Typology Emerges from Computability

Jon Rawski
Linguistics & IACS @ SBU
Linguistics @ SJSU (Fall 2021)

Today's Lecture

- Typology: Scope and limits of linguistic processes
- Computational Typology: Computability as an organizing principle

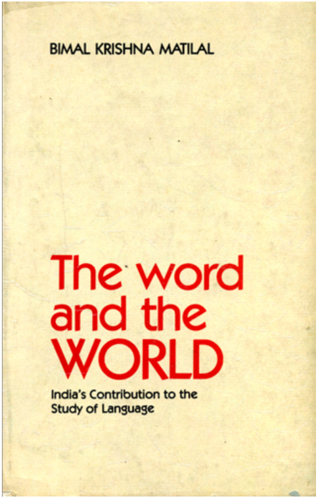
Parts of the Lecture

- Typology and Computability
- Situating processes in types of computation
- Neural interpretability experiments
- Open areas

Collaborators

Jeff Heinz
Thomas Graf
Aniello de Santo
Hossep Dolatian
Alëna Aksenova
Dakotah Lambert
Jane Chandlee
Adam Jardine
Eileen Blum
James Rogers
Dine Mamadou
Chris Oakden
Eric Bakovic
Adam McCollum
Andrew Lamont
Anna Mai
Kevin McMullin
Remi Eyraud
...You?

Typology: Bharthari to von Humboldt



Encyclopedias in Linguistic Typology

Encyclopedia of Types: Processes in Natural Language
Encyclopedia of Categories: Classes of Computable Functions

Data vs Phenomena (Bogen & Woodward 1988)

- Data
  - unstable, perceptually accessible, observable
  - “idiosyncratic to particular investigative contexts"
- Phenomena:
  - “relatively stable, recurrent, general features of the world"
  - “a varied ontological bag that includes objects, states, processes, events, and other features that are hard to classify

Encyclopedia of Types: Possible Linguistic Processes

Today's talk will be illustrated by pieces of phonological and morphological typology

Harmony:

- Aksënova, Rawski, Heinz, Graf. *The Computational Power of Harmony*. To Appear (preprint on LingBuzz)

Reduplication:

- Dolatian & Heinz (2020). *Computing and classifying reduplication with 2-way finite-state transducers*. Journal of Language Modeling
- Dolatian & Heinz (2020). *RedTyp: A Database of Reduplication with Computational Models*. SCiL 2019

Encyclopedia of Types: Possible Linguistic Processes

Harmony

- Progressive
  - iuuu→ iiii
- Regressive
  - uuui→ iiii
- Sour Grapes
  - iuuuu→iiiii
  - iuuau→iuuau
- Circumambient
  - iuui→iiii
    iuuu→iuuu
- Majority Rules
  - iuuii→iiiii
    iuuiu →uuuuu

Reduplication/Copying

- Partial
  - abcd → ababcd
- Total:
  - abcd → abcdabcd
- Triplication:
  - abcd → abcdabcdabcd
- Polynomial $w \to w^{|w|}$:
  - abcd→abcdabcdabcdabcd
- Exponential:
  - abcd→a bb ccc dddd
- Iterated prefix:
  - abcd → a ab abc abcd

Encyclopedia of Types: Possible Linguistic Processes

Harmony

- Progressive
  - iuuu→ iiii
- Regressive
  - uuui→ iiii
- **Sour Grapes**
  - iuuuu→iiiii
  - iuuau→iuuau
- Circumambient
  - iuui→iiii
    iuuu→iuuu
- **Majority Rules**
  - iuuii→iiiii
    iuuiu →uuuuu

Reduplication/Copying

- Partial
  - abcd → ababcd
- Total:
  - abcd → abcdabcd
- Triplication:
  - abcd → abcdabcdabcd
- **Polynomial** $w \rightarrow w^{|w|}$:
  - abcd→abcdabcdabcdabcd
- **Exponential**:
  - abcd→a bb ccc dddd
- **Iterated prefix**:
  - abcd → a ab abc abcd

Attested Reduplication: Data and Phenomena

(1) Total reduplication = unbounded copy (∼83%)

a. wanita→wanita∼wanita
'woman'→'women' (Indonesian)

(2) Partial reduplication = bounded copy (∼75%)

a. C: gen→g∼gen
'to sleep'→'to be sleeping' (Shilh)

b. CV: guyon→gu∼guyon
'to jest'→'to jest repeatedly' (Sundanese)

c. CVC: takki→ tak∼takki
'leg'→'legs' (Agta)

d. CVCV: banagaɲu→bana∼banagaɲu
'return' (Dyirbal)

Phenomenological vs Theoretical Laws (Cartwright 1983)

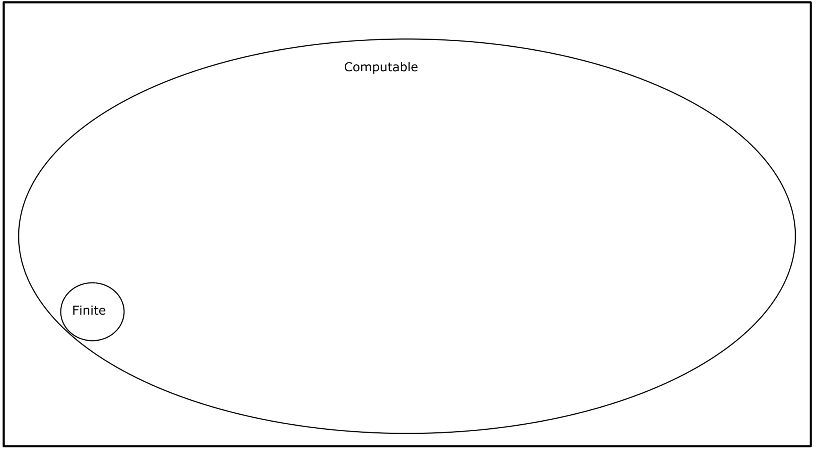
- **Phenomenological Law**: descriptively adequate statements, analytic/approximate predictions within a framework, framework extensions to handle empirical cases
- **Theoretical Law**: explanatory statements about possible/impossible phenomenological laws

Cartwright: "the distinction between theoretical and phenomenological has nothing to do with what is observable and what is unobservable. Instead the terms separate laws which are fundamental and explanatory from those that describe"

Encyclopedia of categories: Computable Functions

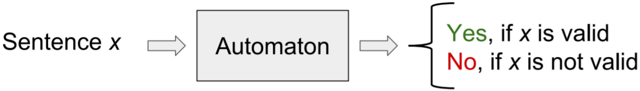
Al-Khwarizmi: “When I consider what people want in computing, it is generally a number"

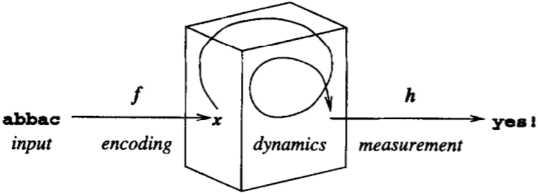
Turing: It is impossible to mechanically enumerate certain sets



Procedural Views of Computability

- **Grammar/Automaton:** Computational device that decides whether a string is in a set (says yes/no)
- Functional perspective: $f : \Sigma^* \to \{0, 1\}$
  - Σ: Alphabet of Symbols
  - Σ^*: set of all possible strings (free monoid on Σ)





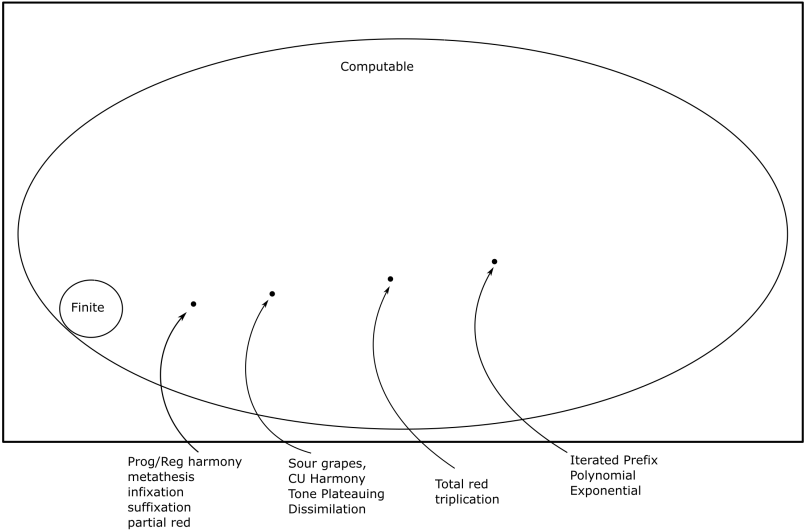
p.c. Casey 1996

mina
@problem_halting

Uncomputability isn't just a mathematical concept. It denotes the possibility of contradiction arrived at not because of the failure of, but because of the success of reason. And this is a different sort of contradiction than one you arrive at from failure of reason.

Computability: Necessary, Not Sufficient

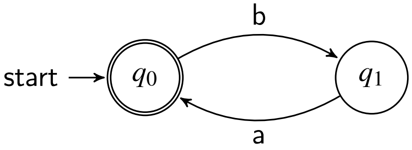
> *[This] condition, on the other hand, has no interest. We learn nothing about a natural language from the fact that its sentences can be effectively displayed,i.e., that they constitute a recursively enumerable set. The reason for this is clear. Along with a specification of the class F of grammars, a theory of language must also indicate how, in general, relevant structural information can be obtained for a particular sentence generated by a particular grammar.*
>
> *Chomsky 1959*


Computable
Finite
Prog/Reg harmony
metathesis
infixation
suffixation
partial red
Sour grapes,
CU Harmony
Tone Plateauing
Dissimilation
Total red
triplication
Iterated Prefix
Polynomial
Exponential

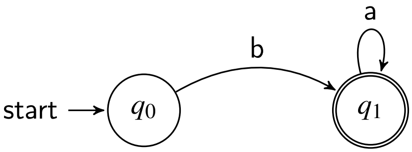
Regular Languages & Finite-State Automata

Regular Language: Memory required is finite w.r.t. input

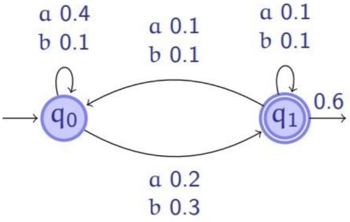
(ba)*: {ba, baba, bababa,...}



b(a*): {b, ba, baaaaa,....}



Regular Languages & Finite-State Automata



Operator Representation

$$\boldsymbol{\alpha} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad \mathbf{A}^{a} = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\boldsymbol{\omega} = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad \mathbf{A}^{b} = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

$$f(ab) = 0.4 \times 0.3 \times 0.6 + 0.2 \times 0.1 \times 0.6 = 0.084$$

$$= \boldsymbol{\alpha}^{\top} \mathbf{A}^{a} \mathbf{A}^{b} \boldsymbol{\omega}$$

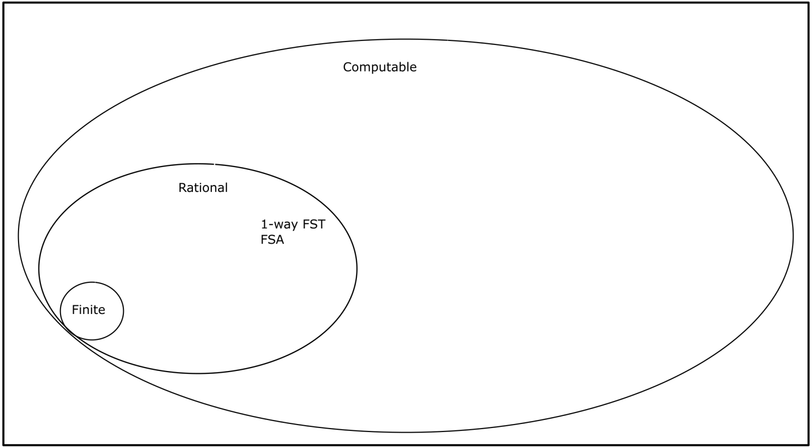
p.c. Guillaume Rabusseau

Sets to Processes via Semirings

We can generalize "regularity" to consider various output semirings, not just Bools or Reals

SEMIRING	*function*	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Boolean	$\phi : \Sigma^* \to \{0,1\}$	$\vee$	$\wedge$	0	1
Natural	$\phi : \Sigma^* \to \mathbb{N}$	$+$	$\times$	0	1
Viterbi	$\phi : \Sigma^* \to [0,1]$	max	$\times$	0	1
Probability	$\phi : \Sigma^* \to \mathbb{R}_+$	$+$	$\times$	0	1
Log	$\phi : \Sigma^* \to \mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\phi : \Sigma^* \to \mathbb{R} \cup \{-\infty, +\infty\}$	$\min$	$+$	$+\infty$	0
String	$\phi : \Sigma^* \to \Sigma^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ε
Language	$\phi : \Sigma^* \to \mathcal{P}(\Sigma^*)$	$\cup$	$\cdot$	$\emptyset$	$\{\varepsilon\}$

Rational vs Computable

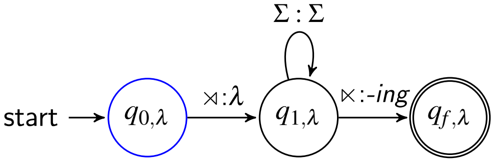


Rational Morphology: suffixation

- Working example: *hold* $\rightarrow$ *hold-ing*

  Input: $\rtimes$ h o l d $\ltimes$

  Output:

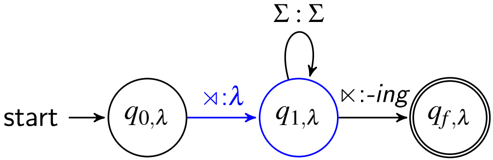


Rational Morphology: suffixation

- Working example: *hold* → *hold-ing*

  Input: ⋊ h o l d ⋉

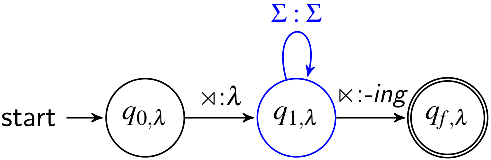
  Output:



Rational Morphology: suffixation

- Working example: *hold* $\rightarrow$ *hold-ing*

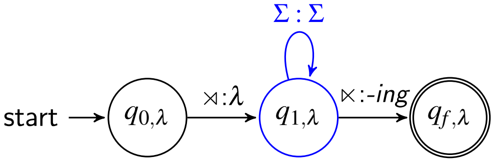
Input:	$\rtimes$	h	o	l	d	$\ltimes$
Output:		h				



Rational Morphology: suffixation

- Working example: *hold* $\rightarrow$ *hold-ing*

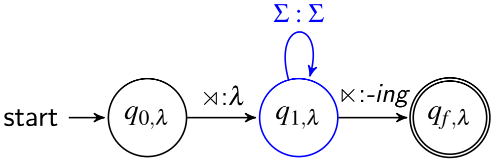
Input:	⋊	h	o	l	d	⋉
Output:		h	o			



Rational Morphology: suffixation

- Working example: *hold* → *hold-ing*

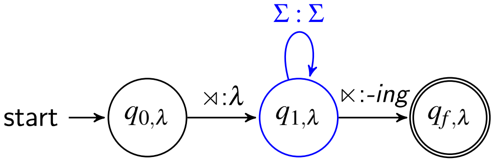
Input:	⋊	h	o	l	d	⋉
Output:		h	o	l		



Rational Morphology: suffixation

- Working example: *hold* → *hold-ing*

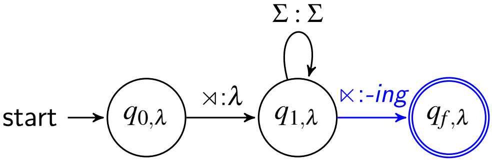
Input:	⋊	h	o	l	d	⋉
Output:		h	o	l	d	



Rational Morphology: suffixation

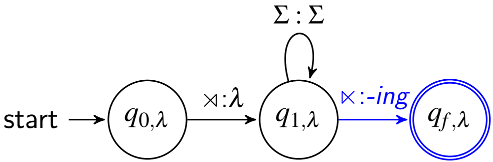
- Working example: *hold* $\rightarrow$ *hold-ing*

Input:	$\rtimes$	h	o	l	d	$\ltimes$			
Output:		h	o	l	d		i	n	g

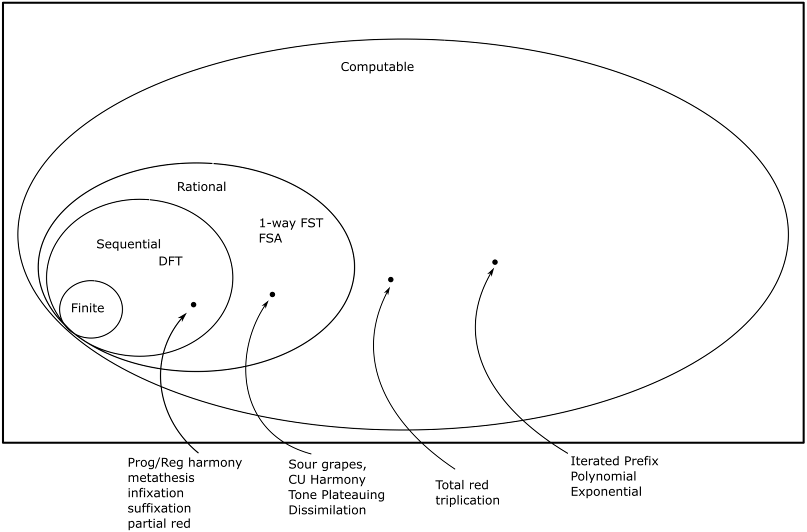


Sequential functions (Schützenberger 1965)

- Computed by Deterministic 1-way FST
- Deterministic: one choice per symbol per state
- Bounded Lookahead
- Examples: prefixation, suffixation, partial copying, progressive/regressive harmony (Chandlee 2017, Heinz & Lai 2013)



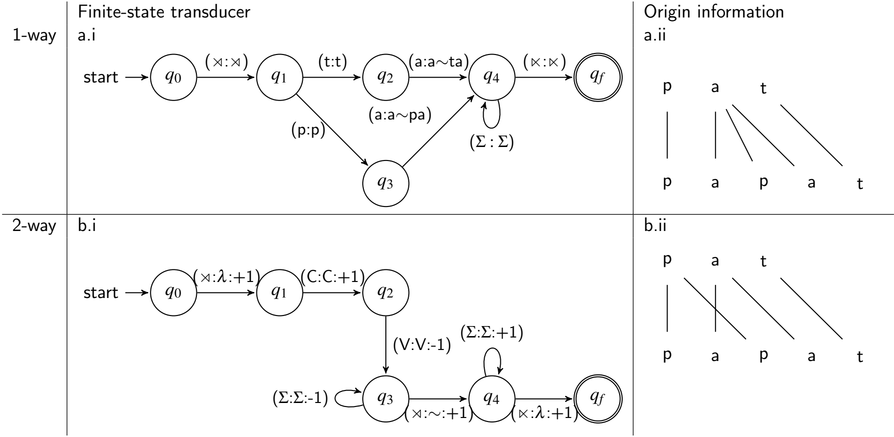
Sequential vs Rational



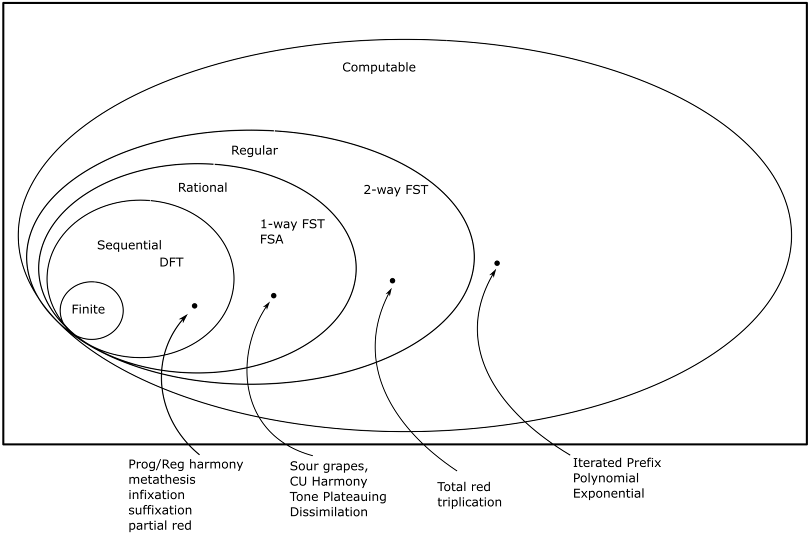
Regular Functions (Engefriet & Hoogeboom 2002)

- image of string of length n has length $(\mathcal{O}(n))$ (Lhote 2018)
- Computable by 2-way FSTs, streaming string transducers
- Examples: Total Reduplication, Triplication, all Rational & Sequential (Dolatian & Heinz 2020)

1-way and 2-way Finite-State Transducers

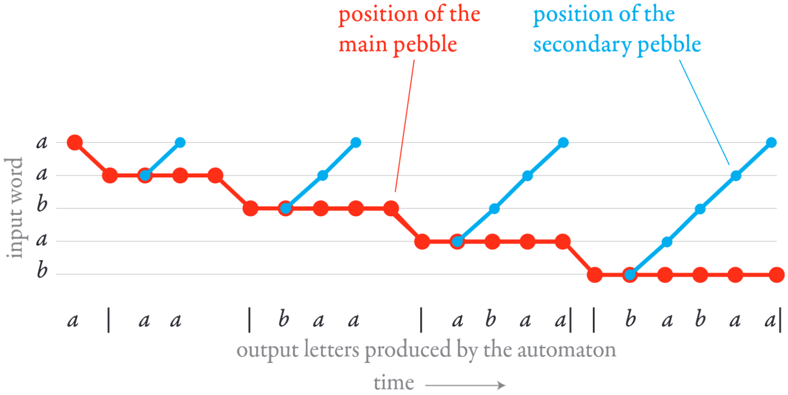


Regular vs Rational

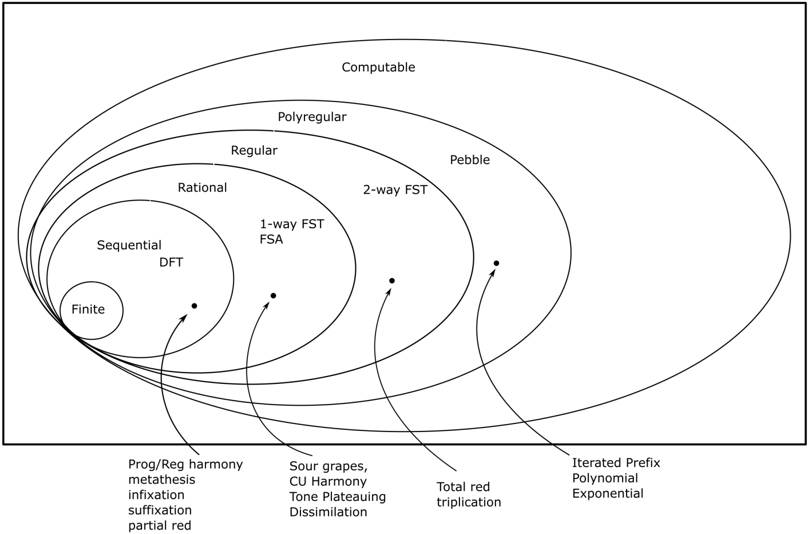


Polyregular Functions (Bojanczyk 2018)

- Image of string has length ($\mathcal{O}(n^k)$)
- Computed by pebble transducers with k pebbles (like stacks)
- Examples: Regular + Iterated Prefix Copy, Polynomial Copy, $w \to w^{|w|}$



Polyregular vs Regular



Rawski & Heinz 2019, *Language*

1. No Free Lunch in Linguistics or Machine Learning
2. Every successful induction system contains biases. Those biases constrain what it can and can't learn
3. "Don't confuse ignorance of biases with absence of biases"

- **Grammatical Inference**: what is the nature of these biases when learning grammars from data?
- **Encyclopedia of Categories**:
  - Necessary and sufficient conditions on computable functions
  - Provide target function classes for generalization/learning

Probing RNN Generalization with Reduplication

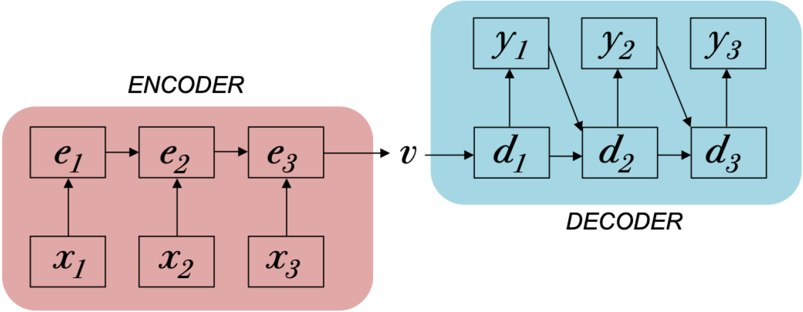
Hossep Dolatian
(Stony Brook)

Max Nelson
(UMass Amherst)

Brandon Prickett
UMass Amherst)

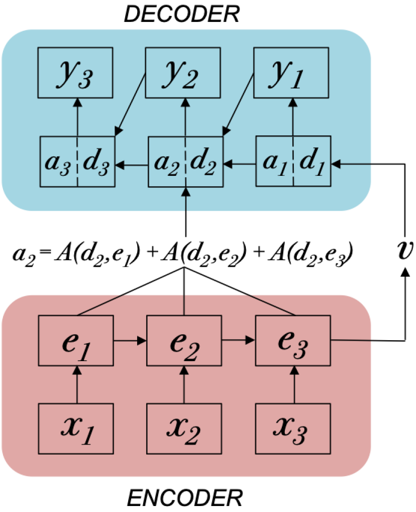
RNN Encoder-Decoder and Transducers

- **Function**: *Given string w, generate $f(w) = v$*
  - = accepted pairs of input & output strings
  - Computed by different classes of grammars (**transducers**)
- Recurrent encoder maps a sequence to $v \in \mathbb{R}^n$, recurrent decoder language model conditioned on v (Sutskever et al., 2014)
- How expressive are they?

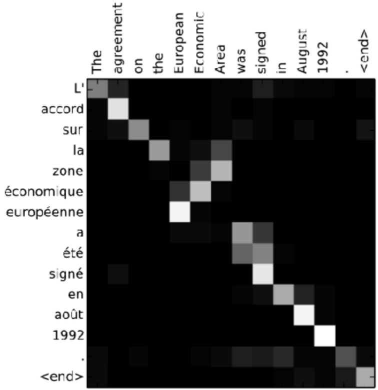


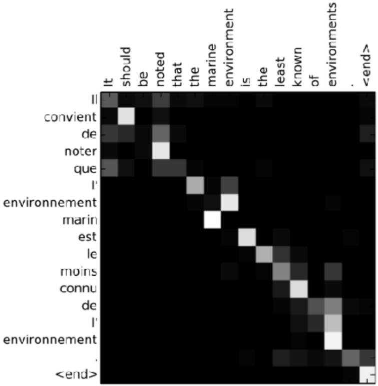
Attention

- In standard ED, the encoded representation is the only link between the encoder and decoder
- **Global attention** allows the decoder to selectively pull information from hidden states of the encoder (Bahdanau et al., 2014)
- **FLT Analog**: 2-way FST has full access to the input by moving back and forth



Attention


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>


It
should
be
noted
that
the
marine
environment
is
the
least
known
of
environments
.
<end>
Il
convient
de
noter
que
l'
environnement
marin
est
le
moins
connu
de
l'
environnement
.
<end>

Test data

- Input-output mappings generated with 2-way FSTs from RedTyp database[1]
  1. Initial-CV tasgati→ta∼tasgati
     Fixed-size reduplicant
  2. Initial two-syllable (C*VC*V)
     tasgati→tasga∼tasgati
     Onset maximizing, fixed over vowels
  3. Total
     tasgati→tasgati∼tasgati
     Variably sized reduplicant
- 10,000 generated for each language, 70/30 train/test split
- Minimum string length 3 - maximum string length varied
- Alphabet of 10, 16, or 26 characters
- Boundary symbols (∼) are not present

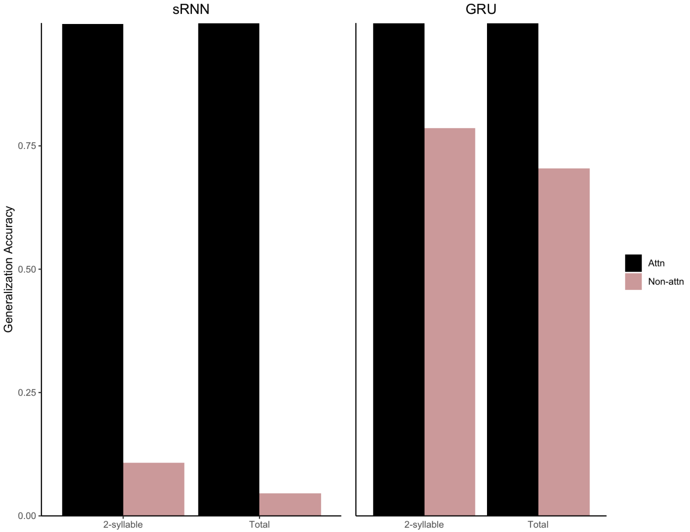
[1] Dolatian and Heinz (2019); also available on GitHub

Experiment 1

- Interaction between reduplication type, recurrence, and attention
  - Total and partial (two-syllable) reduplication
  - sRNN and GRU with and without attention
- Max string length: 9
- 10 symbols alphabet

Attention should improve function generalization across reduplication types and recurrence relations

Experiment 1

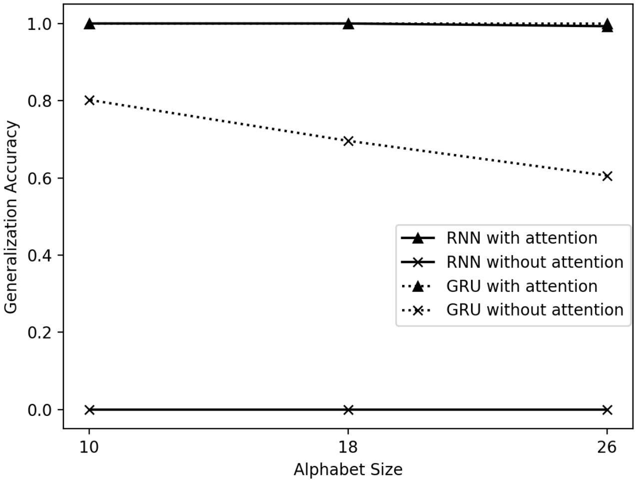


Experiment 2

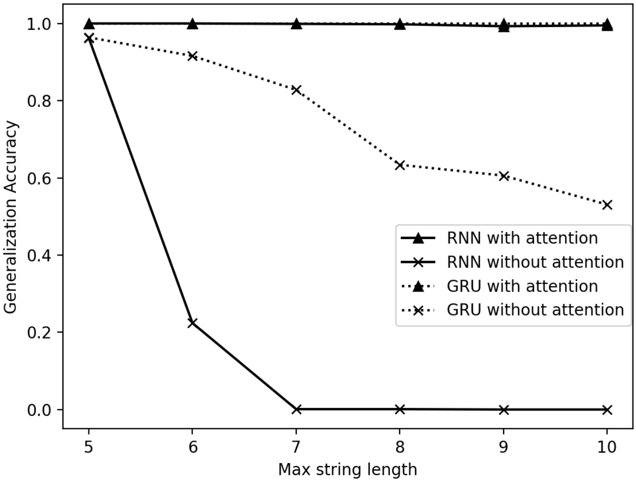
- Effects of alphabet size and range of permitted string lengths
- CV reduplication only
- sRNN/GRU $\times$ attention/non-attention $\times$ 3 alphabet sizes $\times$ 7 length ranges

Network generalization while learning a general reduplication function should be invariant to language composition

Experiment 2



Experiment 2

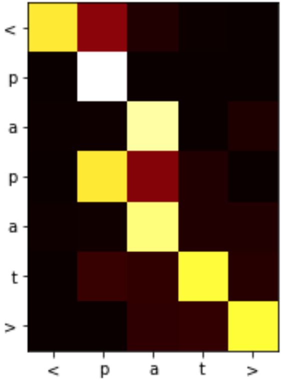


Discussion

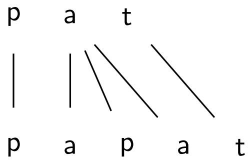
- Networks with global attention learn and generalize all types of reduplication and seem robust to string length and alphabet size
- sRNNs without attention show slightly better generalization of partial reduplication than total reduplication
  - Confound with less attested reduplicant lengths or a bias preferring the regular pattern?
- GRUs perform better than sRNNs across all conditions
  - Without attention not robust to length/alphabet - likely learning heuristics that capture most data rather than a general function

Networks that cannot see material in the input multiple times cannot learn generalizable reduplication

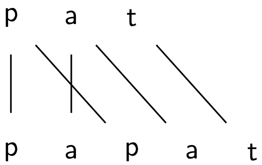
Attention and Origin Semantics

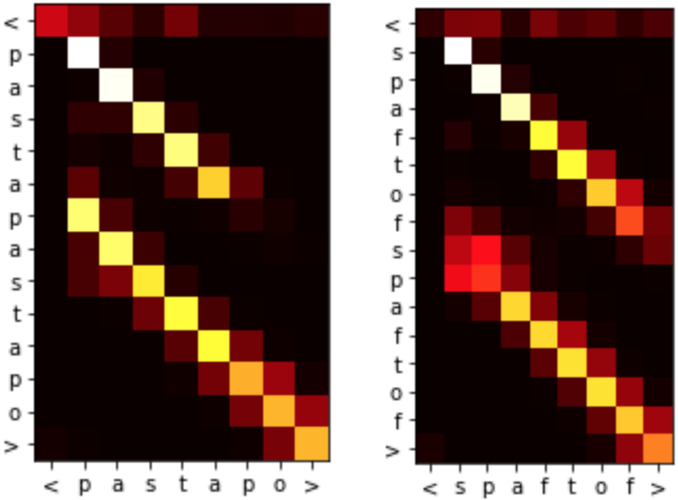


1-Way:



2-Way:





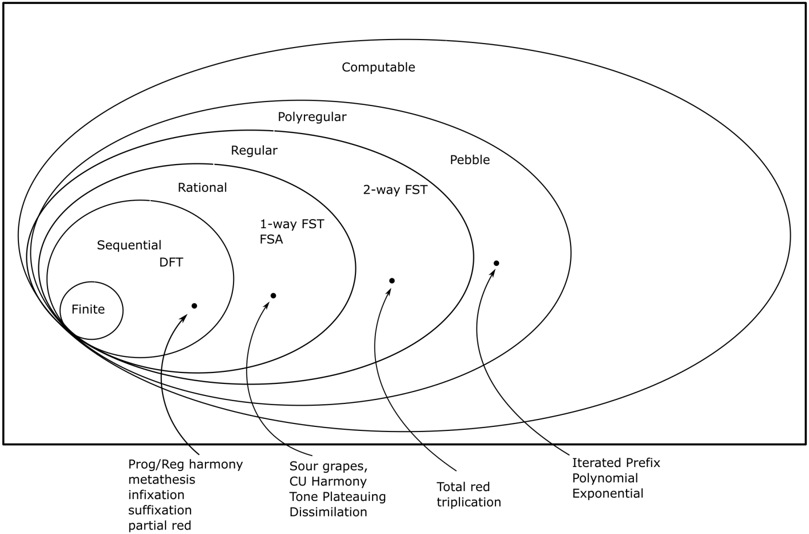
< p a s t a p a s t a p o >
< p a s t a p o >
< s p a f t o f s p a f t o f >
< s p a f t o f >

Summary

- Partial/total reduplication is typologically common, inhabits restricted function classes
- allows testing generalization capacity of neural nets, connecting to 1-way/2-way FSTs
- Attention is necessary and sufficient for robustly learning and generalizing reduplication functions using Encoder-Decoders
- Non-attention networks are limited to a single input pass, approximating 1-way FST.
- Attention networks, approximating 2-way FST, can read the input again during decoding
  - Support for this hypothesis from attention weights
  - IO correspondence relations mirror origin semantics of 2-way FST

Open Areas

- Empirical
- Theoretical
- Experimental


Computable
Polyregular
Regular
Pebble
Rational
2-way FST
1-way FST
FSA
Sequential
DFT
Finite
Prog/Reg harmony
metathesis
infixation
suffixation
partial red
Sour grapes,
CU Harmony
Tone Plateauing
Dissimilation
Total red
triplication
Iterated Prefix
Polynomial
Exponential

Experimental Questions

- Attested and Unattested reduplication patterns
  - What about $w \to w^3$, $w \to ww^r$, $w \to w^w$, ...
- Fine-grained distinctions using phonological harmony patterns (Heinz & Lai 2013)
  - Progressive, regressive, majority rules, ...
- Syntactic transformations (movement, passives, adjunction, ...)
- Different architectures: Transformers (no recurrence, just attention), etc

Global Summary

Three different perspectives

- Typological statements emerge from computability
- Classes of computable functions give principled explanations for attested and unattested processes
- these functions enable interpretability experiments for machines we don't understand

- Linguists can **contribute** and not just **borrow**
- computation has much to study and much to offer typology
- Let a thousand flowers bloom!

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dolatian, H. and Heinz, J. (2019). Redtyp: A database of reduplication with computational models. In *Proceedings of the Society for Computation in Linguistics*, volume 2. Article 3.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.