Class 18: Optimality Theory, part I

To do
- Start working on last assignment (due Thursday, Dec. 8): develop an OT analysis of the Fijian data from last time—more directions will be posted on web page.
- Read Prince & Smolensky (SQs due Tuesday)

1. The “conceptual crisis” (Prince & Smolensky p. 1)
Since Kisseberth 1970, constraints were taking on a bigger and bigger role. But…
- What happens when there’s more than one way to satisfy a constraint? We need to prioritize the rules that could be triggered.
- Why aren’t constraints always obeyed?
- Relatedly, what happens when constraints conflict? What if one constraint wants to trigger a rule, but another wants to block it? We need a way of prioritizing constraints.
- Should a rule be allowed to look ahead in the derivation to see if applying alleviates a constraint violation? Or does the alleviation have to be immediate?
- Can a constraint be against making a certain type of change, rather than against a certain structure?

2. Prince & Smolensky’s solution: Optimality Theory

<table>
<thead>
<tr>
<th>rule-based grammar</th>
<th>OT grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>start with UR/input (from mental lexicon)</td>
<td>apply all possible rules, producing a (large!) set of candidate outputs</td>
</tr>
<tr>
<td>apply rules in sequence—intermediate representation is known at all times</td>
<td>constraints pick the best candidate</td>
</tr>
<tr>
<td>constraints may block or trigger rules</td>
<td>since the candidate outputs are all potential surface forms, there is full look-ahead to the end of each possible derivation</td>
</tr>
<tr>
<td>look-ahead is nonexistent or sketchy</td>
<td>constraints interact through strict domination</td>
</tr>
<tr>
<td>interaction of constraints is nonexistent or sketchy</td>
<td>similarity to UR is enforced by faithfulness</td>
</tr>
<tr>
<td>similarity to UR is the result of not applying too many rules and not having too many constraints</td>
<td></td>
</tr>
<tr>
<td>end with SR/output (send it to the phonetic system)</td>
<td></td>
</tr>
</tbody>
</table>

3. Gen
This is the function that creates the set of candidate outputs from the input.

One way to think of it:¹ apply all possible rules to the input, any number of times: each underlying segment can be deleted or have its features changed; extra segments can be inserted anywhere; underlying segments can change their order

¹ This is what P&S call ‘anharmonic serialism,’ except with a universal set of rules that’s broad enough that the result is the “all possible variants” that P&S propose.
Gen(/ab/) = \{[ab], [a], [b], [ba], [], [ta], [at], [ae], \ldots\}

Why is the resulting set of candidates infinite (assuming a finite alphabet of symbols)?

4. Constraints
In standard OT, we can think of each constraint as a function from a candidate output to a natural number (the number of violations).

\[ \text{NOCODA}([bak]) = 1 \]
\[ \text{NOCODA}([tik.pad]) = 2 \]

Alternatively, we can think of each constraint \( C_i \) as imposing a strict partial ordering \( >_i \) ("is more harmonic than with respect to \( C_i \)"") on a set of candidates, with the following additional properties:

- The ordering is **stratified**: If \( a \succ b \) and \( b \succ a \), then for any \( i > a, i > b \) too; and for any \( j \) such that \( a \succ j, b \succ j \) too. (We can say that if \( a \succ b \) and \( b \succ a \), \( a \) and \( b \) are of equivalent harmony.)
- There exists some \( a \) such that there is no \( i \succ a \). (That is, one or more candidates are the most harmonic; there are not necessarily one or more least harmonic candidates, though.)

A strict partial ordering is **transitive**, **irreflexive**, and **asymmetric**:

- Transitive: if \( a > b \) and \( b > c \), then \( a > c \).
- Irreflexive: \( a \not> a \).
- Asymmetric: If \( a > b \), then \( b \not> a \).

(In Colin Wilson’s targeted-constraints variant of OT, the stratification requirement is relaxed.)

\[ \text{NOCODA} \]

Why does assigning a non-unique natural number (0, 1, 2, \ldots) to each candidate meet the ordering requirements above?

If you’re ahead in the reading, can you recall a case from P&S where numbers of violations weren’t used?

Why are there no least harmonic candidates for NOCODA?
5. Eval
Eval can be thought of as a function from a set of output candidates and an ordered list of constraints to the subset of those candidates that is optimal (typically, that subset contains just one candidate—the chosen output).

Eval does this by taking the orderings imposed by the various constraints and assembling them into one giant ordering. We can think of many ways this could be done…strict ranking is the mechanism used in standard OT for adjudicating harmony disagreements among constraints.

6. Constraint interaction through strict ranking
Strict ranking is like alphabetization: to find the alphabetically earliest member of a list, pick out all the members of the list that have the earliest first letter; from that list, pick the members that have the earliest second letter, etc. Once a word is out of the running, it can’t redeem itself by, e.g., having lots of as later on:

axiom  axiate  tab  axicle  cab
banana  azalea  axolotl  zabaglione  baa

Eval works the same way. If you have \( n \) constraints…
- Pick out the candidates that tie for being ‘best’ on the top-ranked constraint \( C_0 \); discard the rest of the candidates.
- Move to the next constraint, \( C_1 \), and do the same—pick out the candidate(s) that are best with respect to the set that is left.
- Repeat for \( C_2, \ldots, C_n \).
- Whatever candidates are still left at the end are tied for being the winner (if you have enough constraints, there is normally just one winner).

Q: How can that be computable? Wouldn’t you have to go through an infinite list of candidates just to do the first step?

A: For that reason, most computational implementations of OT (Ellison, Eisner, Albro, Riggle) represent the candidate set as a regular expression, which is a finite way to represent a certain class of infinite sets. For example, \( ab^*a \) is the set \( \{aa, aba, abba, abbbba, \ldots\} \). These expressions can then be manipulated algorithmically, either in a fairly literal translation of the above or by other means.

More declaratively, a candidate \( a \) is optimal iff, for any \( b \) and \( C_j \) such that \( b \succeq_j a \), there exists some \( C_i \) such that \( i < j \) (i.e., \( C_i \) is higher ranked than \( C_j \)) and \( a \succeq_i b \). In words, for \( a \) to be optimal, any candidate that does better than \( a \) on some constraint must do worse than \( a \) on some higher-ranked constraint.

- Can you imagine some other ways that constraints could conceivably interact?
7. Two types of constraint

In pre-OT approaches to constraints, constraints were all markedness constraints: they penalized certain surface structures, such as CCC clusters.

So, on first hearing about OT, many people’s second reaction (the first was worrying about infinity) was to wonder why, if it’s all about constraints, every word isn’t maximally unmarked.

- In rule+constraint theories, what prevents every word from coming out [baba] (or whatever the least marked word is)?
- For those who know OT, how do P&S prevent every word from coming out [baba]?

**Markedness** constraints are constraints on the output (they could be thought of as requiring articulatory ease, or perceptual clarity, or some other “natural” type of unmarkedness\(^2\)). The simplest ones can be defined by the structural description that they ban: *[+voice]#, *C]_σ.

You can (and should!) give a constraint a helpful mnemonic name, like NOCODA for *C]_σ, as long as you clearly define the constraint somewhere. A good constraint definition should make it clear not just what is banned, but how the number of violations is assessed.

**Faithfulness** constraints are constraints on the relationship between the input and the output (the standard ones require similarity).

P&S use PARSE (roughly, don’t delete) and FILL (roughly, don’t insert), but this has been superseded by McCarthy & Prince’s correspondence approach, which you’ll learn about in more detail in 201:

- MAX-X: don’t delete X (e.g., MAX-C, MAX-V)
- DEP-X: don’t insert X (e.g., DEP-C, DEP-V)
- IDENT-F: don’t change a segment’s value for the feature F

People often have a hard time at first with IDENT-F. The most common confusion is thinking it means “don’t delete a segment that is +F”. The next most common is thinking it means “don’t alter a segment that is +F (e.g., by changing its values for some other feature G)”.

8. Exposition: the tableau

In a utopic future (which may not be that far away), we will all check our analyses with software that evaluates the infinite candidate set. In the meantime, we illustrate our analyses for the reader with a tableau\(^3\) showing a finite subset of candidates that have been chosen to demonstrate aspects of the constraint ranking. (The danger here is obvious—what if you didn’t think of some important candidate?)

---

\(^2\) Or maybe they are just arbitrary and learned by speakers in response to whatever cards history has dealt them. Or, maybe both natural and unnatural constraints are possible, but learners treat them differently. This is a matter of current debate. See the recent volume *Phonetically based phonology*, ed. by Hayes, Kirchner, & Steriade, and recent papers by Colin Wilson.

\(^3\) French for ‘table’. The singular tableau is pronounced [tablô] in French, [tʰæblóu] in English. The plural tableaux is pronounced [tabló] in French, [tʰæblóu] or [tʰæblóuz] in English.
This tableau shows a ranking argument: we have two candidates that differ in that NoCoda prefers \(a\) (the winner), whereas Dep-V prefers \(b\). If that’s the only difference between the candidates—there is no other constraint that prefers \(a\) over \(b\)—then NoCoda must outrank \(\gg\) Dep-V.

<table>
<thead>
<tr>
<th>/at+ka/</th>
<th>NoCoda</th>
<th>Dep-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\nexists a) [a.tə.kə]</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>(b) [at.kə]</td>
<td>*!</td>
<td></td>
</tr>
</tbody>
</table>

Parts of the tableau:
- input
- output candidates
- constraints
- asterisks
- exclamation marks
- shading
- pointing finger (or you can use an arrow)

These three don’t add any new information, but are there for the convenience of the reader.

9. How do I know which candidates and constraints to include in my tableaux?

Here is a procedure that usually works reasonable well:
- Start with the winning candidate and the fully faithful candidate.
- If the winning candidate \(\neq\) the fully faithful candidate…
  - Add the markedness constraint(s) that rule out the fully faithful candidate.
  - Add the faithfulness constraints that the winning candidate violates.
  - Think of other ways to satisfy the markedness constraints that rule out the fully faithful candidate. Add those candidates, and the faithfulness and markedness constraints that rule them out. You have to use your judgment in deciding how far to take this step.
- If the winning candidate = the fully faithful candidate, then you are probably including this example only to show how faithfulness prevents satisfaction of a markedness constraint that, in other cases, causes deviation from the underlying form.
  - Add that markedness constraint.
  - Add one or more candidates that satisfy that markedness constraint.
  - Add the faithfulness constraints that rule out those candidates.

Let’s try it for /atka/ → [atəkə].

One of the candidates below is unnecessary in arguing for the constraint ranking. Why?

<table>
<thead>
<tr>
<th>/at+ka/</th>
<th>*CC</th>
<th>Dep-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\nexists a) [atəkə]</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>(b) [atka]</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>(c) [atəkə]</td>
<td>**!</td>
<td></td>
</tr>
</tbody>
</table>

A candidate is harmonically bounded if it could not win under any constraint ranking.
10. Comparative tableaux
An innovation of Alan Prince. They convey the same information, but in a different form

<table>
<thead>
<tr>
<th>/at+ka/ → [atɔka]</th>
<th>*CC</th>
<th>DEP-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>a [atɔka] vs. [atka]</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>b [atɔka] vs. [atɔkə]</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

Each line compares the winner to one losing candidate, and shows whether each constraint prefers the winner (W) or the loser (L).

Comparative tableaux are nice because you can easily see if your ranking is correct: the first non-blank cell in each row must say W.
We also see easily why [atɔkə] is irrelevant to the ranking.

11. Exercise: Metaphony (the two easy cases)
- Develop an OT account of these two metaphony systems.

**Foggiano/Pugliese**
- pête ‘foot’
- mójja ‘soft (fem.)’
- kjéna ‘full (fem.)’
- grɔssa ‘big (fem.)’
- píti ‘feet’
- müffu ‘soft (masc.)’
- kjímu ‘full (masc.)’
- grússu ‘big (masc.)’

**Veneto**
- védo ‘I see’
- kóro ‘I run’
- préte ‘priest’
- béllo ‘beautiful (masc. sg.)’
- módo ‘way’
- gáto ‘cat’
- te vídi ‘you see’
- te kúri ‘you run’
- préti ‘priests’
- béli ‘beautiful (masc. pl.)’
- módì ‘ways’
- gáti ‘cats’

12. Another exercise: English regular plurals
- pi-z ‘peas’
- thou-z ‘toes’
- dal-z ‘dolls’
- pʰæn-z ‘pans’
- dag-z ‘dogs’
- læb-z ‘labs’
- kʰlín-z ‘kilns’
- kʰlæsp-s ‘clasps’
- mít-s ‘mitts’
- blouk-s ‘blokes’
- kʰuf-s ‘coughs’
- glas-iz ‘glasses’
- fiz-iz ‘fizzes’
- bæntʃ-iz ‘branches’
- bædʒ-iz ‘badges’
- wʃ-iz ‘wishes’
- gəɾə-iz ‘garages’
13. If we have lots of time, extended exercise: Nanti

Key cases, repeated from assignment:

(i.pì)(ri.ní)ti  
‘he sits’

(i.rì)(pi.rí)(ni.te)  
‘he will sit’

(já.nu)ti  
‘he walked’

(nò.ko)(gá.ko)ta <ro>  
‘I request it’

(o.tà)(sóŋ.kà)kse <ro>  
‘she blew on it’

(ò.ko)(ri.kjì)(tá.ka)  
‘she wore a nose-disc’

(no.sà)(me.rè)(já.ka)  
‘I got a blister’

(yo.biì)(kái.gà)kse  
‘they (m.) drank’

(o.tà)(sóŋ.kà)kse <ro>  
‘she blew on it’

(nòn.ksèn)(tá.ksè) <ro>  
‘I will pierce it (compl.) (w/ an arrow)’

(in.ksèn)(tá.ksè) <ro>  
‘he will pierce it (compl.) (w/ an arrow)’