What Do We Really Mean by (Non)Iterativity?

Jane Chandlee

Haverford College

27 June 2021

NAPhCxi
Rule application

- The concept of iterativity originates with rule-based phonology.
- Johnson (1972): iterative rules reapply to the output string until no targets remain.
- Linear rules (Johnson, 1972) or directional rules (Howard, 1972) apply left-to-right or right-to-left.

$$\text{[-cons]} \rightarrow \text{[+nas]} / \text{[+nas]} \quad \text{(left-to-right)}$$

$$\text{NVVV} \leftrightarrow \text{ÑVVV} \leftrightarrow \text{ÑVVV} \leftrightarrow \text{ÑVVV}$$
Iterativity parameter

\[ [-\text{cons}] \rightarrow [+\text{nas}] / [+\text{nas}] \_ \_ \]

+iterative \hspace{1cm} -iterative

\[ \text{NVVV} \leftrightarrow \text{N}\acute{\text{V}}\acute{\text{V}}\acute{\text{V}} \hspace{1cm} \text{NVVV} \leftrightarrow \text{N}\acute{\text{V}}\acute{\text{V}}\acute{\text{V}} \]

Simultaneous application

‘To apply a rule, the entire string is first scanned for segments that satisfy the environmental constraints of the rule. After all such segments have been identified in the string, the changes required by the rule are applied simultaneously’ (Chomsky and Halle, 1968).

Also called *global application* (Bale and Reiss, 2018).
Simultaneous application

What about NVVV \rightarrow N\ddot{V}\ddot{V}\ddot{V}?
Parenthesis-star notation

\([-\text{cons}] \to [+\text{nasal}] / [+\text{nasal}] ([-\text{cons}])^* \_\]

expands to

\([-\text{cons}] \to [+\text{nasal}] / [+\text{nasal}] \_\]
\([-\text{cons}] \to [+\text{nasal}] / [+\text{nasal}] [-\text{cons}] \_\]
\([-\text{cons}] \to [+\text{nasal}] / [+\text{nasal}] [-\text{cons}] [-\text{cons}] \_\]

\(\text{NVVV} \to \text{N}\text{V}\text{V}\text{V}\text{V} \)
Parenthesis-star notation

\([-\text{cons}] \rightarrow [+\text{nasal}] / [+\text{nasal}] \ ([-\text{cons}])^* \]

Issues:

- Loss of target-trigger adjacency
- Redundancy: specification of target is repeated in the structural description
Optimality Theory (Prince and Smolensky, 1993, 2004) is inherently iterative.

<table>
<thead>
<tr>
<th>/NVVV/</th>
<th>*[+nas][−nas]</th>
<th>IDENT-NAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>[NVVV]</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>[N˜VVV]</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>[N˜V˜VV]</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>[N˜V˜V˜V]</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>☞[NVṼV]</td>
<td></td>
<td>***</td>
</tr>
</tbody>
</table>
Emergent Noniterativity Hypothesis (Kaplan, 2008): No formal entity in phonological grammars may require noniterativity.

Prediction: ‘There is no phenomenon that must be analyzed with a self-feeding rule that is not permitted to apply to its own output’ (4).
Emergent noniterativity

(1) Lango (Nilo-Saharan; Noonan, 1992)
   a. /bɔŋø-ni/ → [bɔŋo-ni], ‘your (sg.) dress’
   b. /cəŋø-ni/ → [cəŋo-ni], ‘your (sg.) beer’

- Positional licensing: suffix [ATR] needs to be linked to the root (Kaplan, 2008).
Noniterativity

Noniterative = ‘should’ iterate, but doesn’t.

1. Classified as a type of process that is considered inherently iterative.

2. Would be described with a self-feeding rule (if we were using rules).
What does FLT say?

- Formal language theory analyses identify computational properties of rule/constraint extensions:
  - \{ (NV, N\tilde{V}), (VN, VN), (NVV, N\tilde{V}V), \ldots \} (‘noniterative’)
  - \{ (NV, N\tilde{V}), (VN, VN), (NVV, N\tilde{V}V), \ldots \} (‘iterative’)

Implementing application modes

- Simultaneous application = match context on the input string
- Iterative application = match context on the output string

Hulden (2009)’s finite-state compiler for phonological rules:

\[ a \rightarrow b \mid \mid a \_ a \] match both contexts on the input

\[ a \rightarrow b \backslash\backslash a \_ a \] left context on input, right context on output

\[ a \rightarrow b \text{ // } a \_ a \] left context on output, right context on input

(See also Kaplan and Kay (1994).)
ISL and OSL

\[ a \rightarrow b / b \]

Input Strictly Local

Output Strictly Local

simultaneous
baaa \rightarrow bbbaa

iterative
baaa \rightarrow bbbb
ISL and OSL
ISL and OSL

noniter

iterative
ISL and OSL

noniterative

iterative
OSL and noniterativity

\[
/\text{NVVV}/ \leftrightarrow [\text{N\text{"}{\text{\`}{\text{"}}}\text{\text{"}}}\text{\text{"}}}]\quad \text{iterative}
\]

\[
/\text{NVVV}/ \leftrightarrow [\text{N\text{"}{\text{\`}{\text{"}}}\text{\text{"}}}\text{\text{"}}}]\quad \text{noniterative}
\]
OSL and noniterativity

\[ /NVVV/ \mapsto [N\breve{N}\breve{N}\breve{N}] \quad \text{OSL} \]

\[ /NVVV/ \mapsto [N\breve{N}V\breve{V}] \quad \text{ISL and OSL} \]
OSL and noniterativity

OSL (iterative)

OSL (noniterative)
ISL and OSL

noniterative

iterative
‘True’ noniterativity

Definition (First attempt)
A noniterative map is a ‘self-feeding’ ISL function.
Triggers and Outputs

Let $\text{Triggers}$ be the set of segments that trigger a process, and let $\text{Outputs}$ be the set of segments that result from that process.

A phonological map is

1. $\text{output-nondistinct}$ if $\text{Outputs} \cap \text{Triggers} \neq \emptyset$.
2. $\text{output-distinct}$ if $\text{Outputs} \cap \text{Triggers} = \emptyset$. 
Triggers and Outputs

\[
/NVVV/ \mapsto [NVVV] \quad /NVVV/ \mapsto [NVVV]
\]

\[
\text{Triggers} = \{N, \tilde{V}\} \quad \text{Triggers} = \{N\}
\]

\[
\text{Outputs} = \{\tilde{V}\} \quad \text{Outputs} = \{\tilde{V}\}
\]

output-nondistinct \quad \text{output-distinct}
<table>
<thead>
<tr>
<th></th>
<th>output-nondistinct</th>
<th>output-distinct</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OSL</strong></td>
<td>iterative</td>
<td>‘noniterative’</td>
</tr>
<tr>
<td></td>
<td>output-nondistinct</td>
<td>output-distinct</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>OSL</td>
<td>iterative</td>
<td>‘noniterative’</td>
</tr>
<tr>
<td>ISL</td>
<td>noniterative</td>
<td>‘noniterative’</td>
</tr>
</tbody>
</table>
ISL and OSL

<table>
<thead>
<tr>
<th></th>
<th>output-nondistinct</th>
<th>output-distinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSL</td>
<td>iterative</td>
<td>‘noniterative’</td>
</tr>
<tr>
<td>ISL</td>
<td>noniterative</td>
<td>‘noniterative’</td>
</tr>
</tbody>
</table>
Revised definition

Definition (Second attempt)
A noniterative map is an output-nondistinct ISL function.
A note on local versus long-distance

- If we assume vowel harmony rules include $C_0$, they are trivially neither ISL nor OSL.

$$V_{-\alpha} \rightarrow +\alpha \ / \ V_{+\alpha} \ C_0$$

- Will instead use strings of vowels, which can be interpreted in two ways:
  1. The map applies on a vowel tier.
  2. The number of consonants is bounded by syllable structure and can be ignored (for presentation purposes).
Bengali ATR harmony

\{ɛ, ɔ\} → \{e, o\} in nouns when they precede a high vowel (Mahanta, 2007).

\kotʰa\ ‘spoken words’ \kotʰito\ ‘uttered’ \kɔthoniyo\ ‘speakable’
\kolpo\ ‘resembling’ \kolpito\ ‘invented’ \kolponiyo\ ‘imaginable’

/ɔːi/ → [ɔːi]
Bengali ATR harmony

- **Triggers** = \{i, u\} (only high vowels can trigger harmony)
- **Outputs** = \{e, o\}
- ✓ output-distinct
- Both ISL and OSL.
Crimean Tatar labial harmony


(2) Central Crimean Tatar (Nmzr-Poss.3S)
   a. /tuz-lwγ-w/ [tuz-luγ-w] ‘salt’
   b. /kyz-lyg-i/ [kyz-lyg-i] ‘autumn’
   c. /toz-lwγ-w/ [toz-luγ-w] ‘dust’
   d. /køz-lyg-i/ [køz-lyg-i] ‘eye’
Crimean Tatar labial harmony

- **Triggers** = \{y, ø, u, o\}
- **Outputs** = \{y, ø, u, o\}
Crimean Tatar labial harmony

- \textbf{Triggers} = \{\#y, \#\emptyset, \#u, \#o\} (only initial vowels are triggers)
- \textbf{Outputs} = \{y, \phi, u, o\}
- \checkmark\text{ output-distinct}
- Both ISL and OSL.
Crimean Tatar labial harmony

\[ V_{-rd} \rightarrow [+rd] / \# V_{+rd} \]

\*[\# [+rd, −rd] (McCollum and Kavitskaya, 2018)
Kazakh labial harmony

[+rd] vowel triggers harmony on following vowel (McCollum and Kavitskaya, 2018):

a. /mojən-də/ [mojʊn-də] ‘neck-Acc’
b. /tʊr-məs-ə-ңəŋ/ [tʊr-mus-ə-ңəŋ] ‘live-NMZR-POSS.3-GEN’
c. /kino-m-əz-dəŋ/ [kino-m-ʊz-dəŋ] ‘movie-POSS.1-PL-GEN’
Kazakh labial harmony

[+rd] vowel triggers harmony on following vowel (McCollum and Kavitskaya, 2018):

a. /mojən-də/ [mojʊn-də] ‘neck-Acc’
b. /tʊr-məs-ə-nəŋ/ [tʊr-mus-ə-nəŋ] ‘live-NMZR-POSS.3-GEN’
c. /kino-məz-dəŋ/ [kino-mʊz-dəŋ] ‘movie-POSS.1-PL-GEN’

- ✔ output-nondistinct
- Necessarily ISL → Noniterative
Dissimilation

(3) \( L \rightarrow R \) / \( L \) 

- **Triggers** = \{L\}
- **Outputs** = \{R\}
- ✔ output-distinct
- Predicts both ISL and OSL will produce a noniterative map.
What is a noniterative dissimilation map?

(4) /LLLL/ \rightarrow [LRLL]?

Simultaneous application gives:

(5) /LLLL/ \rightarrow [LRRR]
Dissimilation

- Tianjin tone sandhi (Chen, 1986; Hung, 1987; Tan, 1987; Zhang, 1987; Chen, 2000; Lin, 2008; Wee, 2010)

(6)  
  a. $R \rightarrow H / ___ R$
  b. $RRR \rightarrow HHR$
Dissimilation

ISL

OSL

LLL LRRR

LLL LRLR
Dissimilation: future work

- Current definitions don’t predict iterativity in OSL dissimilation maps.
- Not surprising, since dissimilation is actually self-bleeding...
Two-sided contexts

- Rules with two-sided contexts are also necessarily ISL.

‘There are a number of well-motivated rules in a large number of languages in which the environment is of the form X___Y...So far as I can tell, the direction of iteration in these rules is irrelevant: one can go equally well from right-to-left or from left-to-right’ (Lightner, 1972, pg. 365).
Two-sided contexts

- Do these rules ever iterate?
- Survey of P-base (Mielke, 2008): 4560 rules
  - 1202 have a two-sided context
  - 243 of these are epenthesis
  - 947 can’t iterate (i.e., aren’t self-feeding)
  - Data not likely available to determine whether the remaining candidates iterate.
Examples

(7) Estonian (Uralic; Harms, 1962)
/e/ → [j] / V — V

/eeee/ ↔ [ejje]? [ejee]? [eeje]?
Examples

(8) Maasai (Nilotic; Hollis, 1971)
    /i/ → ∅ / i ∈ {i, u, s, n, ι, η, l, r, w, j}

(9) a. /iik/, ‘to clean (teeth)’
    b. /i-iik/ → iik, ‘thou cleanest (teeth)’
Conclusions

- Some cases of noniterativity are indeed ‘emergent’ (i.e., do not require ISL).
- Noniterative maps are identified as output-nondistinct and necessarily ISL.
- Establishing whether a map is ISL, OSL, or both adds to our understanding of the respective roles of input- and output-oriented computation in the phonological grammar.
Acknowledgements

Thank you!

(And thanks to earlier audiences ‘at’ Johns Hopkins University and Tel Aviv University.)
Appendix: Delaying output

\[ a \rightarrow b \mid a \_ \_ a \]

**ISL**

```
λ → a → aa → a:b
```

**OSL**

```
λ → a
```

```
References


