Introduction


A constraint-based but derivational view of phonology and its typology

1) An HS schematic:

Theoretical Optimism: Harmonic Serialism has some interesting and desirable properties for accurately capturing phonologies without overgenerating (for a classified bibliography, see McCarthy 2009a)

Assumption: if Harmonic Serialism is a good theory of phonology, it must be associative with a good theory of learning
- see also Staubs and Pater (to appear) on HS learning

Stumbling Block for Learning:
- derivations happen step-by-step, driven by one M constraint at a time
- if error [outputs] drive learning, the learner only sees the end result

So how can learners figure out the steps in between?

Hidden Rankings
- HS sometimes requires rankings M1 >> M2 not revealed by errors!

Roadmap of the Talk

One: Some bare bones of HS, enough to get by in this talk

Crucial properties:
- a restricted GEN function, creating a (quite) finite candidate set
- a single ranking per language
- EVAL applies serially to /input/…
- converging finally on [output] whose local harmony is maximized


- learning just the attested surface forms, as restrictively as possible
- errors made by current grammar used as learning data
- small proposal: in HS, errors must processed slightly differently

Three: The learning challenge of hidden rankings
- in OT, M1 >> M2 rankings are pretty trivial to learn
- in HS they can be completely surprising
- the extent of the problem: examining errors is not enough

Four: The Proposal for Learning Hidden Rankings
- learn phonotactics first, with {M1, M2} left unranked
- then use winners – i.e. stored, observed surface forms
- examine the violation profiles of their failed loser candidates
- … these can iteratively reveal hidden rankings

Five: Implications, Imbroglios, Discussion

Take Home Message:
- HS makes phonotactic learning much more complicated
- but it makes every input’s candidate space much less complicated
- so this trade-off may be a key to successful HS learning
I. Harmonic Serialism in a Nutshell

**HS properties shared with classic OT** (Prince and Smolensky 1993/2004):
- basic architecture: /any input/ → ranked constraints → [optimal output]
- (basic) constraint types: markedness/structural and faithfulness
- EVAL: same method of choosing optima via strict domination

**HS properties shared with derivational theories**
- changes between /input/ → [output] are small and gradual
  i.e. HS has a much more restricted GEN
- HS loops through multiple I→O iterations in the same mapping
  i.e. HS is less parallel than classic OT

**HS mappings work like this:**
- feed any /input1/ to the GEN function
- GEN returns a finite set of output candidates: the fully faithful candidate, plus all the candidates that are ‘one unfaithful mapping away’ from the input (the set of such unfaithful changes being the subject of theoretical and empirical inquiry: see McCarthy, 2009b)

### Table: HS mappings work like this:

<table>
<thead>
<tr>
<th>Constraint</th>
<th>COMP CODA</th>
<th>DEP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ɛgz/</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ɛgz</td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ɛgz ... etc.</td>
<td>*!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- EVAL uses the constraint ranking to map from /input/ → [output1]
- the winning [output1] is now taken as /input2/
- GEN reapplies, and then EVAL reapplies

### Table: HS mappings work like this (continued):

<table>
<thead>
<tr>
<th>Constraint</th>
<th>COMP CODA</th>
<th>DEP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ɛg/</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ɛ</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ɛgz ... etc</td>
<td>*!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- the last cycle is one where /input/ → [fully faithful candidate]
- and that FFC is the [winner]

---

**II. Learning from errors in HS**

Start with a maximally restrictive grammar: \{M\} >> \{F\}
(in OT: Smolensky (1996) and many many others)

4) Toy Initial State: ONSET, *COMPLEXCODA >> MAX, DEP

Then attempt to reproduce an observed form with the current grammar, and analyze anything mapped unfaithfully as an error:

5) ERC vectors, following Prince (2002) and others

<table>
<thead>
<tr>
<th>input</th>
<th>winner ~ loser</th>
<th>*COMPCODA</th>
<th>ONSET</th>
<th>MAX</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>/lost/</td>
<td>[lost] ~ [tos]</td>
<td>*</td>
<td>L</td>
<td>e</td>
<td>W</td>
</tr>
</tbody>
</table>

Every error is stored in a Support table, which grows over time

**To learn** (Prince and Tesar, 2004; Hayes, 2004; Tessier, 2009):
Take all errors in the Support and build a new ranking from scratch:
- one that resolves these errors,
- but otherwise remains as restrictive as possible

**A rather simplified version of this learning algorithm:**

6) Build each stratum as follows:

   Step 1: *Install all M constraints that prefer no Losers in the current stratum*
   If any errors resolved, remove from Support and start next stratum

   Step 2: If Step 1 has done nothing, *install as few of the Faith constraints which prefer Winners as possible, so as to free up Ms constraint(s) at the next stratum*
   - If any errors resolved, remove from Support and start next stratum

Continue cycling through Steps 1 and 2 until no more M constraints:

**Step 3: Install all remaining F constraints and then **terminate.**

**Turn over to see learning at work for this error:**

---

**Note:** Within a language, the ranking always stays the same
- like in classic OT, the HS constraint ranking defines the language
- different outputs win on different iterations because the limited GEN means different inputs give rise to different candidate sets

---

1 A practical textbook-style intro: http://works.bepress.com/john_j_mccarthy/108/
7) Learning from this error:

<table>
<thead>
<tr>
<th>input</th>
<th>winner ~ loser</th>
<th>*COMP CODA</th>
<th>ONSET</th>
<th>MAX</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tost/</td>
<td>[tost] ~ [tos]</td>
<td>L</td>
<td>e</td>
<td>W</td>
<td>e</td>
</tr>
</tbody>
</table>

Stratum 1: Step One installs ONSET (since it hasn’t made any errors)...
Stratum 2: Step Two installs MAX (since it will resolve the errors)...
Error is now resolved, so is ignored...
Stratum 3: Step One installs *COMP CODA (since its error is gone)...
Stratum 4: Step Three installs DEP and terminate.

8) New Grammar: ONSET >> MAX >> *COMP CODA >> DEP

2.1 HS phonotactic learning: how to process errors? How does the learner learn from errors with multiple /I/ → [O] mappings?

9) Suppose Ḣ₁: *COMP CODA >> ONSET >> {MAX, DEP}

10a) /εg/z/  *COMP CODA  ONSET  MAX  DEP

<table>
<thead>
<tr>
<th>eg</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>εgz</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>gεgz</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>etc…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10b) /gεg/  *COMP CODA  ONSET  MAX  DEP

<table>
<thead>
<tr>
<th>gεg</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>εg</td>
<td>*!</td>
</tr>
<tr>
<td>εgz</td>
<td>*!</td>
</tr>
<tr>
<td>etc…</td>
<td></td>
</tr>
</tbody>
</table>

10c) /gεg/  *COMP CODA  ONSET  MAX  DEP

<table>
<thead>
<tr>
<th>gεg</th>
<th>*!</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>εg</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>gεgz</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>etc…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11) Full derivation: /εg/z/ → εg → [gεg]

Question: How do you store this error as a winner~loser pair? Recall that the HS grammar never compares εg z ~ gεg directly!

Proposal: Simply use the first step in the derivation:

<table>
<thead>
<tr>
<th>input</th>
<th>winner ~ loser</th>
<th>*COMP CODA</th>
<th>ONSET</th>
<th>MAX</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>/εg/z/</td>
<td>[εg/z] ~ [εg]</td>
<td>L</td>
<td>e</td>
<td>W</td>
<td>e</td>
</tr>
</tbody>
</table>

This error will teach the learner something small, but right, about the target:

13) Ḣ⁻₂ learned from (12): ONSET >> MAX >> *COMP CODA >> DEP

14a) /εg/z/  ONSET  MAX  *COMP CODA  DEP

<table>
<thead>
<tr>
<th>eg</th>
<th>!</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>εg</td>
<td>!</td>
<td>*</td>
</tr>
<tr>
<td>etc…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14b) /gεg/  ONSET  MAX  *COMP CODA  DEP

<table>
<thead>
<tr>
<th>gεg</th>
<th>!</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>εg</td>
<td>!</td>
<td>*</td>
</tr>
<tr>
<td>etc…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With Ḣ⁻₂: /εg/z/ → [gεg] – one step closer to the target

Summary of progress
Relying on ‘first step’ errors,
- the HS learner can determine phonotactics slowly but effectively
- for those marked structures highlighted by errors

The problem to come
What about marked structures that the learner never sees?
- In OT, a restrictive learner will rule them out from H_initial
- In HS, more learning is necessary... and errors are no help

There are other ways to do this; some might be better, but some are, I think, worse. Certainly there are ways to get more out of such errors, but Tessier (2011) suggests that’s maybe not such a good idea…
III. Hidden M rankings as the challenge for Error-driven HS learning

3.1 The target grammar: feeding among two well-attested processes

Process one: Stops lenite to fricatives intervocally
15a) ex: Tiberian Hebrew /p, t, k/ → [f, θ, x] / V_V - see Gurevich (2004), summarized especially in Kaplan (2010)

M1: *INTERVOCALIC STOPs, or
*V-STOP-V

Process two: Fricative allophony -- palatals before [i], velars elsewhere
15b) ex: Greek (Kazazis 1972) /xi/ → [çi], but elsewhere [x], *[ç] /yi/ → [ji], but elsewhere [γ], *[ɟ]

M2: *[VELAR FRICATIVE] [HIGH V] or
*[xi] ... and here *[xi] >> *PALATAL FRICATIVE, *[ç]

Hypothetical language: both processes in a feeding order (Kiparsky 1968)
16) a) /max/ → [max] b) /maxi/ → [maçi] neutralization /mak/ → [mak] c) /mak/ →

Process Interaction in Harmonic Serialism:
Driving (16c)'s mapping: . /maki/ → maxi → [maçi]
... requires a ranking of M1 >> M2:

17) *V-STOP-V >> *xi >> FAITH

18) Harmonic Improvement on /maki/

<table>
<thead>
<tr>
<th></th>
<th>*V-STOP-V</th>
<th>*xi</th>
<th>IDENT [CONTIN]</th>
<th>IDENT [PLACE]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ma.ki</td>
<td>*</td>
<td>!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>... less harmonic than</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ma.xi</td>
<td>*</td>
<td>!</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>... less harmonic than</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ma.çi</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

This constraint name is a cover name for a proper theory of featural place faithfulness, but the choice of faith constraints does not affect this learning result.

The opposite HS ranking, M2 >> M1, would block any change in /k/

19) Step One: *xi *V-STOP-V IDENT [CONTIN] IDENT [PLACE]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ma.ki</td>
<td>!</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ma.xi</td>
<td>!</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ma.çi</td>
<td>not in the candidate set for /maki/</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparison: in OT, M1 and M2 don’t need to be ranked wrt each other:
20) OT ranking: {*V-STOP-V, *xi} >> *ç >> FAITH correct drives: /maki/ → [maçi]

3.2 The learning problem with hidden M rankings

Assumptions
- an initial state is a partial ordering: {all M} >> {F}
- {all M} means a total ordering M1 >> M2 >> M3..., chosen randomly each time grammar is used
- just acquiring phonotactics – via the ‘Identity Map’

21) ℋ_{initial}: {*V-STOP-V, *xi, *ç} >> {ID-CONT, ID-PLACE}

22) Observed kinds of forms:
 [mak] [max] [maçi] [çik]

Phonotactic errors made by (21) on forms in (22) will just reveal allophony:

23) input winner ~ loser *V-STOP-V *xi *ç ID[CONT] ID[PLACE]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/çi/</td>
<td>çi ~ xi</td>
<td>*V-STOP-V</td>
<td>*xi</td>
<td>*ç</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ID[CONT]</td>
<td>ID[PLACE]</td>
<td>W</td>
</tr>
</tbody>
</table>

24) ℋ_{new} learned from (22):
{*V-STOP-V, *xi} >> *ç >> {ID-CONT, ID-PLACE}

The grammar in (2) maps all of (21) faithfully – no more errors or learning.

Successful properties of the final phonotactic grammar:
- *xi >> *ç >> ID[PLACE] = correct allophony: /çi/ → [çi], otherwise [x], *[ç]
- *V-STOP-V >> ID[CONT] = correct lenition: /*aka/ → [axa]
Failures of this phonotactic grammar:
- stop lenition and fricative allophony do not feed:

24) Language in (23), at the end of phonotactic learning:
   /baxi/ $\rightarrow$ [baçi]  
   but:  /baki/ $\rightarrow$ [baki] OR [baçi]
   /baçi

Upshot: hidden ranking not learned
- grammar vacillates between [baki] and [baçi]
- learner (incorrectly) believes that [Vki] is the unique legal intervocalic stop

Is this final state even a language we want in the typology?
- maybe, maybe not – see section 5.
- but if the language has NO intervocalic stops...
- learning (23)-(24) is a failure of restrictiveness

Broader upshot: errors cannot be relied on to reveal hidden rankings.

Side question: Will alternations help?
Short answer: if they occur, they might; but even so they are not guaranteed.

[Longer answer: even if there ARE alternations, the learner still has hard work to do – because they will need to establish a UR that maps to a winner in more than one step – which means setting up the winner is rather harder than setting up a loser back in (12). This is an independent problem that HS learning has to resolve.]

For more on alternations, see the Appendix, and/or discussion period questions, although I frankly know little on the topic as of yet.

Bermudez-Otero (2003) describes the acquisition of alternations in stratal OT – where the grammar uses multiple derivations (with every stratum’s ranking different) but EVAL remains parallel as in classic OT – but there, alternations do indeed help...

IV. Proposal: Learning Hidden Rankings from Winners

The order of the approach
(i) first, learner acquires F >> M and unhidden M >> M rankings, using errors
(ii) then learner determines remaining hidden M rankings – proposal below
(iii) still later, alternations and unfaithful URs are set up ...
this is a proposal about step (ii)

Starting with (23) again, the end state of phonotactic learning:

23) $\mathcal{H}_{\text{phono}}$ {*V-STOP-V, *XI} >> *Ç >> {ID-CONT, ID-PLACE}

The learner’s unanswered question:

Does restrictiveness require a ranking of \{M1, M2\}?
here:  \{*V-STOP-V, *XI\}

Step One: Examine Failed Candidates and Construct ERCs

Look for winners -- stored observed forms -- whose failed candidate set includes losers to which M1 or M2 assign a W and the other assigns an e:

For more on alternations, see the Appendix, and/or discussion period questions, although I frankly know little on the topic as of yet.

\[\text{Longer answer: even if there ARE alternations, the learner still has hard work to do – because they will need to establish a UR that maps to a winner in more than one step – which means setting up the winner is rather harder than setting up a loser back in (12). This is an independent problem that HS learning has to resolve.}\]
In this case, words like [baçi] have both kinds of losers in their candidate set:

27) /baçi/’s candidate set: baçi, baçi, baxi, paçi, baHi... (**baki)

<table>
<thead>
<tr>
<th>28) /baçi/</th>
<th>*V-STOP-V</th>
<th>*xi</th>
<th>*ç</th>
<th>IDENT [CONT]</th>
<th>IDENT [VELAR]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) baci</td>
<td>*!</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b) bachi</td>
<td>*!</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From which we can construct two ERCs, taking each loser as the winner:

29) input winner~loser  *V-STOP-V *xi  *ç IDENT [CONT] IDENT [VELAR]

a) /baçi/ baçi ~ baci W L L

b) /baxi/ baxi ~ baxi W L L

(Recall that **baksi is not in the candidate set for this winner [baçi]!)

These constructed ERCs in (29) don’t yet teach us anything NEW
- the current grammar already prefers the winners to the losers
- (29a) has simply created evidence for an initial state M >> F rankings

**Step Two: Building the complement of the constructed ERCs**

For each constructed ERC (as in 29):
- take the ‘loser’ as an input, and build its candidate set (30)
- again, consider this loser as the winner
- and find any losers with the opposite violation profile
  e.g. faring worse on M2 but not M1 – as in 31)

30a) /baksi/’s candidate set: [baksi] [baçi], [baksi], [bayi], [bahi] etc...

30b) /baxi/’s candidate set: [baxi], [baçi], [baksi], [baksi], [baxi] etc...

Reasoning from (30a)
- we built an ERC from the loser [baksi] because it fared worse on *xi (28c)
- is there a loser for winner [baksi] which fares worse on *V-STOP-V? Yes:

31) /baksi/ *V-STOP-V *xi  *ç IDENT [CONT] IDENT [VELAR]

baki

baksi

So we have a third constructed ERC:

32) input winner~loser  *V-STOP-V *xi  *ç IDENT [CONT] IDENT [VELAR]

a) /baksi/ baksi ~ baki W L e L e

33) Combining the pre-existing Support and the constructed ERCs

<table>
<thead>
<tr>
<th>/input/</th>
<th>winner~loser  *V-STOP-V *xi  *ç IDENT [CONT] IDENT [VELAR]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(22) /çi/</td>
<td>çi ~ xi W L W</td>
</tr>
<tr>
<td>(29a) /baçi/</td>
<td>baçi ~ baci W L</td>
</tr>
<tr>
<td>(29b) /baxi/</td>
<td>baxi ~ baxi W L L</td>
</tr>
<tr>
<td>(32) /baksi/</td>
<td>baksi ~ baki W L</td>
</tr>
</tbody>
</table>

Stratum 1: Step One installs *V-STOP-V errors (22) and (29a) resolved, can be ignored
Stratum 2: Step One installs *xi errors (29a) and (31) resolved, can be ignored
Stratum 3: Step One installs *ç Stratum 4: Step Three installs all remaining Faith ... and terminate.

34) $H^f = \{ *V-STOP-V >> *xi >> *ç >> \{IDENT-CONT, IDENT-PLACE} \}$

Success – the hidden ranking. Now – to assess.
V. Insufficient Discussion

Three broader questions about this proposal:

Q1. What mistakes could it make?
   - Will it ever assign a ranking that is less restrictive than the target, which errors could then never overcome?

Q2. What is the role of 'negative evidence'?
   - Reasoning back from observed winners to failed losers with certain violation profiles
   - What's the cost of such a search?
   - What sacrifice is this to the error-driven endeavour?

Q3. Should the learner simply be biased to prefer hidden rankings?

With respect to Q3 – here is the opposite, incorrect ranking from (19):

<table>
<thead>
<tr>
<th>19) Step One:</th>
<th>*xi</th>
<th>*V-STOP-V</th>
<th>IDENT CONTIN</th>
<th>IDENT PLACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/maki/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ma.ki</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ma.xi</td>
<td>*!</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ma.çi</td>
<td>not in the candidate set for /maki/</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If this ranking were learned definitely, we'd have this language:

35) Language with (19)'s ranking
   /baxi/ \[baçi\]  but:  /baki/ \[baki\]

Generalization:
- The only intervocalic stops are those which, if lenited, would also palatalize
- Maybe it's just as well to bias the learner away from such languages?

But: here is an example where both rankings seem warranted:

McCarthy (2008a) on cluster simplification:
Q: Why are word-medial clusters /VC1C2V/ usually reduced to VC2V?
A: Because deletion takes two HS steps:
   - First coda debuccalization /pat.ka/ \[paH.ka\]
   - Then placeless-C deletion /paH.ka/ \[pa.ka\]
   - ... and /pat.ka/ \[*[pat.Ha]\] is not harmonically improving

36) Harmonic improvement on /patka/

<table>
<thead>
<tr>
<th>/pat.ka/</th>
<th>CODA Condition</th>
<th>HAVE Place</th>
<th>MAX [PLACE]</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>pat.ka</td>
<td>*!</td>
<td></td>
<td>*!</td>
<td>*!</td>
</tr>
<tr>
<td>... less harmonic than</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paH.ka</td>
<td>*!</td>
<td></td>
<td>*!</td>
<td>*!</td>
</tr>
<tr>
<td>... less harmonic than</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa.ka</td>
<td></td>
<td></td>
<td></td>
<td>(*)</td>
</tr>
</tbody>
</table>

37) The hidden ranking: CODACOND >> HAVEPLACE
   Required to debuccalize before deleting

Result: *patka  *paH.ka  \[pa.ka\]

38) The reverse ranking: HavePlace >> CodaCond
   Blocks debuccalization to begin with...

Result: \[pat.ka\]  *paH.ka  \[pa.ka\]

Upshot: some hidden rankings are reversible, without pathology...
- In fact, local blocking is something HS is in part designed to predict
- But some such patterns are desirable, while others might be pathological
- But which ones?
- And is this a question for the builder of CON? or for the learning algorithm?

VI. Tentative Conclusions – are also the Take Home Message
- HS makes phonotactic learning much more complicated
- But it makes every input's candidate space much less complicated
- So this trade-off may be a key to successful HS learning

Thank you very much.
References


[http://www.people.umass.edu/eelfner/eelfner_2011_Stress-epenthesisHS.pdf]


McCarthy, John J. 2009a. Classified Bibliography of Works on OT with Candidate Chains (OT-CC) and Harmonic Serialism (HS). Ms, UMass Amherst. [http://works.bepress.com/john_j_mccarthy/102]


Appendix: Alternations that reveal hidden rankings?

Suppose the language from sections 3-4 had revealing alternations:

1) a) Bare Stem:  
   /mad/ → [mad]  
   /max/ → [max]  
   /mak/ → [mak]

b) Derived Form  
   /mad + i/ → [madi]  
   /max + i/ → [maçi]  
   /mak + i/ → [maçi]

If we have worked out the correct URs for 1b), could they teach us the hidden ranking, just making errors with the phonotactic grammar?

With an input /maki/, the variable behaviour of the grammar will be observed:

2) /maki/ → [maçi] OR *[maki]

But when looking at the error version, the learner now has to decide how to represent this winner:

3) /mak/ → [maçi]?

This winner is not in this input’s candidate set! So what can we do? In effect, we need to know the middle step.

We could invent a mechanism to create that middle step – reasoning about faith violations to create intermediate members on the chain:

4) /maki/ → ?? → [maçi]

If we knew to go via /maxi/, we would learn the hidden ranking fine:

<table>
<thead>
<tr>
<th>input</th>
<th>winner~ loser</th>
<th>*V-STOP-V</th>
<th>*xi</th>
<th>*Ç</th>
<th>ID [CONT]</th>
<th>ID [PLACE]</th>
</tr>
</thead>
<tbody>
<tr>
<td>/mak + i/</td>
<td>maxi ~ maki</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>/maxi</td>
<td>maçi ~ maxi</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

These two errors will teach the learner everything, just as in (32)

6) Correct Result:  
   *V-STOP-V >> *xi >> *Ç >> ID[CONT], ID[PLACE]

This search for an intermediate candidate is definitely not hopeless - because of the finite candidate set - and it’s probably rather simpler than the one needed in sections 3 and 4 - but you can’t guarantee the relevant alternation will occur

Upshot: alternations cannot be relied on to reveal hidden rankings

And since we need a mechanism for phonotactic hidden rankings anyway, we needn’t equip the learner with two similar devices for finding crucial intermediate losers

The present proposal could help the learner find these hidden rankings before alternations raise the problem of building a winner in (3) here

Of course:  
- real alternations still need to be learnable  
- so the problem of building winners, if not like (3) then how, must be solved

Future, future work.

---

6 Thanks especially to Igor Yanovich for pressing me on this point, and helpful ensuing discussion.