The computational nature of phonological generalizations

Jeffrey Heinz

Rutgers University

April 28, 2017

Today

1. Show that the computational nature of phonological generalizations has implications for
   - Typology
   - Learning
   - Psychology and memory
2. Argue that logic and automata are the best vehicles for expressing phonological generalizations

Part I

What is phonology?

The fundamental insight

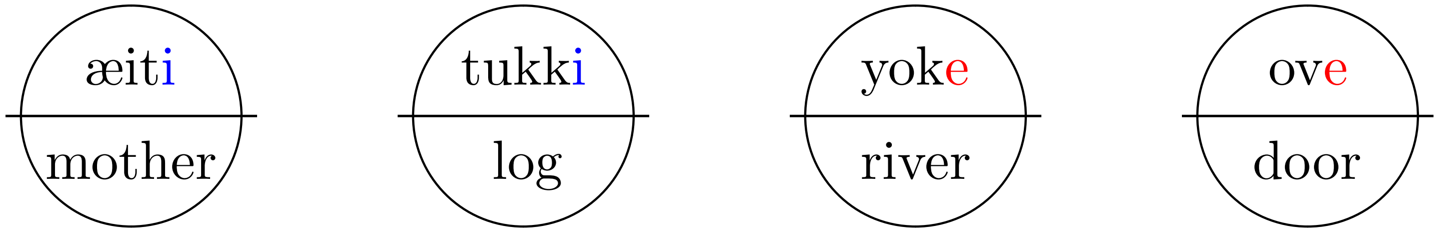
The fundamental insight in the 20th century which shaped the development of generative phonology is that the **best** explanation of the systematic variation in the pronunciation of morphemes is to posit a single underlying mental representation of the phonetic form of each morpheme and to derive its 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 generalizations 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 $\longrightarrow$ [+high] / __ #
2. *e# >> IDENT(HIGH)

Nothing precludes these grammars from operating on words of *any* length. The infinite objects those grammars describe look like this:

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

(manilabanile,manilabanili), ...

Truisms about transformations

1. Different grammars may generate the same transformation.

   Such grammars are *extensionally equivalent*.

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

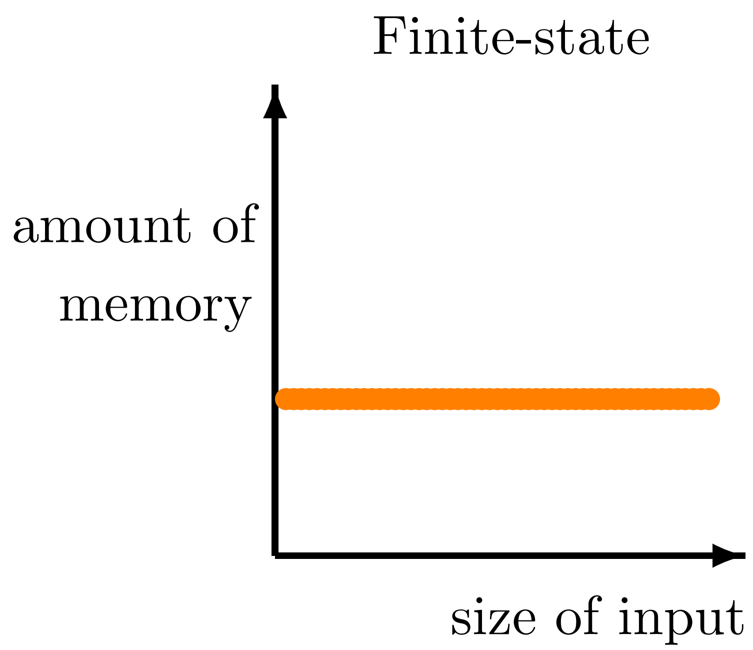
3. Transformations may have properties *largely independent* of their grammars.
   - output-driven maps (Tesar 2014)
   - finite-state functions (Elgot and Mezei 1956, Scott and Rabin 1959)
   - subsequential functions (Oncina et al. 1993, Mohri 1997, Heinz and Lai 2013)
   - strictly local functions (Chandlee 2014)

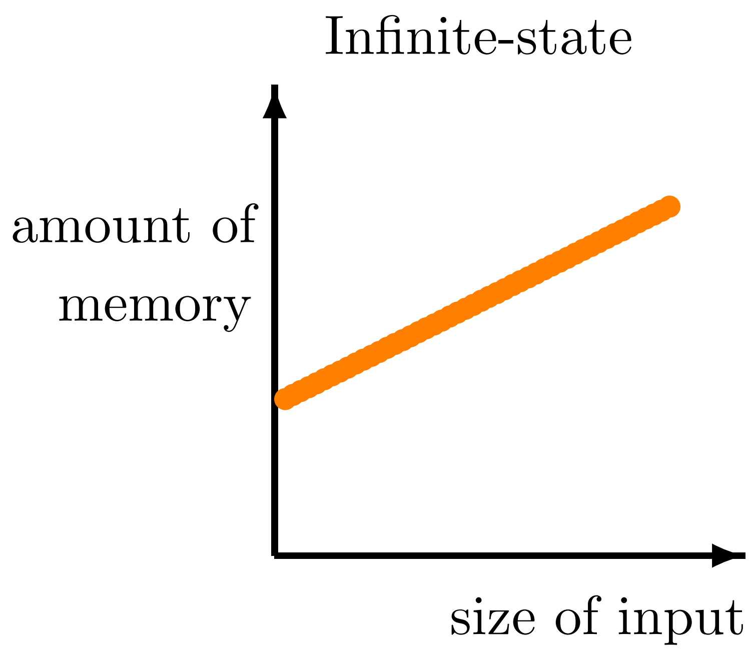
Part II

Phonological Generalizations are Finite-state

What “Finite-state” means

A generalization is finite-state provided **the memory required is bounded by a constant, *regardless of the size of the input.***

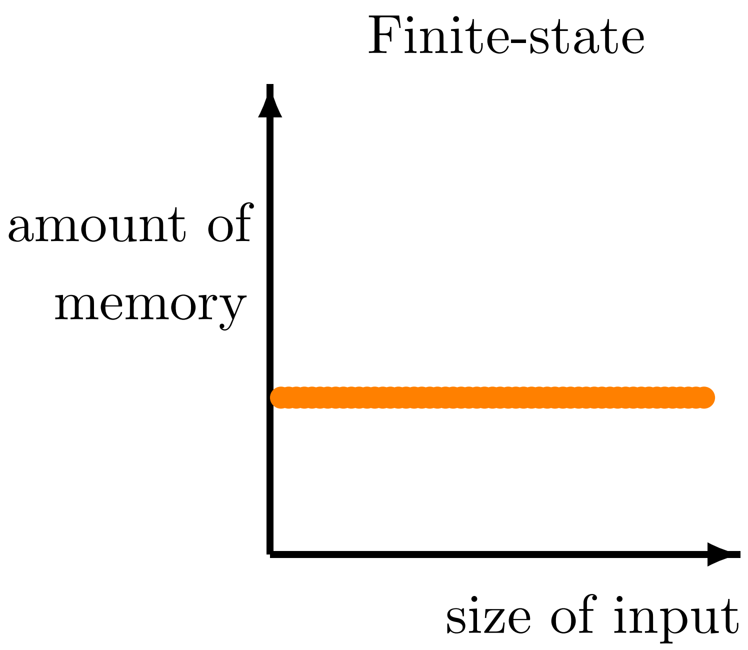


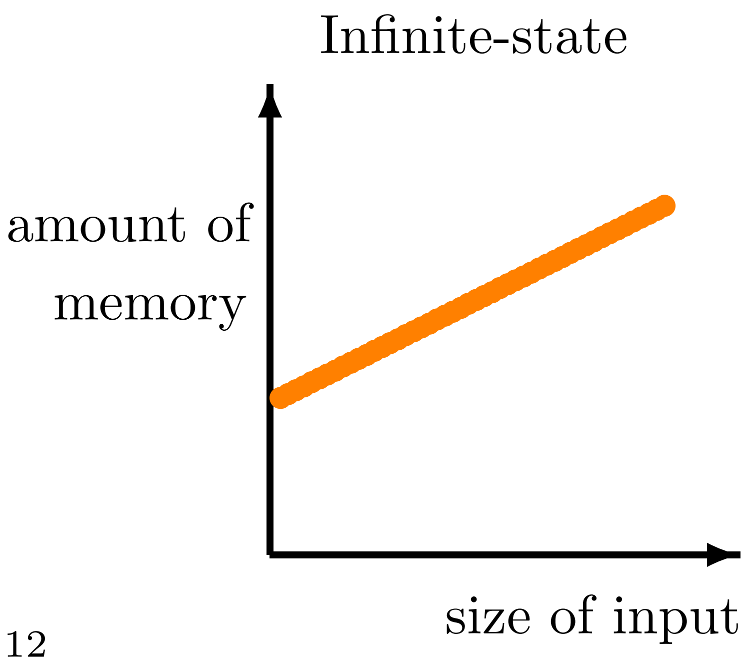


Any finite-state device “processing any input with respect to the generalization” has a finite number of distinct internal states (constant memory capacity).

Processing an input with respect to the generalization

- For given constraint C and any representation w:
  - Does w violate C?
  - (Or, how many times does w violate C?)
- For given grammar G and any underlying representation w:
  - What is the surface representation when G transforms w?





Example: Vowel Harmony

Progressive

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

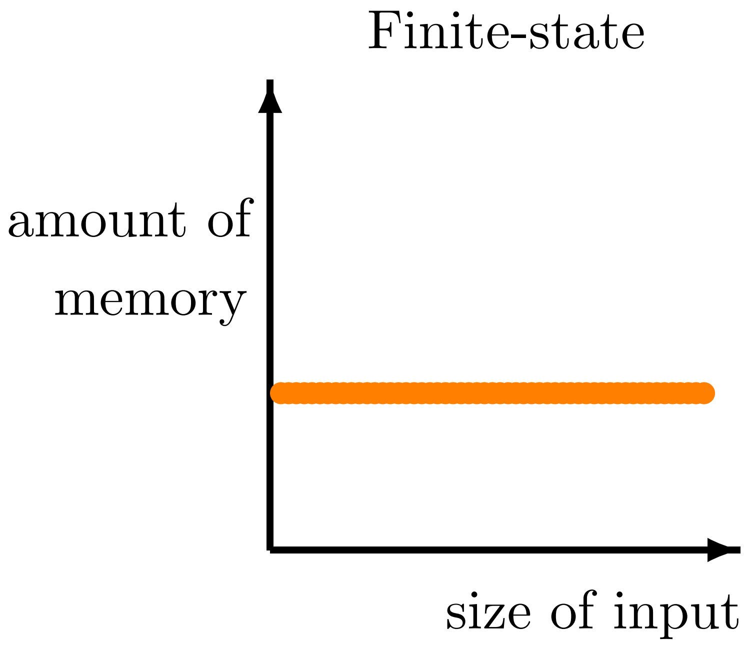
Majority Rules

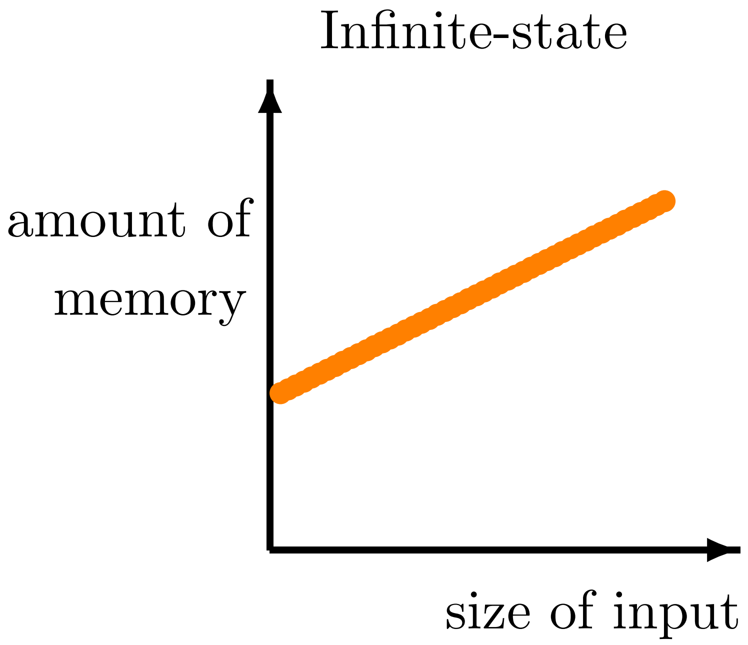
Vowels agree in backness with the majority of vowels in the underlying representation.

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

Discussion

- Majority Rules is not finite-state (Riggle 2004, Heinz and Lai 2013).
- Majority Rules is unattested (Bakovic 2000).
- Human subjects fail to learn Majority Rules in artificial grammar learning experiments, unlike progressive harmony (Finley 2008, 2011).
- There exists a CON and ranking over it which generates Majority Rules: AGREE(BACK)>>IDENTIO[BACK].
- Changing CON may resolve this, but does this miss the forest for the trees?

Phonological generalizations are finite-state

Evidence supporting the hypothesis that phonological generalizations are finite-state originates with Johnson (1972) and Kaplan and Kay (1994), who showed how to translate any SPE-style rewrite rule into a grammar known to be finite-state.

Consequently:

1. Any phonological transformation expressible with SPE-style rewrite rules is finite-state.
2. Phonological grammars defined by an ordered sequence of rules are finite-state (since finite-state functions are closed under composition).
3. Constraints on well-formed surface and underlying representations are finite-state (since the image and pre-image of finite-state functions are finite-state).

(Rabin and Scott 1959)

Finite-state grammar formalisms

- Finite-state automata
- Regular expressions
- Monadic Second-Order logic

Part III

Finite-state Automata and Logic

Finite-state automata and logic

I am going to now argue that these grammar formalisms have much to offer phonology and phonological theory with respect to:

1. generation
2. typological predictions
3. learnability and learning models
4. pyscholinguistic models and memory

…even when compared to rule-based and constraint-based formalisms.

1. Generation

Word-final /e/ raising

1. $e \longrightarrow [+\text{high}]$ / __ #
2. *e# >> Ident(high)

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

(manilabanile,manilabanili), . . .

How do the above intensional grammars above relate to the extension below?

1. Generation with Rules

$$aa \rightarrow b$$

- What is the output of this rule applied to *aaa*?

"*To apply a rule, the entire string is first scanned for segments that satisfy the environmental constraints of the rule. After all such segments have been identified in the string, the changes required by the rule are applied simultaneously.*" Chomsky and Halle (1968, p.344):

1. Generation with OT

Given an OT grammar and an input form, there is a well-defined solution to the generation problem.

Known Issues:

1. Are **all** relevant constraints present in EVAL?
2. Does EVAL consider **all** candidates produced by GEN?
3. Are violations counted properly?

Prince (2002 , p. 276) explains that if a constraint is ignored that must be dominated by some other constraint then the analysis is "*dangerously incomplete.*" Similarly, if a constraint is omitted that may dominate some other constraint then the analysis is "*too strong and may be literally false.*"

1. Generation with OT (continued)

Solutions exist for limited cases

- Kartunnen 1998 (see also Gerdemann and van Noord 2000, Gerdemann and Hulden 2012)
- Riggle 2004

GEN, and the constraints in CON must be finite-state, and optimization must result in a finite-state grammar. (Albro 2005 allows GEN to be infinite-state.)

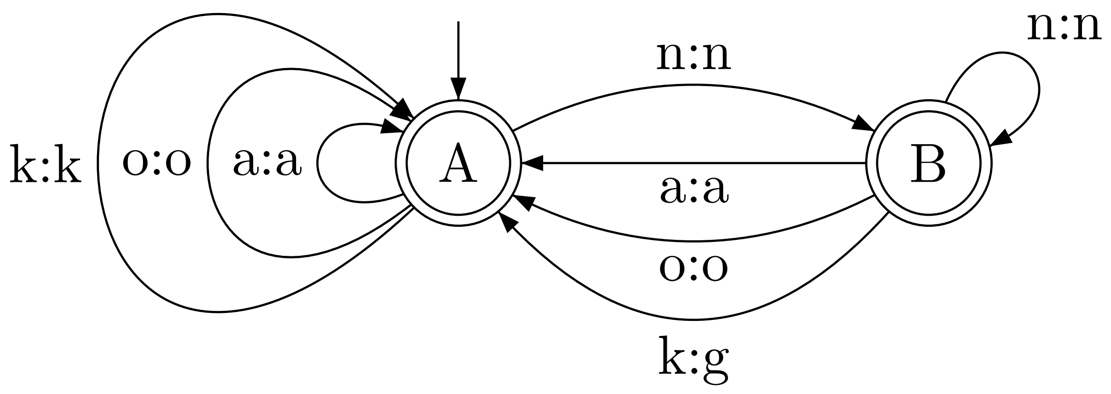
Software in use

- OT-Workplace addresses (1,3) with finite-state grammars.
- OT-Soft and OT-Help don't address these issues.

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing



Input:

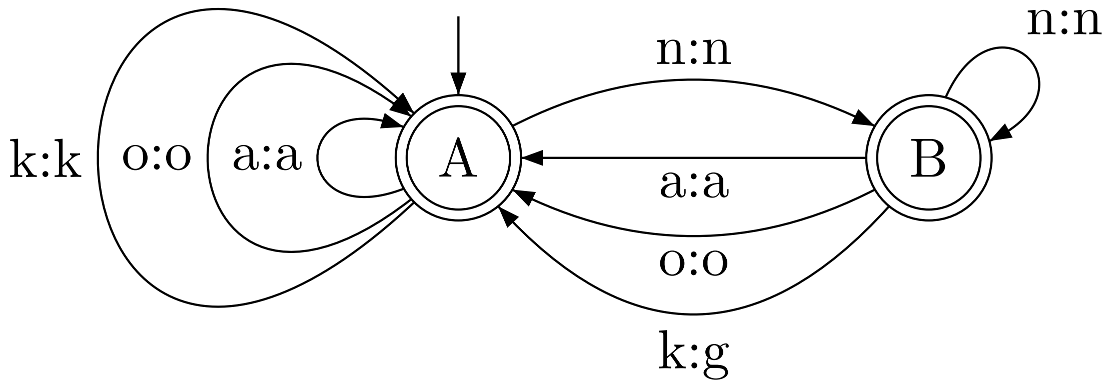
States: A

Output:

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing

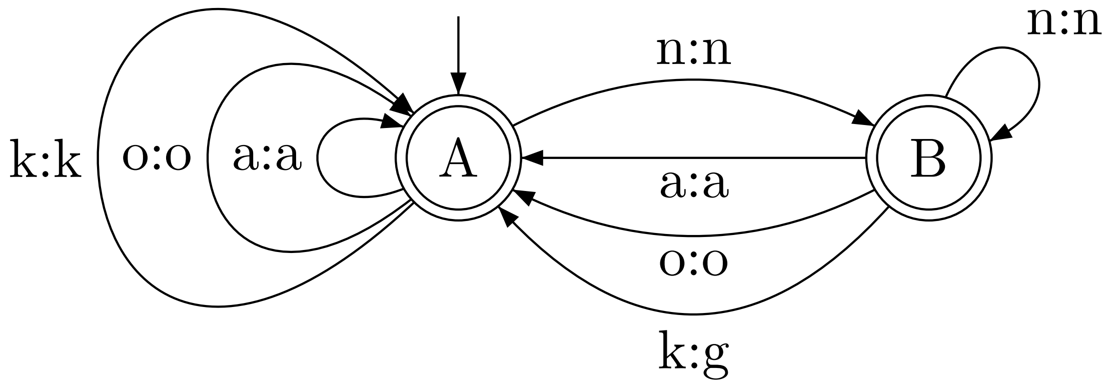


Input:		k	
States:	A	$\rightarrow$	A
Output:		k	

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing

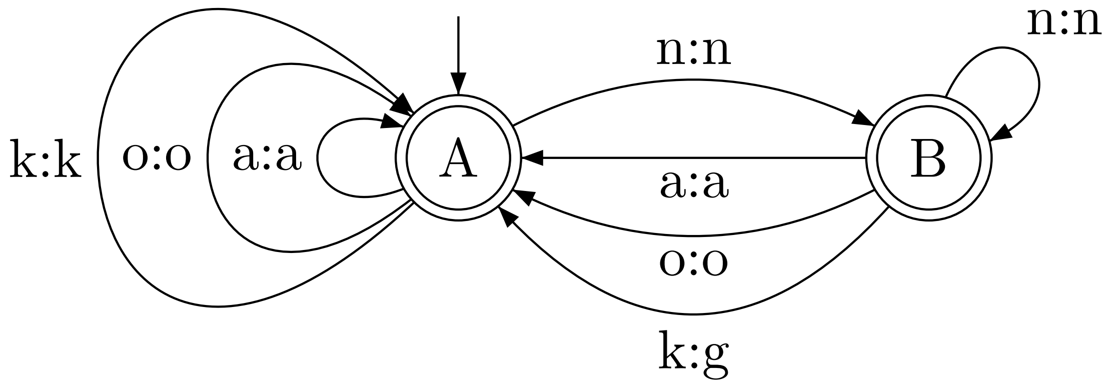


Input:		k		o	
States:	A	→	A	→	A
Output:		k		o	

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing

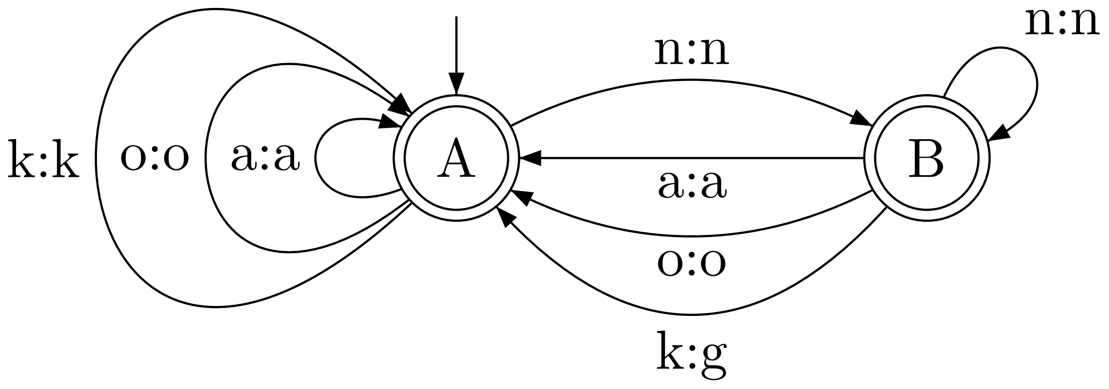


Input:		k		o		n	
States:	A	$\rightarrow$	A	$\rightarrow$	A	$\rightarrow$	B
Output:		k		o		n	

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing

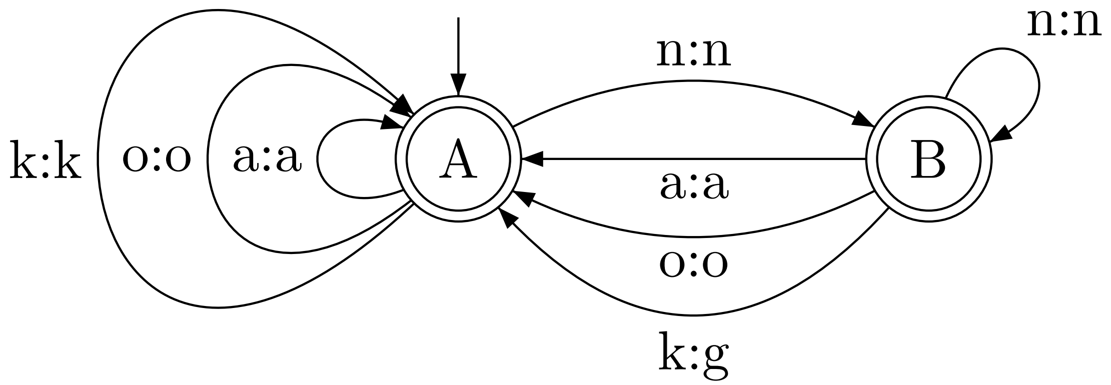


Input:		k		o		n		k	
States:	A	→	A	→	A	→	B	→	A
Output:		k		o		n		g	

1. Generation with finite-state automata

- Well-studied and explained in many textbooks.

Post Nasal Voicing



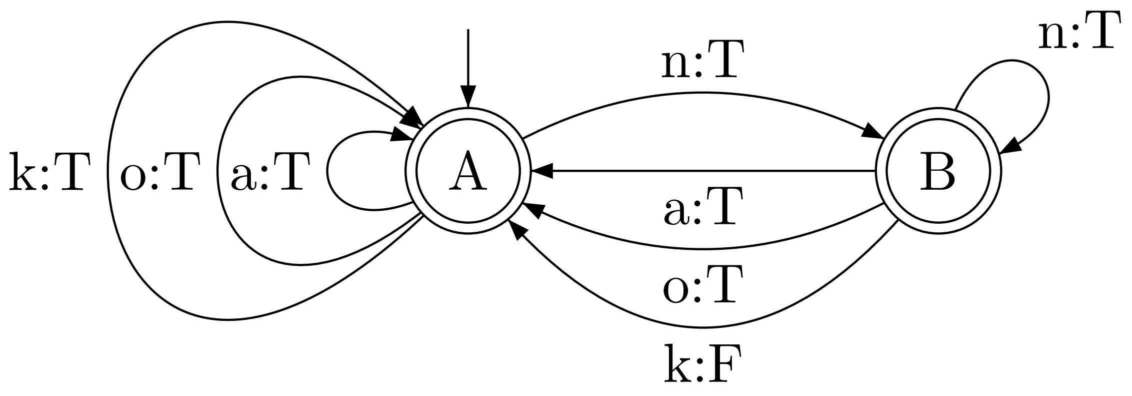
Input:		k		o		n		k		o	
States:	A	→	A	→	A	→	B	→	A	→	A
Output:		k		o		n		g		o	

Concatenating the outputs yields: /konko/ $\mapsto$ [kongo]

1. Generation with finite-state automata (cont.)

- Well-studied and explained in many textbooks.

***Nasal-Voiceless Obstruent (categorical version)**



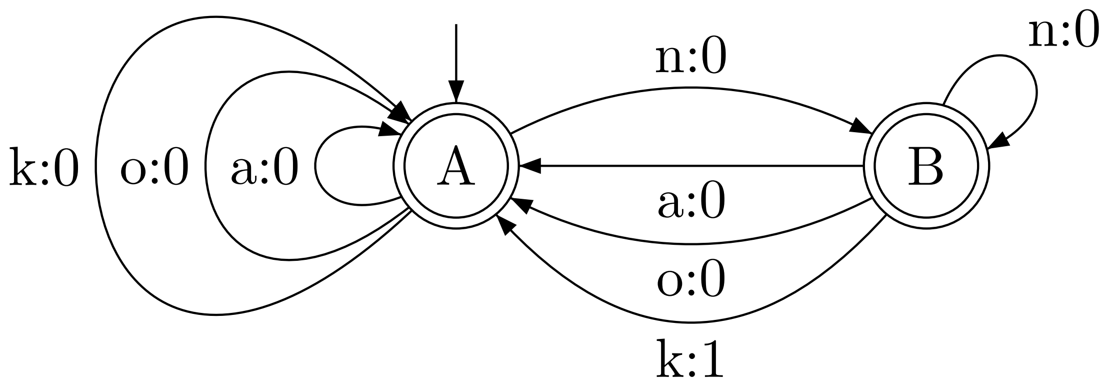
Input:		k		o		n		k		o	
States:	A	→	A	→	A	→	B	→	A	→	A
Output:		T		T		T		F		T	

Conjunction of the outputs yields: /konko/ $\mapsto$ `False`

1. Generation with finite-state automata (cont.)

- Well-studied and explained in many textbooks.

***Nasal-Voiceless Obstruent (counting version)**



Input:		k		o		n		k		o	
States:	A	$\rightarrow$	A	$\rightarrow$	A	$\rightarrow$	B	$\rightarrow$	A	$\rightarrow$	A
Output:		0		0		0		1		0	

Summing the outputs yields: /konko/ $\mapsto$ 1

1. Generation with MSO logic

- Logical expressions can translate one relational structure into another.

$$P_\varphi(x) \quad \overset{\text{def}}{=} \quad Q(x) \tag{1}$$

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

$$\texttt{voiced}_\varphi(x) \quad \overset{\text{def}}{=} \quad \texttt{voiced}(x) \vee (\exists y)[y \triangleleft x \wedge \texttt{nasal}(y)] \tag{2}$$

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

$$x \triangleleft_\varphi y \quad \overset{\text{def}}{=} \quad x \triangleleft y \tag{4}$$

(Courcelle and Engelfriedt 2001, 2011)

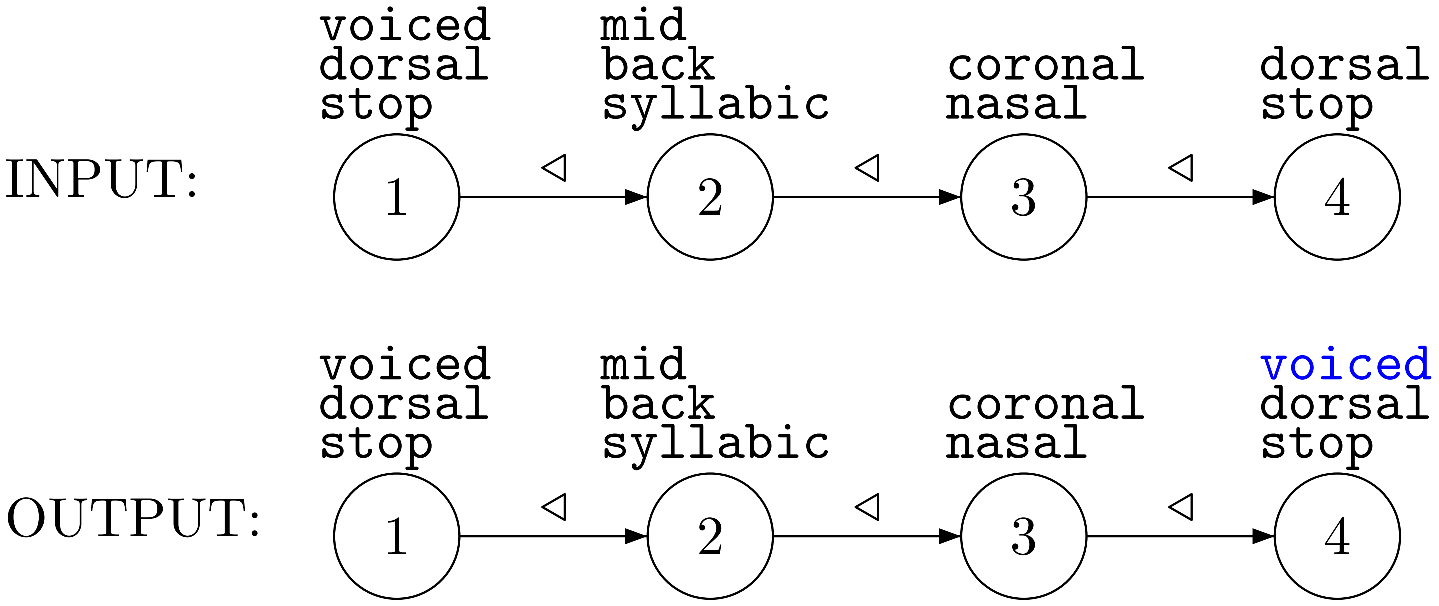
1. Generation with MSO logic (cont.)

$$\texttt{voiced}_\varphi(x) \overset{\text{def}}{=} \texttt{voiced}(x) \vee (\exists y)[y \triangleleft x \wedge \texttt{nasal}(y)] \quad (2)$$

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

$$x \triangleleft_\varphi y \overset{\text{def}}{=} x \triangleleft y \quad (4)$$

Post Nasal Voicing /gonk/ $\mapsto$ [gong]



(Courcelle and Engelfriedt 2001, 2011)

1. Generation with MSO logic (cont.)

***Nasal-Voiceless Obstruent (categorical version)**

$$^{*}\mathtt{NT} \overset{\text{def}}{=} \neg(\exists x, y)\Big[x \triangleleft y \wedge \mathtt{nasal}(x) \wedge \neg\mathtt{voiced}(y) \wedge \neg\mathtt{consonantal}(y)\Big]$$

***Nasal-Voiceless Obstruent (counting version)**

$$^{*}\mathtt{NT} \overset{\text{def}}{=} (\exists x, y)\Big[x \triangleleft y \wedge \mathtt{nasal}(x) \wedge \neg\mathtt{voiced}(y) \wedge \neg\mathtt{consonantal}(y) \wedge 1\Big]$$

(Droste and Gastin 2009)

Discussion

1. Well-studied methods from computer science 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 probabiltiies, count violations, ...
4. One advantage of logic is the flexibility of the representations.
5. Linguists can use and systematically explore different representational primitives like features, syllables, etc (Potts and Pullum 2002, Graf 2010, Jardine 2016).
6. Automata are best understood for strings and trees, but they can be introduced for other representations as well.
7. For the string and tree cases, translations exist between MSO logic and automata.
8. Both logic and automata will still be here in 100, 200 years...!!

Discussion (continued)

So… Phonological generalizations are finite-state. Is that a theory of phonology?

1. No.
2. As a hypothesis, it states a *necessary* condition of phonological generalizations, not a *sufficient* one.
3. So not all finite-statable generalizations are going to be phonological.

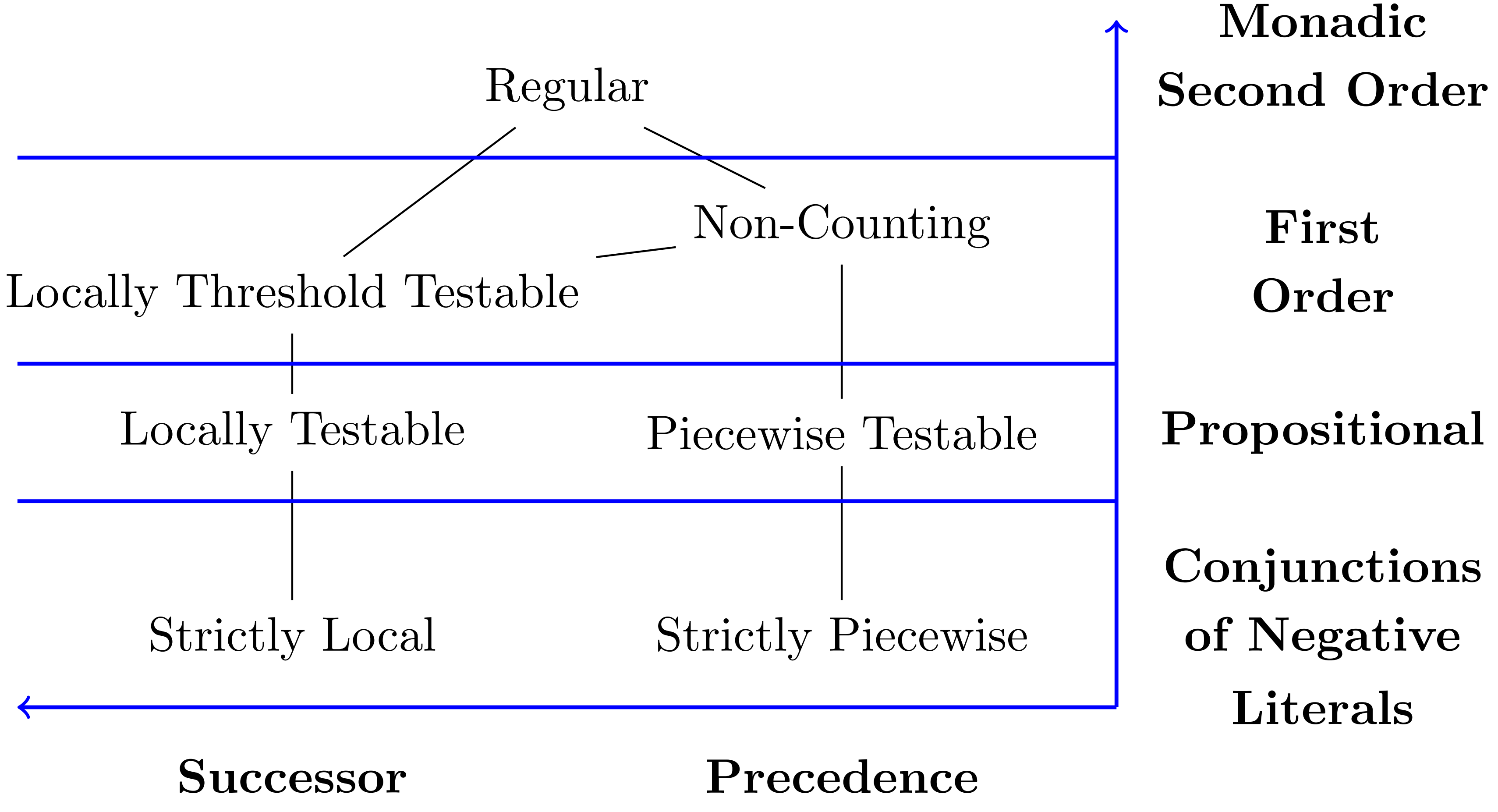
BTW, the word *regular* is often used as a synonym for *finite-state*.

2. Typology 3. Learnability 4. Psychology…

Part IV

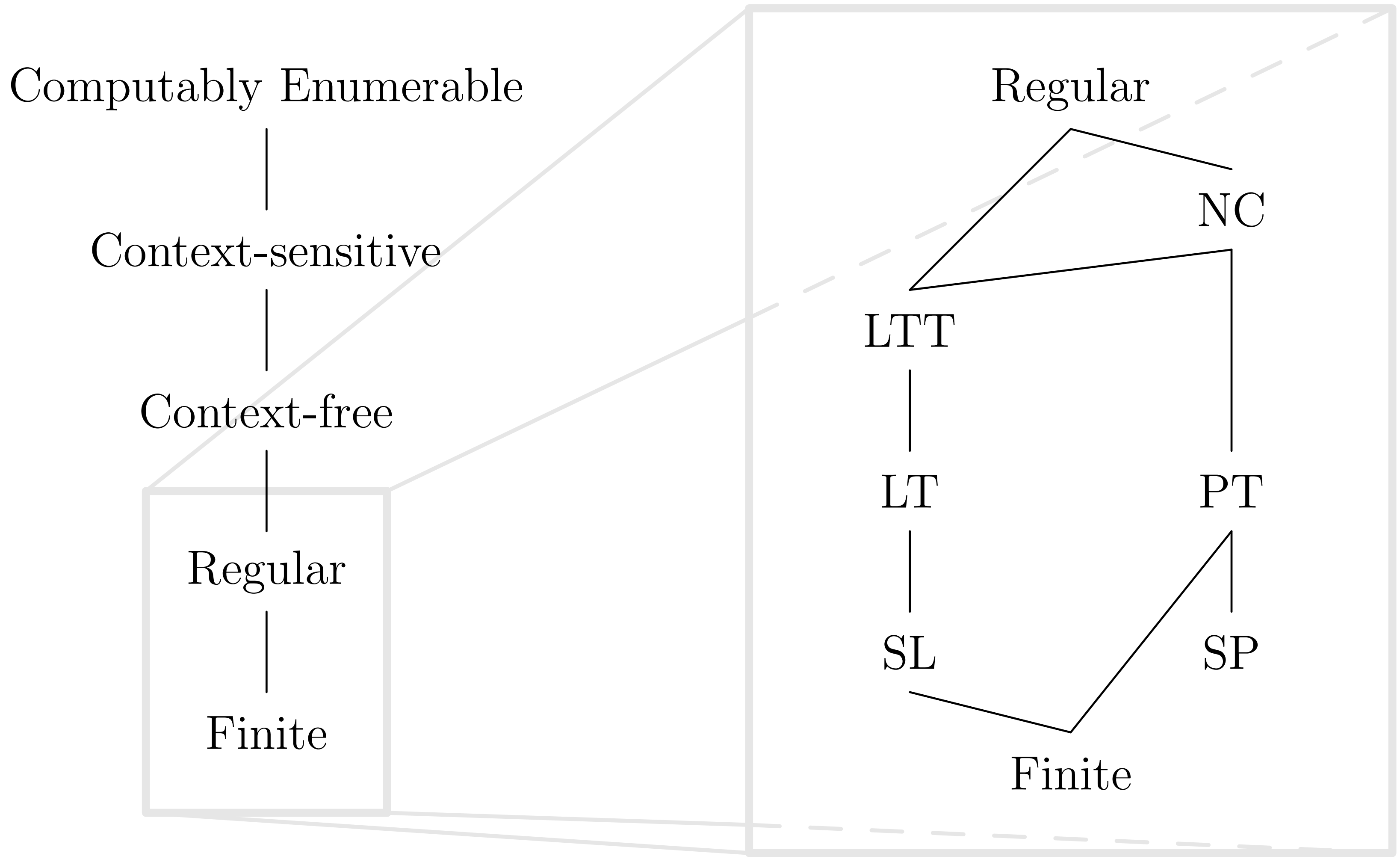
Phonological generalizations are less than Finite-state

Subregular Hierarchies of Stringsets



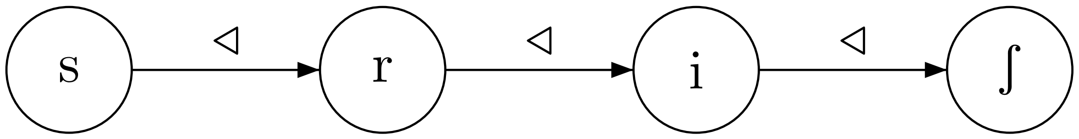
(McNaughton and Papert 1971, Heinz 2010, Rogers and Pullum 2011, Rogers et al. 2013)

The Chomsky Hierarchy



Representing words with successor

hypothetical [sriʃ]



- The information about order is given by the successor (◁) relation.

Sub-structures

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

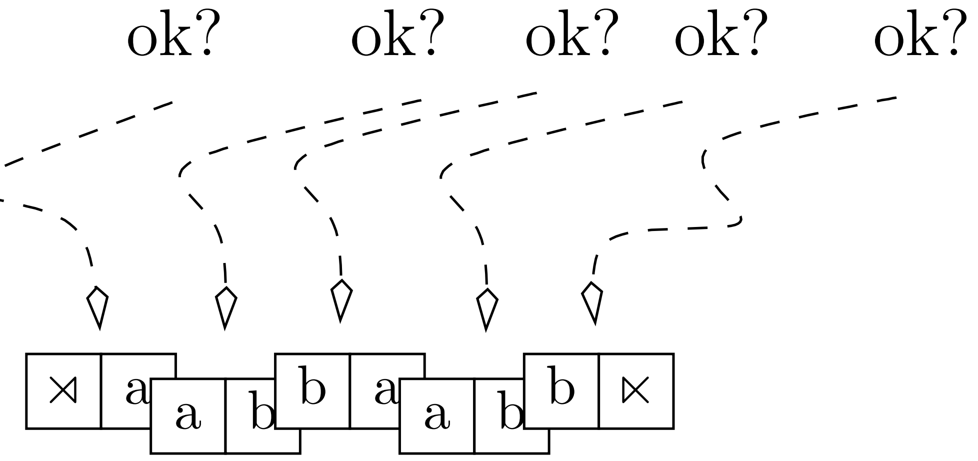
- So [sr] is a sub-structure of [sriʃ]

$$\begin{tikzcd}
\boxed{\mathbf{s}} \arrow[r, very thick, "\lhd"] & \boxed{\mathbf{r}} \arrow[r, "\lhd"] & \text{i} \arrow[r, "\lhd"] & \int
\end{tikzcd}$$

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 sub-structures* (with words represented using the successor relation).
- Here is the string *abab*. If we fix a diameter of 2, we have to check these substrings.

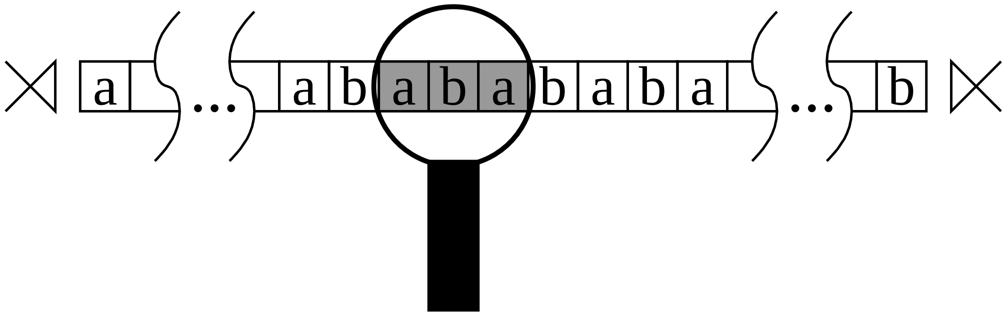


(Rogers and Pullum 2011, Rogers et al. 2013)

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.
- Memory is limited to these specific chunks, taken one at a time.

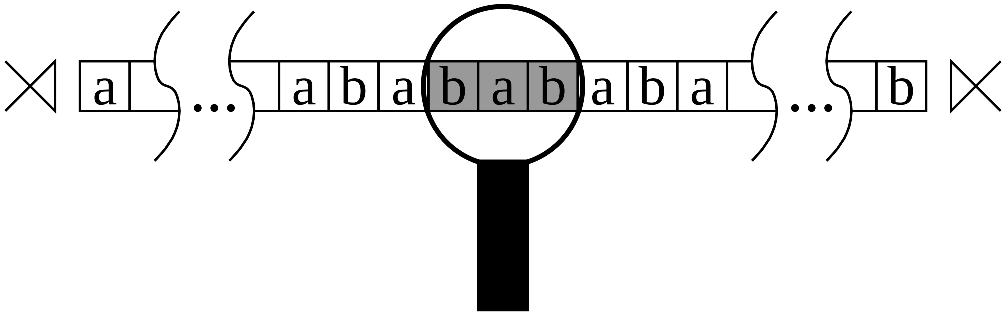


(Rogers and Pullum 2011, Rogers et al. 2013)

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.
- Memory is limited to these specific chunks, taken one at a time.

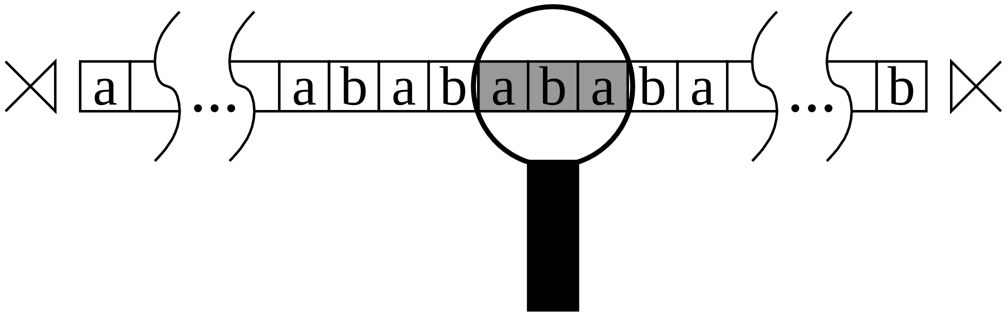


(Rogers and Pullum 2011, Rogers et al. 2013)

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.
- Memory is limited to these specific chunks, taken one at a time.



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

Examples of Strictly Local constraints for strings

- *aa
- *ab
- *NT
- NoCoda

Examples of Non-Strictly Local constraints

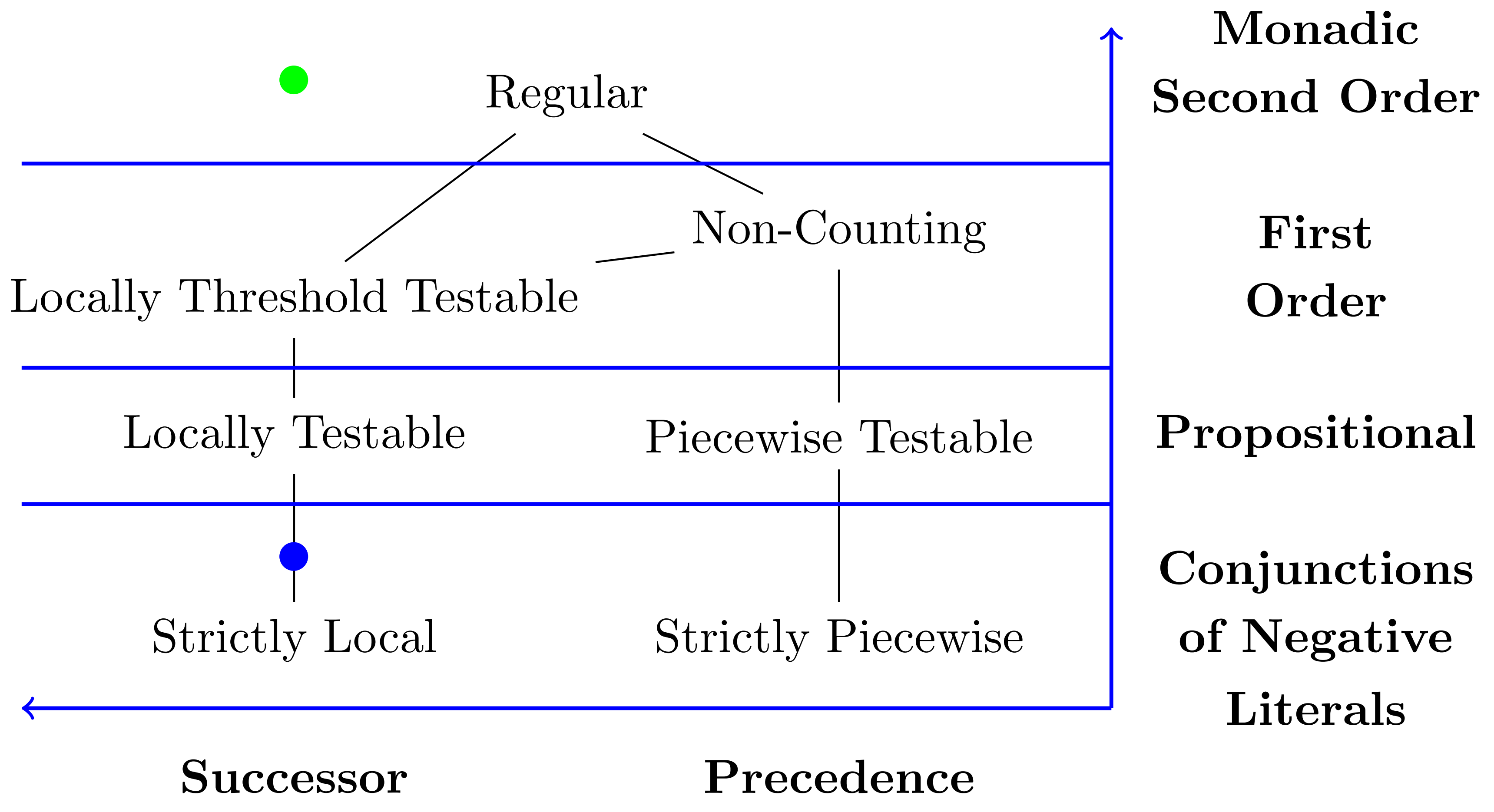
- *s...ʃ (Hansson 2001, Rose and Walker 2004, Hansson 2010, inter alia)
- Obligatoriness: Words must contain one primary stress (Hayes 1995, Hyman 2011, inter alia).

Sarcee (Cook 1978, 1984)

a.	/si-tʃiz-aʔ/	→	**ʃ**í**t**ʃídzàʔ	‘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)

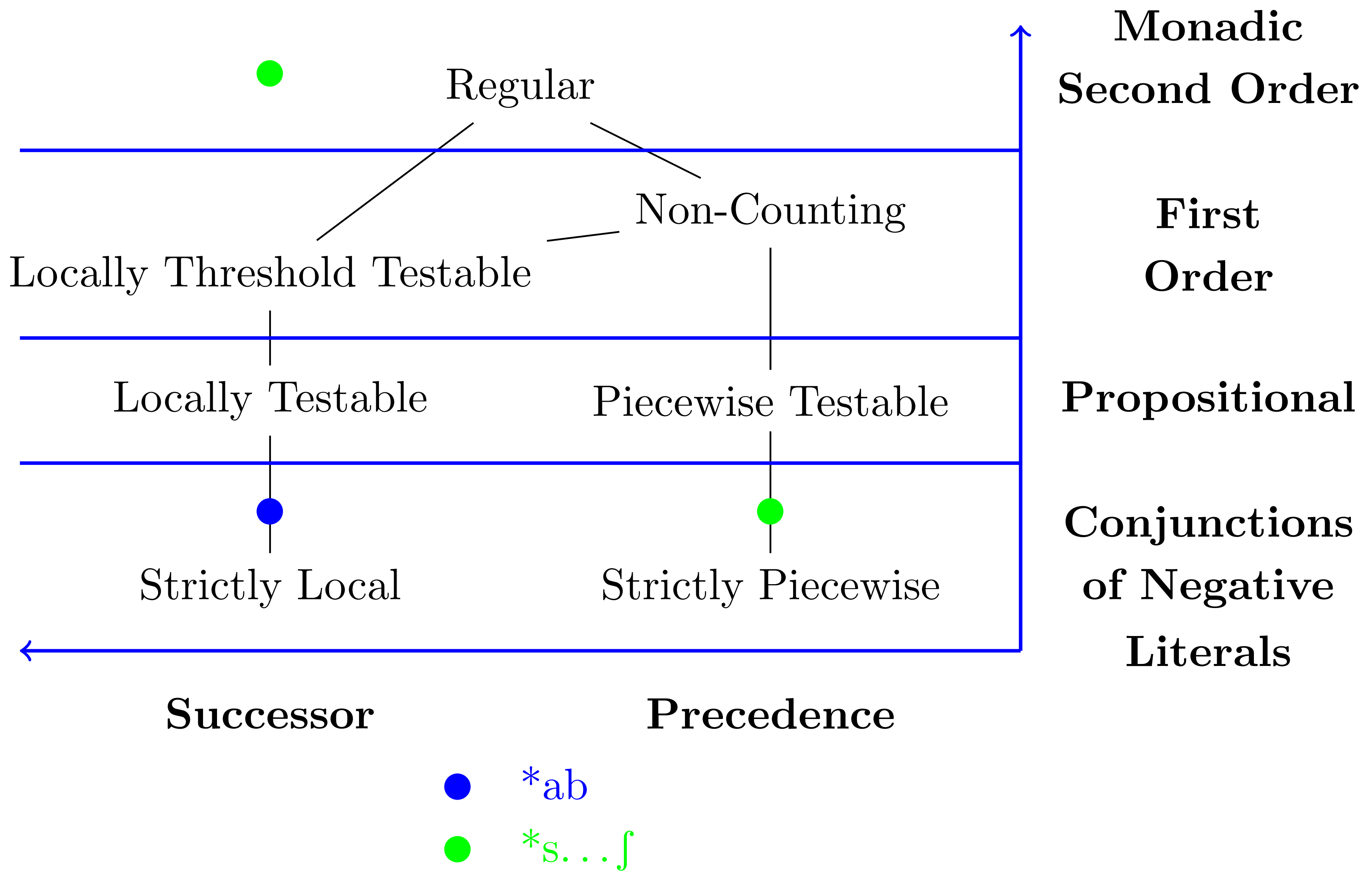
The “MSO with successor” Theory

Is this a good theory of possible constraints in phonology?

NO! Because…

1. Typologically, it overgenerates.
   (a) *EVEN-Sibilants
   (b) 2 NT structures OK, but 3 is forbidden
2. There are no feasible algorithms for learning the whole class of regular stringsets (Gold 1967, inter alia).

Subregular Hierarchies of Stringsets



(Heinz 2010, Rogers et al. 2010)

Representing Order in Sequences

hypothetical [sriʃ]

1. Successor

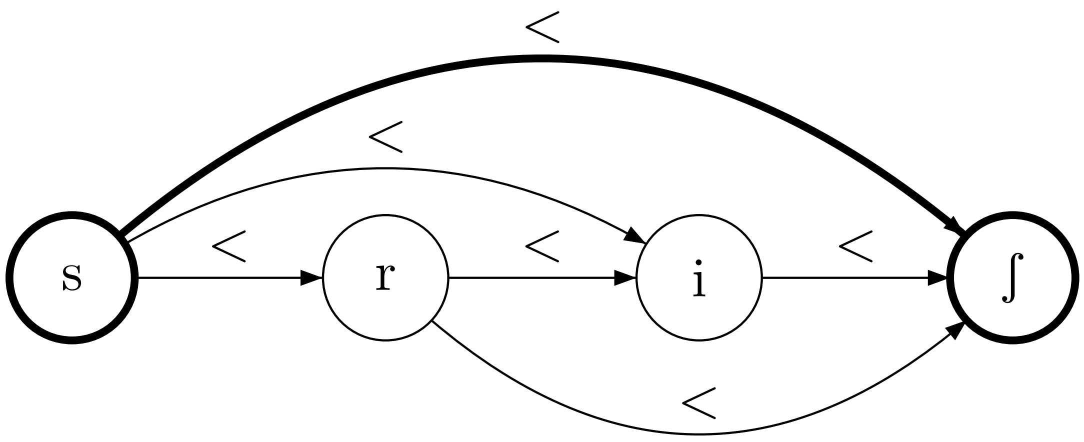
$$\begin{tikzcd}
s \arrow[r, "\triangleleft"] & r \arrow[r, "\triangleleft"] & i \arrow[r, "\triangleleft"] & ʃ
\end{tikzcd}$$

2. Precedece

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

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

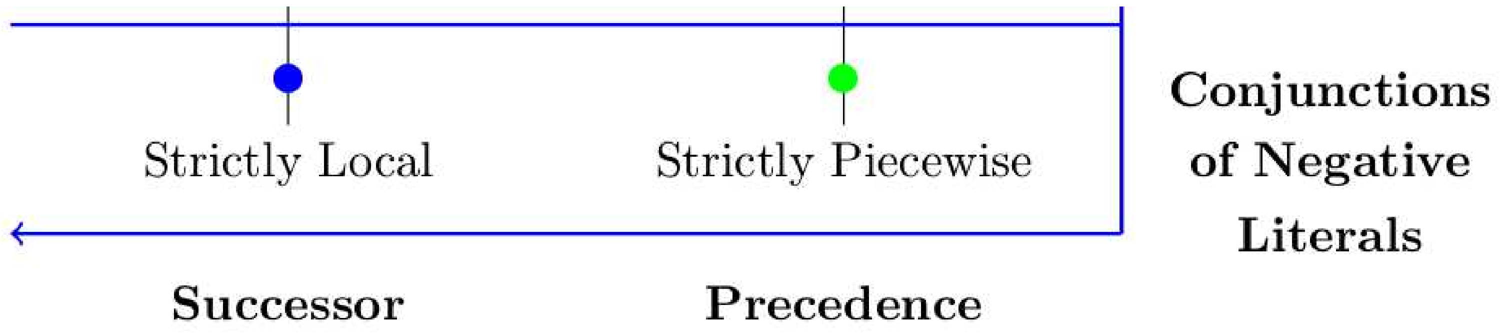
- So $\boxed{\text{s} < \int}$ is a sub-structure of [sriʃ]



$$\begin{tikzcd}
s \arrow[r, "<"] \arrow[rr, bend left, "<"] \arrow[rrr, bend left=40, "<", line width=2pt] & r \arrow[r, "<"] \arrow[rr, bend right, "<"'] & i \arrow[r, "<"] & \int
\end{tikzcd}$$

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

The CNL with Successor and Precedence Theory



- ● *ab
- ● *ʃ. . . s

Is this a better theory of possible constraints in phonology?

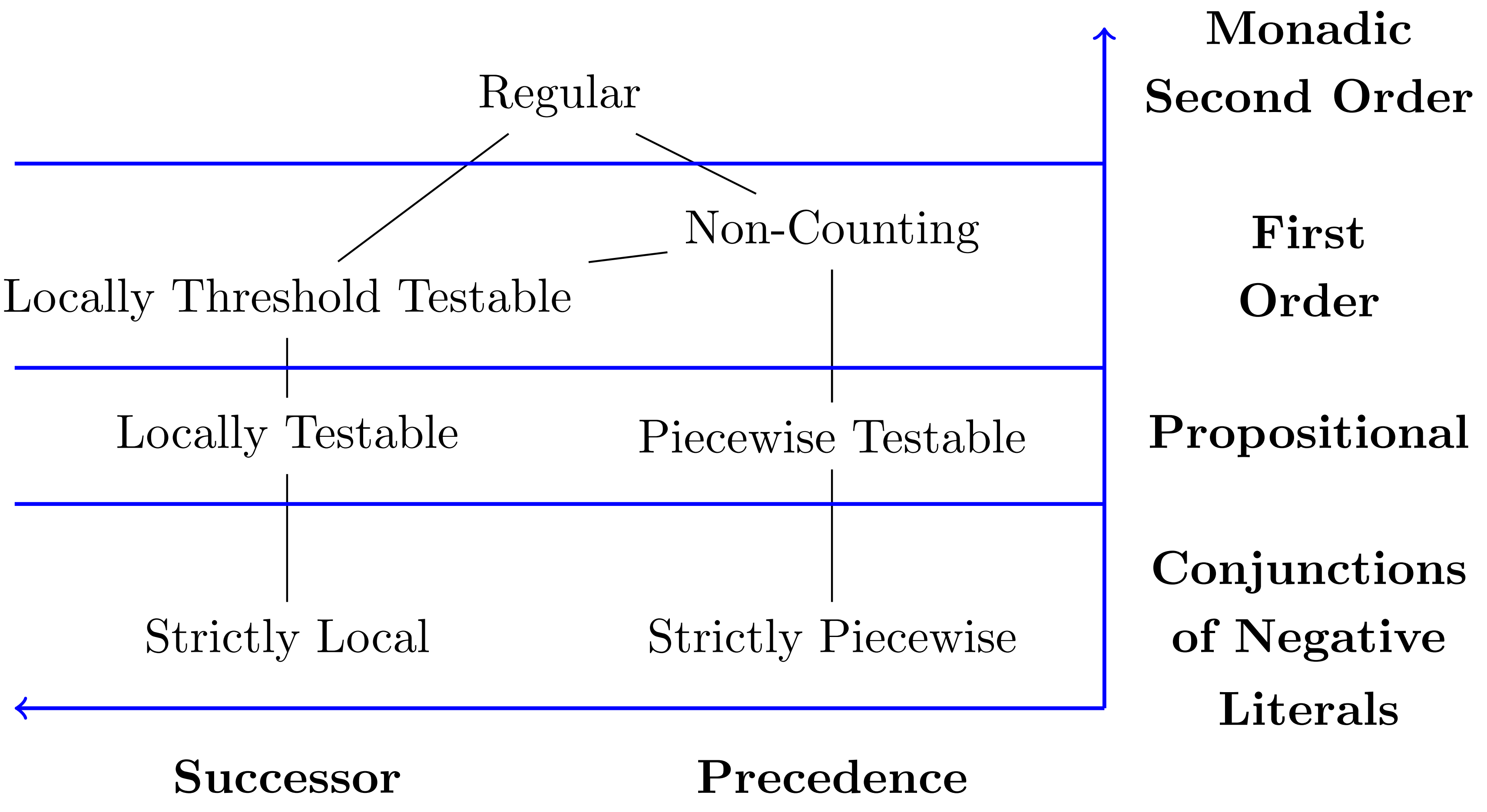
1. Typologically, it is better than “MSO with successor.”
   (a) Admits phonotactic constraints which arguably drive long-distance harmony patterns
   (b) Provably excludes constraints like *EVEN-Sibilants and “2 NT structures OK, but 3 is forbidden”.
2. Both Strictly Local and Strictly Piecewise constraints 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 interplay between representation and computation in linguistic theory can be studied upon a firm mathematical foundation.

Subregular Hierarchies of Stringsets



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

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)

Part V

Conclusion

Conclusion

The computational nature of phonology matters because:

1. It provides well-studied solutions to the generation problem.
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

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