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

Harmony:

Reduplication:
## Encyclopedia of Types: Possible Linguistic Processes

### Harmony

- **Progressive**
  - $iuuu \rightarrow iii$

- **Regressive**
  - $uuui \rightarrow iii$

- **Sour Grapes**
  - $iuuuu \rightarrow iiii$
  - $iuuau \rightarrow iuuau$

- **Circumambient**
  - $iuii \rightarrow iiii$
  - $iuuu \rightarrow iuuu$

- **Majority Rules**
  - $iuiii \rightarrow iiii$
  - $iuuiu \rightarrow uuuuu$

### Reduplication/Copying

- **Partial**
  - $abcd \rightarrow ababcd$

- **Total:**
  - $abcd \rightarrow abcdabcd$

- **Triplication:**
  - $abcd \rightarrow abcdabcdabcd$

- **Polynomial** $w \rightarrow w|w|$
  - $abcd \rightarrow abcdabcdabcdabcd$

- **Exponential:**
  - $abcd \rightarrow a bb ccc dddd$

- **Iterated prefix:**
  - $abcd \rightarrow a ab abc abcd$
## Encyclopedia of Types: Possible Linguistic Processes

### Harmony
- **Progressive**
  - $iuuu \rightarrow iii$  
- **Regressive**
  - $uuui \rightarrow iii$  
- **Sour Grapes**
  - $iuuuu \rightarrow iiii$  
  - $iuuau \rightarrow iuuau$  
- **Circumambient**
  - $iuii \rightarrow iii$  
  - $iuuu \rightarrow iuuu$  
- **Majority Rules**
  - $iuiiii \rightarrow iiiii$  
  - $iuiuiu \rightarrow uuuuuu$

### Reduplication/Copying
- **Partial**
  - $abcd \rightarrow ababcd$  
- **Total:**
  - $abcd \rightarrow abcdabcd$  
- **Triplication:**
  - $abcd \rightarrow abcdabcdabcd$  
- **Polynomial** $w \rightarrow w^{\lfloor w \rfloor}$:
  - $abcd \rightarrow abcdabcdabcdabcd$  
- **Exponential:**
  - $abcd \rightarrow a b b c c c d d d d$  
- **Iterated prefix:**
  - $abcd \rightarrow a a b a b a b c a b c a b c d a b c d a b c d a b c d$
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: banagañu → bana ~ banagañu
      ‘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: $f : \Sigma^* \rightarrow \{0, 1\}$
  - $\Sigma$: Alphabet of Symbols
  - $\Sigma^*$: set of all possible strings (free monoid on $\Sigma$)

---

p.c. Casey 1996
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.
[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
Regular Languages & Finite-State Automata

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

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

\[ q_0 \xrightarrow{a} q_1 \]

\[ q_0 \xrightarrow{b} q_0 \]

b(a*): \{b, ba, baaaaaa,....\}

\[ q_0 \xrightarrow{b} q_1 \]

\[ q_0 \xrightarrow{a} q_1 \]
Regular Languages & Finite-State Automata

Operator Representation

\[ \alpha = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A^a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix} \]

\[ \omega = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A^b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix} \]

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

\[ = \alpha^\top A^a A^b \omega \]

p.c. Guillaume Rabusseau
Sets to Processes via Semirings

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

<table>
<thead>
<tr>
<th>SEMIRING</th>
<th>function</th>
<th>$\oplus$</th>
<th>$\otimes$</th>
<th>$\overline{0}$</th>
<th>$\overline{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>$\phi : \Sigma^* \to {0, 1}$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>Natural</td>
<td>$\phi : \Sigma^* \to \mathbb{N}$</td>
<td>$+$</td>
<td>$\times$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$\phi : \Sigma^* \to [0, 1]$</td>
<td>$max$</td>
<td>$\times$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>Probability</td>
<td>$\phi : \Sigma^* \to \mathbb{R}_+$</td>
<td>$+$</td>
<td>$\times$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>Log</td>
<td>$\phi : \Sigma^* \to \mathbb{R} \cup {-\infty, +\infty}$</td>
<td>$\oplus_{\log}$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
</tr>
<tr>
<td>Tropical</td>
<td>$\phi : \Sigma^* \to \mathbb{R} \cup {-\infty, +\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
</tr>
<tr>
<td>String</td>
<td>$\phi : \Sigma^* \to \Sigma^* \cup {\infty}$</td>
<td>$\land$</td>
<td>$\cdot$</td>
<td>$\infty$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>Language</td>
<td>$\phi : \Sigma^* \to \mathcal{P}(\Sigma^*)$</td>
<td>$\cup$</td>
<td>$\cdot$</td>
<td>$\emptyset$</td>
<td>${\epsilon}$</td>
</tr>
</tbody>
</table>
Rational vs Computable

Diagram:
- Computable
  - Rational
    - 1-way FST
      - FSA
    - Finite
Rational Morphology: suffixation

Working example: *hold* → *hold-ing*

Input:  

Output:

![Diagram](image_url)
Rational Morphology: suffixation

Working example: \( hold \rightarrow hold-ing \)
Input: \( \times h o l d \times \)
Output:

\[
\begin{align*}
\Sigma : \Sigma \\
q_0, \lambda \quad \times : \lambda \quad \times : -ing \\
q_1, \lambda \\
q_f, \lambda
\end{align*}
\]
Rational Morphology: suffixation

► Working example: \( \textit{hold} \rightarrow \textit{hold-ing} \)

Input: \( \times \ h \ o \ l \ d \ \times \)

Output: \( h \)

\[ \begin{align*}
\Sigma : \Sigma \\
\begin{array}{c}
\text{start} \\
q_0, \lambda \\
q_1, \lambda \\
q_f, \lambda
\end{array}
\end{align*} \]

\( \times : \lambda \quad \times : \text{-ing} \)
Rational Morphology: suffixation

Working example: \( \text{hold} \rightarrow \text{hold-ing} \)

Input: \[ \times \ h \ o \ l \ d \ \times \]
Output: \[ \times \ h \ o \]

\[ q_0, \lambda \rightarrow q_1, \lambda \rightarrow q_f, \lambda \]

\[ \Sigma : \Sigma \]

\[ \times : \lambda \rightarrow - \text{ing} \]
Rational Morphology: suffixation

Working example: $\text{hold} \rightarrow \text{hold-ing}$

Input: $\times$ h o l d $\times$

Output: h o l

$\Sigma : \Sigma$

$$q_0,\lambda \quad \times : \lambda \quad q_1,\lambda \quad \times :-ing \quad q_f,\lambda$$
Rational Morphology: suffixation

Working example: $hold \rightarrow hold-ing$

Input: $\times h o l d \not\times$

Output: $hold$

\[
\begin{align*}
\Sigma : \Sigma \\
\text{start} \rightarrow & q_0,\lambda \rightarrow q_1,\lambda \rightarrow q_f,\lambda
\end{align*}
\]
Rational Morphology: suffixation

Working example: \( \text{hold} \rightarrow \text{hold-ing} \)

Input: \( \times \ h \ o \ l \ d \ \times \)
Output: \( \text{hold} \ \text{i} \ \text{n} \ \text{g} \)

\[ \Sigma : \Sigma \]

start \( \rightarrow q_{0,\lambda} \rightarrow q_{1,\lambda} \rightarrow q_{f,\lambda} \)

\( \times : \lambda \rightarrow \times : -i \text{ng} \)
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

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

References
Regular Functions (Engefiert & 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

1-way

Finite-state transducer

- **a.i**
  - Origin information
    - a.ii
      - p
      - a
      - t

2-way

- **b.i**
  - Origin information
    - b.ii
      - p
      - a
      - t

Diagram:

- **1-way**
  - Start: $q_0$
  - Transitions:
    - $q_0 \xrightarrow{\times:x} q_1$
    - $q_1 \xrightarrow{t:t} q_2$
    - $q_2 \xrightarrow{a:a \sim ta} q_4$
    - $q_4 \xrightarrow{\times:x} q_f$
    - $q_2 \xrightarrow{a:a \sim pa} q_3$
    - $q_3 \xrightarrow{p:p} q_1$
    - $q_1 \xrightarrow{p:p} q_2$
    - $q_2 \xrightarrow{\Sigma:\Sigma} q_4$

- **2-way**
  - Start: $q_0$
  - Transitions:
    - $q_0 \xrightarrow{\times:\lambda:+1} q_1$
    - $q_1 \xrightarrow{C:C:+1} q_2$
    - $q_2 \xrightarrow{V:V:-1} q_3$
    - $q_3 \xrightarrow{\Sigma:\Sigma:-1} q_4$
    - $q_4 \xrightarrow{\times:\sim:+1} q_f$
    - $q_3 \xrightarrow{\times:\sim:+1} q_4$
    - $q_4 \xrightarrow{\times:\lambda:+1} q_f$
Regular vs Rational

![Diagram showing the relationship between Regular and Rational types in the context of open areas and references.](image-url)
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, $w \rightarrow w|w|$
Polyregular vs Regular
No Free Lunch in Linguistics or Machine Learning

Every successful induction system contains biases. Those biases constrain what it can and can’t learn

"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 Dolutian  
(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
Test data

▶ Input-output mappings generated with 2-way FSTs from RedTyp database

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

---

1Dolatian 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

![Bar chart comparing sRNN and GRU models. The chart shows generalization accuracy for 2-syllable and total tasks, with a comparison between attention (Attn) and non-attention (Non-attn) conditions.]
Experiment 2

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

Network generalization while learning a general reduplication function should be invariant to language composition
Experiment 2

The graph illustrates the generalization accuracy of different models as a function of alphabet size. The x-axis represents the alphabet size, ranging from 10 to 26. The y-axis shows the generalization accuracy, ranging from 0.0 to 1.0.

- **RNN with attention**: Solid black line with triangles, showing a general decrease in accuracy as alphabet size increases.
- **RNN without attention**: Dashed black line with crosses, also showing a decrease in accuracy with increasing alphabet size.
- **GRU with attention**: Dotted black line with squares, indicating a general decrease in accuracy as alphabet size increases.
- **GRU without attention**: Dashed-dotted black line with asterisks, showing a decrease in accuracy with increasing alphabet size.

The data suggests that models with attention mechanisms may perform better in terms of generalization accuracy compared to those without attention, especially as the alphabet size increases.
Experiment 2

![Graph showing generalization accuracy vs. max string length for RNN with attention, RNN without attention, GRU with attention, and GRU without attention.]
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: \[\text{p a t} \]

2-Way: \[\text{p a t} \]
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
Experimental Questions

- Attested and Unattested reduplication patterns
  - What about $w \rightarrow w^3$, $w \rightarrow w w^r$, $w \rightarrow w^w$, ...

- 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!
