Computational Phonology - Class 5

Jeffrey Heinz (Instructor)
Jon Rawski (TA)

LSA Summer Institute
UC Davis
July 08, 2019

Today

So far, we learned how to write constraints and transformations in MSO logic.

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

TODAY

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

1. MSO logic *is sufficient* to describe phonological generalizations.

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

1. MSO logic *is sufficient* to describe phonological generalizations.
2. MSO logic *is not necessary* to describe phonological generalizations.

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

1. MSO logic *is sufficient* to describe phonological generalizations.
2. MSO logic *is not necessary* to describe phonological generalizations.
3. Even FO logic *is not necessary* to describe phonological generalizations.

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

1. MSO logic *is sufficient* to describe phonological generalizations.
2. MSO logic *is not necessary* to describe phonological generalizations.
3. Even FO logic *is not necessary* to describe phonological generalizations.
4. We argue weaker logics for constraints and transformations *are sufficient.*

Today

So far, we learned how to write constraints and transformations in MSO logic.

So what?

Today we argue:

1. MSO logic *is sufficient* to describe phonological generalizations.
2. MSO logic *is not necessary* to describe phonological generalizations.
3. Even FO logic *is not necessary* to describe phonological generalizations.
4. We argue weaker logics for constraints and transformations *are sufficient.*
5. We contrast this with the dominant paradigm in phonology: Optimality Theory.

Along the way

We also argue that

Along the way

We also argue that

1. Logic is not just another formalism.

Along the way

We also argue that

1. Logic is not just another formalism.
2. It usefully demarcates properties of extensions independent of *any* grammatical formalism.

Along the way

We also argue that

1. Logic is not just another formalism.
2. It usefully demarcates properties of extensions independent of *any* grammatical formalism.
3. Those properties are about what *any computer* (machine or organism) must be able to distinguish or not distinguish.

Along the way

We also argue that

1. Logic is not just another formalism.
2. It usefully demarcates properties of extensions independent of *any* grammatical formalism.
3. Those properties are about what *any computer* (machine or organism) must be able to distinguish or not distinguish.
4. Logic classifies these properties along two dimensions: representation and computational power.

Part I

Regular?

Some equivalences

MSO-definable Constraints

Some equivalences

$\mathrm{MSO}(\triangleleft) \equiv \mathrm{MSO}(<)$

MSO-definable Constraints

Some equivalences

$\text{MSO}(\triangleleft) \equiv \text{MSO}(<)$

$\equiv$ 1NFA $\equiv$ 1DFA $\equiv$ 2NFA $\equiv$ 2DFA

MSO-definable Constraints

Some equivalences

MSO($\triangleleft$) $\equiv$ MSO($<$)

$\equiv$ 1NFA $\equiv$ 1DFA $\equiv$ 2NFA $\equiv$ 2DFA

$\equiv$ RE $\equiv$ GRE

MSO-definable Constraints

Some equivalences

$\mathrm{MSO}(\triangleleft) \equiv \mathrm{MSO}(<)$

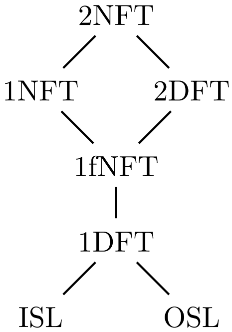
$\equiv$ 1NFA $\equiv$ 1DFA $\equiv$ 2NFA $\equiv$ 2DFA

$\equiv$ RE $\equiv$ GRE

The extensions of these grammars are often called 'regular' or 'rational' formal languages/ constraints/ stringsets.

(Kleene 1956, Scott and Rabin 1959, Büchi 1960, McNaughton and Papert 1971, Hopcroft and Ullman 1979, Thomas 1997)

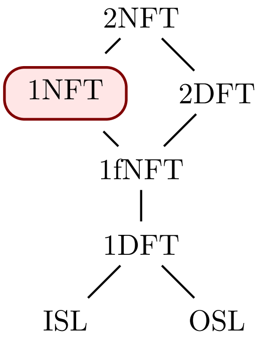
MSO-definable Transformations



Properties separate when studying transformations.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

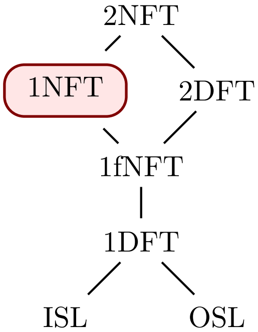
MSO-definable Transformations



Computational linguists call 1NFT **regular relations**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

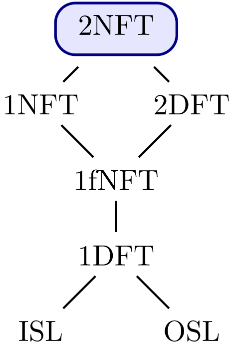
MSO-definable Transformations



French computer scientists call 1NFT **rational relations**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

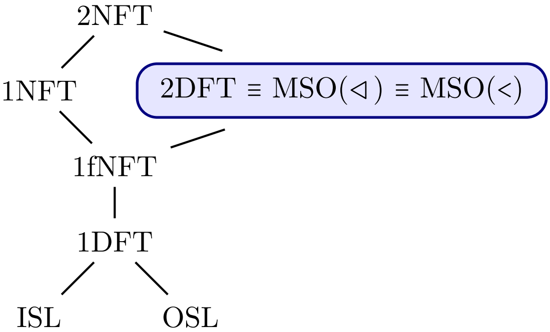
MSO-definable Transformations



French computer scientists call 2NFT **regular relations**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

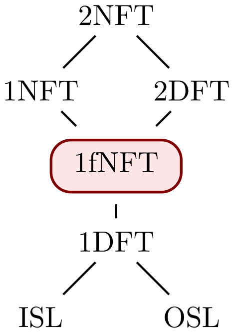
MSO-definable Transformations



French computer scientists call 2DFT **regular functions**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

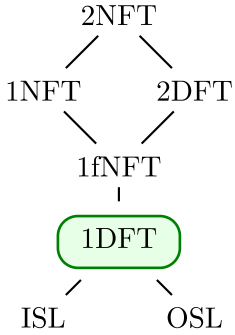
MSO-definable Transformations



French computer scientists call 1fNFT **rational functions**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

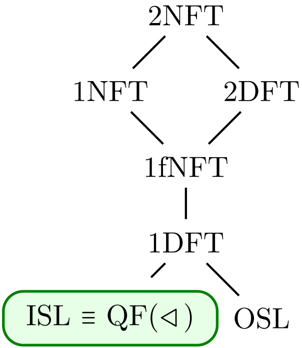
MSO-definable Transformations



French computer scientists call 1DFT **sequential functions**.

(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

MSO-definable Transformations

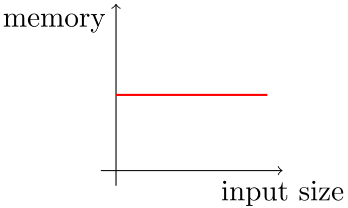


Today we are going to talk a lot about this class of functions!

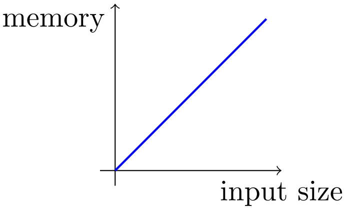
(Engelfriedt and Hoogeboom 2001, Chandlee 2014, Chandlee et al. 2014, 2015, Filiot and Reynier 2016)

What "Regular" means

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



Regular



Non-regular

Example: Vowel Harmony

Progressive

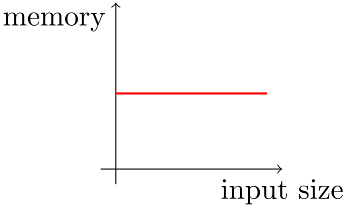
Vowels agree in backness with the first vowel in the underlying representation.

Majority Rules

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

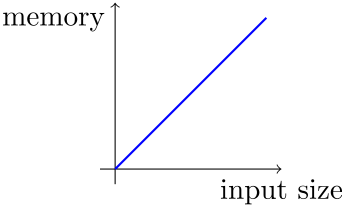
UR	Progressive	Majority Rules
/nokelu/	nokolu	nokolu
/nokeli/	nokolu	nikeli
/pidugo/	pidige	pudugo
/pidugomemi/	pidigememi	pidigememi

Progressive and Majority Rules Harmony



Regular

Progressive



Non-regular

Majority Rules

Some Perspective

Typological: Majority Rules is unattested.

(Lombardi 1999, Bakovic 2000, Bowler 2013)

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.

Part II

Phonological Theories

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

Main Claim

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

Part III

Phonological Generalizations are Regular

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 1NFT (rational relation).

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 1NFT (rational relation).

Consequently:

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 1NFT (rational relation).

Consequently:

1. Constraints on well-formed surface and underlying representations are regular (since the image and pre-image of rational relations are regular). (Rabin and Scott 1959)

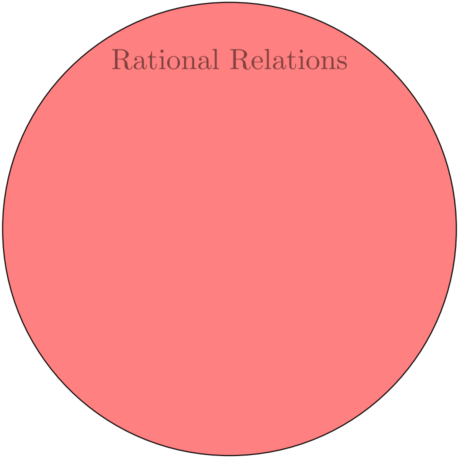
Phonological generalizations are regular

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 1NFT (rational relation).

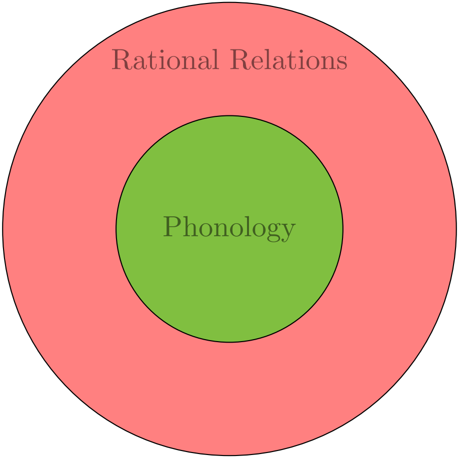
Consequently:

1. Constraints on well-formed surface and underlying representations are regular (since the image and pre-image of rational relations are regular). (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 function that *any* phonological grammar defines.

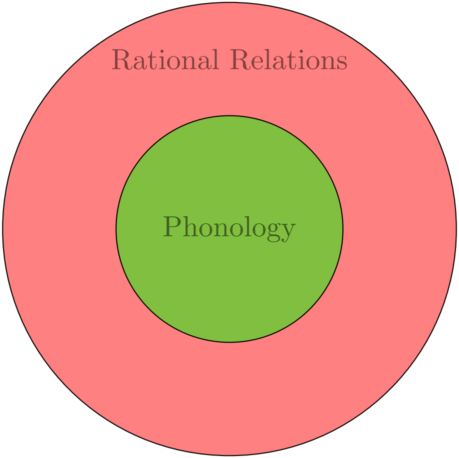
In Pictures



In Pictures



In Pictures



Some argued rule-based grammars overgenerated. Also, nobody knew how to learn them.

Part IV

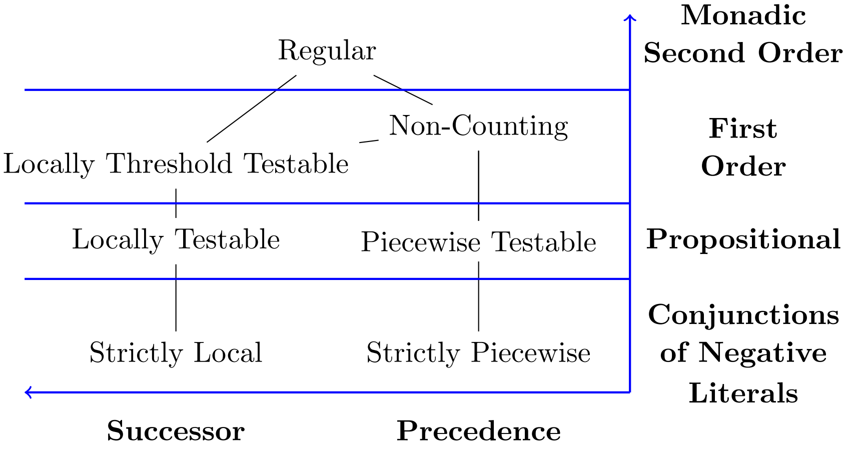
Analytical Framework

Computation is reflected in logical power

Subregular hierarchies organize pattern complexity along two dimensions.

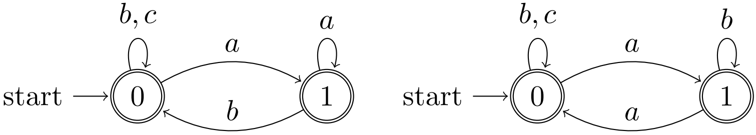
- **logical power** along the vertical axis
- **representational primitives** along the horizontal axis.

Logical Characterizations of Subregular Stringsets



(McNaughton and Papert 1971, Rogers and Pullum 2011, Rogers et al. 2013, Heinz 2018)

Size of automata ∝ complexity? No.



G1 **G2**

- 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").
- G1 generates/recognizes all words except those with a forbidden string [ac]; and G2 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 G2 is properly regular.

(Heinz and Idsardi 2013)

Logic as a high-level language

1. Logical formulas 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
   1. a better fit to the typology than optimization,
   2. have learning results that are as good or better than in OT,
   3. provide equally good or better insights,
   4. and are effectively computable.

Part V

Input Strictly Local Functions

Input Strict Local *Transformations*

This is a class of transformations which. . .

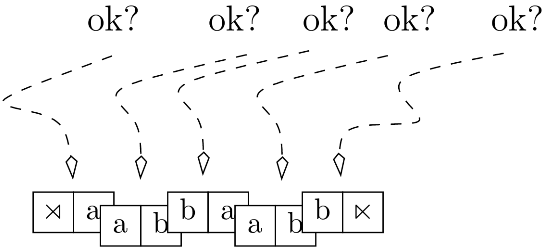
1. generalizes Strictly Local Stringsets,
2. captures a wide range of phonological phenomena,
3. including *opaque* transformations,
4. and is effectively learnable!

(Chandlee 2014, Chandlee et al. 2014, 2015, Chandlee and Heinz 2018, Chandlee et al. 2018)

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.



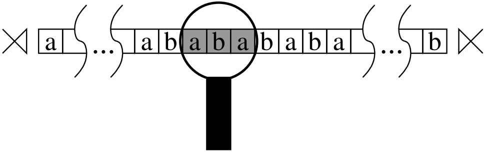
An ill-formed sub-structure is **forbidden**.

(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.

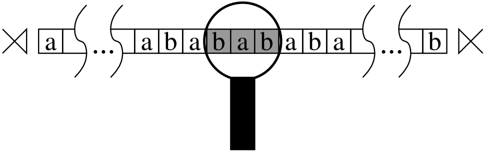


(Rogers and Pullum 2011, Rogers et al. 2013)

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 forbidden or not. The whole structure is well-formed only if each local sub-structure is.

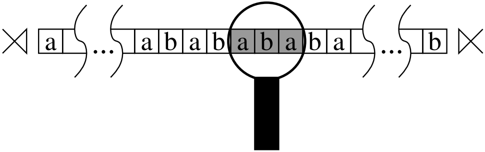


(Rogers and Pullum 2011, Rogers et al. 2013)

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 forbidden or not. The whole structure is well-formed only if each local sub-structure is.



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

Examples

Examples of Strictly Local constraints for strings

- *aa
- *ab
- *NC̥
- NoCoda

Examples of Non-Strictly Local constraints

- *s…ʃ (Hansson 2001, Rose and Walker 2004, Hansson 2010)
- *#s…ʃ# (Lai 2012, 2015)
- Obligatoriness: Words must contain one primary stress (Hayes 1995, Hyman 2011, inter alia).

SL Constraints as Conjunctions of Negative Literals

$$\varphi \stackrel{\text{def}}{=} \neg\text{np} \wedge \neg\text{nk} \wedge \neg\text{mt} \wedge \neg\text{mg}$$

- A SL_k grammar can be thought of as a list of **forbidden** sub-strings whose max length is k.
- As a logical language, this is simply the conjunctions of negative literals:
  - where the literals u are strings
  - and $\mathcal{M}_w^{\triangleleft} \models u$ iff $\mathcal{M}_u^{\triangleleft}$ is a sub-structure of $\mathcal{M}_w^{\triangleleft}$.

Abstract characterizations. . .

1. are independent of logical formulas, grammars, and automata
2. provide laws of *inference* for learning
3. provide ways to show certain stringsets do NOT belong to the class.

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

SL stringsets - Abstract characterization

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

Theorem. k-Local Suffix Substitution Closure

For all $L \subseteq \Sigma^*$, $L \in \text{SL}$ iff there exists k such that for all $u_1,\ v_1,\ u_2,\ v_2,\ x \in \Sigma^*$ it is the case that

$$\begin{array}{rl} \text{if} & u_1 x v_1, u_2 x v_2 \in L \text{ and } |x| = k-1 \\ \text{then} & u_1 x v_2 \in L \end{array}$$

Using Suffix Substitution Closure

- The theorem provides a law which simultaneously
  - provides a basis for *inference*
  - provides a method for establishing non-SL_k stringsets.

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

In-class exercises

1. Consider a Strictly 2-Local stringset L which contains the words aa and ab. Using Suffix Substitution Closure, explain what other words must be in L.
2. Consider the constraint *s. . . ʃ. Show this is not SL_k for any k.

Cognitive Interpretation of SL

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

(Rogers et al. 2013)

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 et al. 2014, 2015, Chandlee and Heinz 2018)

Input Strict Locality: Main Idea

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

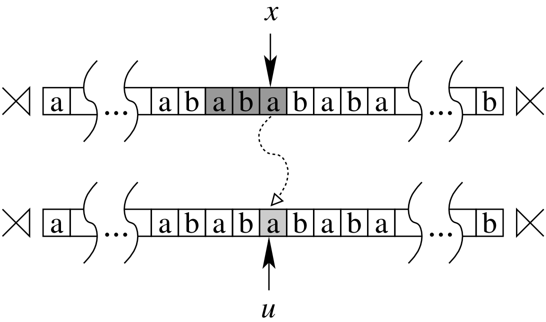


Figure: For every Input Strictly k-Local function, the output string u of each input element x depends only on x and the $k-1$ input elements previous to x. In other words, the contents of the lightly shaded cell only depends on the contents of the darkly shaded cells.

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

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

input:	⋊	o	v	e	⋉
output:	⋊	o	v	λ	i ⋉

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

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

input:	⋊	o	v	e	⋉
output:	⋊	o	v	λ	i ⋉

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

/ove/ ↦ [ovi]

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

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

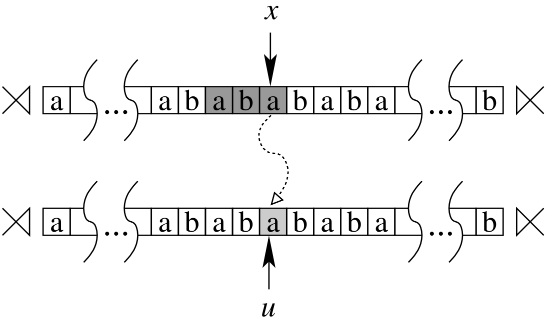
/ove/ $\mapsto$ [ovi]

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

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.



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. 2014, AMP).
   - 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 VI

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 VII

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, 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)

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. 2018)

Example: Counterbleeding in Yokuts

	'might fan'
	/ʔiliː+l/
[+long] → [-high]	ʔileːl
V ⟶ [-long] / __ C#	ʔilel
	[ʔilel]

Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ ↦ [ʔilel]

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

Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiːlil/ ↦ [ʔilel]

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

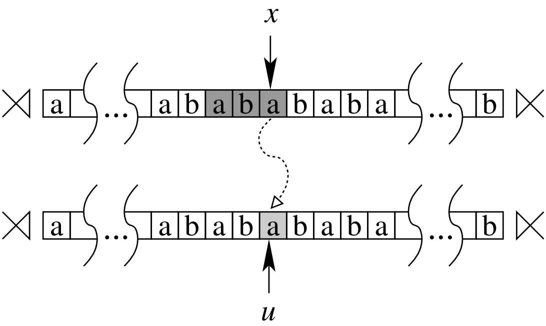
Example: Counterbleeding in Yokuts is ISL with k=3

/ʔiliːl/ ↦ [ʔilel]

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.



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

Logical Characterization of ISL Functions

Theorem.

ISL functions correspond exactly to the functions definable with Quantifier-Free logic with successor: QF($\triangleleft$).

(Lindell and Chandlee, LICS 2016)

Logical Characterization of ISL Functions

Theorem.

ISL functions correspond exactly to the functions definable with Quantifier-Free logic with successor: QF($\triangleleft$).

Quantifier-free formulas are ones without *any* quantification!

(Lindell and Chandlee, LICS 2016)

Locality and Quantification

Compare:

1. $P(x) \stackrel{\text{def}}{=} Q(x) \wedge \exists y[R(y)]$ (First Order Definable)

   Requires scanning whole word for such a y!!

2. $P(x) \stackrel{\text{def}}{=} Q(x) \wedge R(\texttt{predecessor}(x))$ (QF Definable)

   Information to decide P is local to x in the input!!

Autosegmental Input Strictly Local Functions

- Defined as Quantifier Free Transformations with Autosegmental Representations (ASR)
- To what extent does ASRs make non-local processes local?

Pattern	Language	ISL	A-ISL
Bounded shift (§4.1, 5.2)	Rimi	✓	✓
Bounded spread (§6.1)	Bemba	✓	✓
Bounded Meussen's Rule (§6.3)	Luganda	✓	✗
Unbounded shift (§4.2,5.3)	Zigula	✗	✓
Unbounded deletion (§6.2)	Arusa	✗	✓
Alternating Meussen's Rule (§6.4)	Shona	✗	✗
Unbounded spread (§6.5)	Ndebele	✗	✗

Table 1: Summary of analyses.

(Chandlee and Jardine 2019, TACL)

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 2018)

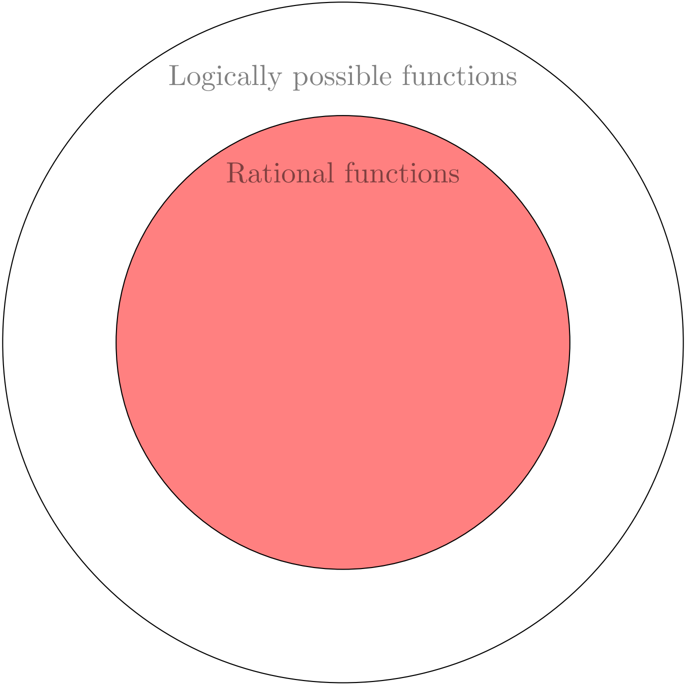
Typological Summary of ISL

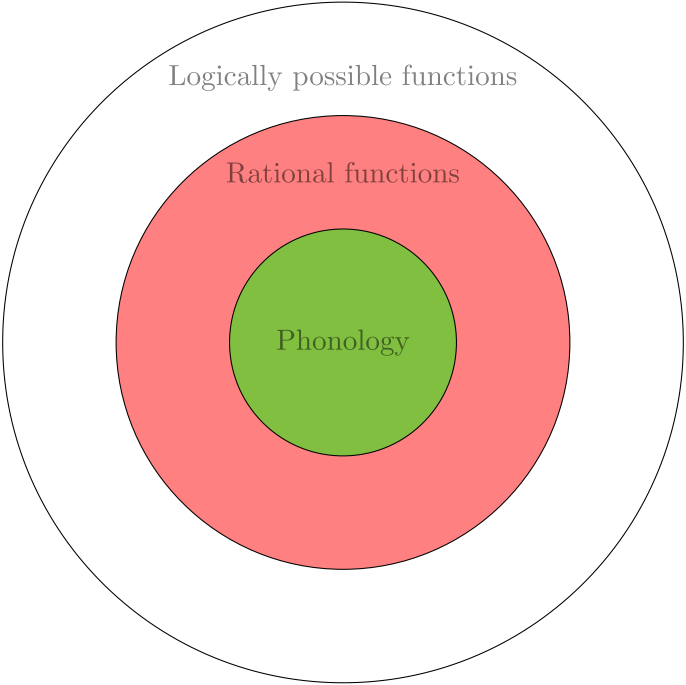
Undergeneration? Yes, for now. . .

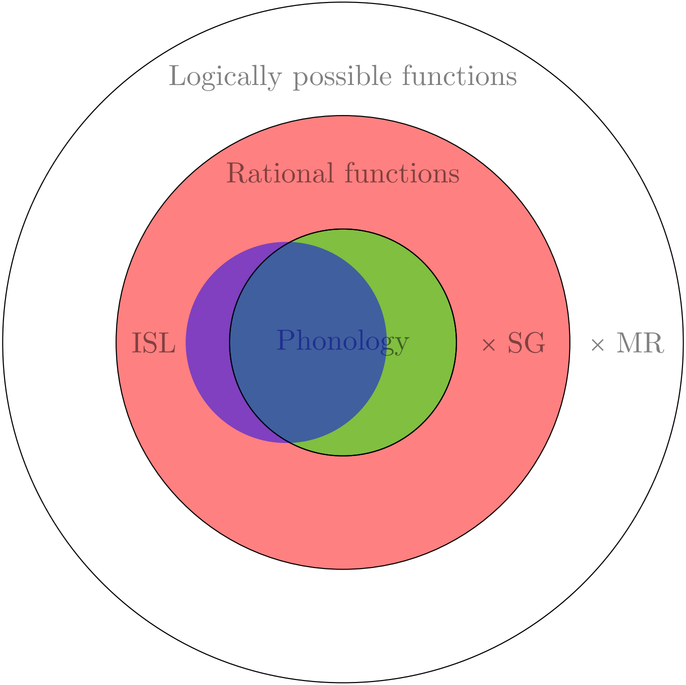
- ISL functions are insufficiently expressive for spreading and long-distance harmony. I will discuss these later (or in Q&A).

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 functions
Rational functions


Logically possible functions
Rational functions
Phonology


Logically possible functions
Rational functions
ISL
Phonology
× SG
× MR

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 aforementioned 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)

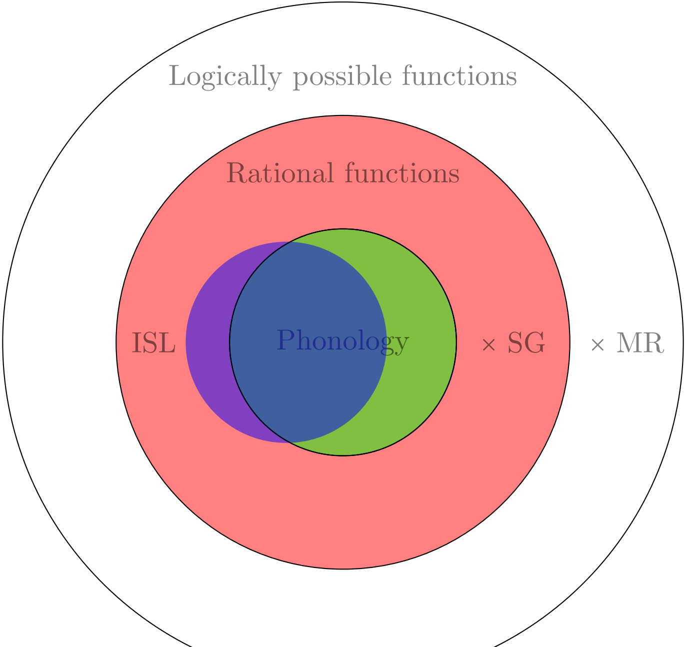
Optimization misses an important generalization

- 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.

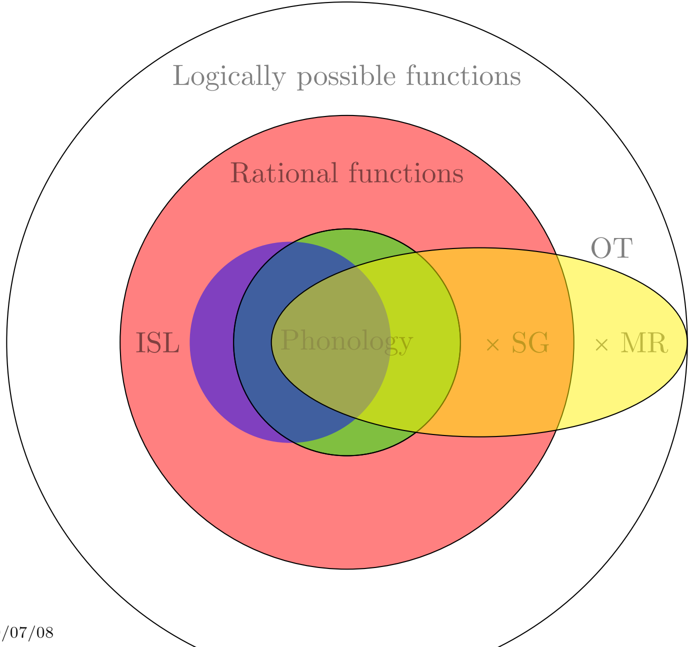
OT's greatest strength is its greatest weakness.

- 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.

Classic OT and Typology



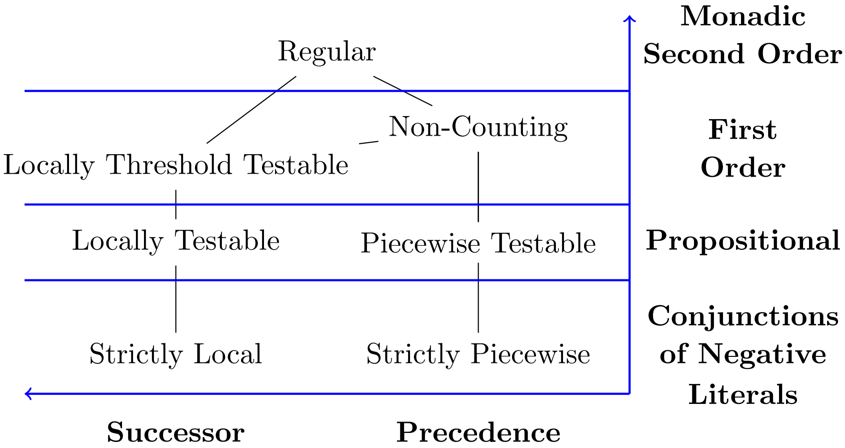
Classic OT and Typology



Part VIII

Conclusion

Logical Characterizations of Subregular Stringsets



(McNaughton and Papert 1971, 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.

Homework for Thursday

1. Read Strother-Garcia (2018) "Imdlawn Tashlhiyt Berber Syllabification is Quantifier-Free" (8 pages)
2. Read Chandlee and Jardine (2019) "Autosegmental Input Strictly Local Functions" (12 pages)

Supplemental reading

- Chandlee and Heinz (2018) "Strict Locality and Phonological Maps"
- Rawski (under review) "The Logical Nature of Phonology Across Speech and Sign"