1. The problem: Getting from a lexicon to a grammar
Phonological processes that are restricted to certain lexical items typically apply stochastically to novel items.

The behavior of novel items reflects lexical trends (Hayes & Londe 2006, Albright & Hayes 2003, Zuraw 2000, and several others)

→ We need a way to project a stochastic grammar from the lexicon

GLA (Boersma 1997):
The GLA doesn’t distinguish existing words from novel words.
The GLA can’t use a lexicon to learn a stochastic grammar, as it will wrongly promote general faithfulness (Hayes & Londe 2006, Tessier 2007).

USELISTED (Zuraw 2000):
Distinguishes existing items from novel ones, but doesn’t derive the patterning of novel items from the trend created by the listed items.

2. Case study: Turkish voicing alternations
Turkish has intervocalic voicing in some words, but not others:

<table>
<thead>
<tr>
<th>Turkish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>tat</td>
<td>taste</td>
</tr>
<tr>
<td>kanat</td>
<td>wing</td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>Turkish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>dut</td>
<td>mulberry</td>
</tr>
<tr>
<td>sepet</td>
<td>basket</td>
</tr>
</tbody>
</table>

Factors that correlate with the relative proportion of alternating stops:

- Size: Monosyllables don’t usually alternate, polysyllables usually do
- Place: final [t]’s don’t usually alternate, final [p], [k] usually do

Lexical statistics and experimental results that confirm speakers’ knowledge of the pattern are in Becker, Ketrez and Nevins (2007).

How do speakers learn the proportion of alternating stops for each size and place?

3. The solution: Generalized Cloning
When lexical items demand conflicting rankings, BCD (Prince & Tesar 1999) detects inconsistency and stalls:

<table>
<thead>
<tr>
<th></th>
<th>OO-Ident(voice)</th>
<th>*VTV</th>
</tr>
</thead>
<tbody>
<tr>
<td>dut-u ➔ dut-u</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>tad-1 ➔ tad-1</td>
<td>L</td>
<td>W</td>
</tr>
</tbody>
</table>

The Pater (2006) solution: Clone a constraint to resolve the inconsistency.
Generalized cloning: All clones are lexically specific.

<table>
<thead>
<tr>
<th></th>
<th>OO-Ident(voice)_{dut}</th>
<th>*VTV</th>
<th>OO-Ident(voice)_{tat}</th>
</tr>
</thead>
<tbody>
<tr>
<td>dut-u ➔ dut-u</td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>tad-1 ➔ tad-1</td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

Result: A categorical grammar for listed lexical items:

\[
\text{OO-Ident(voice)}_{\text{dut}, \ldots} \gg *\text{VTV} \gg \text{OO-Ident(voice)}_{\text{tat}, \ldots}
\]

Generalization: Lexically-specific grammar ➔ Stochastic grammar

\[
\text{OO-Ident(voice)}_{60\%} \gg *\text{VTV} \gg \text{OO-Ident(voice)}_{40\%}
\]
4. **Generalized Cloning: Specific constraints first**

Speakers keep track of monosyllables independently of polysyllables. Speakers can do so thanks to the existence of initial-syllable faithfulness. OO-Ident(voice)\(_{\sigma 1}\) accounts for fewer lexical items, i.e. it is more specific:

<table>
<thead>
<tr>
<th></th>
<th>OO-Ident(voice)(_{\sigma 1})</th>
<th>OO-Ident(voice)</th>
<th>*VTV</th>
</tr>
</thead>
<tbody>
<tr>
<td>dut-u ~ dud-u</td>
<td>W</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>tad-i ~ tat-i</td>
<td>L</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>seped-i ~ seped-i</td>
<td>W</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>kanat-1 ~ kanat-1</td>
<td>L</td>
<td>L</td>
<td>W</td>
</tr>
</tbody>
</table>

If the learner wrongly clones the general OO-Ident(voice) first, the general constraint will account for all exceptions, and the size effect will not be learned:

\[\square \text{OO-Ident(voice)}_{\{\text{dut, sepet, ...}\} } \Rightarrow \text{*VTV} \Rightarrow \text{OO-Ident(voice)}_{\{\text{tad, kanat, ...}\} }\]

The learner must clone OO-Ident(voice)\(_{\sigma 1}\) first to list monosyllables, then clone the general OO-Ident(voice) to list polysyllables:

\[\text{OO-Ident(voice)}_{\sigma 1\{\text{dut, ...}\} } \Rightarrow \text{OO-Ident(voice)}_{\{\text{sepet, ...}\} } \Rightarrow \text{*VTV} \Rightarrow \text{OO-Ident(voice)}_{\{\text{tad, ...}\} } \Rightarrow \text{OO-Ident(voice)}_{\{\text{kanat, ...}\} }\]

5. **The learner**

Universal Grammar = a set of universal constraints + a theory of representations + GEN

Language-specific data = surface forms, arranged into paradigms

RCD & Cloner

Language-specific grammar = a ranking of the universal constraint set, with cloned constraints as necessary

The learner reads in the words of the given language one by one, and runs them through the grammar, creating a candidate set according to principles of OT-CC (McCarthy 2007). If the winner is different from the surface form, a winner-loser pair is formed and submitted to the RCD algorithm.

If RCD detects inconsistency, the learner clones a constraint that assigns the non-zero minimum of both W’s and L’s to the set of inconsistent ERC’s. This continues recursively, until the data becomes consistent, or can’t be made consistent by cloning.

The resulting grammar is categorical relative to existing lexical items, but can apply stochastically to novel items:

\[\text{OO-Ident(voice)}_{\sigma 1\{\text{dut, ...}\} }, \text{OO-Ident(voice)}_{\{\text{sepet, ...}\} } \Rightarrow \text{*VTV} \Rightarrow \text{OO-Ident(voice)}_{\{\text{tad, ...}\} }, \text{OO-Ident(voice)}_{\{\text{kanat, ...}\} }\]

The program uses code from JavaTableau (Becker, Potts & Pater 2007), CCamelOT (Becker 2005) and OT-Help (Becker, Pater & Potts 2007).

6. **References**


