
Weeks 14-15 – Optimality Theory

December 2, 2008

1 Optimality Theory: Overview

- (1) Prince and Smolensky (1993, 2004)
- (2) The “conceptual crisis” (Prince & Smolensky 1993, p. 1)
- (3) 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?
- (4) Prince & Smolensky’s solution: Optimality Theory

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

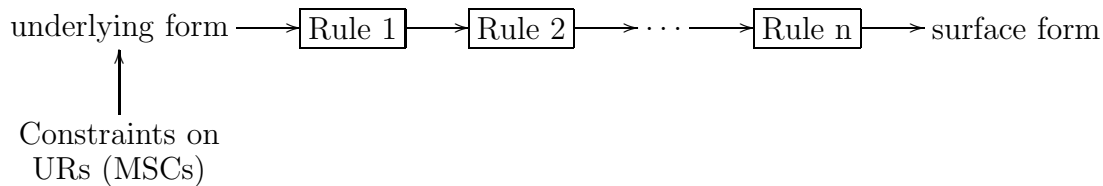


Figure 1: Traditional Derivational Generative Phonology

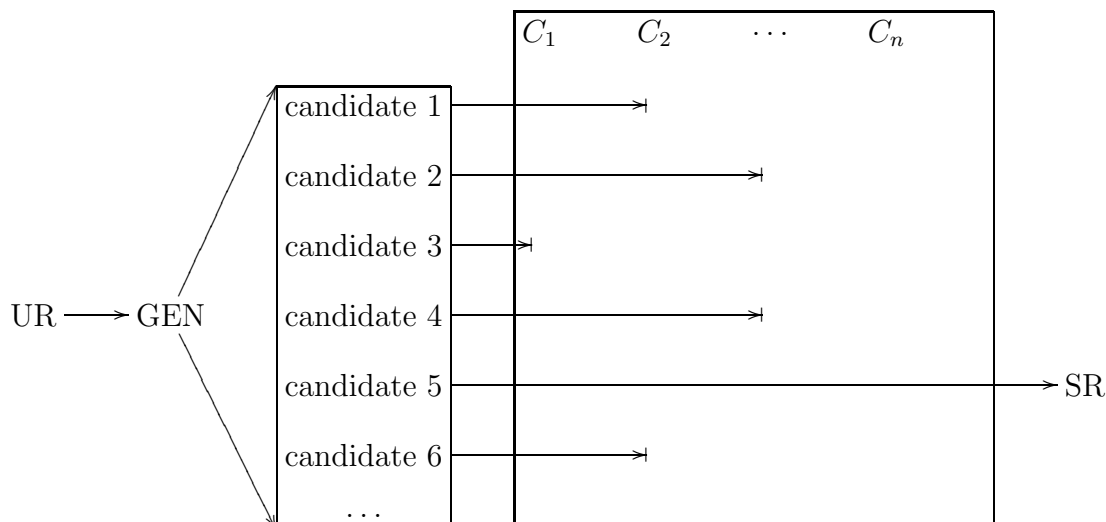


Figure 2: Optimality-theoretic Generative Phonology

1.1 OT Illustration

2 Architecture of OT

- (5) The components of an OT grammar are GEN, CON, AND EVAL, all which are considered to be universal.

2.1 Gen

- (6) GEN 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.

$$\text{Gen}(/ab/) = [ab], [a], [b], [ba], [], [ta], [at], [ae], \dots, [abbbbbbbbbb], \dots$$

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

2.2 Constraints

- (7) In standard OT, we can think of each constraint as a function from a (input,output) pair to a natural number (the number of violations).

$$\begin{aligned} \text{NoCODA}(/bak/, [bak]) &= 1 \\ \text{NoCODA}(/tikpad/, [tik.pad]) &= 2 \end{aligned}$$

- ★ In the example above, does the input to the constraint play a role in determining the number of violations?

- (8) For a particular input, we can think of each constraint C_i as imposing a strict partial ordering \succ_i (“is more harmonic than with respect to C_i ”) on the set of candidates, with the following additional properties:

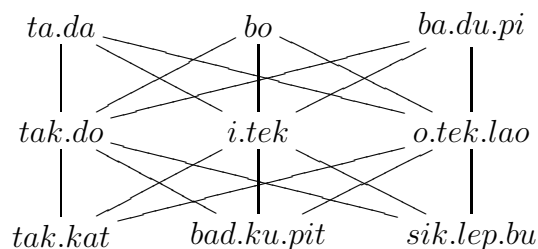
- The ordering is stratified: If $a \not\succeq b$ and $b \not\succeq a$, then for any $i \succ a$, $i \succ b$ too; and for any j such that $a \succ j$, $b \succ j$ too. (We can say that if $a \not\succeq b$ and $b \not\succeq a$, a and b are of equivalent harmony.)

¹This 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.

- There exists some a such that there is no $b \succ a$. (That is, one or more candidates are the most harmonic; there are not necessarily one or more least harmonic candidates, though.)

(9) (In Colin Wilson's targeted-constraints variant of OT, the stratification requirement is relaxed (ColinWilson 2001).)

(10) Then NOCODA:²



...

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

- ★ Why are there no least harmonic candidates for NOCODA?

2.3 Eval

- (11) 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 are the most harmonic—i.e. optimal. (Typically, that subset contains just one candidate).
- (12) 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.

²Really, this is for some particular input, say /tad/. It turns out of course that for markedness constraints like NOCODA, it is easier to think of the harmonic ordering imposed by them without reference to the input, but only because the imposed ordering will be the same no matter what the input is.

2.4 Defining Best Satisfaction

- (13) •Rank constraints in a hierarchy
 •Recursive procedure (see Algorithm 1)

Algorithm 1 General procedure for determining the winning candidate in OT

Input: an underlying form UR, a set of ranked constraints CON

Output: the optimal candidate(s) (the surface form(s))

With GEN, generate a set of output candidates CANDIDATES

while there are unexamined constraints in CON **do**

1. Submit CANDIDATES to the highest-ranked constraint yet unexamined.
2. Keep only those candidates in CANDIDATES which violate this constraint *the least*. (Discard the other candidates.)
3. Mark this highest ranked constraint as ‘examined’.

end while

Designate all candidates remaining in CANDIDATES as winners (in practice there is usually one, but there could be more than one in principle in which case we need a way to interpret this, e.g. free variation).

- (14) Strict Constraint Domination, a.k.a. Lexicographic Order
- a. Let candidate C1 violate the strictest constraint once.
 - b. Let candidate C2 violate constraints 2 through 1,000,000 a million times each.
 - c. The winner is C2.
- (15) Analogy: the lexicographic ordering of AZZZZZZZZZZZZZZZZZZZ before BAAAAAAAAAAAAAAAAA
- (16) This is controversial; see for instance Coleman and Pierrehumbert (1999), <http://www.phon.ox.ac.uk/~jcoleman/colepier.ps>), who attempt to refute strict domination outright.
- (17) Q: How can that be computable? Wouldn't you have to go through an infinite list of candidates just to do the first step?
- (18) A: For that reason, most computational implementations of OT (Ellison 1994, Eisner 1997, Karttunen 1998, Frank and Satta 1998, Albro 1998, Riggle 2004, Albro 2005) 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 $aa, aba, abba, abbbba, \dots$. These expressions can then be manipulated algorithmically, either in a fairly literal translation of the above or by other means.
- (19) More declaratively, a candidate a is optimal iff, for any b and C_j such that $b \succ_j a$, there exists some C_i such that $i < j$ (i.e., C_i is higher ranked than C_j) and $a \succ_i b$.
- a. 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?

- (20) Finally, note the big picture:
- a. Both the traditional ordered-rule and the OT approach map URs to SRs.
 - b. This relation as far as is known is *regular*! (Johnson 1972, Kaplan and Kay 1981, 1994)
 - c. This means we can use *regular rewrite grammars* (or finite state transducers), or many other ways of defining regular grammar (see Kracht (2003) for example).
 - d. This does NOT mean any regular relation is a possible phonological one.
 - e. Rather it means the focus of our investigation is this relation, which has this property (and that relations with this property have been extensively studied in other fields).
 - f. What other properties do attested phonological relations have? And which may be useful to learning?

3 OT in Practice

- (21) The above is a framework for OT in general. Here we talk about some particular OT conventions, i.e. OT as it is practiced.

3.1 Two types of constraint

- (22) In pre-OT approaches to constraints, constraints were all *markedness* constraints: they penalized certain surface structures, such as CCC clusters.
- (23) 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)?

3.1.1 Faithfulness Constraints

- (24) Faithfulness Constraints
- a. We factor the ways that two representations could differ.
 - b. To make the differences utterly explicit, we put an index on every segment. This is called correspondance theory (McCarthy and Prince 1995).

- (25) IDENT = differ in one feature value
 a. /p₁ a₂ k₃/, candidate [b₁ a₂ k₃] violates IDENT(VOICE).
- (26) MAX = an underlying segment of some natural class (specified with features) is missing in the first form.
 ★ For UR /p₁ a₂ k₃/, what does the candidate [b₁ a₂] violate? (multiple answers)
- (27) DEP = a surface segment of some natural class (specified with features) is missing in the underlying form.
 ★ For UR /a₂ k₃/, what does the candidate [ʔ₁ a₂ k₂] violate? (multiple answers)
- (28) LINEARITY, violated when the linear order of any pair of segments is switched.
 ★ Count the violations for each of the candidates below for UR /p₁ a₂ k₃/: [k₃ a₂ p₁], [p₃ a₂ k₁], and [p₃, a₂, k₁].
- (29) Not a standard Faithfulness constraint IDENT(P) “Don’t change anything about [p] so it isn’t [p] any more.”
- (30) Fundamental principle of clear analytic presentation: always declare all Faithfulness constraints violated by winners.

3.2 Defining Constraints

- (31) Constraints are *functions* from (UR,SR) pairs to natural numbers.
 a. IDENT(VOICE)(/p₁ a₂ k₃/, [b₁ a₂ k₃]) = 1.
- (32) Note that when defining a markedness constraint, the UR doesn’t matter.
 a. *NOCODA(/p₁ a₂ k₃/, [p₁ a₂ k₃]) = 1
 b. *NOCODA(/p₁ a₂/, [p₁ a₂ k₃]) = 1
 c. *NOCODA(/p₁ a₂ k₃ a₄/, [p₁ a₂ k₃]) = 1
- (33) For this reason, we can define markedness constraints as follows:
 a. A markedness constraint M is a constraint such that for any SR, UR₁, and UR₂, it is the case that M(UR₁,SR) = M(UR₂,SR).
 ★ Why is this not true for faithfulness constraints?

- (34) Fundamental principles of clear analytic presentation: whenever defining a constraint, it should be clear how to determine the number of violations for any (UR,SR) pair.
- (35) *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 unmarkedness³). The simplest ones can be defined by the structural description that they ban: $*[+voice]_\#, *C]_\sigma$.
- (36) If you define a new constraint, you should give a constraint a helpful mnemonic name, like NOCODA for $*C]_\sigma$, 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.
- (37) *Faithfulness constraints* are constraints on the relationship between the input and the output (the standard ones require similarity).
- (38) P&S use PARSE (roughly, don’t delete) and FILL (roughly, don’t insert), but this has been superseded by McCarthy & Prince’s 1995 correspondence approach, which you’ll learn about in more detail next semester:

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

- (39) 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)”.

3.3 Exposition: the tableau

- (40) 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 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?)
- (41) 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 ($>>$) DEP-V.

³[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 (Hayes et al. 2004), and recent papers by Colin Wilson, e.g. Wilson (2006).

/at+ka/	NoCODA	DEP-V
☞ a. a.ta.ka		*
b. at.ka	*!	

- (42) Parts of the tableau:
- a. Crucial elements
 - (i) input
 - (ii) output candidates
 - (iii) constraints
 - (iv) asterisks
 - b. Extra bits which make it easier on the reader (but add no new information)
 - (i) exclamation marks
 - (ii) shading
 - (iii) pointing finger (or you can use an arrow)
- (43) How do I know which candidates and constraints to include in my tableaux?
- (44) Here is a procedure that usually works reasonable well:
- a. Start with the winning candidate and the fully faithful candidate.
 - b. If the winning candidate \neq the fully faithful candidate...
 - (i) Add the markedness constraint(s) that rule out the fully faithful candidate.
 - (ii) Add the faithfulness constraints that the winning candidate violates.
 - (iii) 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.
 - c. 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.
 - (i) Add that markedness constraint.
 - (ii) Add one or more candidates that satisfy that markedness constraint.
 - (iii) Add the faithfulness constraints that rule out those candidates.

★ Let's try it for /atka/ \rightarrow [aka].

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

/at+ka/	*CC	DEP-V
☞ a. a.ta.ka		*
b. at.ka	*!	
c. a.ta.kaa		**!

(45) A candidate is *harmonically bounded* if it cannot win under any constraint ranking.

(46) Exercise: Metaphony (the two easy cases)

★ Develop an OT account of these two metaphony systems.

3.3.1 Foggiano/Pugliese (i,e,ɛ,a,u,o,ɔ)

kjéna	‘full (fem.)’	kjínu	‘full (masc.)’
péte	‘foot’	píti	‘feet’
móffa	‘soft (fem.)’	múffu	‘soft (masc.)’
gróssa	‘big (fem.)’	grússu	‘big (masc.)’

Assume that [a] does not alternate.

3.3.2 Veneto (i,e,ɛ,a,u,o,ɔ)

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

References

- Albro, Dan. 1998. Evaluation, implementation, and extension of Primitive Optimality Theory. Master’s thesis, University of California, Los Angeles.
- Albro, Dan. 2005. A Large-Scale, LPM-OT Analysis of Malagasy. Ph.D. thesis, University of California, Los Angeles.
- ColinWilson. 2001. Consonant cluster neutralisation and targeted constraints. *Phonology* :147–197.
- Eisner, Jason. 1997. What constraints should OT allow? Talk handout, Linguistic Society of America, Chicago. ROA#204-0797. Available at <http://roa.rutgers.edu/>.
- Ellison, Mark. 1994. Phonological Derivation in Optimality Theory. In *COLING 94*, volume 2. pages 1007–1013. Kyoto, Japan.
- Frank, Robert and Giorgio Satta. 1998. Optimality Theory and the Generative Complexity of Constraint Violability. *Computational Linguistics* 24(2):307–315.

- Hayes, Bruce, Robert Kirchner, and Donca Steriade, eds. 2004. *Phonetically-Based Phonology*. Cambridge University Press.
- Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.
- Kaplan, Ronald and Martin Kay. 1981. Phonological Rules and Finite State Transducers. Paper presented at ACL/LSA Conference, New York.
- Kaplan, Ronald and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.
- Karttunen, Lauri. 1998. The proper treatment of Optimality Theory in computational phonology. *Finite-state methods in natural language processing* :1–12.
- Kracht, Marcus. 2003. *The Mathematics of Language*. Mouton de Gruyter.
- McCarthy, John and Alan Prince. 1995. Faithfulness and Reduplicative Identity. In *Papers in Optimality Theory*, edited by Jill Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk, number 18 in University of Massachusetts Occasional Papers in Linguistics. pages 249–384.
- Prince, Alan and Paul Smolensky. 1993. Optimality Theory: Constraint Interaction in Generative Grammar. Technical Report 2, Rutgers University Center for Cognitive Science.
- Prince, Alan and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.
- Riggle, Jason. 2004. Generation, Recognition, and Learning in Finite State Optimality Theory. Ph.D. thesis, University of California, Los Angeles.
- Wilson, Colin. 2006. Learning Phonology With Substantive Bias: An Experimental and Computational Study of Velar Palatalization. *Cognitive Science* 30(5):945–982.