

Optimality Theory in Linguistics

Kie Zuraw

Department of Linguistics

University of Southern California

3601 Watt Way, GFS 301

Los Angeles, CA 90089-1693

Short title: Optimality Theory

phone: 213-740-3884

fax: 213-740-9306

email: zuraw@usc.edu

INTRODUCTION

Prince and Smolensky (1993) introduced Optimality Theory (OT) as a framework for linguistic analysis. Kager (1999) gives an entry-level introduction to OT, McCarthy (2001) surveys advanced topics, and the Rutgers Optimality Archive, (<http://ruccs.rutgers.edu/roa.html>) contains hundreds of OT papers. Within phonology, OT has largely supplanted rule-based frameworks. OT has also been applied to syntax and semantics, although not as widely; Legendre, Grimshaw, and Vikner (2001) provide an overview of current work in OT syntax.

Rule-based frameworks account for linguistic patterns through the sequential application of transformations to lexical entries. For example, variation between two pronunciations of the English plural suffix—[s] in *cats* but [z] in *dogs*—is explained by a rule that devoices the suffix after voiceless consonants (like [t]). The input *cat* + /z/, assembled from entries in the speaker's mental dictionary, is transformed by rule into the output *cat[s]*. In OT, the output is instead chosen through competition with other candidates: a constraint requiring adjacent consonants to match in voicing favors *cat[s]* over *cat[z]*.

Generation of utterances in OT involves two functions, *Gen* and *Eval*. *Gen* takes an input and returns a (possibly infinite) set of output candidates. Some candidates might be identical to the input, others modified somewhat, others unrecognizable. *Eval* chooses the candidate that best satisfies a set of ranked constraints; this optimal candidate becomes the output.

The constraints of *Eval* are of two types. *Markedness* constraints enforce well-formedness of the output itself, prohibiting structures that are difficult to produce or comprehend, such as consonant clusters or phrases without overt heads. *Faithfulness* constraints enforce similarity between input and output, for example requiring all input consonants to appear in the output, or all morphosyntactic features in the input to be overtly realized in the output. Markedness and faithfulness constraints can conflict, so the constraints' ranking—which differs from language to language—determines the outcome. One language might eliminate consonant

clusters by deleting consonants, despite the resulting faithfulness violations; another might retain all input consonants, violating the markedness constraint.

In standard OT, constraints are strictly ranked and violable. *Strict ranking* means that a candidate violating a high-ranking constraint cannot redeem itself by satisfying lower-ranked constraints (constraints are not numerically weighted, and lower-ranked constraints cannot gang up on a higher-ranked constraint). *Violability* means that the optimal candidate need not satisfy all constraints. *Eval* can be viewed as choosing the subset of candidates that best satisfy the top-ranked constraint, then, of this subset, selecting the sub-subset that best satisfy the second-ranked constraint, and so on. Another way of describing *Eval* is that a candidate *i* is optimal if and only if, for any constraint that prefers another candidate *j* to *i*, there is a higher-ranked constraint that prefers *i* to *j*.

The *tableau* (a standard expositional device in OT) in Figure 1 illustrates output selection for the input /ilp/ in a hypothetical mini-language. Each of the four output candidates is flawed: *c*, the most faithful, has a consonant cluster; violating the markedness constraint *CC, as indicated by the asterisk at the intersection of *CC's column and *c*'s row. Candidate *b* has deleted a segment, and *a* has inserted a segment; these candidates violate the Faithfulness constraints DON'TDELETE and DON'TINSERT, respectively (phonologists' MAX and DEP). Candidate *d* has inserted a segment without breaking up the consonant cluster, violating both DON'TINSERT and *CC.

*CC is the highest-ranked constraint (ranking is indicated by left-to-right ordering of the constraints' columns—we can also write *CC >> DON'TDELETE >> DON'TINSERT). *Eval* first eliminates *c* and *d* from the competition (exclamation mark represents elimination) because they alone violate *CC. The shading in the cells to the right represents the irrelevance of *c*'s and *d*'s performance on any lower-ranked constraints. *Eval* next eliminates *b*, because it violates DON'TDELETE; the remaining candidate, *a*, is optimal, as indicated by the pointing finger. In this language, an input string /ilp/ is pronounced [ilip]; in another language, the constraint ranking, and thus the output, might be different. There are rankings that would choose *a* or *b* as the

optimal candidate. Candidate *d*, however is *harmonically bounded* by *a*, and by *c*: its violations are a proper superset of both *a*'s and *c*'s. Therefore, *d* cannot be the optimal candidate under any ranking of just these three constraints, though it could be optimal with a larger constraint set.

Wilson (2001) proposes an alternative formulation of *Eval*, in which markedness constraints are 'targeted': they compare only candidates that are maximally perceptually similar and impose only pairwise preferences on candidates. For each constraint, starting with the highest ranked, *Eval* adds any new pairwise preferences that do not contradict those imposed by higher-ranked constraints, and constructs the transitive closure.

OT IN LINGUISTIC THEORY

This section reviews why OT has been so widely adopted, and its advantages and disadvantages. See McCarthy (2001) for discussion of these issues.

OT was developed as a response to a "conceptual crisis at the center of phonological thought" (Prince & Smolensky 1993; 1) concerning the role of output constraints. In a 1970 *Linguistic Inquiry* article, Charles Kisseberth identified a 'conspiracy' in Yawelmani: rules of vowel insertion and deletion conspire to place every consonant adjacent to a vowel. Kisseberth proposed introducing constraints (such as *CCC, forbidding three-consonant clusters) to block or trigger rules, which could then be simplified and made more similar across languages. Output constraints were increasingly exploited in the literature, but many aspects of their use were unclear. How should a constraint be designated to blocker or trigger a rule? What if output constraints conflicted? How could non-absolute preferences be expressed? For example, Yawelmani allows the sequences *CiCC* and *CCiC*, but underlying CCC is repaired to *CiCC*. Therefore, in addition to the constraint *CCC and the rule of *i*-insertion, there must be a constraint preferring *CiCC* over *CCiC*. But this second constraint is violable, because *CCiC* sequences do occur. OT addressed these problems by eliminating rules entirely in favor of constraints, and specifying how constraints interact.

One advantage of OT over rule-based theories is that it predicts the emergence of the unmarked ('TETU'): a markedness constraint that is frequently violated in a language may still affect outputs. The constraint favoring *CiCC* over *CCiC* in Yawelmani, for example, is not surface-true (*CCiC* sequences do occur, because high-ranking faithfulness constraints preserve them), but when **CCC* forces a vowel to be inserted, *CiCC* is preferred over *CCiC*. A major contribution of OT has been focusing attention on TETU, of which many new cases have been found.

Another advantage of OT is its straightforward account of what McCarthy calls 'homogeneity of target/heterogeneity of process'. A rule specifies the structure that it applies to (target), and the operation to be performed on that structure (process). It has long been observed, however, that rules applying different processes to the same target tend to occur, cross-linguistically and within the same language. A rule-based theory has no explanation for why a structure should be a recurring target. In OT, however, the explanation is straightforward: there is a markedness constraint against the target, but whether and how the target is repaired depends on interaction with other constraints. In Figure 1, for example, permuting the constraint ranking yields three mini-languages: one that allows *CC* clusters, one that eliminates them by vowel insertion, and one that eliminates them by consonant deletion. The set of predicted languages that results from permuting the ranking of a group of constraints is its *factorial typology*. A proposed set of interacting constraints is considered viable only if its factorial typology matches the typology of observed languages—that is, it predicts all existent and no non-existent patterns.

In some cases, OT's prediction of heterogeneity of process may be overly exuberant. For example, all else being equal, languages that resolve intervocalic *CC* clusters by deletion delete the first consonant, not the second. Wilson's targeted constraints close this gap in the factorial typology: with targeted constraints, deleting the second consonant cannot be optimal under any constraint ranking.

OT is at a disadvantage in dealing with opacity. In a rule-based framework, opacity occurs when a later rule either eliminates the structure that caused an earlier rule to apply

(obscuring why the earlier rule applied), or creates a structure that would have caused an earlier rule to apply (obscuring why the earlier rule failed to apply). Standard OT, however, is unable to capture most opacity. Several additional proposals have therefore been made, including harmonic serialism, turbid output representations, output-output faithfulness, sympathy, targeted constraints, and constraint conjunction (see McCarthy 2001, chapter 3, for a survey). The computability consequences of these proposals, in learning and/or generation, remain to be established.

COMPUTABILITY OF OT

Generation

In rule-based frameworks, generation—mapping input to output, the speaker’s task—is straightforward. Each rule identifies target structures in a representation, makes the required change, and passes the result to the next rule. In OT, generation presents a computational challenge, because the candidate set may be infinite (in phonology, it is always infinite, because insertions are unlimited). In that case, *Eval* cannot proceed in the obvious way, by first going through all candidates and totaling violations of the highest-ranked constraint, because that first step would never end.

Eisner (1997), building on earlier work of Mark Ellison, proposes a simple way of dealing with the infinite candidate set: at every point in his generation algorithm, the candidate set is represented as a finite-state automaton (FSA), rather than as a list. This is possible if the candidates and constraints are expressed in Eisner’s Primitive Optimality Theory (OTP) formalism.

The winning candidate in OTP can be defined recursively. *Repns* is an FSA that accepts all syntactically well-formed OTP representations of input-output mappings. *Input* is an FSA that accepts mappings from the given input to any output. Intersecting *Repns* and *Input* produces an FSA, S_0 , that accepts well-formed mappings from the given input. S_0 is the initial candidate set.

Further, define an FSA C_i for each of the n constraints in the hierarchy, where C_i corresponds to the highest-ranked constraint. Each C_i accepts any mapping, but the arcs that a mapping traverses when it violates CONSTRAINT_i are weighted. C_i is intersected with S_0 to produce an FSA that accepts S_0 , but with the arcs corresponding to violations of CONSTRAINT_i weighted. Dijkstra's Best Paths algorithm, which finds the least-weighted path(s) through an FSA, is then applied to $C_i \cap S_0$ to yield an FSA (S_1) that accepts the representations in S_0 that minimally violate CONSTRAINT_i —i.e., the set of candidates left after CONSTRAINT_i has applied. Repeating this procedure for all n constraints, the winning candidate (or set of candidates, if there are not enough constraints to select a unique winner) is S_n .

Comprehension

Comprehension—the listener's task—has been little addressed for standard OT, though Eisner (2000) proposes an algorithm for comprehension under 'directional constraint evaluation'. A comprehension algorithm would yield, for a given output form, the (possibly infinite) set of inputs that would map to that output under the given grammar. The problem is not trivial: the input may contain a markedness violation not found in the output just in case the constraint ranking is such that the violation would have been repaired by a higher-ranking faithfulness constraint, and the result of the repair would be the observed output.

Learning

Learning—the child's task—includes (at least) two subtasks: building a lexicon and determining the constraint ranking of the target language. If the constraint set is not universal, the learner must also determine what the constraints of her language are; see Boersma (1998) for a model of learning articulatory and acoustic constraints, and Albright and Hayes (1999) for an algorithm that learns morphophonological constraints.

Little work exists on the learnability of the lexicon. Prince and Smolensky (1993) propose 'lexicon optimization': where possible, learners construct lexical representations that are

identical to the surface representations they hear. When the learner encounters alternations, such as the different pronunciations of the English plural suffix, she must construct a single lexical representation. Curtin (2000) presents evidence that children's early lexical representations are phonetically detailed and do not strip out redundancies; this suggests that lexical consolidation of different pronunciations of the same morpheme occurs relatively late in learning, perhaps after most of the constraint ranking is established.

The problem of establishing a constraint ranking has been addressed more thoroughly. Tesar and Smolensky's Constraint Demotion Algorithm and its variants, described in Tesar and Smolensky (2000), rank a set of constraints given a set of outputs. The algorithm compares an observed output (presumed to come from a mature speaker) to any candidate erroneously rated as optimal under the learner's current constraint ranking. In order to make the observed output optimal, for every constraint *B* that prefers the spurious output, some higher-ranked constraint *A* must prefer the observed output. If this is not already the case, the learner demotes *B* below *A*. The learner must know the input form in order to evaluate faithfulness constraints; in a more realistic model, some interleaving of input-learning and ranking-learning would be necessary. Variants of the algorithm accommodate the common proposal that the learner ranks markedness above faithfulness unless she encounters evidence to the contrary.

The Constraint Demotion Algorithm finds a ranking consistent with the learning data, if one exists. The algorithm has not been successfully generalized to learn variable grammars (see below), however, and is not robust to occasional errors in the learning data.

PROBABILISTIC AND VARIABLE OT

Intra-speaker variation is common in language: a speaker may produce an utterance differently on different occasions. For example, American English speakers optionally produce [nt] as a nasalized flap (so that 'winter' sounds similar to 'winner'). The desire to capture variability in OT has led to proposals of variable constraint ranking.

Anttila (1995) proposes that a ‘stratified’ constraint ranking is equivalent to all the linear hierarchies that are consistent with it, and the predicted frequency of a variant is the proportion of linear rankings that generate it. Suppose a language has the stratified ranking $A \gg \{B, C, D\}$ (i.e., B , C , and D are freely ranked, but below A), and a candidate a is optimal under both $A \gg B \gg C \gg D$ and $A \gg B \gg D \gg C$. The stratified ranking collapses six linear rankings, two of which produce a , so a should be observed 33% of the time. In a corpus study of Finnish genitive plurals, Anttila found a good match between predicted and observed frequencies of variants. No learning algorithm has been proposed, however, for grammars with free rankings.

Boersma (1998) proposes stochastic constraint ranking—ranking that is neither absolutely fixed nor absolutely free, but probabilistic. Each constraint in an individual’s grammar has a ranking value in arbitrary units. For every utterance, the speaker generates effective values for each constraint by randomly perturbing each ranking value slightly. Each constraint is thus associated with a probability density function centered on its ranking value. Figure 2 illustrates a mini-grammar in which constraint C_1 is nearly always top-ranked, and C_4 is nearly always bottom-ranked, but C_2 and C_3 are variably ranked, with a preference for the ranking $C_2 \gg C_3$.

Stochastic constraint ranking captures fine-grained frequencies. In an Anttilian grammar with three variably ranked constraints, a variant can occur only 0%, 33%, 67%, or 100% of the time, depending on which rankings produce that variant. In a Boersmian grammar with the same three constraints, the variant can occur at any frequency, depending on the ranking values of the constraints. Boersma and Hayes (2001) suggest some cases of very infrequent variants that would be difficult to capture in an Anttilian model, though firm data remain to be gathered.

An advantage of Boersma’s model is its learnability. Boersma’s Gradual Learning Algorithm can learn stochastic grammars from variable learning data (if the learning data are not variable, the rankings values learned are so far apart that the ranking is effectively fixed). In each learning trial of the algorithm, the learner compares its production to an adult target form. If there is a mismatch, the learner increments the ranking values of all constraints that prefer the learner’s incorrect form, and decrements the ranking values of all constraints that prefer the adult

form. The algorithm is robust to errors in the learning data; if an erroneous learning datum nudges a constraint in the wrong direction, subsequent data push it back. The Gradual Learning Algorithm can also model the course of acquisition. Curtin (2000) shows how, for the acquisition of prosody, the Gradual Learning Algorithm successfully models variability in children's productions, stage-like progression, and the order in which markedness constraints are demoted.

The Gradual Learning Algorithm can learn rates of variation because conflicting variants exert opposite influences on ranking values. The more frequent variant occurs in more learning trials, so the relevant constraints' ranking values are separated to the degree that the variants differ in frequency. The algorithm is also able to learn rates of lexical variation (situations in which each word's pronunciation is stable, but certain words undergo some phonological phenomenon and others do not), as shown in Zuraw (2000). In Zuraw's model, the resulting grammar has high-ranking faithfulness constraints that ensure the correct pronunciation of existing words, with lexical variation encoded in low-ranked constraints that come in to play in the production and comprehension of new words.

DISCUSSION

OT was partly inspired by neural networks. The ideas of optimization, parallel evaluation, competition, and soft, conflicting constraints are familiar. Prince and Smolensky (1997) discuss the implementation of OT in a neural network. Constraints are implemented as connection weights, and the network implements a Lyapunov function that maximizes 'harmony' ($\sum_{ij} a_i w_{ij} a_j$: the sum, for all pairs i, j of neurons, of the product of the neurons' activations and their connection weight). Hierarchically structured representations (e.g., consonants and vowels grouped into syllables) can be represented as matrices of neurons, where each matrix is the tensor product of a vector for a linguistic unit and a vector for its position in the hierarchy. Implementing strict domination (rather than the usual numerical weighting) of constraints remains unsolved, so translation between OT grammars and neural networks is not in general possible, however.

REFERENCES

Albright, A. and B. Hayes, 1999. An automated learner for phonology and morphology, manuscript, University of California, Los Angeles.

Anttila, A. 1997. Deriving variation from grammar: a study of Finnish genitives, in Variation, Change, and Phonological Theory, (F. Hinskens, R. van Hout, and L. Wetzels, eds.), Amsterdam: John Benjamins, pp. 35-68.

Boersma, P. 1998. Functional Phonology, The Hague: Holland Academic Graphics.

*Boersma, P. and B. Hayes. 2001. Empirical tests of the Gradual Learning Algorithm, Linguistic Inquiry, 32: 45-86.

Curtin, S. 2001. Enriched Lexical Representations and Constraint Organization in a Developing System, Dissertation, University of Southern California.

Eisner, J. 1997. Efficient generation in primitive Optimality Theory, in Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Association for Computational Linguistics, San Francisco: Morgan Kaufmann, pp. 313-320.

Eisner, J. 2000. Directional constraint evaluation in Optimality Theory, Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000), San Francisco: Morgan Kaufmann, pp. 257-263.

*Kager, R. 1999. Optimality Theory. Cambridge: Cambridge University Press.

To appear in Michael Arbib, ed., *Handbook of Brain Theory and Neural Networks*, 2nd edition. MIT Press. 12

*Legendre, G., J. Grimshaw and S. Vikner, eds., 2001. Optimality-Theoretic Syntax. Cambridge, MA: MIT Press.

McCarthy, J. 2001. A Thematic Guide to Optimality Theory, Cambridge: Cambridge University Press.

Prince, A. and P. Smolensky, 1993. Optimality Theory: Constraint interaction in generative grammar, Rutgers Center for Cognitive Science Technical Report TR-2 [see Kager 1999 for textbook treatment].

*Prince, A. and P. Smolensky, 1997. Optimality: from neural networks to universal grammar, Science, 275: 1604-1610.

Tesar, B. and P. Smolensky, 2000. Learnability in Optimality Theory, Cambridge, MA, MIT Press.

Wilson, C. 2001. Consonant cluster neutralisation and targeted constraints, Phonology, 18: 147-197.

Zuraw, K. 2000. Patterned exceptions in phonology, Dissertation, University of California, Los Angeles.

Figure 1




	/ilp/	*CC	DON'TDELETE	DON'TINSERT
<i>a</i>	 [ilp]			*
<i>b</i>	[il]		*!	
<i>c</i>	[ilp]	*!		
<i>d</i>	[ilpi]	*!		*

Figure 2 

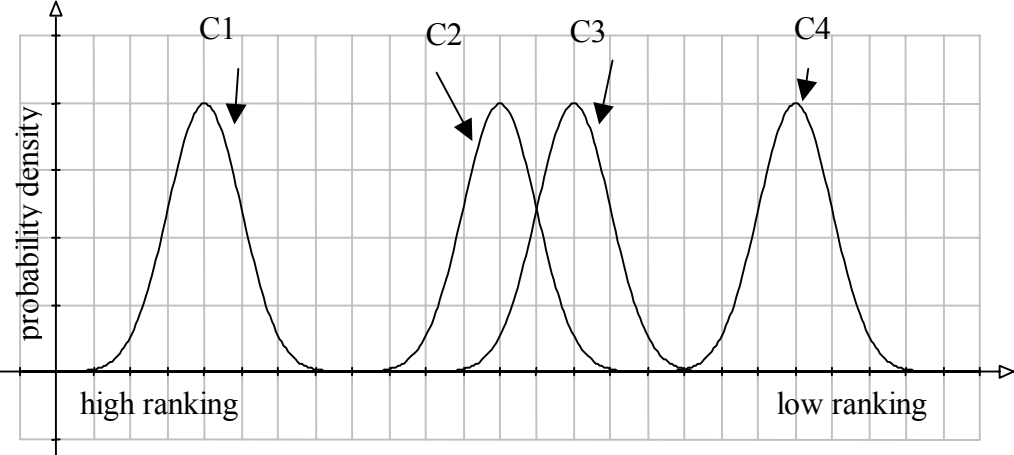


Figure 1 caption: OT tableau

Figure 2 caption: Stochastic constraint ranking