Locality in Phonological Interactions

Jane Chandlee

Haverford College

Rutgers University
April 28, 2017

Introduction

- Central goal: to understand the computational nature of phonological maps
  - Why?
    - Typological predictions
    - Implications for learning

Previous result

- Phonological maps describable with A $\rightarrow$ B / C __ D are *regular relations* provided they do not re-apply to their own structural change (Johnson 1972, Koskenniemi 1983, Kaplan & Kay 1994)

(1) $\emptyset \rightarrow$ ab / __ b

  - ab $\mapsto$ aabb $\mapsto$ aaabbb $\mapsto$ ... $a^n b^n$
  - ab $\mapsto$ aabb $\mapsto$ aababb $\mapsto$ ... $a(ab)^+b$

- Such maps are then *finite state describable*.

Previous result

(2) $A \rightarrow B \,/\, C \_\_ D$

- A, C, D are regular languages, which admits maps such as:

  (3) Nasalize a vowel after an even number of nasal consonants.

- In addition, the regular relations are not learnable from positive data in the sense of Gold (1967).

Proposal

- Phonological maps in fact belong to *subregular* classes of functions.
- Specifically, maps with *local triggers* are *Input Strictly Local (ISL) functions*.
- Maps with multiple generalizations, including opaque maps, also have the property of Input Strict Locality.

Outline

- What is ISL?
  - Finite state characterization
  - Language-theoretic characterization
- What kinds of maps are ISL?
  - Maps with local triggers (i.e., CAD is finite)
  - Map interactions, including opaque ones
  - Some morphological maps
- Bigger picture: subregular hierarchy of functions

Phonological maps

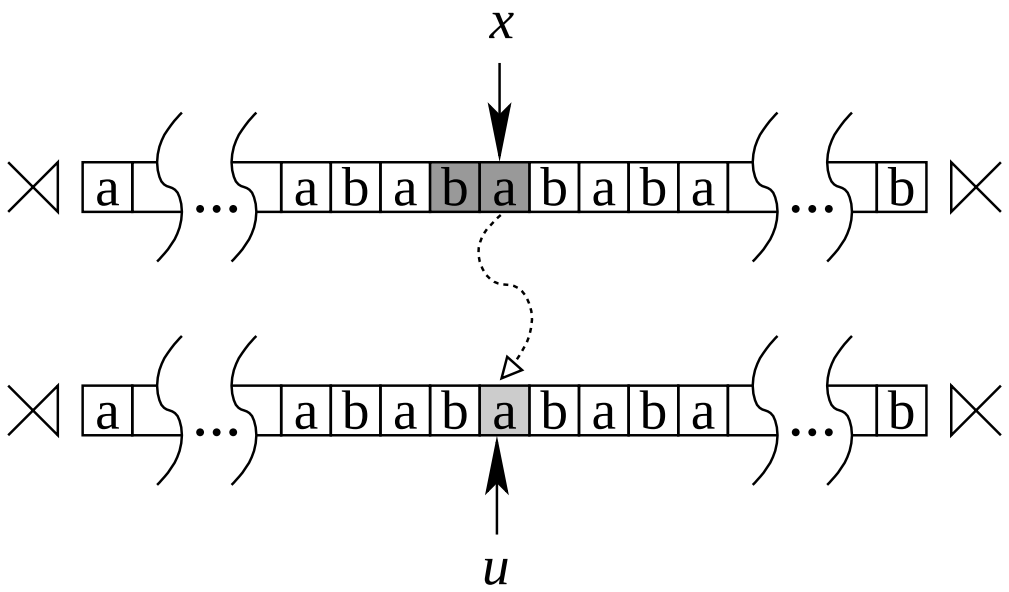
- A relation/function/map is a set of string *pairs*.
- A phonological map (i.e., 'process') is a (UR, SR) pair.

(4) $n \rightarrow m \; / \; \_\_ \; b$

(5) *nb $\gg$ IDENT-PLACE (Baković 2013, see also Tesar 2014)

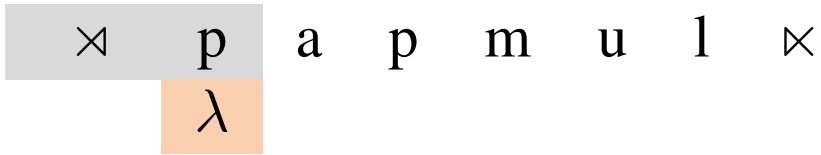
(6) { (anba, amba), (nanta, nanta), (ama, ama), ... }

Input Strictly Local (ISL) function



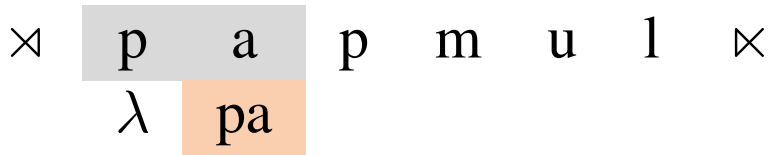
Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] 'rice water'



Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’



Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’

$\rtimes$	p	a	p	m	u	l	$\ltimes$
	λ	pa	λ				

Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] 'rice water'

$\rtimes$	p	a	p	m	u	l	$\ltimes$
	λ	pa	λ	mm			

Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’

⋊	p	a	p	m	u	l	⋉
	λ	pa	λ	mm	u		

Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’

⋊	p	a	p	m	u	l	⋉
	λ	pa	λ	mm	u	l	

Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’

$\rtimes$	p	a	p	m	u	l	$\ltimes$
	λ	pa	λ	mm	u	l	λ

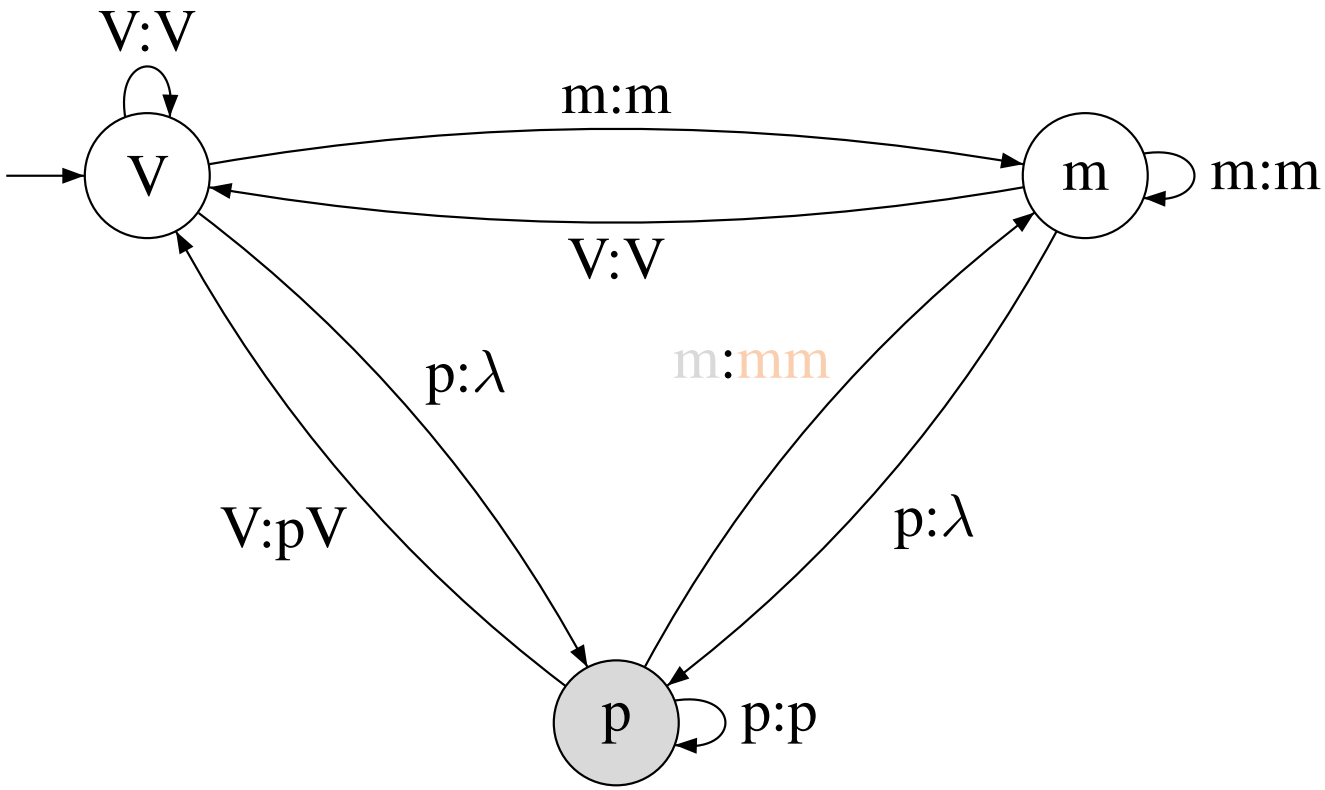
Example: Input Strictly Local (ISL) function

(7) Korean (Lee and Pater 2008)
/pa**pm**ul/ $\mapsto$ [pa**mm**ul] ‘rice water’

Window size is 2: this map is 2-ISL

Input Strictly Local (ISL): FST characterization

(8) /pa**pm**ul/ ↦ [pa**mm**ul]



⋊	p	a	p	m	u	l	⋉
	λ	pa	λ	mm			

Input Strictly Local (ISL): FST characterization

- A k-ISL FST
  - is deterministic on the input
  - has a state for each alphabet sequence up to length $k-1$
  - is always in the state that represents the most 'recent' $(k-1)$ segments of the input

Input Strictly Local (ISL): language-theoretic characterization

- Two input strings with the same $k-1$ suffix have the same tails.

(9) mp $\mapsto$ mb

$k-1$ suffix:	a**m**, pa**m**, mapa**m**, bamapa**m**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	a**p**, ba**p**, maba**p**, pamaba**p**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	ba**b**, baba**b**, mababa**b**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	b**a**, bab**a**, mabab**a**
tails:	(p, b), (b, b), (a, a), (m, m), ...

Input Strictly Local (ISL): language-theoretic characterization

$k-1$ suffix:	a**m**, pa**m**, mapa**m**, bamapa**m**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	a**p**, ba**p**, maba**p**, pamaba**p**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	ba**b**, baba**b**, mababa**b**
tails:	(p, b), (b, b), (a, a), (m, m), ...
$k-1$ suffix:	b**a**, bab**a**, mabab**a**
tails:	(p, b), (b, b), (a, a), (m, m), ...

Correspondence to FSTs: states are $k-1$ suffixes and tails are available paths from each state.

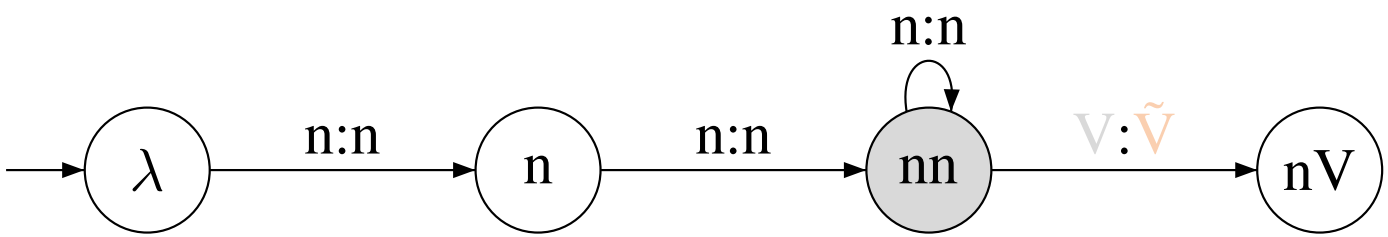
A non-ISL map

(9) Nasalize a vowel after an even number of nasal consonants.

- This map is not ISL...but let's see why.
- To start, if is is ISL, what is k?

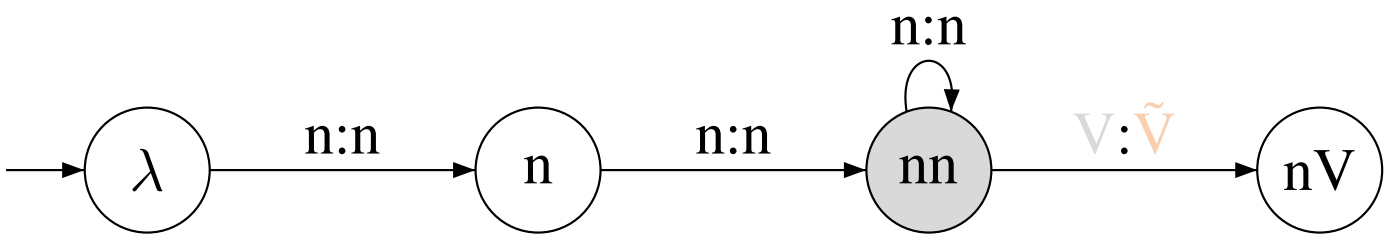
(10) $\text{nnV} \mapsto \text{nn}\tilde{\text{V}}$

A non-ISL map



⋊	n	n	V
	n	n	Ṽ

A non-ISL map



⋊	n	n	n	V
	n	n	n	Ṽ

A non-ISL map

- If it's a k-ISL function, two input strings with the same $k-1$ suffix must have the same tails.

(11) $\text{nnV} \mapsto \text{nn}\tilde{\text{V}}$

$k-1$ suffix:	V**nn**
tails:	(V, $\tilde{\text{V}}$), ...
$k-1$ suffix:	Vn**nn**
tails:	(V, V), ...

What maps are ISL?

(12) $A \rightarrow B \,/\, C \_\_ D$

- Conjecture: a map is ISL if CAD is finite (and the rule applies simultaneously)
- This includes 'bounded' assimilation, dissimilation, epenthesis, deletion, metathesis (Chandlee 2014, Chandlee and Heinz, to appear)

What maps are ISL?

(13) $A \rightarrow B \,/\, C \_\_ D$

- Conjecture: a map is ISL if CAD is finite (and the rule applies simultaneously)
- C = strings with an even number of consonants is an infinite regular language

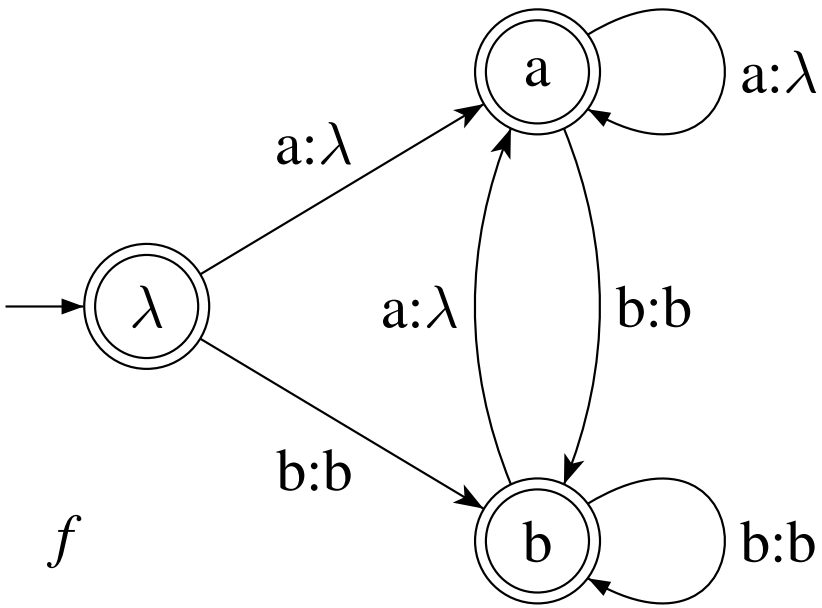
Advantages for learning

- ISL functions $\subset$ Regular relations
  - The restrictiveness of ISL enables efficient learning from positive data (Chandlee et al. 2014, Jardine et al. 2014)

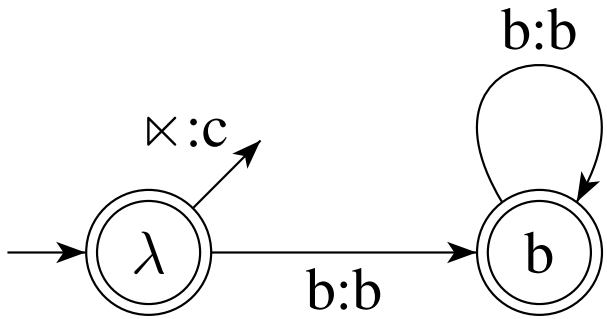
Closure properties

- The regular relations are closed under composition - are the ISL functions?
- As originally defined...no.

Closure properties



f



g

$$g \circ f(ba^k) = g(f(ba^k)) = g(b) = b$$
$$g \circ f(a^k) = g(f(a^k)) = g(\lambda) = c$$

Closure properties

- However, ISL functions without *null cycles* **are** closed under composition (Chandlee & Lindell, in prep).
- Excludes maps that delete arbitrarily long sequences:

(14) $a^*a \mapsto a$

Map interactions

What about map interactions?

Case study: opaque phonological generalizations

- Opaque phonological generalizations result from the interaction of two processes, one of which is not 'surface-true'.
- The 7 distinct types of opacity discussed by Bakovič (2007) and Kavitskaya & Staroverov (2010) can all be described with single ISL maps (Chandlee, Heinz, Jardine, under review).
- The needed information for determining the correct output at any point in the computation is found in a contiguous window of the input.

Opaque phonological maps

(15) Counterbleeding

a. Lowering: [+long] $\rightarrow$ [$-$high]

b. Shortening: V $\rightarrow$ [$-$long] / ___ C #

(16) Yokuts (McCarthy 1999)

/ʔili:l/ $\mapsto$ [ʔilel], 'might fan'

- Lowering over-applies.

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] $\rightarrow$ [$-$high]

b. Shortening: V $\rightarrow$ [$-$long] / __ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ $\mapsto$ [ʔilel], ‘might fan’

⋊	ʔ	i	l	i:	l	⋉
	ʔ					

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] → [−high]

b. Shortening: V → [−long] / __ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ ↦ [ʔilel], ‘might fan’

⋊	ʔ	i	l	i:	l	⋉
	ʔ	i				

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] → [−high]

b. Shortening: V → [−long] / ___ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ ↦ [ʔilel], ‘might fan’

⋊	ʔ	i	l	i:	l	⋉
	ʔ	i	l			

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] $\rightarrow$ [$-$high]

b. Shortening: V $\rightarrow$ [$-$long] / $\_\_$ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ $\mapsto$ [ʔilel], ‘might fan’

⋊	ʔ	i	l	i:	l	⋉
	ʔ	i	l	λ		

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] $\rightarrow$ [$-$high]

b. Shortening: V $\rightarrow$ [$-$long] / $\_\_$ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ $\mapsto$ [ʔilel], ‘might fan’

$\rtimes$	ʔ	i	l	i:	l	$\ltimes$
	ʔ	i	l	λ	λ	

Opaque phonological maps

(17) Counterbleeding

a. Lowering: [+long] $\rightarrow$ [$-$high]

b. Shortening: V $\rightarrow$ [$-$long] / $\_\_$ C #

(18) Yokuts (McCarthy 1999)

/ʔili:l/ $\mapsto$ [ʔilel], ‘might fan’

⋊	ʔ	i	l	i:	l	⋉
	ʔ	i	l	λ	λ	el

Opaque phonological maps

(19) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(20) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

- Deletion underapplies.

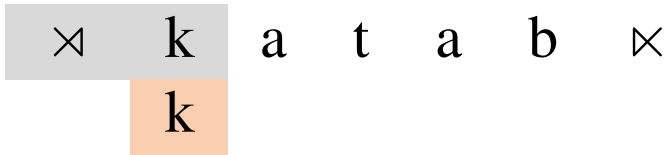
Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'



Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

$\rtimes$	k	a	t	a	b	$\ltimes$
	k	λ				

Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

$\rtimes$	k	a	t	a	b	$\ltimes$
	k	λ	λ			

Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

$\rtimes$	k	a	t	a	b	$\ltimes$
	k	λ	λ	it		

Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

$\rtimes$	k	a	t	a	b	$\ltimes$
	k	λ	λ	it	λ	

Opaque phonological maps

(21) Counterfeeding on focus

a. Deletion: i $\rightarrow \emptyset$ / __ CV

b. Raising: a $\rightarrow$ i / __ CV

(22) Bedouin Arabic (McCarthy 1999)
/katab/ $\mapsto$ [kitab], 'he wrote'

$\rtimes$	k	a	t	a	b	$\ltimes$
	k	λ	λ	it	λ	ab

Opaque phonological maps

(23) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(24) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

- Deletion destroys the environment for epenthesis.

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b							

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b	e						

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b	e	b					

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b	e	b	e				

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b	e	b	e	λ			

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

$\rtimes$	b	e	b	e	k	+	n	$\ltimes$
	b	e	b	e	λ	λ		

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

$\rtimes$	b	e	b	e	k	+	n	$\ltimes$
	b	e	b	e	λ	λ	λ	

Opaque phonological maps

(25) Self-destructive feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / C(+)__C#

b. Deletion: k $\rightarrow \emptyset$ / V__+V

(26) Turkish (Sprouse 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

⋊	b	e	b	e	k	+	n	⋉
	b	e	b	e	λ	λ	λ	in

Summary: ISL Opaque Maps

Opacity Type	**Example**	**k-value**
cross-derivational feeding	Lithuanian	$k = 2$
counterbleeding	Yokuts	$k = 3$
fed counterfeeding	Tundra Nenets	$k = 3$
counterfeeding on environment	Bedouin Arabic	$k = 3$
counterfeeding on focus	Bedouin Arabic	$k = 3$
non-gratuitous feeding	Classical Arabic	$k = 4$
self-destructive feeding	Turkish	$k = 5$

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉		⋊	ä	i	t	i	⋉
	v	e	λ	λ				ä				

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉		⋊	ä	i	t	i	⋉
	v	e	λ	λ	si			ä				

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉
	v	e	λ	λ	si

⋊	ä	i	t	i	⋉
	ä	i			

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉
	v	e	λ	λ	si

⋊	ä	i	t	i	⋉
	ä	i	λ		

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉
	v	e	λ	λ	si

⋊	ä	i	t	i	⋉
	ä	i	λ	ti	

Non-derived environment blocking

(27) Finnish (Kiparsky 1973, 1993)

a. t → s / __ i

b. e → i / __ #

(28) a. vete ↦ vesi, ‘water’

b. äiti ↦ äiti, ‘mother’

⋊	v	e	t	e	⋉
	v	e	λ	λ	si

⋊	ä	i	t	i	⋉
	ä	i	λ	ti	λ

Morphological maps

- In addition to these phonological maps, many (but not all) morphological 'maps' are also ISL.

Affixation

run ↦ running

⋊	ɹ	ʌ	n	⋉
	ɹ	ʌ	n	ŋ

do ↦ redo

⋊	d	u	⋉
ɹi			

Chickasaw
lakna ↦ iklakno
(‘It is yellow’ ↦ ‘It isn’t yellow’)

⋊	l	a	k	n	a	⋉
ik	l	a	k	n	λ	o

Tagalog
sulat ↦ sumulat
(‘write’ ↦ ‘to write’)

⋊	s	u	l	a	t	⋉
	λ	sumu				

Local partial reduplication

Tagalog
sulat ↦ susulat
(‘write’ ↦ ‘will write’)

⋊	s	u	l	a	t	⋉
	λ	susu			t	

Marshallese
ebbok ↦ ebbokbok
(‘to make full’ ↦ ‘puffy’)

⋊	e	b	b	o	k	⋉
	e	b	b	o	k	bok

Pima
mavit ↦ mamvit
(‘lions’ ↦ ‘lions’)

⋊	m	a	v	i	t	⋉
	λ	mam				

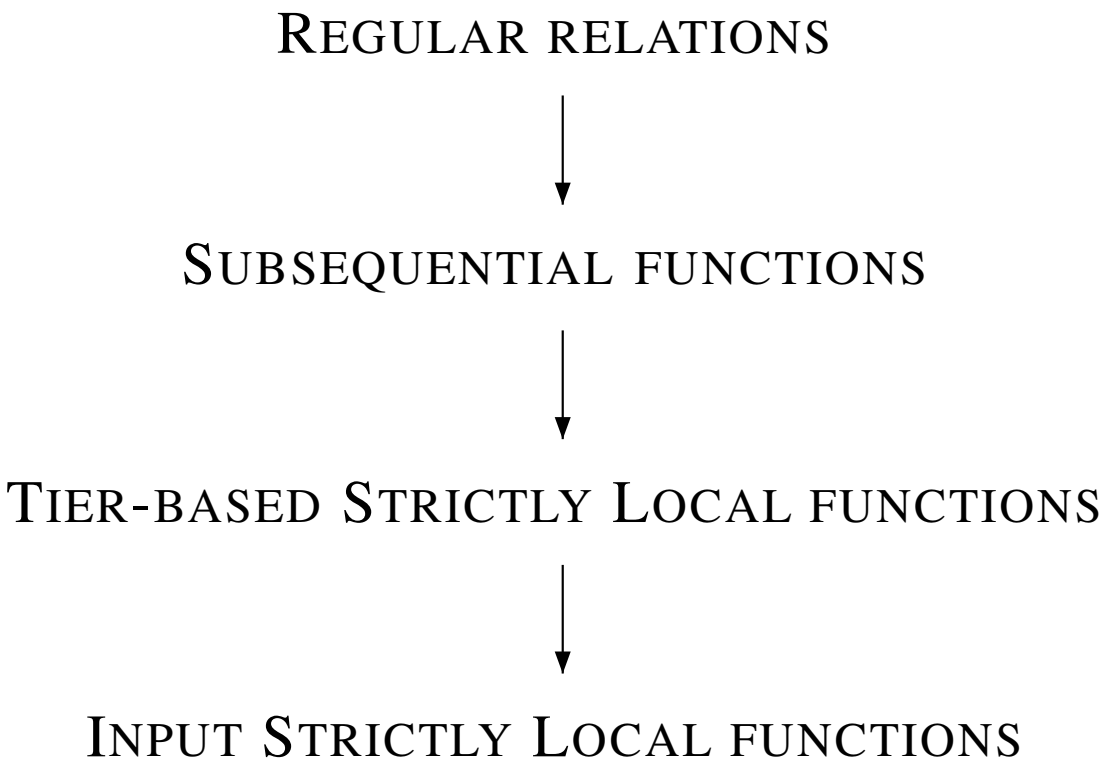
Non-local partial reduplication is not ISL

(29) Chukchee (Bogoras 1969)
nute $\mapsto$ nutenut, 'land.ABS'

(30) Madurese (McCarthy & Prince 1995, Inkelas & Zoll 2005)
mõwã $\mapsto$ wãmõwã, 'face' $\mapsto$ 'faces'

- These maps are not ISL
- Neither is full reduplication

Computational complexity of phonological maps



(Johnson 1972, Koskenniemi 1983, Kaplan & Kay 1994, Mohri 1997, Chandlee 2014, Chandlee et al. 2014, Chandlee, Heinz, Jardine, McMullin 2017)

Conclusion

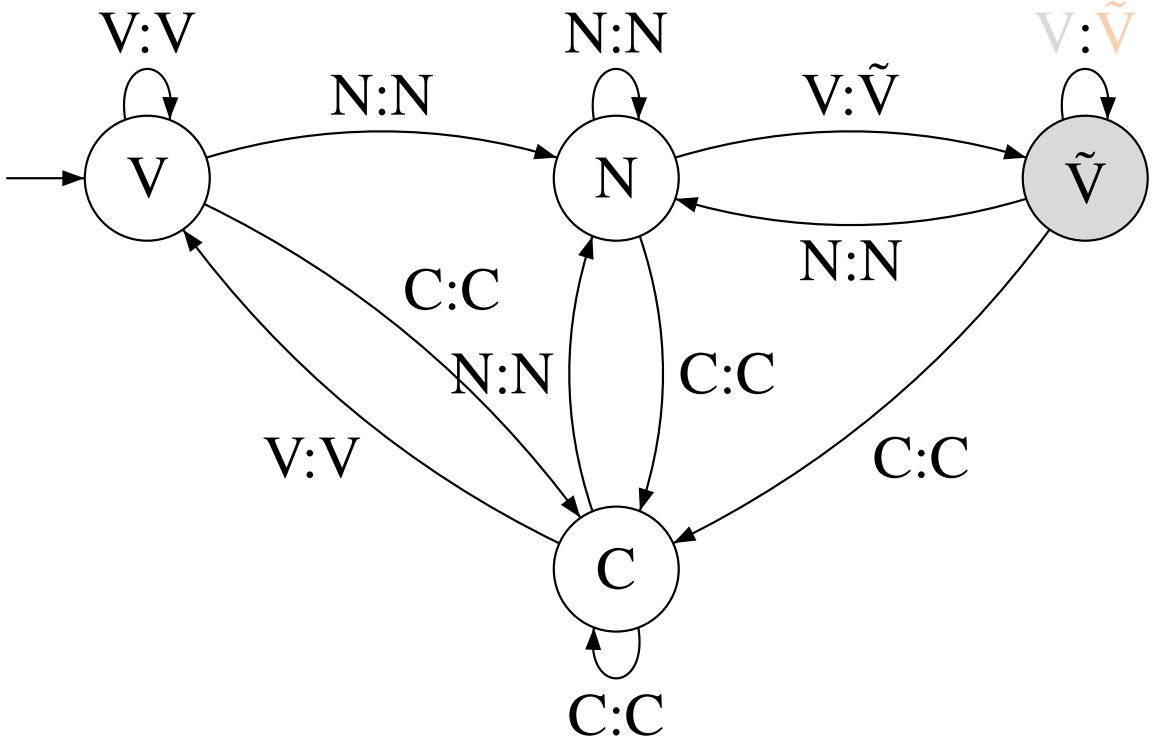
- The ISL functions are a subregular class of functions that are efficiently learnable from positive data.
- Many phonological maps, including those comprised of opaque process interactions, are describable with ISL functions.
- The subregular hierarchy of functions provides a framework for investigations into the computational nature of phonological and morphological maps.

Acknowledgements

- Jeffrey Heinz (University of Delaware)
- Adam Jardine (Rutgers University)
- Jim Rogers (Earlham College)
- Rémi Eyraud (Laboratoire D'Informatique Fondamentale de Marseille)
- Kevin McMullin (University of Ottawa)
- Steven Lindell (Haverford College)

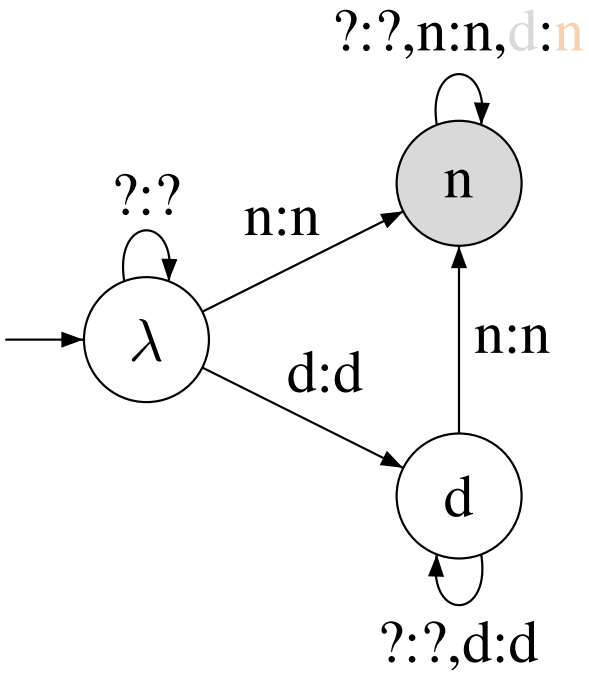
Output Strictly Local (OSL): FST characterization

(31) /pəŋawasan/ ↦ [pəŋãw̃ãsan]



Tier-based Strictly Local (TSL): FST characterization

(32) /tu**n**iki**d**i/ ↦ [tu**n**iki**n**i]



⋊	t	u	n	i	k	i	d	i	⋉
	t	u	n	i	k	i	n		

Additional examples of opacity

(33) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(34) Bedouin Arabic (McCarthy 1999)
/badw/ $\mapsto$ [badu], 'Bedouin'

- Raising underapplies.

Additional examples of opacity

(35) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(36) Bedouin Arabic (McCarthy 2009)
/badw/ $\mapsto$ [badu], 'Bedouin'

$\rtimes$	b	a	d	w	$\ltimes$
	b				

Additional examples of opacity

(35) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(36) Bedouin Arabic (McCarthy 2009)
/badw/ $\mapsto$ [badu], 'Bedouin'

$\rtimes$	b	a	d	w	$\ltimes$
	b	λ			

Additional examples of opacity

(35) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(36) Bedouin Arabic (McCarthy 2009)
/badw/ $\mapsto$ [badu], 'Bedouin'

$\rtimes$	b	a	d	w	$\ltimes$
	b	λ	λ		

Additional examples of opacity

(35) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(36) Bedouin Arabic (McCarthy 2009)
/badw/ $\mapsto$ [badu], 'Bedouin'

$\rtimes$	b	a	d	w	$\ltimes$
	b	λ	λ	ad	

Additional examples of opacity

(35) Counterfeeding on environment

a. Raising: a $\rightarrow$ i / __ CV

b. Vocalization: G $\rightarrow$ V / C __ #

(36) Bedouin Arabic (McCarthy 2009)
/badw/ $\mapsto$ [badu], 'Bedouin'

$\rtimes$	b	a	d	w	$\ltimes$
	b	λ	λ	ad	u

Additional examples of opacity

(37) Non-gratuitous feeding (Baković 2007)

a. Vowel epenthesis: $\emptyset \rightarrow V_i$ / # __ CCV_i

b. Glottal epenthesis: $\emptyset \rightarrow$ ʔ / #__V

(38) Classical Arabic (McCarthy 2007)
/ktub/ $\mapsto$ [ʔuktub], 'write.MASCSG!'

⋊	k	t	u	b	⋉
	λ				

Additional examples of opacity

(37) Non-gratuitous feeding (Baković 2007)

a. Vowel epenthesis: $\emptyset \rightarrow V_i$ / # __ CCV_i

b. Glottal epenthesis: $\emptyset \rightarrow$ ʔ / #__V

(38) Classical Arabic (McCarthy 2007)
/ktub/ $\mapsto$ [ʔuktub], 'write.MASCSG!'

⋊	k	t	u	b	⋉
	λ	λ			

Additional examples of opacity

(37) Non-gratuitous feeding (Baković 2007)

a. Vowel epenthesis: $\emptyset \rightarrow V_i$ / # __ CCV_i

b. Glottal epenthesis: $\emptyset \rightarrow$ ʔ / #__V

(38) Classical Arabic (McCarthy 2007)
/ktub/ $\mapsto$ [ʔuktub], 'write.MASCSG!'

⋊	k	t	u	b	⋉
	λ	λ	ʔuktu		

Additional examples of opacity

(39) Cross-derivational feeding (Baković 2007)

a. Epenthesis: $\emptyset \rightarrow$ i / K_i__K_i

b. Assimilation: K $\rightarrow$ [+voice] / __ D

(40) Lithuania (Odden 2005)

a. /at-taiki:ti/$\rightarrow$[atitaiki:ti], ‘to make fit well’

b. /ap-gauti/ $\mapsto$ [abgauti], ‘to deceive’

c. /ap-berti/ $\mapsto$ [apiberti], ‘to strew all over’

⋊	a	p	b	e	r	t	i	⋉
	a	λ	pib					

Additional examples of opacity

(41) Fed Counterfeeding (Kavitskaya & Staroverov 2010)

a. Glottalization: {t, d, s, n, ŋ} → ʔ / ___ #

b. Deletion: ʌ → ∅ / ___ (ʔ) #

(42) Tundra Nenets

a. /tasʌ/ ↦ [tas], 'whole'

b. /tʲimjʌs/ ↦ [tʲimjʔ], 'it rotted'

⋊	t	a	s	ʌ	⋉
	t	a	λ	λ	s

⋊	tʲ	i	m	j	ʌ	s	⋉
	tʲ	i	m	j	λ	λ	ʔ