Locality in Phonological Interactions: A Computational Account

Jane Chandlee

Haverford College

University of California - San Diego
January 18, 2017
Many phonological input-output maps can be characterized by a small, computationally-restrictive and learnable class of functions, including *opaque interactions*.

This result is part of a larger research program that aims to understand the computational nature of phonological patterns and how their computational properties can inform how they are learned.
Establish a framework for classifying phonological maps based on their computational complexity.

- Subregular hierarchy of function classes: ISL, OSL, TSL

Use this framework to present a computational analysis of opaque map interactions.

- 7 distinct cases of opacity are shown to be ISL functions.
What do we know about the computational complexity of phonotactics and phonological maps?

Starting point: both are *regular* (i.e., finite state describable)

Proposal: both are in fact *subregular* (i.e., describable with proper subsets of regular)

Why is this restriction desirable?
- Better fit to the typology
- Learnability advantages
A formal language is a set of strings.  
A phonotactic constraint can be modeled with the set of strings that do not violate it.

(1) \{ amba, nanta, ama, ... \}

What is the complexity of the formal languages that model phonotactics?
Subregular hierarchy of languages

Regular

Star-free

Tier-based Strictly Local

Locally Threshold Testable

Locally Testable

Strictly Local

Piecewise Testable

Strictly Piecewise

(Rogers and Pullum, 2011; Rogers et al., 2013; Heinz, 2010; Heinz et al., 2011; McMullin, 2016)
Subregular hierarchy of languages

Tier-based Strictly Local

(Rogers and Pullum, 2011; Rogers et al., 2013; Heinz, 2010; Heinz et al., 2011; McMullin, 2016)
Phonological maps

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

\[(2)\quad \{ \text{(anba, amba)}, \text{(nanta, nanta)}, \text{(ama, ama)}, \ldots \}\]

- What is the *complexity* of phonological maps?
Subregular hierarchy of maps

Regular relations

Subsequential functions

Tier-based Strictly Local functions

Strictly Local functions

(Johnson, 1972; Kaplan and Kay, 1994; Mohri, 1997; Chandlee, 2014; Chandlee et al., 2017)
Computational complexity of phonological maps

**Regular relations** (Johnson, 1972; Kaplan and Kay, 1994)

\[ \downarrow \]

**Subsequential functions** (Mohri, 1997)

\[ \downarrow \]

**Tier-based strictly local functions** (Chandlee et al., 2017)

\[ \downarrow \]

**Strictly local functions** (Chandlee, 2014)
Strictly Local functions

- Output string is computed \textit{locally}, depending only on a bounded number of previous segments.
- Two varieties:
  - Input Strictly Local: output depends on previous input symbols
  - Output Strictly Local: output depends on previous output symbols
Input Strictly Local (ISL) function
Example: Input Strictly Local (ISL) function

(3) Korean (Lee and Pater, 2008)
\[ /\text{papmul}/ \mapsto [\text{pammul}] \] ‘rice water’
Example: Input Strictly Local (ISL) function

(3) Korean (Lee and Pater, 2008)

/papmul/ $\mapsto$ [pammul] ‘rice water’
Example: Input Strictly Local (ISL) function

(3)  Korean (Lee and Pater, 2008)
/papmul/ $\mapsto$ [pammul]  ‘rice water’
Example: Input Strictly Local (ISL) function

(3) Korean (Lee and Pater, 2008)

/papmul/ $\mapsto$ [pammmul] ‘rice water’
Example: Input Strictly Local (ISL) function

(3) Korean (Lee and Pater, 2008)

/papmul/ $\mapsto$ [pammmul] ‘rice water’
Example: Input Strictly Local (ISL) function

\(3\)  
Korean (Lee and Pater, 2008)  
/pamul/ \(\mapsto\) [pammul]  
‘rice water’

\(\times\) p a p m u l \(\times\)

\(\lambda\) pa \(\lambda\) mm u l
Example: Input Strictly Local (ISL) function

(3)  Korean (Lee and Pater, 2008)
/papmul/ $\mapsto [pammmul]$  ‘rice water’

Window size is 2: this map is 2-ISL.
Output Strictly Local function
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

/pɛŋjawasan/ \(\mapsto\) [pɛŋãũwãsan] ‘supervision’
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)
\[ /pəŋjawasan/ \mapsto [pəŋãwãsan] \quad \text{‘supervision’} \]
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

\[ /\text{pərjawasan}/ \rightarrow [\text{pərjāwāsān}] \] ‘supervision’

\[ \times \text{ p e n a w a s a n } \times \]

\[ \text{ p e n a w a s a n } \]
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)
/pəŋjawasan/ \(\mapsto\) [pəŋãwãsan] ‘supervision’

\(\times\) peŋja\(\tilde{a}\)wan\(\tilde{a}\)

\(\times\) peŋja\(\tilde{a}\)wan\(\tilde{a}\)
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

/pəŋjawasan/ $\mapsto$ [pəŋãwãsãn]  ‘supervision’

$\times$  p e ŋ a w a s a n  $\times$

p e ŋ ã  w
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

/pəŋjawasan/ $\mapsto$ [pəŋãwãsan] ‘supervision’
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

\[\text{/pəŋjawasan/} \rightarrow [pəŋãwãsan] \quad \text{‘supervision’}\]
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)
\[ /\text{jawasan} / \rightarrow [p\text{e\text{j}a\text{w}a\text{s}a\text{n}}] \]
‘supervision’

\[ \times \text{p e \text{j} a \text{w a s a n} } \times \]
\[ \text{p e \text{j} \text{\=a} \text{\=w a s a} } \]
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)

/pəŋjawasan/ → [pəŋañwäsan] ‘supervision’
Example: Output Strictly Local (OSL) function

(4) Johore Malay (Onn, 1980)
\[pəŋjawasan\] → [pəŋãwãsan] ‘supervision’

Window size is 2: this map is 2-OSL.
Long-distance (unbounded) assimilation

(5) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ $\mapsto$ [tunikini] ‘we ground’

b. /kudumukisila/ $\mapsto$ [kudumukisina] ‘to cause to jump for’
Long-distance (unbounded) assimilation

(5) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ $\mapsto$ [tunikini] ‘we ground’
b. /kudumukisila/ $\mapsto$ [kudumukisina] ‘to cause to jump for’
(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ $\mapsto$ [tunikini] ‘we ground’

b. /kudumukisila/ $\mapsto$ [kudumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the tier: $T = \{n, m, d, l\}$
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ \(\rightarrow\) [tunikini] ‘we ground’

b. /kudumukisila/ \(\rightarrow\) [kudumukisina] ‘to cause to jump for’

\[\mathbb{V} \triangleleft \{n, m, d, l\} \triangleright \mathbb{T}\]

- Designate a subset of the alphabet as the tier: \(T = \{n, m, d, l\}\)
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ \(\rightarrow\) [tunikini] ‘we ground’

b. /kudumukisila/ \(\rightarrow\) [kudumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the tier: \(T = \{n, m, d, l\}\)
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ \(\mapsto\) [tunikini] ‘we ground’

b. /kudumukisila/ \(\mapsto\) [kudumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the tier: \(T = \{n, m, d, l\}\)
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ $\mapsto$ [tunikini] ‘we ground’

b. /kudumukisila/ $\mapsto$ [kudumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the *tier*: $T = \{n, m, d, l\}$

\[
\times t u n i k i d i \times
\]

\[
\times t u n i k \times
\]
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tunikidi/ $\mapsto$ [tunikini] ‘we ground’

b. /kudumukisila/ $\mapsto$ [kudumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the tier: $T = \{n, m, d, l\}$
Example: Tier-based Strictly Local (TSL) function

(6) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

a. /tuŋikidi/ $\rightarrow$ [tuŋikini] ‘we ground’

b. /kuɗumukisila/ $\rightarrow$ [kuɗumukisina] ‘to cause to jump for’

- Designate a subset of the alphabet as the tier: $T = \{n, m, d, l\}$
Long-distance assimilation with blocking

Slovenian (Jurgec, 2011; McMullin, 2016)

a. spif $\mapsto$ ñspif ‘(you) sleep’

b. zaklônîstfe $\mapsto$ ñaklônîstfe ‘bomb shelter’

c. nasiti$\mathbf{j}$ $\mapsto$ nasiti$\mathbf{j}$ ‘(you) feed’

▶ The blocking segments go on the tier.

$\times$ n a s i t i $\mathbf{j}$ $\times$
Long-distance assimilation with blocking

Slovenian (Jurgec, 2011; McMullin, 2016)

a. spiʃ ←> ʃpiʃ  ‘(you) sleep’
b. zaklɒnɪʃtʃe ←> ʒaklɒnɪʃtʃe  ‘bomb shelter’
c. nasitiʃ ←> nasitiʃ  ‘(you) feed’

- The blocking segments go on the tier.

\[
\times \text{ n a s i t i } \quad i \quad i \quad \times
\]

\[
\times \text{ i } \quad i \quad \times
\]
Long-distance assimilation with blocking

Slovenian (Jurgec, 2011; McMullin, 2016)

a. spiʃ → śpiʃ ‘(you) sleep’

b. zaklənɨstʃe → ʒaklənɨstʃe ‘bomb shelter’

c. nasitiʃ → nasitiʃ ‘(you) feed’

▷ The blocking segments go on the tier.
Long-distance assimilation with blocking

Slovenian (Jurgec, 2011; McMullin, 2016)

a. $\text{spiʃ} \rightarrow \text{jpiʃ}$ ‘(you) sleep’
b. $\text{zaklɔniʃtʃe} \rightarrow \text{žaklɔniʃtʃe}$ ‘bomb shelter’
c. $\text{nasitʃ} \rightarrow \text{nasitiʃ}$ ‘(you) feed’

- The blocking segments go on the tier.
Long-distance assimilation with blocking

Slovenian (Jurgec, 2011; McMullin, 2016)

a. spifs \rightarrow spifs \quad \text{‘(you) sleep’}

b. zaksfantsife \rightarrow zaksfantsife \quad \text{‘bomb shelter’}

c. nasitsif \rightarrow nasitsif \quad \text{‘(you) feed’}

- The blocking segments go on the tier.

\[ \times \quad n \quad a \quad s \quad i \quad t \quad i \quad f \quad \times \]

\[ \times \quad s \quad i \quad t \quad i \quad f \]
Example classifications

- ISL and OSL are argued to include all local phonological processes (assimilation, dissimilation, insertion, deletion, metathesis) (Chandlee, 2014; Chandlee et al., 2015)
- Many morphological ‘maps’ are also ISL (Chandlee, under review)
- TSL is conjectured to include long-distance processes
Characterizations

<table>
<thead>
<tr>
<th>Class</th>
<th>Finite-state</th>
<th>Language-theoretic</th>
<th>Logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OSL</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TSL</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

(Chandlee (2014); Chandlee et al. (2014, 2015, 2017), Chandlee and Lindell (to appear))
Input Strictly Local (ISL): FST characterization

(7) \(/p\text{apmul}/ \mapsto [p\text{ammmul}]\)
Output Strictly Local (OSL): FST characterization

(8) /pəŋjawasan/ $\mapsto$ [pəŋãwãsan]
Tier-based Strictly Local (TSL): FST characterization

(9) /tunikidi/ $\mapsto$ [tunikini]
FST Characterizations

Regular relations (describable with FSTs)

Subsequential functions (describable with deterministic FSTs)

Tier-based strictly local functions (states are $T^{\leq k-1}$)

Strictly local functions (transitions follow recent input/output)
Map interactions

What about map interactions?
Opaque phonological generalizations result from the interaction of two processes, one of which is not ‘surface-true’.

The 7 distinct types of opacity from McCarthy (2007); Baković (2007); Kavitskaya and Staroverov (2010) can all be described with ISL maps.

The needed information for determining the correct output at any point in the computation is found in a contiguous window of the input.
Case study: opaque phonological generalizations

(10) Counterbleeding

a. Lowering: [+long] → [−high]
b. Shortening: V → [−long] / _ C #

(11) Yokuts (McCarthy, 1999)

/ili:l/ → [ilel], ‘might fan’

- Lowering over-applies.
Case study: opaque phonological generalizations

(12) Counterbleeding
   a. Lowering: [+long] $\rightarrow$ [−high]
   b. Shortening: V $\rightarrow$ [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)
/ili:l/ $\mapsto$ [ilel], ‘might fan’
Case study: opaque phonological generalizations

(12) Counterbleeding
   a. Lowering: [+long] $\rightarrow$ [−high]
   b. Shortening: V $\rightarrow$ [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)
    /?ili:l/ $\leftrightarrow$ [?ilel], ‘might fan’
Case study: opaque phonological generalizations

(12) Counterbleeding
   a. Lowering: [+long] → [−high]
   b. Shortening: V → [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)
    /?i:l/ → [?ilel], ‘might fan’
Case study: opaque phonological generalizations

(12) Counterbleeding

a. Lowering: [+long] → [−high]
b. Shortening: V → [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)

/ʔiliːl/ → [ʔilel], ‘might fan’
Case study: opaque phonological generalizations

(12) Counterbleeding
   a. Lowering: [+long] → [−high]
   b. Shortening: V → [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)
    /?ili:l/ → [?ilel], ‘might fan’
Case study: opaque phonological generalizations

(12) Counterbleeding
   a. Lowering: [+long] $\rightarrow$ [−high]
   b. Shortening: V $\rightarrow$ [−long] / ___ C #

(13) Yokuts (McCarthy, 1999)
    /?ili:l/ $\mapsto$ [?ilel], ‘might fan’

$\times$    ? i  l    i:   l   $\times$

?    i  l  λ  λ   el
Case study: opaque phonological generalizations

(14) Counterfeeding on focus
   a. Deletion: i → ∅ / ___ CV
   b. Raising: a → i / ___ CV

(15) Bedouin Arabic (McCarthy, 1999)
   /katab/ → [kitab], ‘he wrote’

- Deletion underapplies.
Case study: opaque phonological generalizations

(16) Counterfeeding on focus
  a. Deletion: i → ∅ / ___ CV
  b. Raising: a → i / ___ CV

(17) Bedouin Arabic (McCarthy, 1999)
/katab/ → [kitab], ‘he wrote’

\[
\text{\texttimes} \quad k \quad a \quad t \quad a \quad b \quad \text{\texttimes} \\
\text{\texttimes} \quad k
\]
Case study: opaque phonological generalizations

(16) Counterfeeding on focus

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

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

(17) Bedouin Arabic (McCarthy, 1999)

/katab/ $\mapsto$ [kitab], ‘he wrote’

\[
\begin{array}{cccccc}
\times & k & a & t & a & b & \times \\
\text{k} & \lambda
\end{array}
\]
Case study: opaque phonological generalizations

(16) Counterfeeding on focus
   a. Deletion: i → ∅ / ___ CV
   b. Raising: a → i / ___ CV

(17) Bedouin Arabic (McCarthy, 1999)
/katab/ ↦ [kitab], ‘he wrote’

       ⊗ k a t a b ⊗
      k λ λ
Case study: opaque phonological generalizations

(16) Counterfeeding on focus
   a. Deletion: i → ∅ / __ CV
   b. Raising: a → i / __ CV

(17) Bedouin Arabic (McCarthy, 1999)
    /katab/ → [kitab], ‘he wrote’

      × k a t a b ×
      k λ λ it
Case study: opaque phonological generalizations

(16) Counterfeeding on focus

a. Deletion: i → ∅ / ___ CV
b. Raising: a → i / ___ CV

(17) Bedouin Arabic (McCarthy, 1999)
/katab/ → [kitab], ‘he wrote’

×
k   a   tab   ×

k   λ   λ   it   λ
Case study: opaque phonological generalizations

(16) Counterfeeding on focus
a. Deletion: i → ∅ / __ CV
b. Raising: a → i / __ CV

(17) Bedouin Arabic (McCarthy, 1999)
/katab/ → [kitab], ‘he wrote’

\[\begin{array}{cccccc}
\times & k & a & t & a & b & \times \\
& k & \lambda & \lambda & it & \lambda & ab
\end{array}\]
Case study: opaque phonological generalizations

(18) Self-destructive feeding (Baković, 2007)

a. Epenthesi s: $\emptyset \rightarrow i$ / C(+)___C#

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

(19) Turkish (Sprouse, 1997)

/bebek+n/ $\mapsto$ [bebein], ‘your baby’

- Deletion destroys the environment for epenthesi s.
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
   a. Epenthesis: $\emptyset \rightarrow i / C(+)_C#$
   b. Deletion: $k \rightarrow \emptyset / V_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\mapsto [bebein]$, ‘your baby’

\[
\begin{array}{cccccccccc}
\otimes & b & e & b & e & k & + & n & \otimes \\
\otimes & b & & & & & & & \\
\end{array}
\]
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
   a. Epenthesis: $\emptyset \rightarrow i / C(+)\_\_C$
   b. Deletion: $k \rightarrow \emptyset / V\_\_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\mapsto$ [bebein], ‘your baby’

\begin{array}{cccccccc}
\times & b & e & b & e & k & + & n & \times \\
\times & b & e
\end{array}
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
a. Epenthesis: $\emptyset \rightarrow i / C(+)\_C#$
b. Deletion: $k \rightarrow \emptyset / V\_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\rightarrow$ [bebein], ‘your baby’
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)

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

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

(21) Turkish (Sprouse, 1997)

/bebek+n/ $\mapsto$ [bebein], ‘your baby’
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
   a. Epenthesis: $\emptyset \rightarrow i / C(+)\_C$
   b. Deletion: $k \rightarrow \emptyset / V\_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ \rightarrow [bebein], ‘your baby’

\begin{table}[h]
  \centering
  \begin{tabular}{ccccccc}
    & b & e & b & e & k & + & n & ×
  \end{tabular}
\end{table}
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
a. Epenthesis: $\emptyset \rightarrow i / C(+)_C#$
b. Deletion: $k \rightarrow \emptyset / V_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\mapsto$ [bebein], ‘your baby’

× b e b e k + n ×

b e b e λ λ
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
    a. Epenthesis: $\emptyset \rightarrow i / C(+)\_C$
    b. Deletion: $k \rightarrow \emptyset / V\_+V$

(21) Turkish (Sprouse, 1997)
    /bebek+n/ $\mapsto$ [bebein], ‘your baby’
Case study: opaque phonological generalizations

(20) Self-destructive feeding (Baković, 2007)
  a. Epenthesis: $\emptyset \rightarrow i / C(+)\_C\#$
  b. Deletion: $k \rightarrow \emptyset / V\_+V$

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\mapsto$ [bebein], ‘your baby’
Summary: ISL Opaque Maps

<table>
<thead>
<tr>
<th>Opacity Type</th>
<th>Language</th>
<th>$k$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross-derivational feeding</td>
<td>Lithuanian</td>
<td>$k = 2$</td>
</tr>
<tr>
<td>counterbleeding</td>
<td>Yokuts</td>
<td>$k = 3$</td>
</tr>
<tr>
<td>fed counterfeeding</td>
<td>Tundra Nenets</td>
<td>$k = 3$</td>
</tr>
<tr>
<td>counterfeeding on environment</td>
<td>Bedouin Arabic</td>
<td>$k = 3$</td>
</tr>
<tr>
<td>counterfeeding on focus</td>
<td>Bedouin Arabic</td>
<td>$k = 3$</td>
</tr>
<tr>
<td>non-gratuitous feeding</td>
<td>Classical Arabic</td>
<td>$k = 4$</td>
</tr>
<tr>
<td>self-destructive feeding</td>
<td>Turkish</td>
<td>$k = 5$</td>
</tr>
</tbody>
</table>
Implications for phonological learning

- The regular relations are not learnable from positive data...
- but the ISL functions are (Chandlee et al., 2014; Jardine et al., 2014)
Conclusion

- There is an increasing amount of evidence that phonological maps are subregular in nature, and this property is not dependent on analyses of individual generalizations.
- In particular, opaque map interactions belong to one computationally restrictive and learnable subregular class, the ISL functions.
Future work

- Complete the hierarchy of subregular maps, particularly the regions for long-distance phenomena
- Further explore the logical characterizations of these classes
- Investigate other types of map interaction
  - Blocking
  - Co-existing long-distance maps with and without distinct tiers
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)
Appendix: Additional examples of opacity

(22) Counterfeeding on environment
    a. Raising: a → i/ ___ CV
    b. Vocalization: G → V / C ___ #

(23) Bedouin Arabic (McCarthy, 1999)
    /badw/ ↔ [badu], ‘Bedouin’

  Raising underapplies.
Appendix: Additional examples of opacity

(24) Counterfeeding on environment
   a. Raising: $a \rightarrow i / \_ \_ \text{CV}$
   b. Vocalization: $G \rightarrow V / C \_ \_ \#$

(25) Bedouin Arabic (McCarthy, 1999)
    /badw/ $\rightarrow$ [badu], ‘Bedouin’

\[ \times \quad b \\ a \\ d \\ w \\ \times \]
\[ \times \quad b \]
Appendix: Additional examples of opacity

(24) Counterfeeding on environment
a. Raising: a → i / ___ CV
b. Vocalization: G → V / C ___ #

(25) Bedouin Arabic (McCarthy, 1999)
/badw/ → [badu], ‘Bedouin’

\[ \times \ b \ a \ d \ w \ \times \]
\[ \times \ b \ \lambda \]
Appendix: Additional examples of opacity

(24) Counterfeeding on environment

a. Raising: a → i / __ CV
b. Vocalization: G → V / C __ #

(25) Bedouin Arabic (McCarthy, 1999)
/badw/ → [badu], ‘Bedouin’

\[
\begin{array}{ccccccc}
\times & b & a & d & w & \times \\
\times & b & \lambda & & \lambda & \times
\end{array}
\]
Appendix: Additional examples of opacity

(24) Counterfeeding on environment

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

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

(25) Bedouin Arabic (McCarthy, 1999)

/badw/ $\mapsto$ [badu], ‘Bedouin’
Appendix: Additional examples of opacity

(24) Counterfeeding on environment
a. Raising: a → i / ___ CV
b. Vocalization: G → V / C ___ #

(25) Bedouin Arabic (McCarthy, 1999)
/badw/ → [badu], ‘Bedouin’

\[ \times \quad b \quad a \quad d \quad w \quad \times \]

\[ b \quad \lambda \quad \lambda \quad \lambda \quad \text{ad} \quad \text{u} \]
Appendix: Additional examples of opacity

(26) Non-gratuitous feeding (Baković, 2007)
   a. Vowel epenthesis: $\emptyset \rightarrow V_i / \# \phantom{a} CCV_i$
   b. Glottal epenthesis: $\emptyset \rightarrow ? / \# \phantom{a} V$

(27) Classical Arabic (McCarthy, 2007)
    /ktub/ $\mapsto \left[?uktub\right]$, ‘write.MASC.SG!’
Appendix: Additional examples of opacity

(26) Non-gratuitous feeding (Baković, 2007)
   a. Vowel epenthesis: $\emptyset \rightarrow V_i / \# \ _ CCV_i$
   b. Glottal epenthesis: $\emptyset \rightarrow ? / \# \_ V$

(27) Classical Arabic (McCarthy, 2007)
    /ktub/ $\rightarrow$ [ʔuktub], ‘write.MASC.SG!’
Appendix: Additional examples of opacity

(26) Non-gratuitous feeding (Baković, 2007)

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

b. Glottal epenthesis: $\emptyset \rightarrow ? / \# \_ \_ V$

(27) Classical Arabic (McCarthy, 2007)

/ktub/ $\mapsto$ [ʔuktub], ‘write.MASC.SG!’

\[
\begin{array}{ccccccc}
\times & k & t & u & b & \times \\
\lambda & \lambda & \text{ʔuktu}
\end{array}
\]
Appendix: Additional examples of opacity

(28) Cross-derivational feeding (Baković, 2007)
   a. Epenthesis: $\emptyset \rightarrow i / K_i^\_K_i$
   b. Assimilation: $K \rightarrow [+\text{voice}] / _D$

(29) Lithuania (Odden, 2005)
   a. /at-taiki:ti/ $\rightarrow$ [atitaiki:ti], ‘to make fit well’
   b. /ap-gauti/ $\rightarrow$ [abgauti], ‘to deceive’
   c. /ap-berti/ $\rightarrow$ [apiberti], ‘to strew all over’

\[\times \quad a \quad p \quad b \quad e \quad r \quad t \quad i \quad \times\]
\[a \quad \lambda \quad \text{pib}\]
Appendix: Additional examples of opacity

(30) Fed Counterfeeding (Kavitskaya and Staroverov, 2010)
   a. Glottalization: \{t, d, s, n, η\} → ? / ___ #
   b. Deletion: \(\Lambda\) → ∅ / ___ (?) #

(31) Tundra Nenets
   a. /tasΛ/ → [tas], ‘whole’
   b. /t̂jimjΛs/ → [t̂jimj?], ‘it rotted’

\[\times\] t a s Λ × \times t̂j i m j Λ s ×
  t a λ λ s  t̂j i m j λ λ ?
References I


References III


