Beyond Optimality:

The computational nature of phonological maps and constraints

Jeffrey Heinz (Delaware) and William Idsardi (Maryland)

Whither OT?

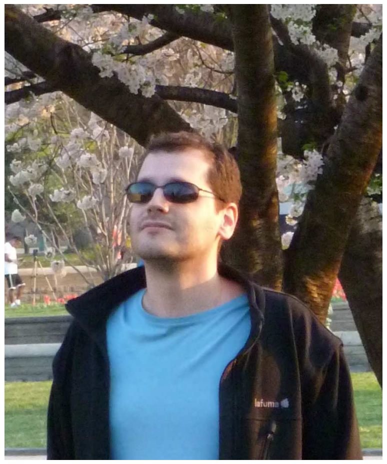
Workshop at the 23rd Manchester Phonology Meeting

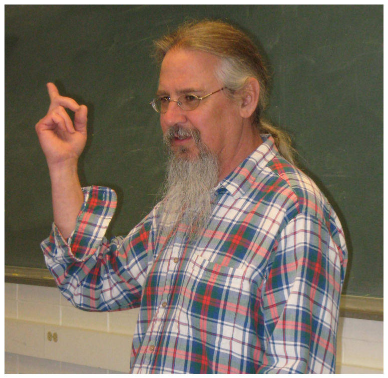
May 27, 2015

University of Manchester

Primary Collaborators

- Dr. Jane Chandlee, UD PhD 2014 (Haverford, as of July 1)
- Prof. Rémi Eryaud (U. Marseilles)
- Adam Jardine (UD, PhD exp. 2016)
- Prof. Jim Rogers (Earlham College)





Main Claim

- Particular *sub-regular* computational properties—and not optimization—**best** characterize the nature of phonological generalizations.

Part I

What is phonology?

The fundamental insight

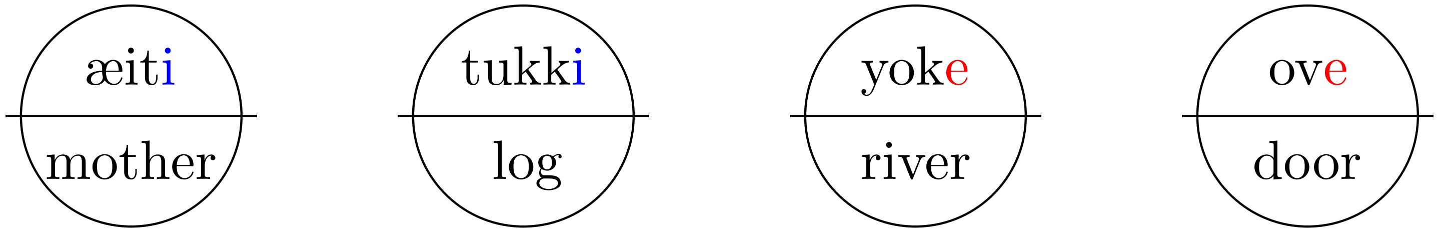
The fundamental insight in the 20th century which shaped the development of generative phonology is that the **best** explanation of the systematic variation in the pronunciation of morphemes is to posit a single underlying mental representation of the phonetic form of each morpheme and to derive its pronounced variants with context-sensitive transformations.

(Kenstowicz and Kisseberth 1979, chap 6; Odden 2014, chap 4)

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.*

Desiderata for phonological theories

1. Provide a theory of typology
   - Be sufficiently expressive to capture the range of cross-linguistic phenomenon
     (explain what is there)
   - Be restrictive in order to be scientifically sound
     (explain what is not there)
2. Provide learnability results
   (explain how what is there could be learned)
3. Provide insights
   (for example: grammars should distinguish marked structures from their repairs)
4. Effectively computable

Part II

Transformations

Phonological 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 $\longrightarrow$ [+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), ...

Likelihood and Well-formedness

- Some would equate probability with well-formedness.

Unless **all** words which violate some markedness constraint have probabiltity zero, this effectively changes the object of inquiry from an infinite set to a finite one.

Why?

- If there are infinitely many words that violate no markedness constraints and at least one word that violates a markedness constraint (like [bzaʃrk]) that has probabilty $\epsilon > 0$ . . .
- Then at some point the probabilities must decrease exponentially in order to sum to 1.
- Therefore, there are infinitely many words violating no markedness constraints which have probability $< \epsilon$ (including perhaps [kapalatʃapoʊlapinisiwaki]).

Truisms about transformations

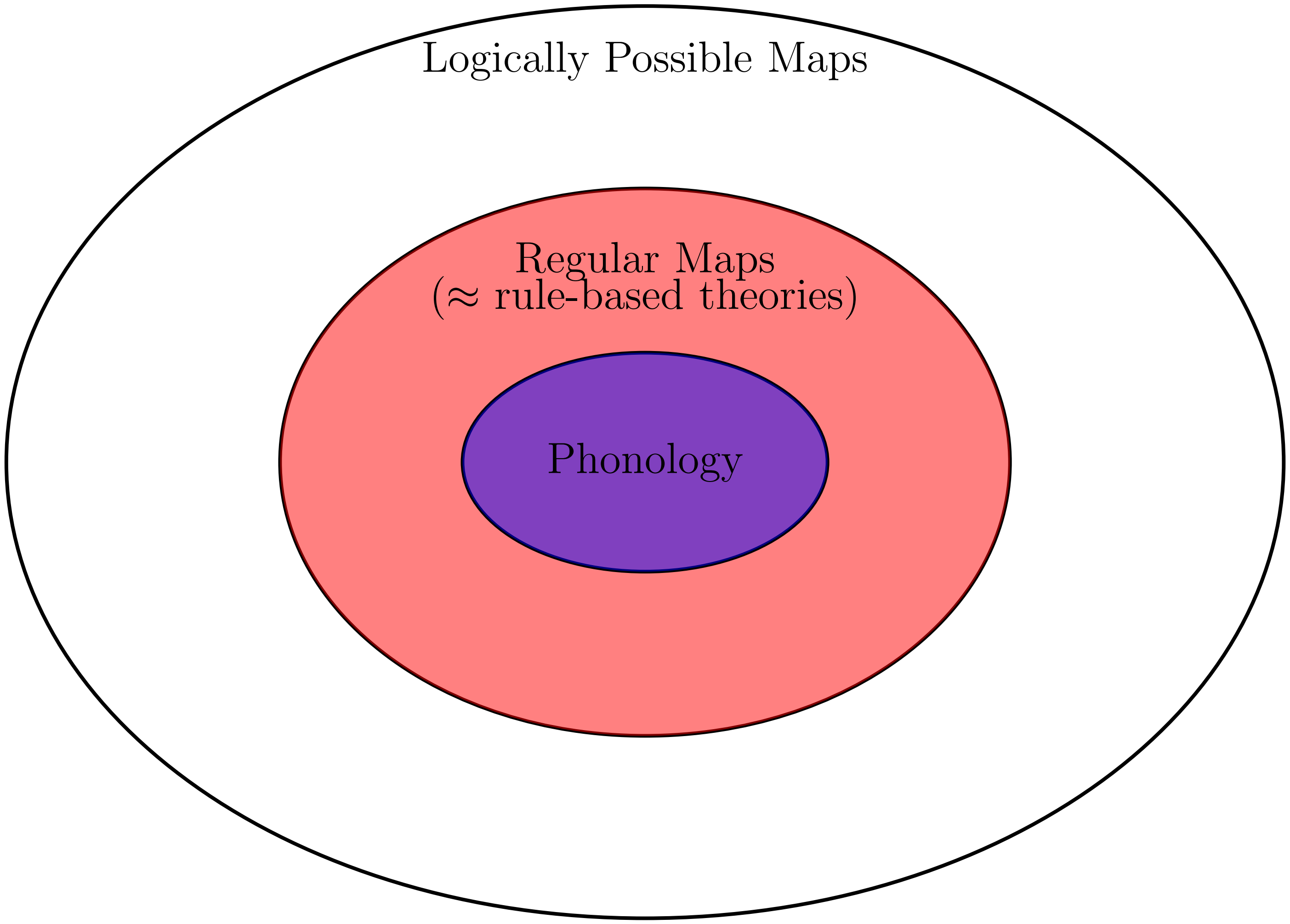
1. Different grammars may generate the same transformation.

   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)
   - regular functions (Elgot and Mezei 1956, Scott and Rabin 1959)
   - subsequential functions (Oncina et al. 1993, Mohri 1997, Heinz and Lai 2013)



1. Rule-based grammars were shown to be extensionally equivalent to regular transductions (Johnson 1972, Kaplan and Kay 1994).
2. Some argued they overgenerated and nobody knew how to learn them.

Part III

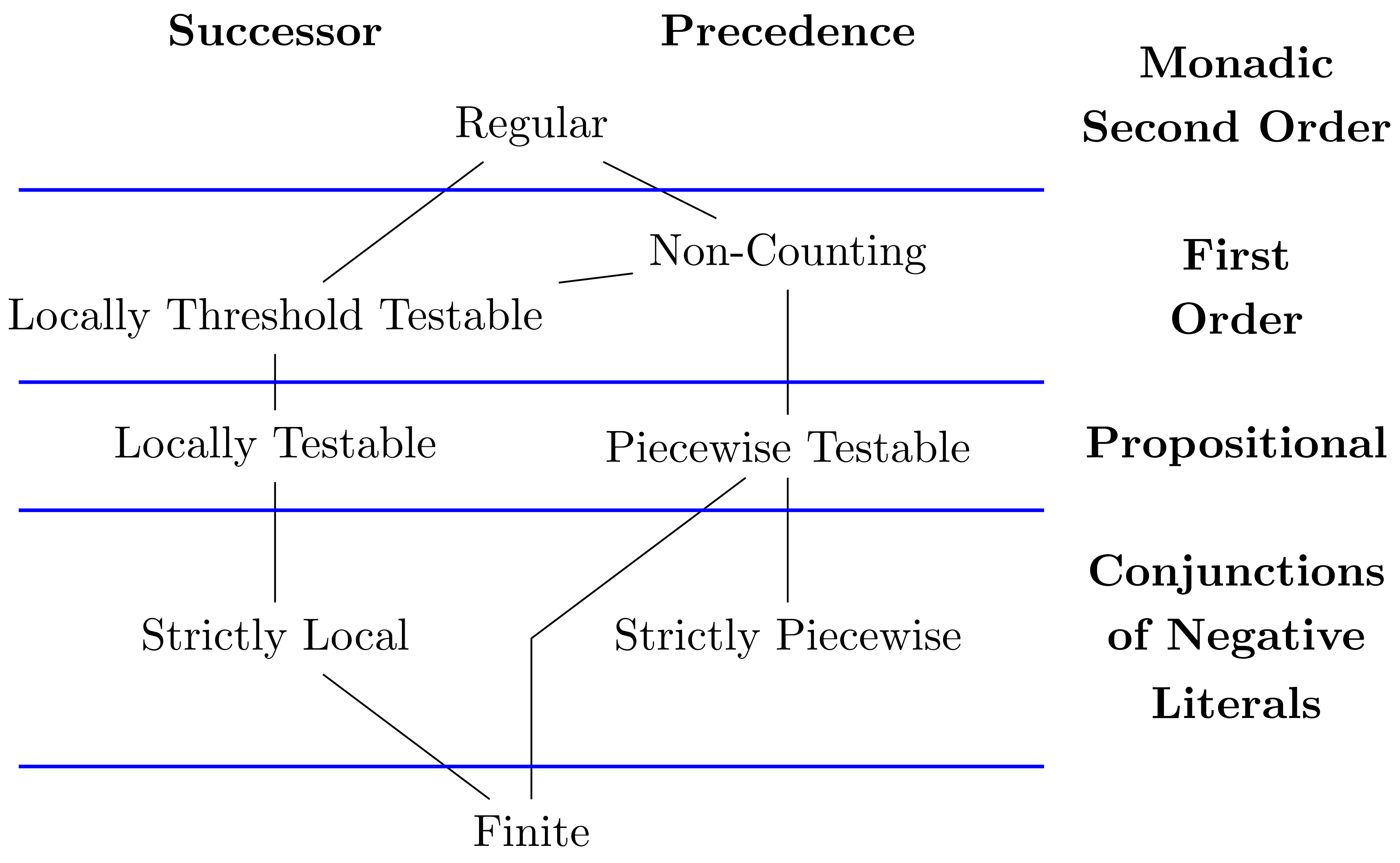
Analytical Framework

Computation is reflected in logical power

Subregular hierarchies organize pattern complexity along two dimensions.

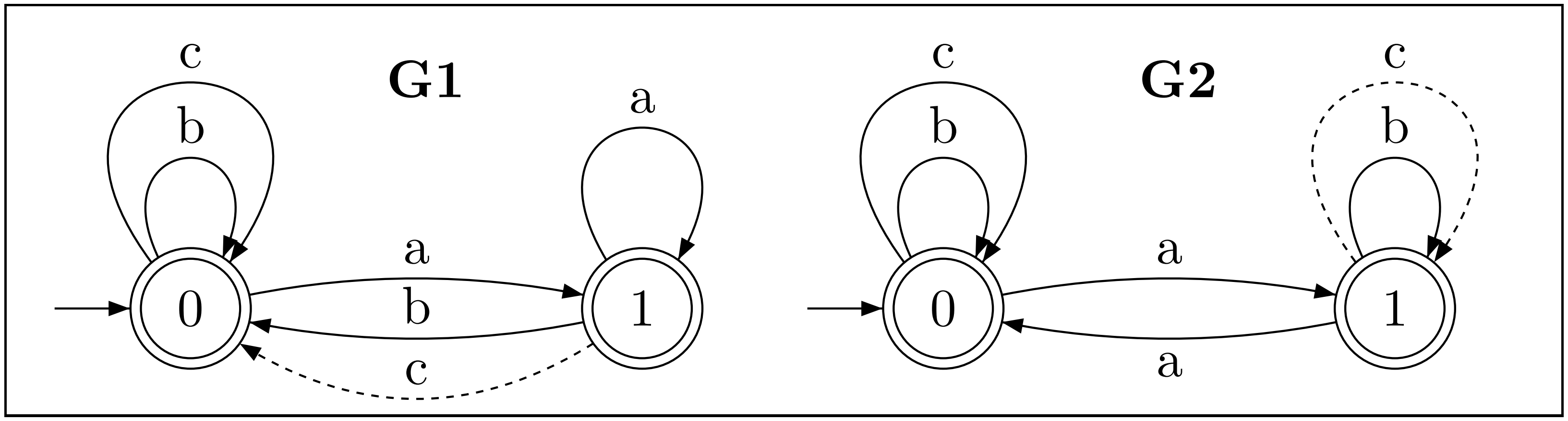
- **logical power** along the vertical axis
- **representational primitives** along the horizontal axis.

Logical Characterizations of Subregular Stringsets



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

Size of automata $\propto$ complexity? No.



- G1 maintains a short term memory w.r.t. [a] (i.e. State 1 means "just observed [a]").
- G2 maintains a memory of the even/odd parity of [a]s (i.e. State 1 means "observed an even number of [a]s").
- If dashed transitions are omitted, then G1 generates/recognizes all words except those with a forbidden string [ac]; and G3 generates/recognizes all words except those with a [c] whose left context contains an even number of [a]s. G1 is Strictly 2-Local, and G3 is Counting.

Finite-state automata are a low-level language

Automata can serve as a *lingua franca* because different grammars can be translated into them.

$$\begin{tikzcd}
 & & \text{RULE GRAMMARS} \arrow[dl, shift left, "**"] \\
\text{MSO FORMULA} \arrow[r, shift right] & \text{AUTOMATA} \arrow[l, shift right, "*"'] \arrow[ur, shift left] & \\
 & & \text{OT GRAMMARS} \arrow[ul, "***"]
\end{tikzcd}$$

*Büchi 1960.

**Johnson 1972, Kaplan and Kay 1994, Beesley and Karttunen 2003.

***Under certain conditions (Frank and Satta 1998, Kartunnen 1998, Gerdemann and van Noord 2000, Riggle 2004, Gerdemann and Hulden 2012)

Logic as a high-level language

1. Logical formulae over relational structures (model theory) provide a high-level description language (which are easy to learn to write—even for whole grammars).

2. We argue these levels of complexity yield hypotheses characterizing phonology that provide
   (a) a better fit to the typology than optimization,
   (b) have learning results that are as good or better than in OT,
   (c) provide equally good or better insights,
   (d) and are effectively computable.

Part IV

Input Strictly Local Functions

Input Strict Local *Transformations*

This is a class of transformations which. . .

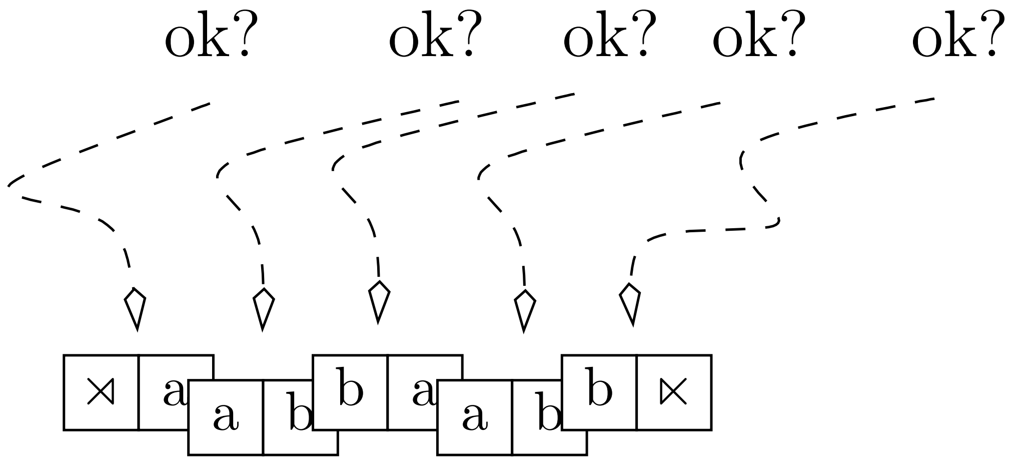
1. generalizes Strictly Local Stringsets,
2. captures a wide range of phonological phenomena,
3. including *opaque* transformations,
4. and is effectively learnable!

(Chandlee 2014, Chandlee and Heinz, under revision)

Strictly Local constraints for strings

When words are represented as strings, **local sub-structures are sub-strings** of a certain size.

Here is the string *abab*. If we fix a diameter of 2, we have to check these substrings.



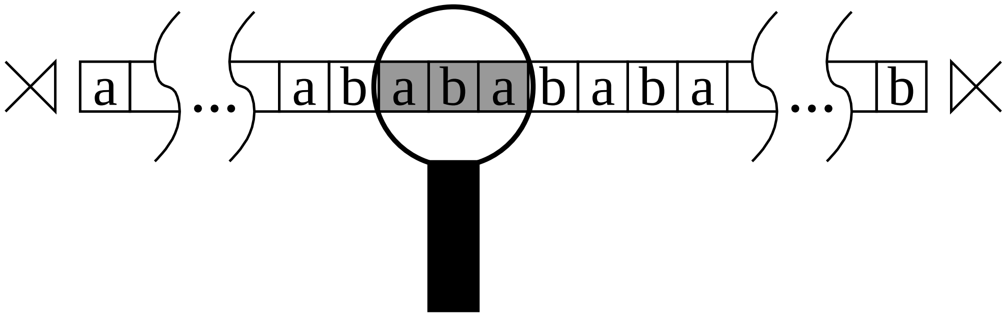
An ill-formed sub-structure is **forbidden**.

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

Strictly Local constraints for strings

When words are represented as strings, **local sub-structures are sub-strings** of a certain size.

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

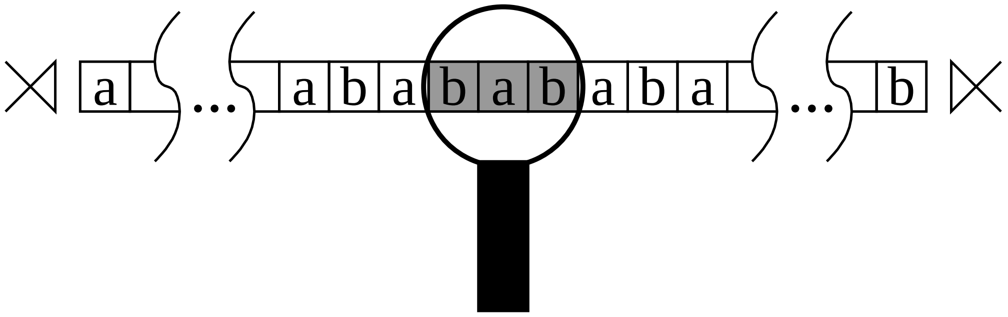


(Rogers and Pullum 2011, Rogers et al. 2013)

Strictly Local constraints for strings

When words are represented as strings, **local sub-structures are sub-strings** of a certain size.

- We can imagine examining each of the local-substructures, checking to see if it is well-formed. The whole structure is well-formed only if each local sub-structure is.

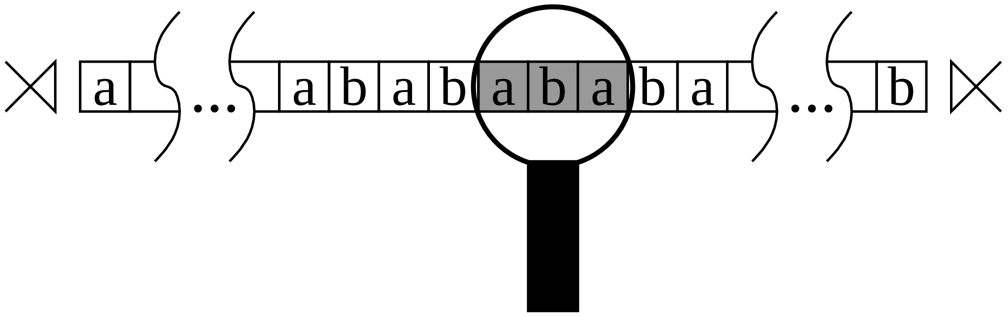


(Rogers and Pullum 2011, Rogers et al. 2013)

Strictly Local constraints for strings

When words are represented as strings, **local sub-structures are sub-strings** of a certain size.

- We can imagine examining each of the local-substructures, checking to see if it is well-formed. The whole structure is well-formed only if each local sub-structure is.



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

Examples of Strictly Local constraints for strings

- *aa
- *ab
- *N$\mathring{\text{C}}$
- NoCoda

Examples of Non-Strictly Local constraints

- *s. . . ʃ (Hansson 2001, Rose and Walker 2004, Hansson 2010, inter alia)
- *#s. . . ʃ# (Lai 2012, to appear, LI)
- Obligatoriness: Words must contain one primary stress (Hayes 1995, Hyman 2011, inter alia).

Input Strict Local *Transformations*

This is a class of transformations which...

1. generalizes Strictly Local Stringsets,
2. captures a wide range of phonological phenomena,
3. including *opaque* transformations,
4. and is effectively learnable!

Input Strict Locality: Main Idea

(Chandlee 2014, Chandlee and Heinz, under revison)

These transformations are Markovian in nature.

$$x_0 \; x_1 \; \dots \; x_n$$

$$\downarrow$$

$$u_0 \; u_1 \; \dots \; u_n$$

where

1. Each x_i is a single symbol ($x_i \in \Sigma_1$)
2. Each u_i is a *string* ($u_i \in \Sigma_2^*$)
3. 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.

(so u_i is a function of $x_{i-k+1}x_{i-k+2}\dots x_i$)

Input Strict Locality: Main Idea in a Picture

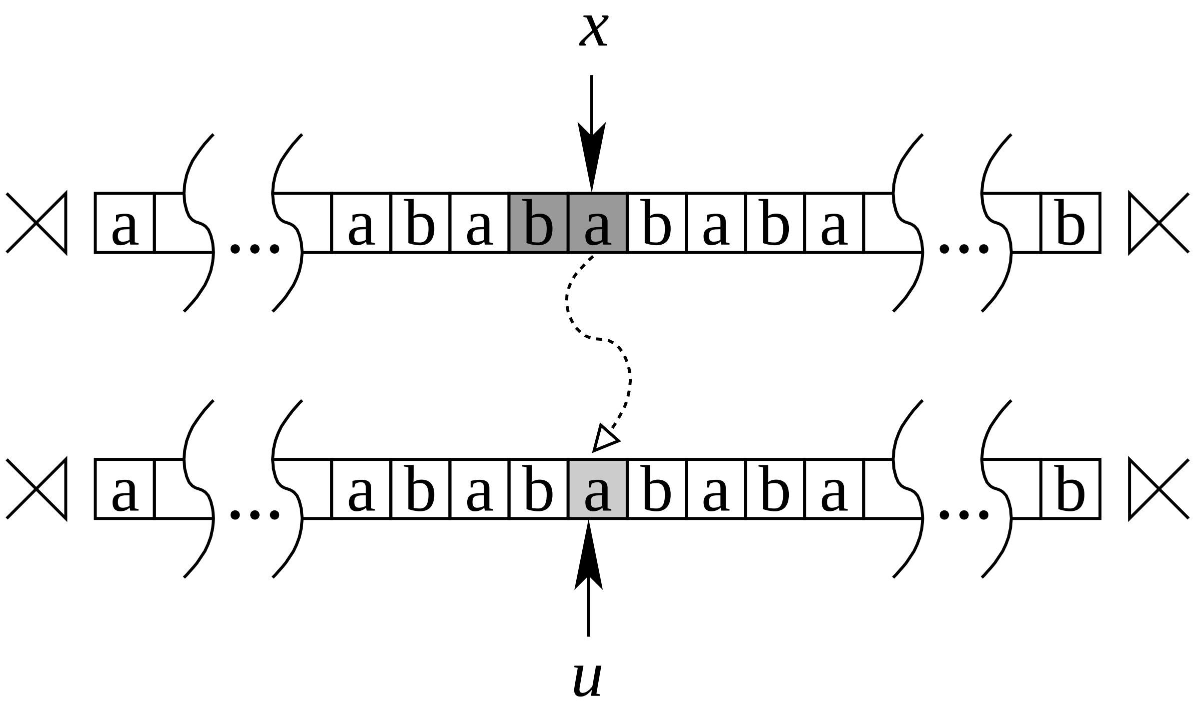

x
⋊ a ... a b a b a b a b a ... b ⋉
⋊ a ... a b a b a b a b a ... b ⋉
u

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 $k = 2$

/ove/ $\mapsto$ [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 $k = 2$

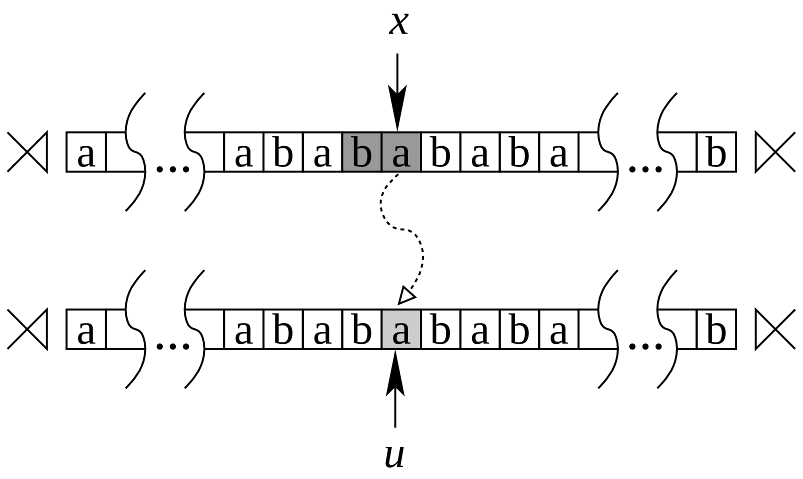
$$/\text{ove}/ \mapsto [\text{ovi}]$$

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

What this means, generally.

The *necessary information* to decide the output is contained within a *window of bounded length* on the *input* side.

- This property is largely independent of whether we describe the transformation with constraint-based grammars, optimization-based grammars, rule-based grammars, or other kinds of grammars.


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

How does this relate to traditional phonological grammatical concepts?

1. Like OT, k-ISL functions do not make use of intermediate representations.
2. Like OT, k-ISL functions separate marked structures from their repairs (Chandlee et al. to appear, AMP 2014).
   - k-ISL functions are sensitive to all and only those markedness constraints which could be expressed as $*x_1x_2 \ldots x_k$, $(x_i \in \Sigma)$. (So Strictly k-Local markedness constraints)
   - In this way, k-ISL functions model the "homogeneity of target, heterogeneity of process" (McCarthy 2002)

Part IV

Learning ISL functions

Results in a nutshell

- Particular finite-state transducers can be used to represent ISL functions.
- Grammatical inference techniques (de la Higuera 2010) are used for learning.
- **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)

Comparison of learning results in classic OT

- Recursive Constraint Demotion (RCD) is guaranteed to give you a consistent grammar in reasonable time.
- Exact convergence is not guaranteed for RCD because the nature of the data sample needed for exact convergence is not yet known.
- On the other hand, we are able to characterize a sample which yields exact convergence.

Part V

ISL Functions and Phonological Typology

What can be modeled with ISL functions?

1. Many individual phonological processes.

   (local substitution, deletion, epenthesis, and 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, in revision)

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, in revision)

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. 2015, GALANA & GLOW workshop on computational phonology)

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/ $\mapsto$ [ʔiliːl]

input:	$\rtimes$	ʔ	i	l	iː	l	$\ltimes$
output:	$\rtimes$	ʔ	i	l	λ	λ	el $\ltimes$

Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiːlil/ ↦ [ʔilel]

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

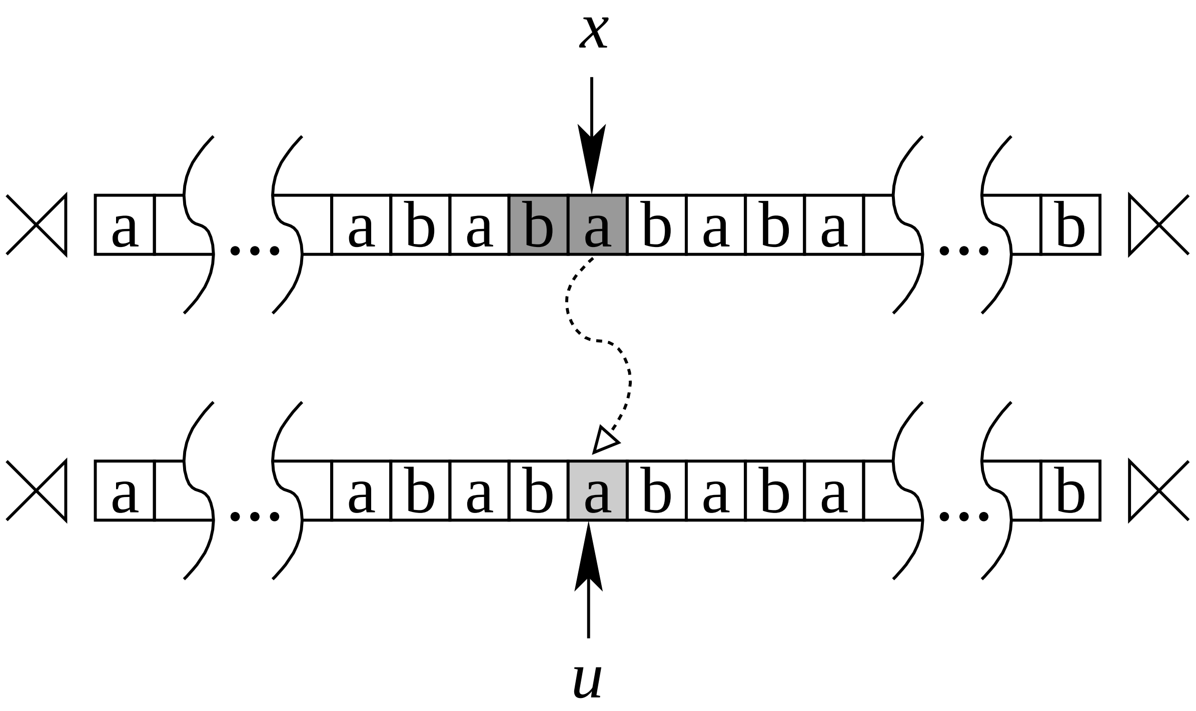
Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ $\mapsto$ [ʔilel]

input:	$\rtimes$	ʔ	i	l	iː	l	$\ltimes$
output:	$\rtimes$	ʔ	i	l	λ	λ	el $\ltimes$

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.


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

And can thus be learned by the ISLFLA and SOSFIA algorithms!

What CANNOT be modeled with ISL functions

1. progressive and regressive spreading
2. long-distance (unbounded) consonant and vowel harmony
3. non-*regular* transformations like Majority Rules vowel harmony and non-*subsequential* transformations like Sour Grapes vowel harmony (Baković 2000, Finley 2008, Heinz and Lai 2013)

(Chandlee 2014, Chandlee and Heinz, in revision)

Undergeneration? Yes, for now...

- ISL functions are insufficiently expressive for spreading and long-distance harmony. I will discuss these later (or in Q&A).

Overgeneration? Not so much.

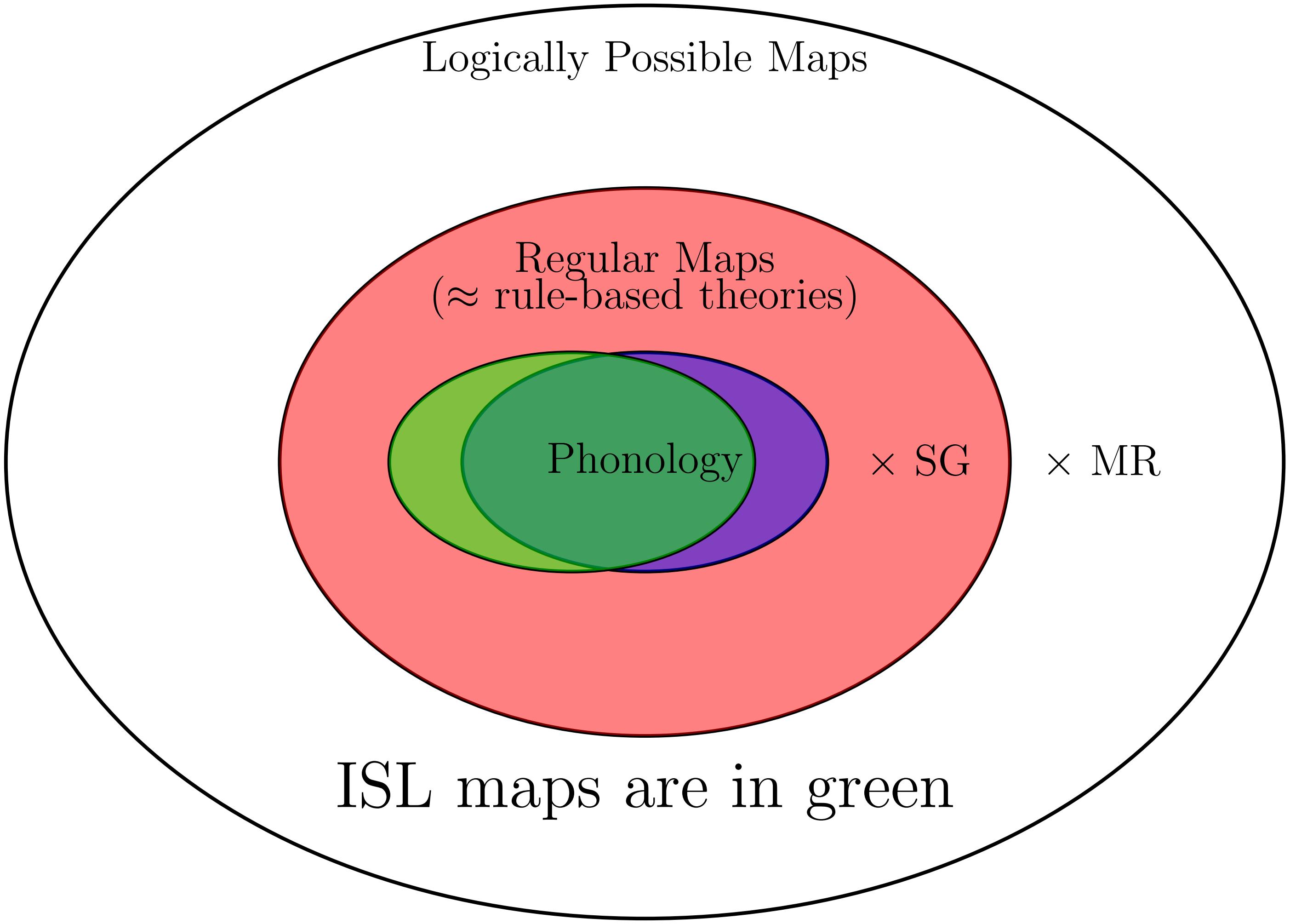
- **Theorem:** ISL is a proper subclass of left and right subsequential functions.

  (Chandlee 2014, Chandlee et al. 2014)

- **Corollary:** SG and MR are not ISL for any k.

  (Heinz and Lai 2013)

- So MR and SG are correctly predicted to be outside the typology.


Logically Possible Maps
Regular Maps
(≈ rule-based theories)
Phonology
× SG
× MR
ISL maps are in green

Undergeneration in Classic OT

- It is well-known that classic OT cannot generate opaque maps (Idsardi 1998, 2000, McCarthy 2007, Buccola 2013)

  (though Baković 2007, 2011 argues for a more nuanced view).

- Many, many adjustments to classic OT have been proposed.
  - constraint conjunction (Smolensky), sympathy theory (McCarthy), turbidity theory (Goldrick), output-to-output representations (Benua), stratal OT (Kiparsky, Bermudez-Otero), candidate chains (McCarthy), harmonic serialism (McCarthy), targeted constraints (Wilson), contrast preservation (Łubowicz) comparative markedness (McCarthy) serial markedness reduction (Jarosz), . . .

See McCarthy 2007, *Hidden Generalizations* for review, meta-analysis, and more references to these earlier attempts.

Adjustments to Classic OT

The aforemetioned approaches invoke different representational schemes, constraint types and/or architectural changes to classic OT.

- The typological and learnability ramifications of these changes is not yet well-understood in many cases.
- On the other hand, *no special modifications are needed* to establish the ISL nature of the opaque maps we have studied.

Overgeneration in Classic OT

- It is not controversial that classic OT generates non-regular maps with simple constraints (Frank and Satta 1998, Riggle 2004, Gerdemann and Hulden 2012, Heinz and Lai 2013)

  (Majority Rules vowel harmony is one example.)

Simple constraints in OT generate non-regular maps

Ident, Dep >> *ab >> Max

$$a^n b^m \mapsto a^n, \text{ if } m < n$$

$$a^n b^m \mapsto b^m, \text{ if } n < m$$

(Gerdemann and Hulden 2012)

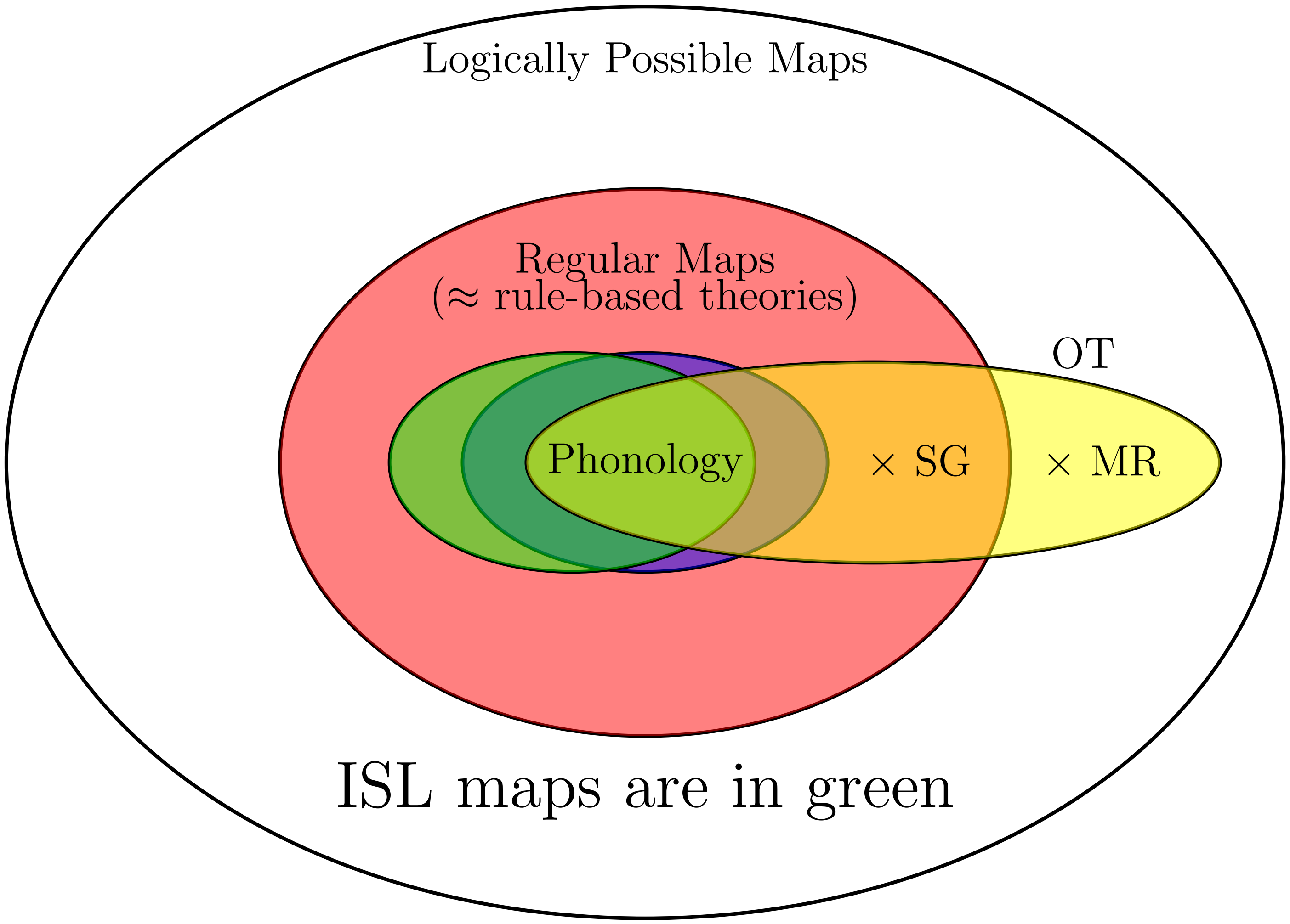
Optimization misses an important generalization

- When computing the output of phonological transformations, the *necessary information* is contained within *sub-structures of bounded size*.
- This is neither expected nor predicted under global optimization.
- On the other hand, it is one of the defining characteristics of k-ISL.

OT's greatest strength is its greatest weakness.

- The signature success of a successful OT analysis is when complex phenomena are understood as the interaction of simple constraints.
- But the overgeneration problem is precisely this problem: complex—but weird—phenomena resulting from the interaction of simple constraints (e.g. Hansson 2007, Hansson and McMullin 2014, on ABC).
- As for the undergeneration problem, opaque candidates are not optimal in classic OT.

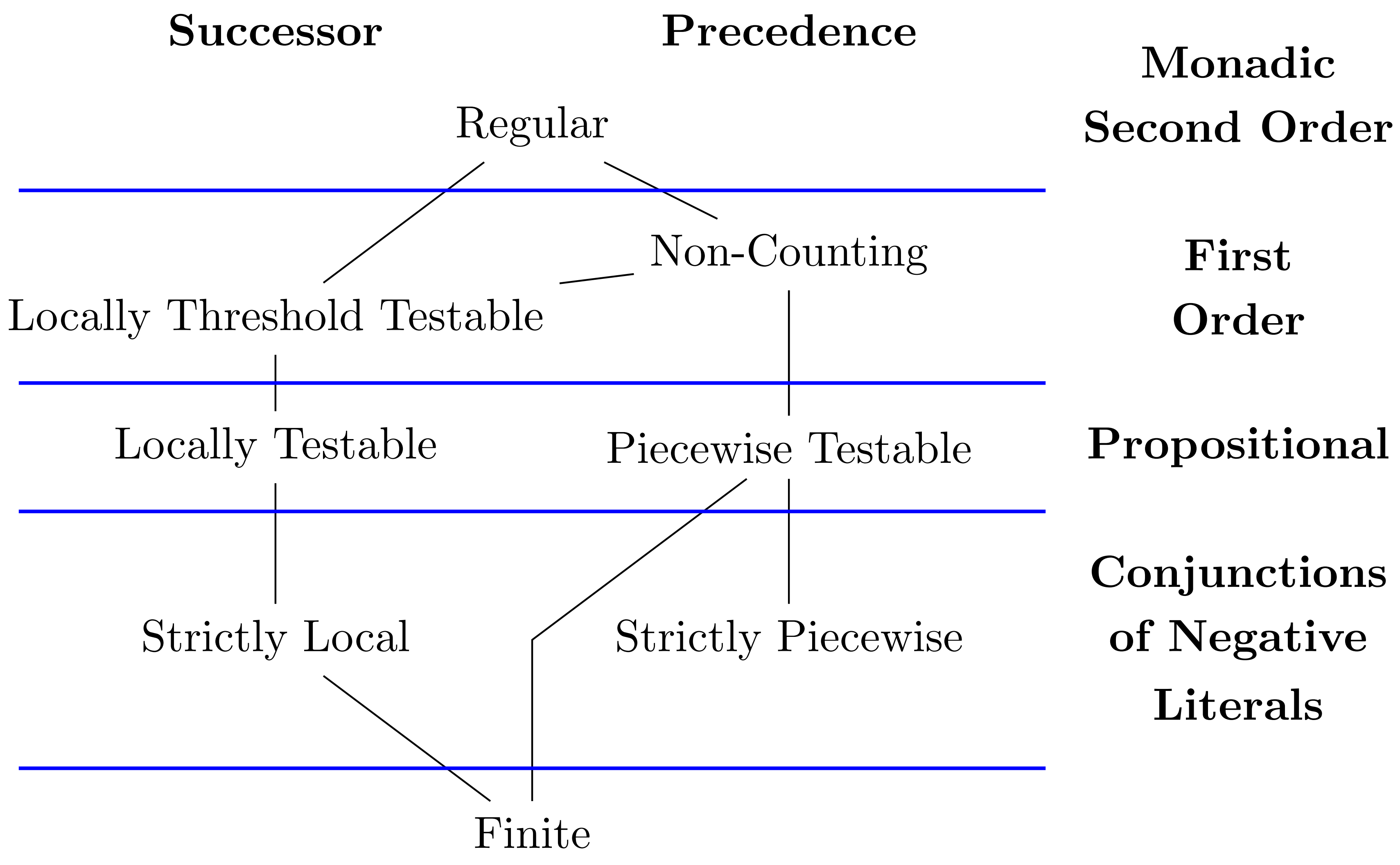
Comparing ISL with classic OT w.r.t. typology



Part VI

Conclusion

Logical Characterizations of Subregular Stringsets



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

Some conclusions

- k-ISL functions provide both a more expressive and restrictive theory of typology than classic OT, which we argue **better matches** the attested typology.
  - In particular: Many phonological transformations, **including opaque ones**, can be expressed with them, but non-subsequential transformations cannot be.
- k-ISL functions are feasibly learnable.
- Like classic OT, there are no intermediate representations, and k-ISL functions can express the "homogeneity of target, heterogeneity of process" which helps address the conspiracy and duplication problems.
- Unlike OT, *subregular computational properties* like ISL—and not optimization—form the core computational nature of phonology.

Questions and Thanks

† We thank Iman Albadr, Joe Dolatian, Rob Goedemans, Hyun Jin Hwangbo, Cesar Koirala, Regine Lai, Huan Luo, Kevin McMullin, Taylor Miller, Amanda Payne, Curt Sebastian, Kristina Strother-Garcia, Bert Tanner, Harry van der Hulst, Irene Vogel and Mai Ha Vu for valuable discussion and feedback.

We also thank the organizers of the “Whither OT?” workshop.

EPILOGUE

(EXTRA SLIDES)

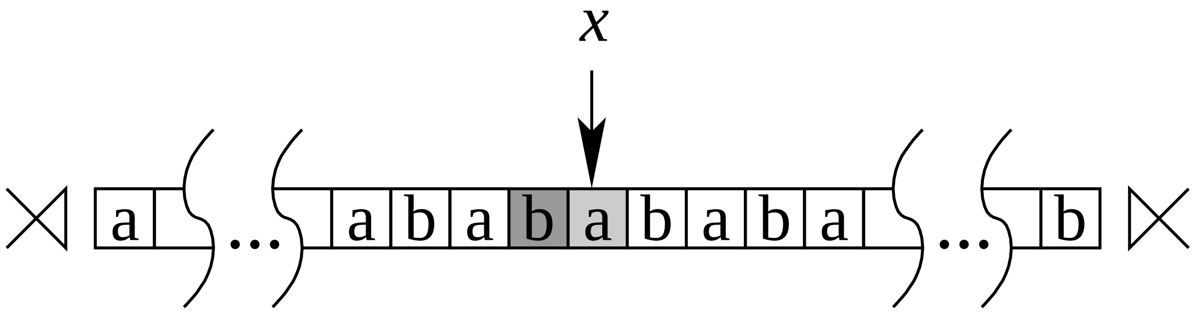
- What about spreading and long-distance phonology?
- How do I write a grammar?
- More examples of ISL, please.
- How do the learning algorithms work exactly?
- ...

QUESTION

Well, what about long-distance phonology?

Formal Language Theory

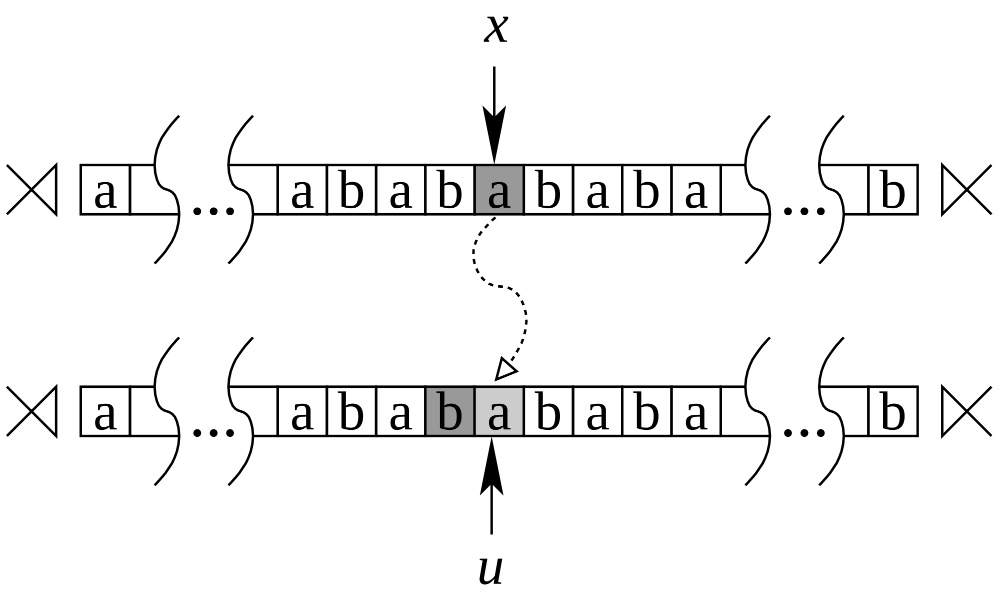
- ISL functions naturally extend Strictly Local (SL) stringsets in Formal Language Theory.
- For SL stringsets, well-formedness can be decided by examining windows of size k.
- SL stringsets are the extensions of *local phonotactic constraints* (Heinz 2010, Rogers et al. 2013)



(McNaughton and Papert 1971, Rogers and Pullum 2011)

What about spreading?

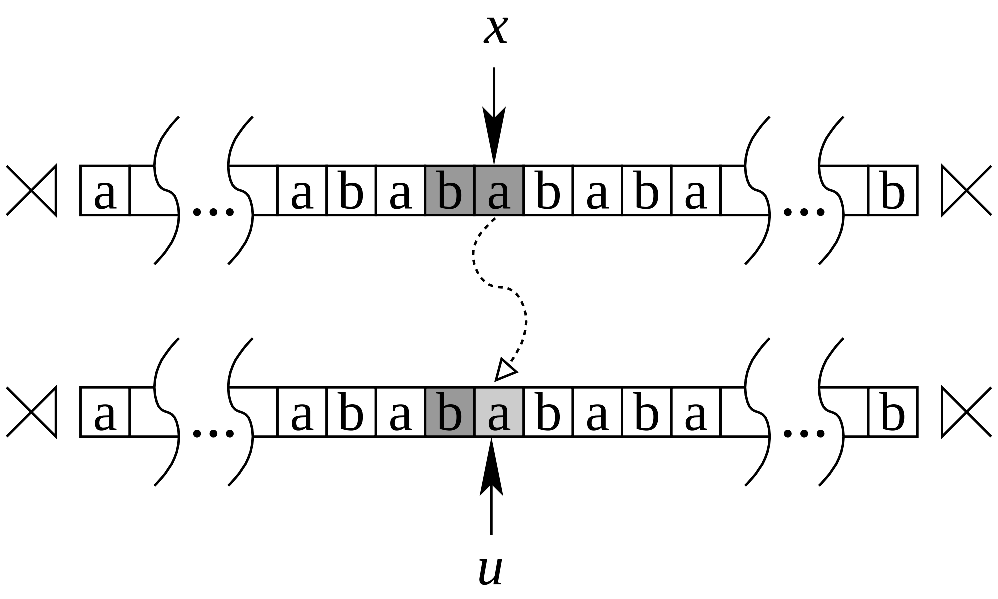
- *Left (and Right) Output* SL functions are other generalizations of SL stringsets which model precisely progressive and regressive spreading (Chandlee et al., MOL 2015).



Unfortunately, these OSL functions cannot model transformations with two-sided contexts.

Input-Output Strictly Local functions

Ultimately, we need a way to combine ISL and OSL. The combination will not be functional composition, but a hybrid (Chandlee, Eyraud and Heinz, work in progress).



- We expect this will be exactly the right computational notion of locality in phonological transformations.
- IOSL transformations will still not describe long-distance phenomena.

Long-distance transformations

- Strictly-Piecewise (SP) and Tier-based Strictly Local (TSL) stringsets model *long-distance phonotactics* (Heinz 2010, Heinz et al. 2011).
- The logical power is the same as for SL stringsets but *representations of words* are different.
  - SL stringsets model words with the successor relation.
  - SP stringsets model words with the precedence relation.
  - TSL stringsets model words with order relations among elements common to a phonological tier (cf. ABC).
  - SL, SP, and TSL each ban sub-structures, but the sub-structures themselves are different.
- We expect functional characterizations of SP and TSL stringsets will model long-distance maps (work-in-progress).

More word representations (expanding the horizontal axis. . .)

- Adam Jardine examines the implications of this way of thinking for richer word models used in phonology, such as autosegmental representations.

  (Jardine 2014 AMP, Jardine 2014 NECPHON, Jardine and Heinz 2015 CLS, Jardine and Heinz, MOL 2015)

- For instance under certain conditions, the No Crossing Constraint and the Obligatory Contour Principle can be obtained by banning sub-structures of autosegmental representations (so they are like SL in this respect).

QUESTION

Well, how am I supposed to write a grammar?

Use logical formula. Example: Finnish /e/ raising

Here is how we can express in first-order logic which elements in the output string are [+high].

$$\begin{aligned}\varphi_{\texttt{high}}(x) \quad \overset{\text{def}}{=} \quad & \texttt{high}(x) \vee \\ & \Big(\texttt{front}(x) \wedge \texttt{nonlow}(x) \wedge \texttt{nonround}(x) \ \wedge \\ & \ (\exists y)[\texttt{after}(x, y) \wedge \texttt{boundary}(y)]\Big)\end{aligned}$$

Essentially, this reads as follows: "Element x in the output string will be [+high] only if its corresponding x in the input string is either [+high] or /e/ followed by a word boundary."

(See also Potts and Pullum 2002)

Why logical formula?

1. They are a high-level language.
   (a) They are very expressive.
   (b) They are precise.
   (c) They are easy to learn with only a little practice.

   (I would argue in each of these respects they are superior to rule-based and constraint-based grammars).
2. Linguists can use (and systematically explore) different representational primitives like features, syllables, etc.
3. They can be translated **to and from** finite-state automata (Büchi 1960).

Finite-state automata are a low-level language

Automata can serve as a *lingua franca* because different grammars can be translated into them and then equivalence can be checked.

$$\begin{tikzcd}
 & & \text{RULE GRAMMARS} \arrow[dll, shift left, "**"] \\
\text{MSO FORMULA} \arrow[r, shift right] & \text{AUTOMATA} \arrow[l, shift right, "*"'] \arrow[ur, shift left] & \\
 & & \text{OT GRAMMARS} \arrow[ul, "***"]
\end{tikzcd}$$

*Büchi 1960.

**Johnson 1972, Kaplan and Kay 1994, Beesley and Karttunen 2003.

***Under certain conditions (Frank and Satta 1998, Kartunnen 1998, Gerdemann and van Noord 2000, Riggle 2004, Gerdemann and Hulden 2012)

Workflow

My idea is this:

1. The phonologist writes in the high-level logical language exactly the grammar they want, using the representational primitives they think are important.
2. This can be automatically translated (compiled) into a low-level language (like automata) for examination.
3. Algorithms can process the automata and determine whether the generalization is ISL, OSL, IOSL, or possesses some other subregular property.

QUESTION

Can I see more examples of Input Strictly Local transformations?

Example: Post-nasal voicing

$$/\text{imka}/ \mapsto [\text{imga}]$$

input:	⋊	i	m	k	a	⋉
output:	⋊	i	m	g	a	⋉

Left triggers are more intuitive.

Example: Post-nasal voicing

$$/\text{imka}/ \mapsto [\text{imga}]$$

input:	⋊	i	m	k	a	⋉
output:	⋊	i	m	g	a	⋉

Left triggers are more intuitive.

Example: Post-nasal voicing

/imka/ $\mapsto$ [imga]

input:	$\rtimes$	i	m	k	a	$\ltimes$
output:	$\rtimes$	i	m	g	a	$\ltimes$

Left triggers are more intuitive.

Example: Intervocalic Spirantization

/pika/ $\mapsto$ [pixa] and /pik/ $\mapsto$ [pik]

input:	$\rtimes$	p	i	k	a	$\ltimes$
output:	$\rtimes$	p	i	λ	xa	$\ltimes$

But if there is a right context, the ‘empty string trick’ is useful to see it is ISL.

Example: Intervocalic Spirantization

/pika/ $\mapsto$ [pixa] and /pik/ $\mapsto$ [pik]

input:	$\rtimes$	p	i	k	a	$\ltimes$
output:	$\rtimes$	p	i	λ	xa	$\ltimes$

But if there is a right context, the ‘empty string trick’ is useful to see it is ISL.

Example: Intervocalic Spirantization

/pika/ $\mapsto$ [pixa] and /pik/ $\mapsto$ [pik]

input:	$\rtimes$	p	i	k	$\ltimes$
output:	$\rtimes$	p	i	λ	k $\ltimes$

But if there is a right context, the ‘empty string trick’ is useful to see it is ISL.

QUESTION

What is the automata characterization of Input k-Strictly Local transformations?

Automata characterization of k-ISL functions

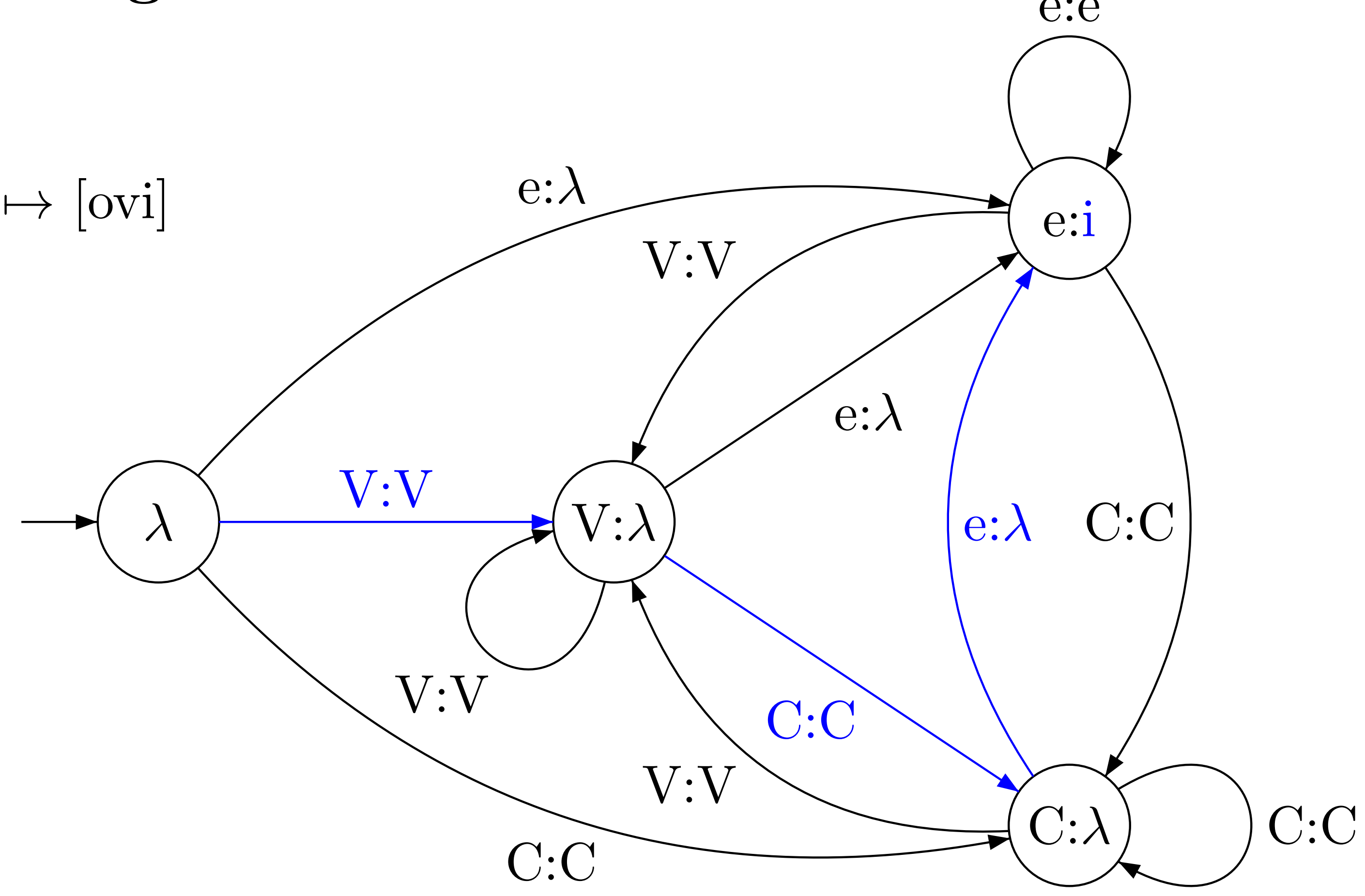
Theorem Every k-ISL function can be modeled by a *k-ISL transducer* and every *k-ISL transducer* represents a k-ISL function.

The state space and transitions of these transducers are organized such that two input strings with the same $k-1$ suffix always lead to the same state.

(Chandlee 2014, Chandlee et. al 2014)

Example: Fragment of 2-ISL transducer for /e/ raising

/ove/ ↦ [ovi]



V represents any vowel that is not /e/ and C any consonant. So the diagram collapses states and transitions. The nodes are labeled `name:output_string`.

QUESTION

How do the learning algorithms work?

ISLFLA: Input Strictly Local Function Learning Algorithm

- The input to the algorithm is k and a finite set of (u, s) pairs.
- ISLFLA builds a input prefix tree transducer and merges states that share the same $k - 1$ prefix.
- Provided the sample data is of sufficient quality, ISLFLA provably learns any function k-ISL function in quadratic time.
- Sufficient data samples are quadratic in the size of the target function.

(Chandlee et al. 2014, TACL)

SOSFIA: Structured Onward Subsequential Function Inference Algorithm

SOSFIA takes advantage of the fact that every k-ISL function can be represented by an *onward* transducer with the *same* structure (states and transitions).

- Thus the input to the algorithm is k-ISL transducer with empty output transitions, and a finite set of (u, s) pairs.
- SOSFIA calculates the outputs of each transition by examining the longest common prefixes of the outputs of prefixes of the input strings in the sample (onwardness).
- Provided the sample data is of sufficient quality, SOSFIA provably learns any function k-ISL function in *linear* time.
- Sufficient data samples are *linear* in the size of the target function.

(Jardine et al. 2014, ICGI)