immediately preceding x_i.

Input Strict Locality: Main Idea in a Picture



x
⋊ a ... a b a b a b a b a ... b ⋉
⋊ a ... a b a b a b a b a ... b ⋉
u

Figure 1: For every Input Strictly 2-Local function, the output string u of each input element x depends only on x and the input element previous to x. In other words, the contents of the lightly shaded cell only depends on the contents of the darkly shaded cells.

Example: Word-Final /e/ Raising is ISL with $k = 2$

$$/\text{ove}/ \mapsto [\text{ovi}]$$

input:	$\rtimes$	o	v	e	$\ltimes$
output:	$\rtimes$	o	v	λ	i $\ltimes$

Example: Word-Final /e/ Raising is ISL with $k = 2$

$$/\text{ove}/ \mapsto [\text{ovi}]$$

input:	$\rtimes$	o	v	e	$\ltimes$
output:	$\rtimes$	o	v	λ	i $\ltimes$

Example: Word-Final /e/ Raising is ISL with $k = 2$

$$/\text{ove}/ \mapsto [\text{ovi}]$$

input:	$\rtimes$	o	v	e	$\ltimes$
output:	$\rtimes$	o	v	λ	i $\ltimes$

Example: Word-Final /e/ Raising is ISL with $k = 2$

$$/\text{ove}/ \mapsto [\text{ovi}]$$

input:	$\rtimes$	o	v	e	$\ltimes$
output:	$\rtimes$	o	v	λ	i $\ltimes$

What can be modeled with ISL functions?

1. Many individual phonological processes.

   (local substitution, deletion, epenthesis, and synchronic metathesis)

   Theorem: Transformations describable with a rewrite rule R: A $\longrightarrow$ B / C __ D where

   - CAD is a finite set,
   - R applies simultaneously, and
   - contexts, but not targets, can overlap

   are ISL for k equal to the longest string in CAD.

(Chandlee 2014, Chandlee and Heinz 2018)

What can be modeled with ISL functions?

2. Approximately 95% of the individual processes in P-Base (v.1.95, Mielke (2008))
3. Many *opaque* transformations without any special modification.

(Chandlee 2014, Chandlee and Heinz 2018, Chandlee et al. to appear)

Opaque ISL transformations

- Opaque maps are typically defined as the extensions of particular rule-based grammars (Kiparsky 1971, McCarthy 2007). Tesar (2014) defines them as non-*output-driven*.
- Baković (2007) provides a typology of opaque maps.
  - Counterbleeding
  - Counterfeeding on environment
  - Counterfeeding on focus
  - Self-destructive feeding
  - Non-gratuitous feeding
  - Cross-derivational feeding
- Each of the examples in Baković's paper is ISL.

(Chandlee et al. to appear)

Example: Counterbleeding in Yokuts

	‘might fan’
	/ʔiliː+l/
[+long] $\rightarrow$ [-high]	ʔileːl
V $\longrightarrow$ [-long] / __ C#	ʔilel
	[ʔilel]

Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ ↦ [ʔiliːl] ‘might fan’

input:	⋊	ʔ	i	l	iː	l	⋉
output:	⋊	ʔ	i	l	λ	λ	el ⋉

Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ $\mapsto$ [ʔilel] ‘might fan’

input:	⋊	ʔ	i	l	iː	l	⋉
output:	⋊	ʔ	i	l	λ	λ	el ⋉

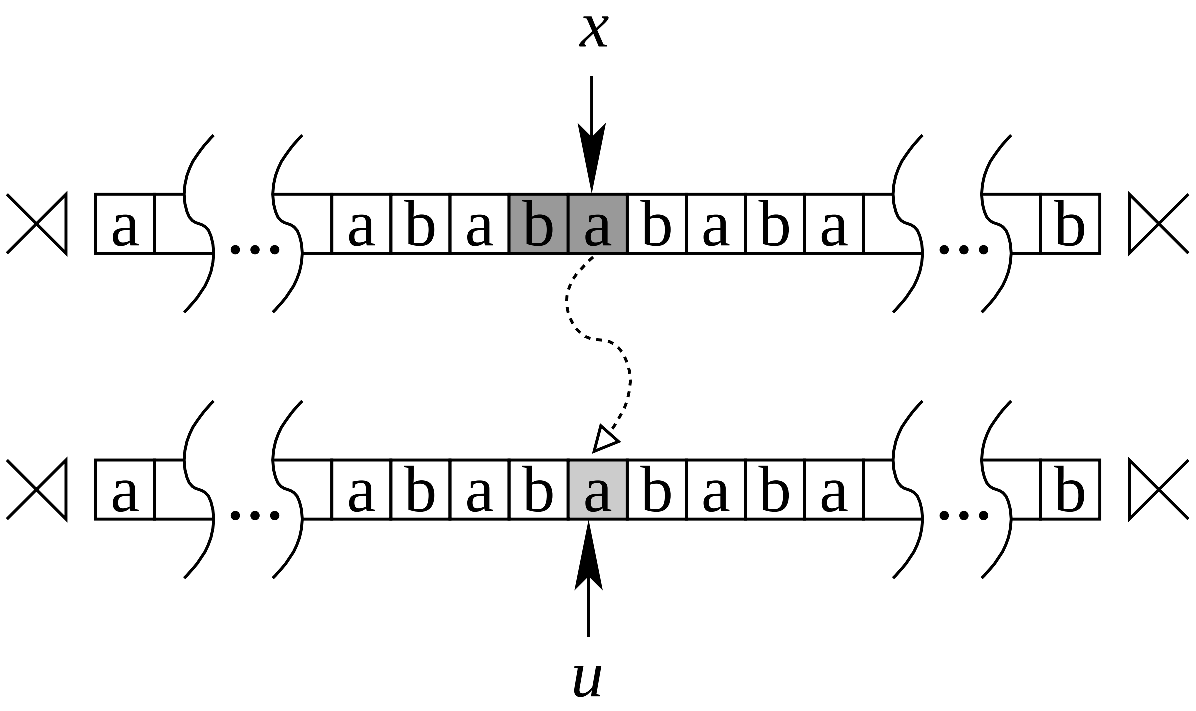
Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ ↦ [ʔilel] ‘might fan’

input:	⋊	ʔ	i	l	iː	l	⋉
output:	⋊	ʔ	i	l	λ	λ	el ⋉

Interim Summary

Many phonological patterns, including many opaque ones, have the *necessary information* to decide the output contained within a *window of bounded length* on the *input* side.



What CANNOT be modeled with ISL functions

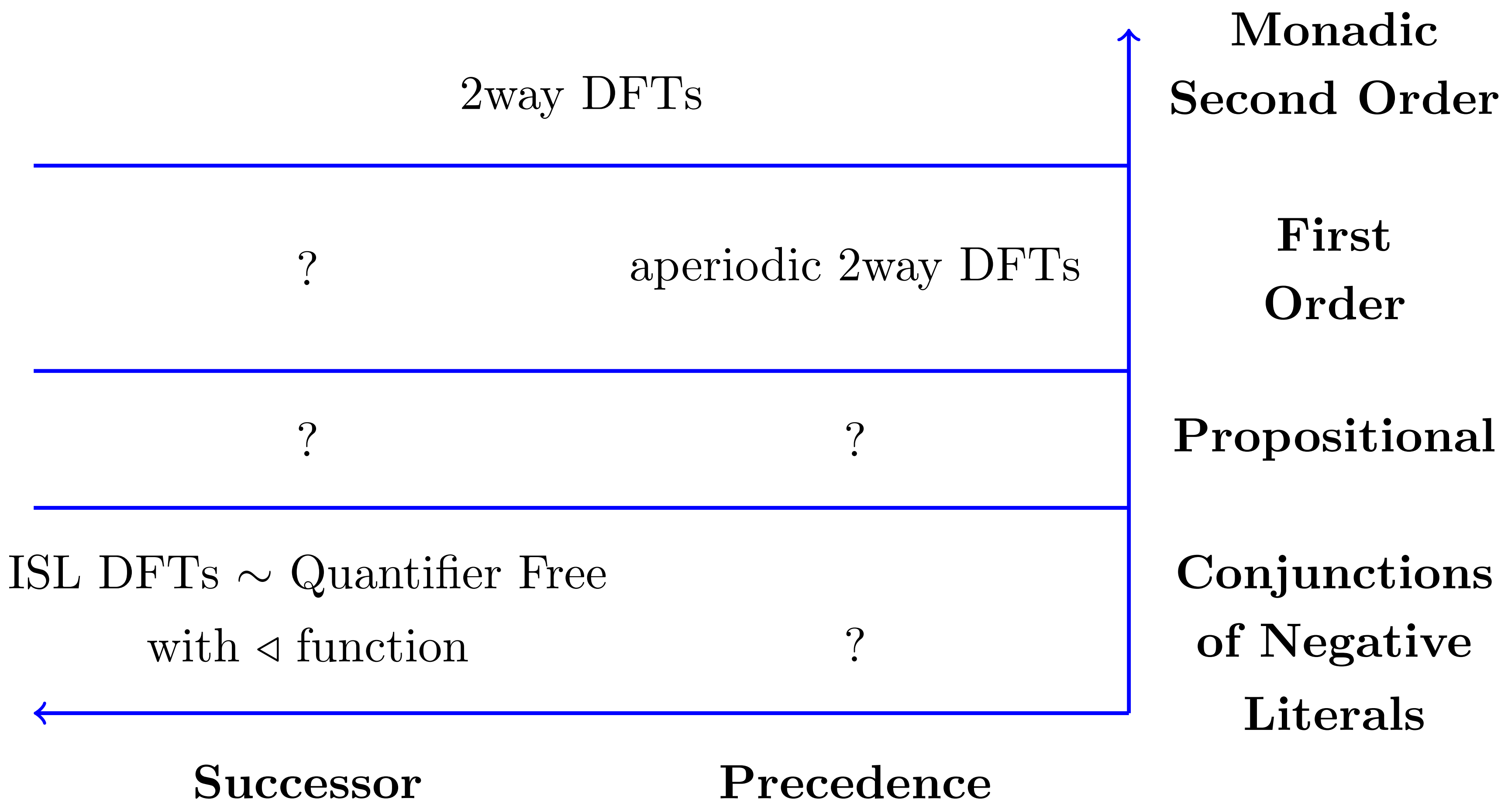
1. progressive and regressive spreading
   - But these are Output Strictly Local (Chandlee et al. 2015)!
2. long-distance (unbounded) consonant and vowel harmony
   - Stay tuned!

(Chandlee 2014, Chandlee and Heinz, 2018)

Learning Results in a nutshell

- Particular finite-state transducers can be used to represent ISL functions.
- Automata-inference techniques (de la Higuera 2010) are used to learn these transducers.
- **Theorems:** Given k and a sufficient sample of (u, s) pairs any k-ISL function can be *exactly* learned in *polynomial* time and data.
  - ISLFLA (Chandlee et al. 2014, TACL) (quadratic time and data)
  - SOSFIA (Jardine et al. 2014, ICGI) (linear time and data)

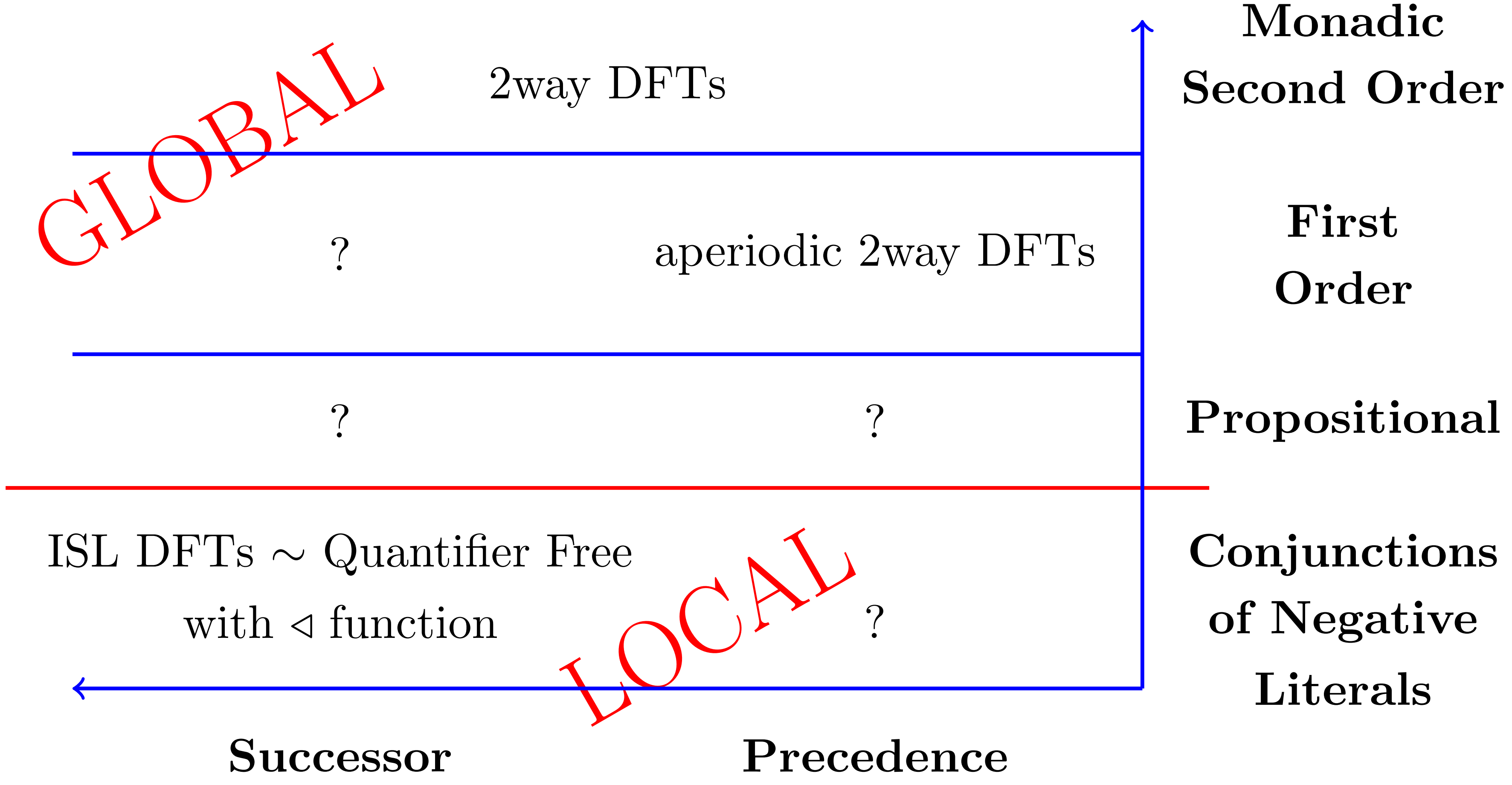
Logical Characterizations of Subregular *Functions* for *Transformations*



	Successor	Precedence
Monadic Second Order	2way DFTs	
First Order	?	aperiodic 2way DFTs
Propositional	?	?
Conjunctions of Negative Literals	ISL DFTs $\sim$ Quantifier Free with $\triangleleft$ function	?

(Chandlee and Lindell 2016, Filiot and Reynier 2016)

Logical Characterizations of Subregular *Functions* for *Transformations*



(Chandlee and Lindell 2016, Filiot and Reynier 2016)

Part V

Logical Characterizations of Transductions

Defining Transductions Logically

- Logical expressions can translate one relational structure into another.

$$P'(x) \quad \overset{\text{def}}{=} \quad Q(x) \tag{1}$$

"Position x has property P in the *output* only if corresponding position x in the *input* has property Q."

(Courcelle 1994, Courcelle and Engelfriedt 2001, 2011)

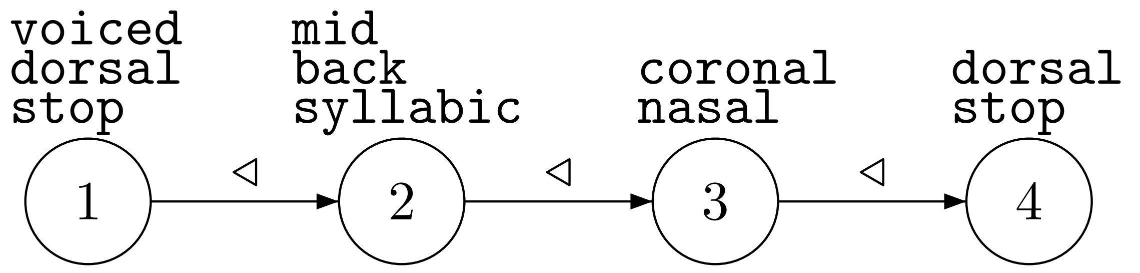
Defining "Post Nasal Voicing" Logically /gonk/ ↦ [gong]

$$\texttt{voiced'}(x) \overset{\text{def}}{=} \texttt{voiced}(x) \vee [\texttt{pred}(x) = y \wedge \texttt{nasal}(y)] \quad (2)$$

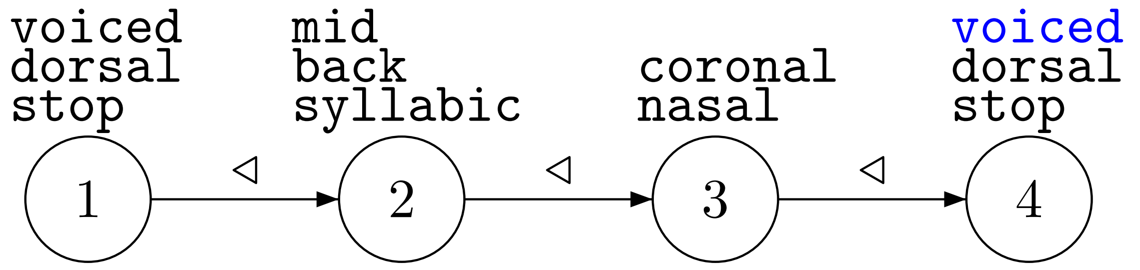
$$\texttt{feature'}(x) \overset{\text{def}}{=} \texttt{feature}(x) \text{ (for other features } \texttt{feature}) \quad (3)$$

$$\texttt{pred'}(x) \overset{\text{def}}{=} \texttt{pred}(x) \quad (4)$$

INPUT:



OUTPUT:



(Courcelle 1994, Courcelle and Engelfriedt 2001, 2011)

Quantifier Free (QF) transductions

Chandlee and Lindell show that ISL transductions from a logical perspective are QF.

Compare:

- $P'(x) \overset{\text{def}}{=} Q(x) \wedge (\exists y)[R(y)]$ (First Order Definable)
- $P'(x) \overset{\text{def}}{=} Q(x) \wedge R(\texttt{succ}(x))$ (QF Definable)

Transformation incorporating phonological representations

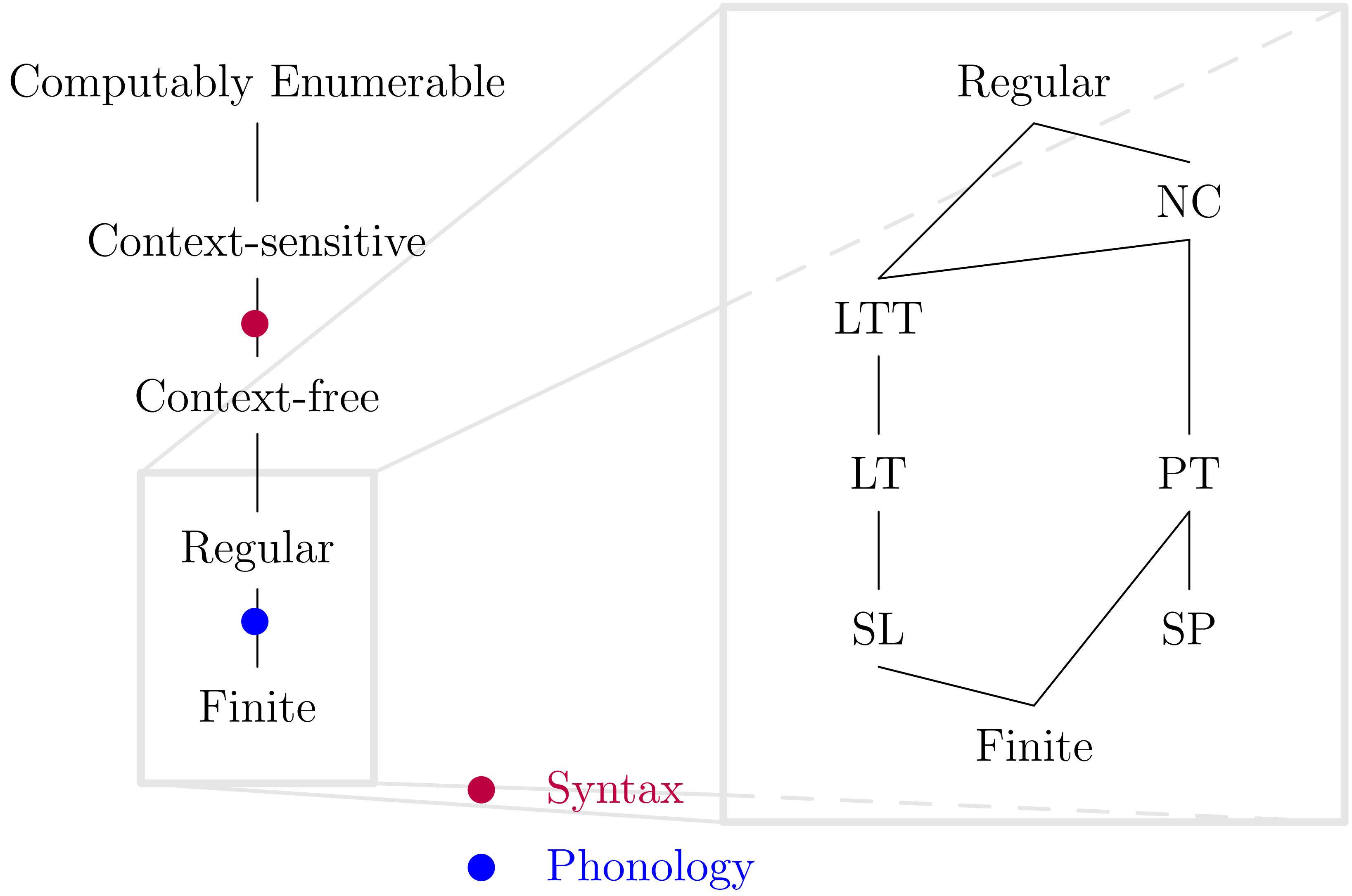
Strother-Garcia (to appear) shows

1. Translations between different syllabic representations is QF.
2. Syllabification in IT Berber is QF, with a “window size” of 3.

She concludes

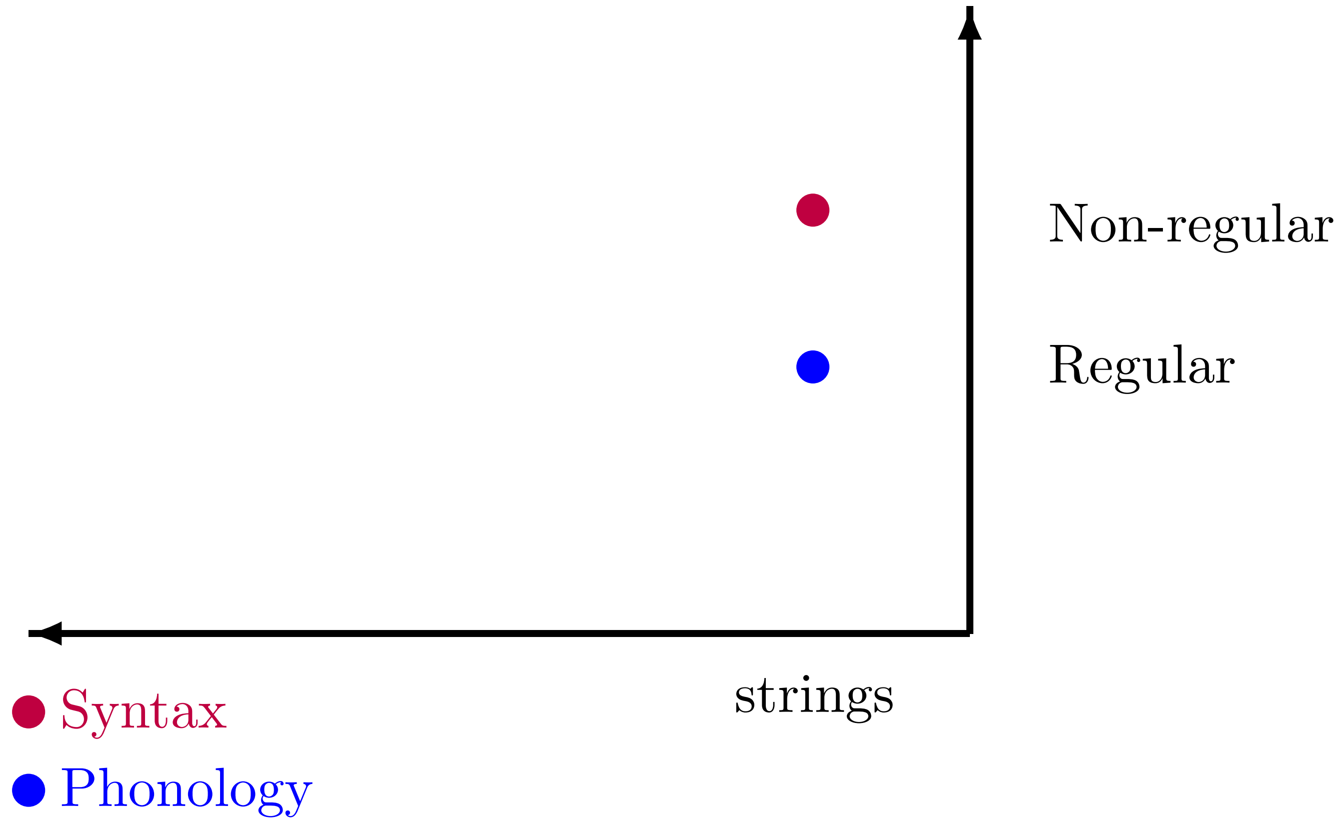
- “. . . syllabification in ITB can be represented by a QF graph transduction, a formalism restricted to substantially lower computational complexity than [traditional] phonological grammars. . . Establishing that ITB syllabification is QF highlights an insight not apparent from [those traditional] grammatical formalisms. . . ”

This matters for syntax too.



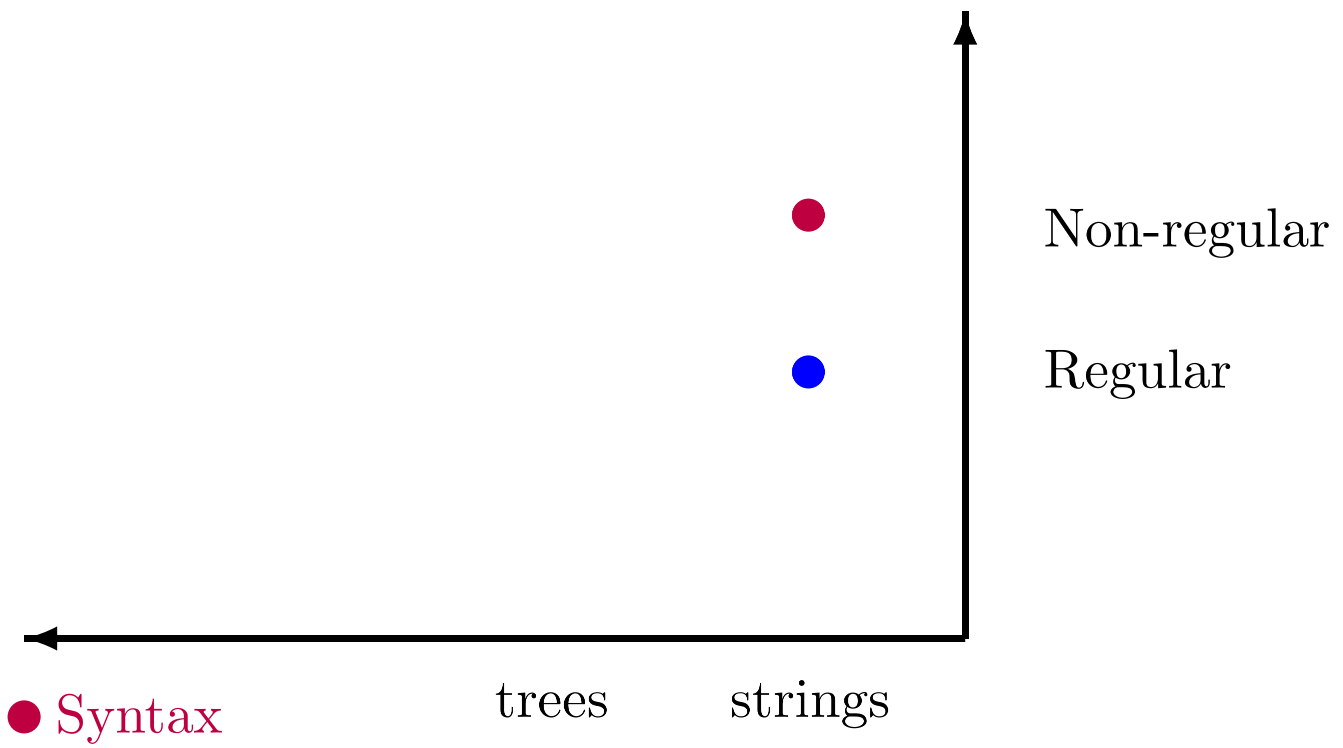
(Heinz and Idsardi 2011, 2013)

Syntax



(work with Thomas Graf)

Syntax

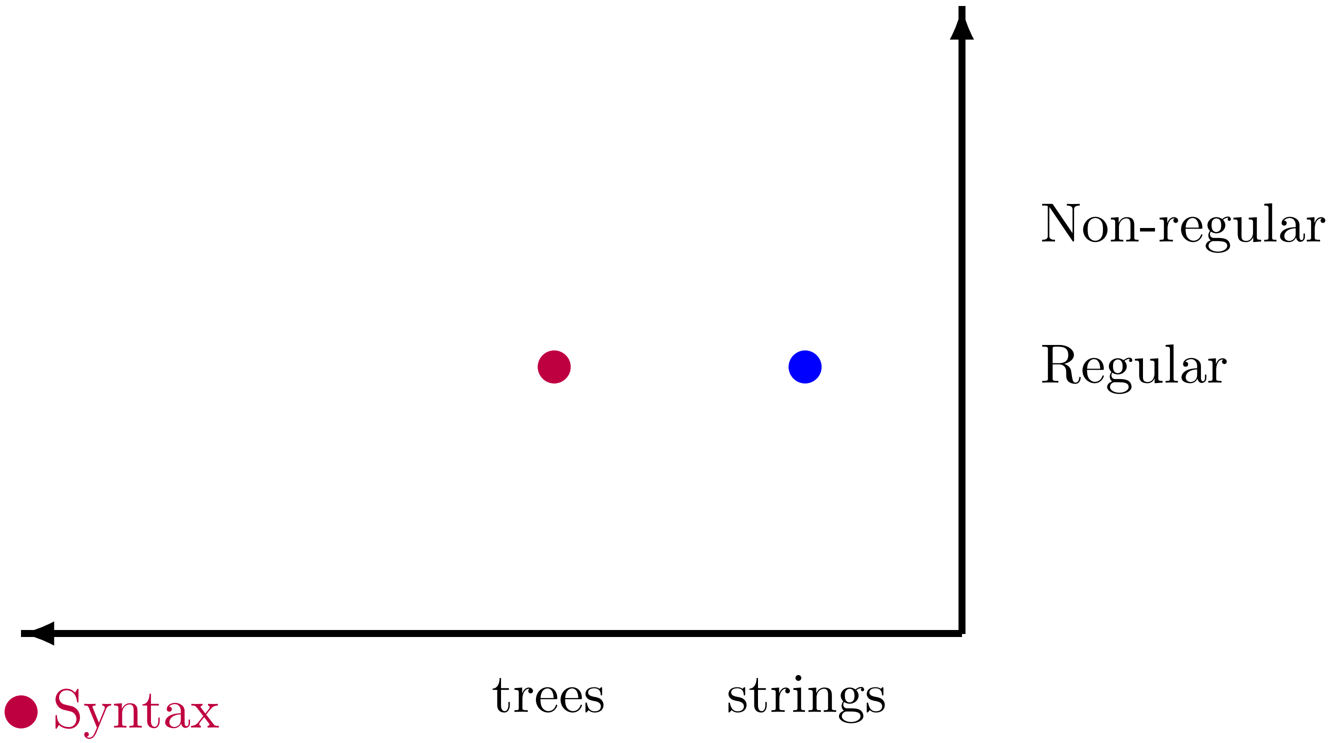


● Syntax

● Phonology

(work with Thomas Graf)

Syntax

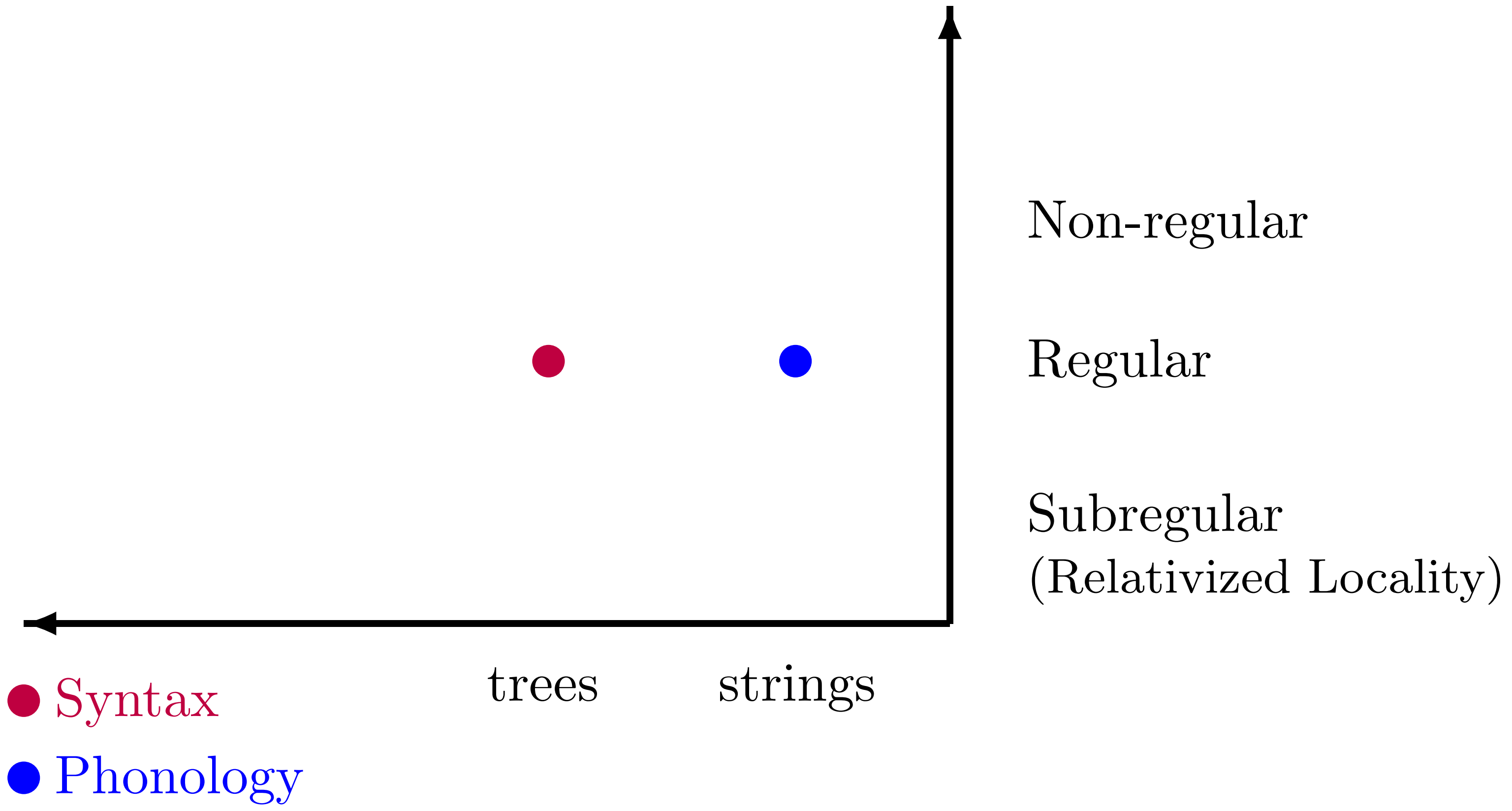


● Syntax

● Phonology

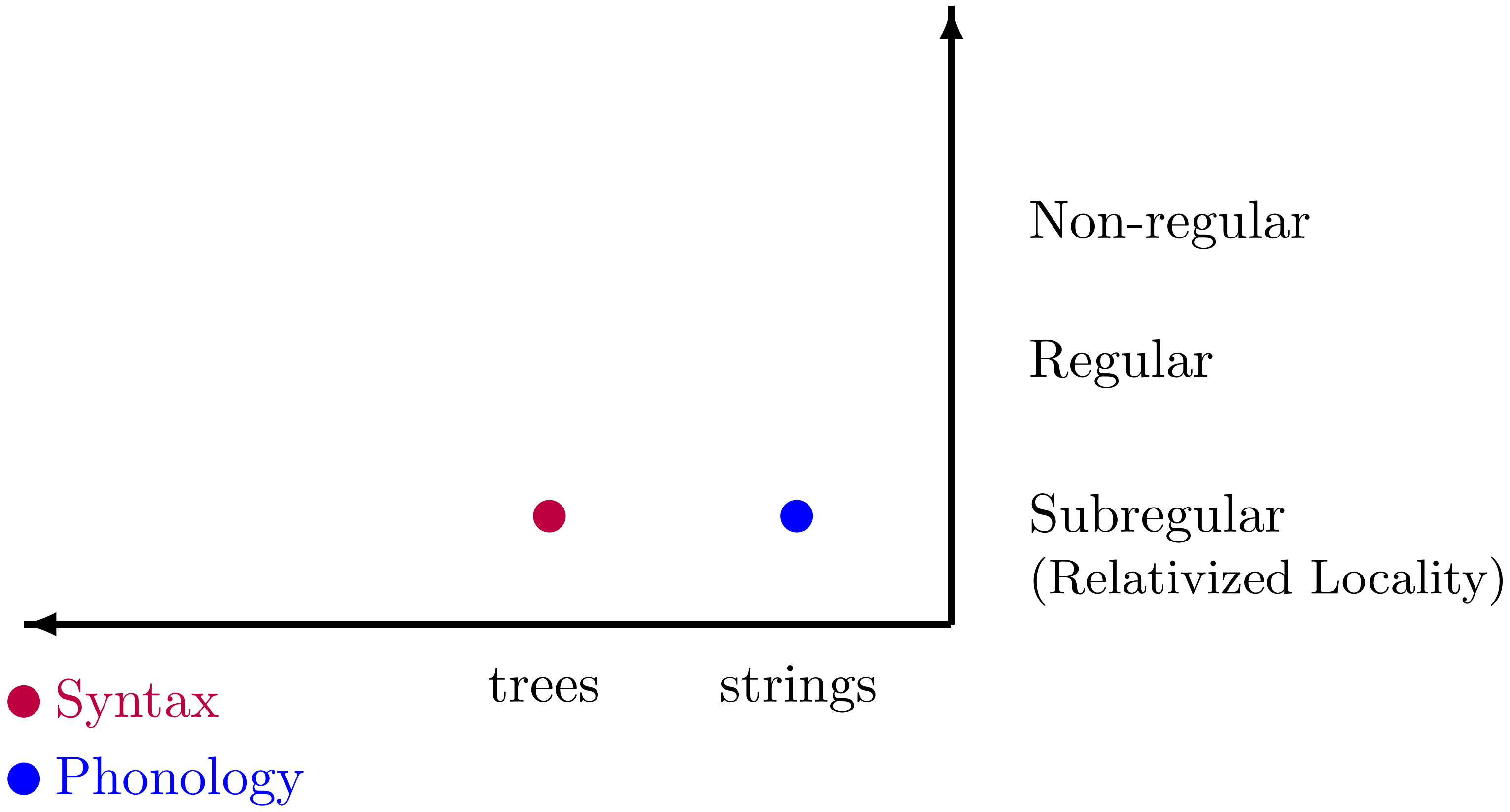
(work with Thomas Graf)

Syntax



(work with Thomas Graf)

Syntax



(work with Thomas Graf)

Additional Areas of Inquiry

1. Systematically compare representations of features, syllables, feet, . . .
2. Phonetics/Phonology Interface
3. Morphology/Phonology Interface
4. Learning lexicons, grammars, exceptions, variation
5. Learning representations

Discussion

1. Well-studied methods from computer science (logic and automata) can be used to express phonological generalizations precisely, accurately, and completely.
2. They are easy to learn with only a little practice.
3. They can be *weighted* to compute probabilties, count violations, handle optionality, . . .
4. One advantage of logic is the flexibility of the representations.
5. One advantage of automata is the previous literature on learning them.
6. Both logic and automata will still be here in 100, 200 years. . . !!

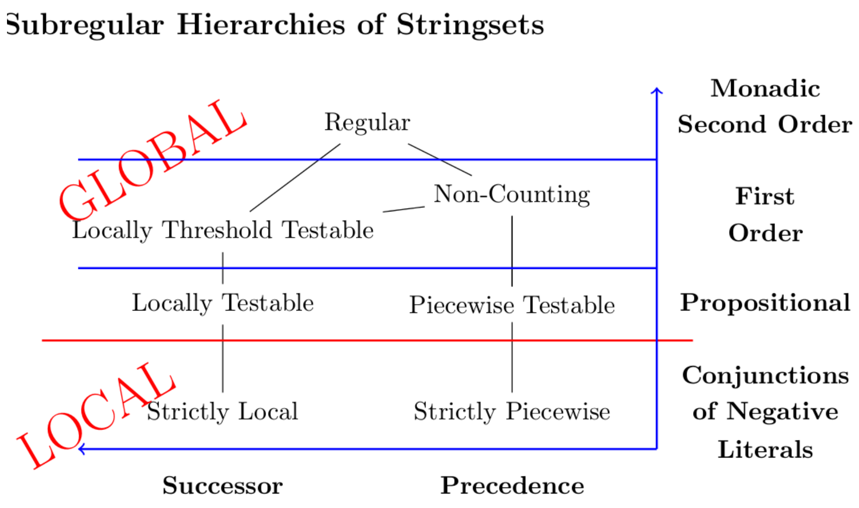
Conclusion

The computational nature of phonology matters because:

1. It provides well-studied methods for relating extensional and intensional descriptions of generalizations.
2. It provides a mathematical foundation for comparing representation and logical power.
3. It often directly leads to psychological models of representation, memory and processing.
4. These models specify what learners must attend to, and thus explains the kinds of phonological generalizations that can be learned.
5. It makes typological predictions and provides explanations for the phonological generalizations we do and do not observe.

Acknowledgments

- Alëna Askenova (Stony Brook)
- Jane Chandlee (Haverford)
- Aniello DeSantos (Stony Brook)
- Hossep Dolatian (Stony Brook)
- Rémi Eryaud (Marseilles)
- Thomas Graf (Stony Brook)
- Hyun Jin Hwangbo (UD)
- Bill Idsardi (UMCP)
- Adam Jardine (Rutgers)
- Regine Lai (HKIEd)
- Kevin McMullin (Ottawa)
- Jon Rawski (Stony Brook)
- Jim Rogers (Earlham)
- Kristina Strother-Garcia (UD)
- Herbert G. Tanner (UD)


Subregular Hierarchies of Stringsets
Monadic Second Order
Regular
GLOBAL
Non-Counting
First Order
Locally Threshold Testable
Locally Testable
Piecewise Testable
Propositional
LOCAL
Strictly Local
Strictly Piecewise
Conjunctions of Negative Literals
Successor
Precedence