

The Logic of Optimality Theory

Michael Hammond
University of Arizona

1 Introduction

This paper is an attempt to determine the underpinnings of Optimality Theory (henceforth OT; Prince and Smolensky, 1993; McCarthy and Prince, 1993), and what logic might have to do with it.¹ This is important because, without this kind of work, very strange misunderstandings of the theory crop up. The work reported here is preliminary, but nothing of this sort has previously been attempted for OT, even though the theory has been around since 1993 and is the predominant framework for phonological research in North America.

The goals of the paper are as follows. First, I develop a logical statement of Optimality Theory. Next, I go on to prove some theorems. Third, I go on to show how the framework allows us to reason about partial derivations, and provides promise as the basis of a theory of acquisition and a theory of parsing.

The organization of the paper is as follows. First, I provide a review of OT. Next, I introduce the basic logical formulation for a pruned-down version of OT with only one constraint. In the following section, I extend the formulation to treat real OT, where there is more than one constraint. I go on in the next section to prove several theorems. Some of these are simply to show that the formalization is doing the right thing, but some of these are quite important in their own right. Finally, I go on to show how the formalization allows us to reason about derivations given incomplete information.

There are a number of results of this paper, but one real important one is that I establish that OT allows for multiple winners. That is, a tableau can have several winning candidates. This may seem obvious to the reader familiar with OT, but many presentations of OT imply or even state that this isn't so. For example:

As mentioned before, for each underlying form (input_i) there is a surface form (output_i) which is the candidate from the set
($\text{candidate}_1, \text{candidate}_2, \dots, \text{candidate}_n$)
that best satisfies the constraint ranking (Rosenthal (1994); p.10).

Likewise:

This set of candidate forms is then submitted in parallel to a hierarchy of constraints for evaluation, and the candidate form which 'best sat-

isfies' the hierarchy emerges as the optimal form. The optimal form is the output form, and vice versa.

$\text{Eval}\{\text{cand}_1, \text{cand}_2, \dots\} \rightarrow \text{cand}_k(\text{the output, given input}_i)$
(Hung (1994); p.2)

Finally, Tesar (1995) is most explicit:

The idea is that by examining the marks assigned by the universal constraints to all the candidate outputs for a given input, there is one which is least marked, or optimal: this is the *one and only* (my emphasis: MH) well-formed description that may be assigned to the input by the grammar (Tesar (1995); p.3).

My point in citing these authors is not to take them to task. Rather, my point is to demonstrate that this understanding of OT is widespread. Thus, a formalization that demonstrates the falsity of this position is of even more use.

For the reader not familiar with logic, the formalism used here may appear quite daunting. However, the basic ideas are rather straightforward. Moreover, I will in all cases recast formal statements in ordinary language.

2 Optimality Theory

Let's take phonology, or any component of grammar, as exhibiting a "tension". What I mean by this is that there are two forces that conflict in generating the observed phonology of a language.

Basically, each word needs to be different, but there are generalizations that tie together the expressions of a language. For example, in English, syllable-initial voiceless stops are aspirated. For example: *tack* [t^hæk], *cat* [k^hæt], and *pat* [p^hæt].

Optimality Theory is one way of accounting for this relation. What are the basic claims of OT? It is actually not altogether clear. One could answer this in terms of what Prince and Smolensky thought they were proposing, but it is surely more meaningful to understand this in terms of what linguists have actually concluded.²

There are three central claims. First, all phonological generalizations can be modeled with constraints. Second, constraints can interact only by strict ranking. Third, all constraints are universal. Prince and Smolensky actually maintain that they only really propose the second, but things have turned out differently. I go through each of these below.

There are three central components to the theory: GEN, CON, and EVAL. The idea is that any lexical representation can be mapped to any pronounceable thing.

This general mapping is performed by GEN. The correct output for some form is achieved by constraints. These constraints govern what is a well-formed mapping. For example, an input form like /kæt/ for *cat* would undergo GEN as follows.³

- (1) /kæt/ → {[kæt], [k^hæt], [dɔg], [karandaf], [k^hitɪ], etc.}

Lexical entries are mapped to surface forms in every imaginable way (GEN) and constraints (CON) limit the set of acceptable mappings. According to Prince and Smolensky, the constraints are finite and universal, but this is all that is known. The process whereby the correct candidate is chosen is called EVAL.


Consider now how the system works with a simple example. Let's go through some of what is required to get aspiration in English to come out correctly.⁴ The first thing we need is a constraint to enforce the inertia of lexical representations: FAITH. This constraint penalizes any candidate that differs from the input.

- (2) FAITH

Input and output should be identical.

If this were all there were to it, the lexical representation would win, as exemplified below. OT "derivations" are presented in constraint tableaux. The input is given in the upper left. Candidates are given along the left side and constraints along the top. Constraint violations are given as asterisks and the winning candidate is indicated with a pointing hand. Here, the form [kæt] wins.

- (3)

/kæt/	FAITH
 [kæt]	
[k ^h æt]	*!
[dɔg]	*!
[karandaf]	*!
[k ^h itɪ]	*!

This is, of course, incorrect. The initial consonant must surface as aspirated. We need a constraint that expresses this generalization about English. Presumably, this can be expressed in more general terms and is a reflection of some more general phonetic tendency regarding laryngeal gestures, but I won't attempt this here.

- (4) ASPIRATION

Word-initial voiceless stops are aspirated.

The ASPIRATION constraint is exemplified in the following tableau. Notice how this constraint also fails to pick out the correct candidate. Instead, it allows any

candidate that satisfies the constraint, including those candidates that are otherwise unfaithful to the input.

(5)

	/kæt/	ASPIRATION
	[kæt]	*!
☞	[k ^h æt]	
☞	[dɔg]	
	[karandaf]	*!
☞	[k ^h ɪtɪ]	

To get the right results, we need both constraints. However, simply having both constraints fails, since there is no candidate which passes both. We need both constraints plus the notion of “strict ranking”. Strict ranking says that constraints are ordered and violations of higher constraints are more important than violations of lower constraints. In this case, the ASPIRATION constraint outranks the FAITH constraint. This is written as follows.

(6) ASPIRATION ≫ FAITH

This can be exemplified in the following tableau. Ranking is indicated with left-to-right ordering of constraints. The interaction of constraints via strict ranking selects the correct candidate.

(7)

	/kæt/	ASP	FAITH
☞	[k ^h æt]		*
	[kæt]	*!	
	[dɔg]		**!*

There are several things to notice about this. First, notice that the winning candidate need not be perfect. For example, [k^hæt] wins, but violates FAITH. Second, ranking is strict. This means that a single violation of a higher constraint overpowers any number of violations of a lower constraint. Finally, notice that there will always be *at least* one winning candidate. In the logical formalization I propose below, these properties will follow.

A clear example of how the ranking is strict is given in the following hypothetical tableau. Notice that candidate [y] wins even though it has more total violations and more violations of constraint B than either of the other candidates.

(8)

/X/	A	B
[y]	*	***
[z]	**!	*
[w]	**!	

Notice also that there may be more than one winning candidate given the system as presented. A tableau showing how this works is given below. The two winners tie on all constraints, neither violating the higher constraint and both violating the lower constraint. I'll return to this below.⁵

(9)

/X/	A	B
[y]		*
[z]		*
[w]	*!	

The review of OT presented here has necessarily focused on the issues of relevance to this paper. In addition, I have not discussed any of the phonological substance of the model. For further detail, see Prince and Smolensky (1993), McCarthy and Prince (1993), or Archangeli and Langendoen (1997).

3 Logic

Let's now consider how this might be expressed in terms of first order logic. There have been various attempts to treat phonology in terms of logic before. See, for example, Bird (1995), Calder and Bird (1991), and Oehrle (1991). There have as yet been no attempts to do this for OT. The only discussion of logic in the context of OT is with regard to "constraint conjunction", e.g. Smolensky (1993) and Crowhurst and Hewitt (1997). These papers treat the question of whether independent constraints can be combined into a new constraint using logical conjunction or disjunction; they do not treat the larger question of a logical interpretation of constraint interaction.

The key move in the formulation I propose is to define winning candidates as true with respect to a predicate we can think of as "is a winner".

(10) Truth

The candidates that win with respect to a constraint, or a ranked set of constraints, are true with respect to the predicate "is a winner".

In what follows, I will sympolize this predicate as " ϕ ".

We now define a language in which we can talk about constraints, candidates, ranking, and violations. The basic ideas here are mostly quite straightforward. First, we want to have the power of first order logic so we can prove things. Second, we want to define terms and syntax for ranking and violations.

First, we need the usual logical connectives.

$$(11) \rightarrow, \wedge, \vee, \neg, \leftrightarrow, \exists, \forall$$

These have their usual interpretations: implication, conjunction, disjunction, negation, bidirectional implication, and the existential and universal quantifiers. In addition, we'll need some additional connectives which will be defined below. These allow us to formalize the comparison of violations and ranking.

$$(12) =, >, \equiv, \gg, \Leftarrow$$

Finally, I'll use lowercase greek letters, α, β, γ , etc., to refer to cells in a tableau, and capital greek letters, A, B, Γ , etc., to refer to candidates.

Let's now consider the auxilliary predicates of (12). First, we use the normal " \gg " symbol to indicate a difference in ranking, but we also need something to refer to two cells that have the same ranking: " \equiv ". We write $\alpha \gg \beta$ if α and β are cells for the same candidate (but different constraints) and α outranks β . We write $\alpha \equiv \beta$ if α and β are cells for the same constraint (but different candidates).

These relations have the expected properties. The " \equiv " relation is symmetric, reflexive, and transitive.

- (13) a. $\alpha \equiv \alpha$.
- b. If $\alpha \equiv \beta$, then $\beta \equiv \alpha$.
- c. If $\alpha \equiv \beta$ and $\beta \equiv \gamma$, then $\alpha \equiv \gamma$.

The " \gg " relation is transitive, but not symmetric, and not reflexive.

- (14) a. $\neg(\alpha \gg \alpha)$
- b. $\neg((\alpha \gg \beta) \wedge (\beta \gg \alpha))$
- c. If $\alpha \gg \beta$ and $\beta \gg \gamma$, then $\alpha \gg \gamma$.

We need to compare violations numerically, and the simplest way to do that is with successor notation: we define $0, 0', 0'', 0''', \dots$, where $0 = 0, 0' = 1, 0'' = 2, 0''' = 3$, etc. The " $>$ " and " $=$ " relations are straightforward. Using successor notation, " $>$ " relation is defined recursively. The " $=$ " relation can then be defined in terms of " $>$ ". Let x, y, z be numbers in successor notation.

$$(15) x > y \text{ if } x \text{ is } y' \text{ or if } (x > z) \wedge (z > y).$$

$$(16) \quad x = y \text{ if } \neg(x > y) \wedge \neg(y > x).$$

All the other comparisons can then be defined in terms of these. The key point here is that we can do the necessary math within the formalization proposed.

The other numerical comparisons can be treated as abbreviations.

$$(17) \quad \begin{array}{ll} \text{abbreviation} & \text{stands for} \\ \alpha < \beta & \beta > \alpha \\ \alpha \geq \beta & (\alpha > \beta) \vee (\alpha = \beta) \\ \alpha \leq \beta & (\alpha < \beta) \vee (\alpha = \beta) \end{array}$$

Here's a tableau so we can see how these can be used. I've marked individual cells with specific greek letters.

(18)

input	constraint A	constraint B
candidate 1	κ : *	λ : ***
candidate 2	μ : **	ν : *

The ranking relations we want are: $\kappa \equiv \mu$, $\lambda \equiv \nu$ and $\kappa \gg \lambda$, $\mu \gg \nu$. The numerical comparisons from the tableau above are: $\kappa = \nu$, $\mu > \kappa$, $\lambda > \kappa$, $\mu > \nu$, $\lambda > \nu$, $\lambda > \mu$.

The logical properties of the “ \Leftarrow ” operator are obviously key, and are treated in the next sections. For the moment, let “ \Leftarrow ” be a unary operator that should be interpreted as “is a winner”, or “is true”.

The essential points of the formalization given so far can be easily summarized in general terms. First, cells can be compared in terms of ranking relations. Second, cells can be compared in terms of the number of violations they exhibit. Let's now consider how this works. I'll do this in two passes. First, I'll show what we would need if there were only one constraint. Then we enrich the system to deal with normal OT where there is more than one constraint.

4 One Constraint

For a single constraint, the idea is simple: the candidate or candidates exhibiting the fewest violations win. This can be expressed formally as follows.

$$(19) \quad \Leftarrow \alpha \leftrightarrow \neg \exists \beta ((\beta \equiv \alpha) \wedge (\beta < \alpha))$$

This can be expressed straightforwardly in normal language as well. This says that some cell α is interpreted as true if and only if there does not exist some other

cell β that has fewer violations, where α and β are cells for the same constraint. This can also be expressed as follows.

$$(20) \quad \Leftarrow \alpha \leftrightarrow \forall \beta ((\beta \equiv \alpha) \rightarrow (\beta \geq \alpha))$$

Let's look at an example. Here the first and third candidates have the fewest violations and so they are both winners or “true with respect to \Leftarrow ”. In tableaux, I will mark cells that are true with a \top and cells that are false with \perp .

(21)

a.	<table border="1"> <tr> <th>/kæt/</th> <th>ASP</th> </tr> <tr> <td>[k^hæt]</td> <td></td> </tr> <tr> <td>[kæt]</td> <td>*!</td> </tr> <tr> <td>[dɔg]</td> <td></td> </tr> </table>	/kæt/	ASP	[k ^h æt]		[kæt]	*!	[dɔg]	
/kæt/	ASP								
[k ^h æt]									
[kæt]	*!								
[dɔg]									
b.	<table border="1"> <tr> <th>/kæt/</th> <th>ASP</th> </tr> <tr> <td>[k^hæt]</td> <td>\top</td> </tr> <tr> <td>[kæt]</td> <td>\perp</td> </tr> <tr> <td>[dɔg]</td> <td>\top</td> </tr> </table>	/kæt/	ASP	[k ^h æt]	\top	[kæt]	\perp	[dɔg]	\top
/kæt/	ASP								
[k ^h æt]	\top								
[kæt]	\perp								
[dɔg]	\top								

Recall that the winners aren't necessarily perfect. The formalization covers this case as well. Here's a sample tableau showing how this works. The key move is that the logical formulation looks for the cells with the fewest violations, not for cells with no violations. Here the first cell has the fewest violations and is thus true.⁶

(22)

a.	<table border="1"> <tr> <th>/X/</th> <th>A</th> </tr> <tr> <td>[y]</td> <td>*</td> </tr> <tr> <td>[z]</td> <td>**!</td> </tr> <tr> <td>[w]</td> <td>**!</td> </tr> </table>	/X/	A	[y]	*	[z]	**!	[w]	**!
/X/	A								
[y]	*								
[z]	**!								
[w]	**!								
b.	<table border="1"> <tr> <th>/X/</th> <th>A</th> </tr> <tr> <td>[y]</td> <td>\top</td> </tr> <tr> <td>[z]</td> <td>\perp</td> </tr> <tr> <td>[w]</td> <td>\perp</td> </tr> </table>	/X/	A	[y]	\top	[z]	\perp	[w]	\perp
/X/	A								
[y]	\top								
[z]	\perp								
[w]	\perp								

Recall that there can be more than one winner and the schema captures these cases as well. This is exemplified in the following tableau. Here the first and second cells satisfy the requirement that there is no other cell for the same constraint that has fewer violations. They are then both true.

(23)

a.	<table border="1"> <tr> <th>/X/</th> <th>A</th> </tr> <tr> <td>[y]</td> <td></td> </tr> <tr> <td>[z]</td> <td></td> </tr> <tr> <td>[w]</td> <td>*!</td> </tr> </table>	/X/	A	[y]		[z]		[w]	*!
/X/	A								
[y]									
[z]									
[w]	*!								
b.	<table border="1"> <tr> <th>/X/</th> <th>A</th> </tr> <tr> <td>[y]</td> <td>\top</td> </tr> <tr> <td>[z]</td> <td>\top</td> </tr> <tr> <td>[w]</td> <td>\perp</td> </tr> </table>	/X/	A	[y]	\top	[z]	\top	[w]	\perp
/X/	A								
[y]	\top								
[z]	\top								
[w]	\perp								

What happens if all candidates tie? The theory says that in such a case, all the candidates win. Here is a tableau showing how this looks. All cells satisfy the requirement that there is no other cell that has fewer violations. Hence all are true.

(24)

a.	<table border="1"> <tr><th>/X/</th><th>A</th></tr> <tr><td>[y]</td><td>*</td></tr> <tr><td>[z]</td><td>*</td></tr> <tr><td>[w]</td><td>*</td></tr> </table>	/X/	A	[y]	*	[z]	*	[w]	*
/X/	A								
[y]	*								
[z]	*								
[w]	*								

b.	<table border="1"> <tr><th>/X/</th><th>A</th></tr> <tr><td>[y]</td><td>⊤</td></tr> <tr><td>[z]</td><td>⊤</td></tr> <tr><td>[w]</td><td>⊤</td></tr> </table>	/X/	A	[y]	⊤	[z]	⊤	[w]	⊤
/X/	A								
[y]	⊤								
[z]	⊤								
[w]	⊤								

5 More than One Constraint

Let's now consider full OT where there is more than one constraint. The key move here is to restrict the choice of winning/true candidates with respect to some constraint to only those candidates that are true with respect to higher constraints

This can be put in prose as follows: when a constraint is ranked below other constraints, the truth values of its cells are a function of the truth values of higher-ranked constraints. Putting this in formal terms is a little complex, so I've broken it into two parts. The following general statement says that a cell α is true if two conditions hold.

$$(25) \quad \models \alpha \leftrightarrow (A \wedge B)$$

The first condition says that α is true if and only if there is no β where β has fewer violations than α , and α and β are with respect to the same constraint, and all cells that outrank β are true.

$$(26) \quad A: \neg \exists \beta ((\beta < \alpha) \wedge (\beta \equiv \alpha) \wedge \forall \delta ((\delta \gg \beta) \rightarrow \models \delta))$$

The second condition says that all cells that outrank α must also be true.

$$(27) \quad B: \forall \gamma ((\gamma \gg \alpha) \rightarrow \models \gamma)$$

Both of these can be cast in prose as well. Condition A is true if and only if there is no cell β such that i) β has fewer violations than α , ii) β is for the same constraint as α , and iii) all cells dominating β are true. Condition B is true if and only if all cells dominating α are true.

Notice that the one-constraint case is handled successfully by this more complex statement as well. There are no higher-ranked cells so the conditions on them are vacuously satisfied. If a constraint is top-ranked, then there are no higher-ranked cells. In this case, the conditions on higher-ranked cells are vacuously satisfied.

Let's go through a case.

(28)

a.

/kæt/	ASP	FAITH
[k ^h æt]		*
[kæt]	*!	
[dɔg]		**!*

The higher-ranked constraint is straightforward; The first and third candidates win because they have the fewest violations. Turning to the second constraint, the second candidate is false because it fails to satisfy condition B.

(29)

b.

/kæt/	ASP	FAITH
[k ^h æt]	⊤	*
[kæt]	⊥	⊥
[dɔg]	⊤	**!*

Both the first and third candidates satisfy condition B, but the third candidate does not satisfy condition A because the third candidate has more violations. Hence, the third candidate is false for the second constraint.

(30)

c.

/kæt/	ASP	FAITH
[k ^h æt]	⊤	⊤
[kæt]	⊥	⊥
[dɔg]	⊤	⊥

This is pretty straightforward and suggests that the truth of a candidate in toto can be seen as the conjunction of the truth values of its cells. In other words, a candidate is true if and only if all its cells are true. This is given formally below.

$$(31) \quad \models \Omega \leftrightarrow \left(\bigwedge_{n=1}^m \models \alpha_n \right)$$

A candidate Ω is a winner if and only if all of its cells are assigned \top .

6 Consequences

Let's now look at the consequences of all this. The basic idea will be to show that with this relatively simple formulation we can prove some important results about OT. I do this by proving several theorems.⁷ The general idea is what is most

important here. If we understand the logic of OT, we can understand the properties of OT more clearly. Consider first a relatively simple idea: all inputs have at least one output.

Theorem 1 *All inputs have at least one output.*

This can be proved relatively easily by mathematical induction. In the case at hand, the proof comes from the observation that given some set of positive integers, there will always be some subset that contains the elements that are smaller than all the rest. Moreover, the effect of constraint ranking, as defined here, is to narrow that further, but not to eliminate all candidates.

Proof:

- Assume that the candidate set has at least one candidate and all candidates exhibit some finite number of constraint violations.
- If there is only one constraint, then there will be a set of cells, $\{\alpha_1, \dots, \alpha_n\}$, representing constraint violations for the different candidates. The relation “<” will always pick out a set of at least one cell, $\{\alpha_i, \dots, \alpha_k\}$ where all members have the same number of violations, but where no other cells have as few. Hence there will be at least one winner in the case of a single constraint.
- Assume this is true for a system with n constraints.
- We must now show that it is true for a system with $n + 1$ constraints. Let us assume that the system with n constraints has resulted in a set $\{\alpha_1, \dots, \alpha_n\}$ of winners. We add another constraint C, such that all the other constraints outrank C. Constraint C assigns violations to all candidates. The schema entails that we only need to consider those candidates that are within the set of winners determined by the topmost n constraints. Among these, we again apply the relation “<” to pick out a set of cells that satisfy the same conditions. Though this only applies to the reduced set of winners $\{\alpha_1, \dots, \alpha_n\}$, it will still be the case that this set contains at least one cell with a minimum number of violations.
- Therefore a system with $n + 1$ constraints will have at least one winner.

□

Notice that this might seem to create problems for cases where phonologists want to rule out any output. For example, McCarthy and Prince (1993) discuss the case of Yidin^y where prosodic constraints interact so that in some morphological categories no output results. McCarthy and Prince get around this by allowing the candidate set to include the null parse. The distribution of such a candidate is

limited by a constraint M-PARSE, judiciously ranked. Thus even in cases where the facts tell us there is no output, the theory has been constructed so that the derivation results in a “null” output. Even when the winning candidate is one that violates a constraint like M-PARSE (McCarthy and Prince, 1993), there is still a winning candidate; it is simply a candidate that has no pronunciation.

Let’s do another theorem. This is simply to show that the formalization as presented gives us strict ranking, as desired.

Theorem 2 *Constraint ranking is strict.*

Recall that strict ranking is the claim that no number of violations of a lower-ranked constraint is sufficient to overpower a single violation of a higher-ranked constraint. This is exemplified in the following tableau.

(32)

	/X/	A	B
☞	[y]		***
	[z]	*!	

The theorem follows from the fact that the two conditions A and B are conjoined.

Proof:

- Consider the case of two constraints, referring to the tableau above. For a candidate to win, its cells must be assigned \top for all constraints. If some cell α has fewer violations than some cell β for some constraint A, then $\Rightarrow \beta$ will be assigned \perp . Because the condition that all higher-ranked cells must be assigned \top must generally hold, it follows that no number of violations for a lower-ranked constraint will have an effect.
- Assume that it is true for two constraints in a system of n constraints with lots of other irrelevant constraints ranked above and below the key constraints.
- For these other constraints to be irrelevant, they must assign \top to both of the candidates in question.
- Increasing the constraint set to $n + 1$ by adding another such constraint does not alter the conclusion. \square

Here is a trivial one. This theorem is simply the expression of the fact that numerical comparison defined over the numbers allowed by the successor notation we’ve adopted will always select a cell with no violations as a winner.

Theorem 3 *If a candidate violates no constraints, it is a winning candidate.*

Proof:

- If CON has only a single constraint, then the smallest number of violations a candidate can have is 0. Such a candidate will be assigned \top and will therefore be in the set of winning candidates.
- Assume this is true when CON has n constraints. That is, if there are n constraints and some candidate $[x]$ violates none of them, it will be in the winning set.
- Now add one more constraint Z and assign $[x]$ no violations of it. With no loss of generality, we can assume that Z is bottom-ranked. If $[x]$ has no violations of Z , then $[x]$ will be assigned \top for Z , since 0 is the smallest number of possible violations.
- Because candidate $[x]$ was a winner when there were only n constraints, it must have had been assigned \top for all n constraints. With $n + 1$ constraints, it is still assigned \top in all cells. Hence $[x]$ is in the winning set. \square

Here is the most interesting case. Interestingly, the proof is the simplest.

Theorem 4 *There can be multiple winners.*

Proof:

- Nothing in the formal system prevents some constraint A from assigning violations to all but 2 candidates ($\exists\alpha\exists\beta(\alpha = \beta = 0)$).
- Assume this is true for n constraints. That is, the first constraint rules out all but two and none of the $n - 1$ remaining constraints distinguish the two surviving candidates ($((\gamma \equiv \delta) \wedge ((\alpha \gg \gamma) \wedge (\beta \gg \delta))) \rightarrow (\gamma = \delta)$).
- To complete the induction, we simply add one more constraint that does not distinguish the two candidates. \square

The proof of this one relies on the existence of at least two candidates that are not distinguished by any constraints.

This may seem unlikely, but notice that this intuition (which I actually share!) is an intuition about the kinds of constraints in CON. The theory simply says that these are universal and finite. Nothing about OT as it stands forces us one way or the other with regard to this situation.

7 Partial Information

Let's now consider the question of partial information. The idea here is to think about whether the formulation might help us in other domains, e.g. parsing and

acquisition. The idea is that if a speaker is confronted with partial information about some phonological pattern in parsing an utterance or in determining the nature of their phonology, the logic of OT can help. The system we have developed here will allow us to reason about tableaux with partial information.

Let's consider the problem with a partially filled-in tableau. Imagine that all we know is that there are three constraints ranked $A \gg B \gg C$. Moreover we know that input /X/ is pronounced [y].

(33)

/X/	A	B	C
[y]			
[z]			
[w]			

What do we know in such cases? We know that [y] must be true for all cells. I put letters in each cell so we can refer to individual cells. We immediately know that all the cells of candidate [y] must be assigned \top .

(34)

/X/	A	B	C
[y]	\top_a	\top_b	\top_c
[z]	d	e	f
[w]	g	h	i

We also know that at least one cell of [z] and [w] is assigned false. We also know that a false cell will never outrank a true cell. The former is given formally below.

$$(35) \quad \neg(\varphi_d \wedge \varphi_e \wedge \varphi_f) \wedge \neg(\varphi_g \wedge \varphi_h \wedge \varphi_i)$$

If we want to think about the number of violations that occurs in each cell, we can reason further. For example: imagine we know that cell e has fewer violations than cell b . It follows that cell a must have fewer violations than cell d .

This follows because we know all cells for [y] are assigned \top . Hence if candidate [z] were assigned a \top in cell d for constraint A, the fact that cell e has fewer violations than cell b would entail that [y] cannot be in the winning set. Therefore [z] cannot be assigned a \top in cell d for constraint A. Since A is the topmost constraint, the fact that [y] is true and [z] is false must follow from a different number of violations. Hence, if cell e has fewer violations of B than cell b , then cell a must have fewer violations of A than cell d .

Reasoning about partial information is a very important result because it paves the way for OT-based theories of acquisition and parsing.

In acquisition, the idea would be that the child is confronted with partial information and must learn other information. The logical structure proposed provides a means to do that.

In parsing, a similar problem obtains. The listener is confronted with partial information and must deduce(!) additional information. Again, the logical structure proposed provides one possible mechanism by which a parser might proceed.

8 Conclusion

I've tried to develop a formulation of at least some aspects of OT in terms of first order logic. Essentially, a cell α is true if all higher-ranked cells are true and there is no other cell β for the same constraint where i) all cells dominating β are true, and ii) β has fewer violations than α . This formulation is surely naive in some regards, but it is the first attempt at this in the literature.

The formulation has allowed us to state and prove several theorems. First, all inputs have at least one output. Second, constraint ranking is strict. Third, if a candidate violates no constraints, it is a winning candidate. Finally, there can be multiple winners.

The most interesting theorem is the last as this is a domain where there has been some misunderstandings in the literature. Notice that there are a variety of ways one could respond to this.

One possibility that I pursued in earlier work Hammond (1994) was to show that one needs multiple winners to handle variation. This would put the issue on satisfyingly familiar empirical grounds.

The other possibility would be to revise the theory of constraints so that we guarantee that there will be no more than one winner in any particular case. This, in fact, is what the practice of most phonologists has been, but it would be really nice to put some theoretical teeth to this.

Finally, the framework proposed has implications for how we might view acquisition and parsing, where the subject is confronted with partial information about phonological representations.

I want to reiterate that the formalization is only partial, but I hope to have demonstrated that there are some useful consequences to this kind of work and hopefully this will encourage others to continue this kind of work.

9 Notes

¹This paper is for my former colleague Dick Oehrle, who recently left Arizona for greener pastures. Thanks also to Colleen Fitzgerald, James Myers, Diane Ohala, and the audience at WECOL for useful discussion. All errors are my own.

²It is also, of course, exceedingly difficult to know what the authors thought!

³I assume Correspondence theory here (McCarthy and Prince, 1995).

⁴There's much more required than this; see Hammond (1999) for more details.

⁵I addressed the issue of whether there is empirical support for allowing OT derivations to result in more than one candidate in an earlier unpublished paper (Hammond, 1994). See also Idsardi (1992).

⁶Of course, as we will see later on, a cell with no violations is necessarily in the set of cells with the fewest violations.

⁷The theorems are a compromise in terms of the degree of formality. I've done this for readability.

10 References

- Archangeli, Diana, and Terry Langendoen, ed. 1997. *Optimality Theory*. New York: Blackwell.
- Bird, Steven, ed. 1991. *Declarative perspectives on phonology*. University of Edinburgh.
- Bird, Steven. 1995. *Computational phonology: a constraint-based approach*. Cambridge: Cambridge University Press.
- Calder, Jo, and Steven Bird. 1991. Defaults in underspecification phonology. In Bird (1991), 107–125.
- Crowhurst, M., and M. Hewitt. 1997. Boolean operations and constraint interactions in optimality theory. Ms., U. of N. Carolina and Brandeis U, ROA.
- Hammond, Michael. 1994. An OT account of variability in Walmatjari stress. U. of Arizona, ROA.
- Hammond, Michael. 1999. *The Phonology of English*. Oxford: Oxford University Press.
- Hung, Henrietta J. 1994. The rhythmic and prosodic organization of edge constituents. Doctoral Dissertation, Brandeis University.
- Idsardi, W. J. 1992. The computation of prosody. Doctoral Dissertation, MIT.
- McCarthy, John, and Alan Prince. 1993. Prosodic morphology. U. Mass.

McCarthy, John, and Alan Prince. 1995. Faithfulness and reduplicative identity. In *Papers in Optimality Theory*, ed. J. Beckman, L. Dickey, and S. Urbanczyk, volume 18 of *U. Mass. Occasional Papers in Linguistics*, 249–384. [ROA].

Oehrle, Richard T. 1991. Prosodic constraints on dynamic grammatical analysis. In Bird (1991), 167–195.

Prince, Alan, and Paul Smolensky. 1993. Optimality theory. U. Mass and U. of Colorado.

Rosenthal, Samuel. 1994. Vowel/glide alternation in a theory of constraint interaction. Doctoral Dissertation, University of Massachusetts.

Smolensky, Paul. 1993. Harmony, markedness, and phonological activity. Presented at the Rutgers Optimality Workshop 1, [ROA].

Tesar, Bruce. 1995. Computational optimality theory. Doctoral Dissertation, University of Colorado. ROA.

Michael Hammond
Department of Linguistics
University of Arizona
Tucson, AZ 85721
hammond@u.arizona.edu