LINGUIST 726: Mathematical Methods in Linguistics
September 30, 2004

Formalizing Optimality Theory: Some Preliminaries[1]

Rajesh Bhatt

1 The Basics of Optimality Theory

1.1 What an OT Grammar Consists Of

- A set of inputs
- A function GEN that given an input generates a set of candidates
- A set of constraints
- A constraint hierarchy
- A function EVAL that given a candidate set and a constraint hierarchy returns the set of optimal candidates.

1.2 Strictness of Domination

An OT grammar is a special case of the computational problem known as constraint satisfaction / optimization.

• Input:

- a set of constraints ($C = \{c_1, \ldots, c_n\}$) and a set of weights associated with the constraints ($W = \{w_1, \ldots, w_n\}$).
- a set of candidates ($K = \{k_1, \ldots, k_n\}$)

• Goal: To determine which candidate satisfies the constraints optimally

[1]Thanks to Andries Coetzee and Joe Pater for answering my questions. The good ideas within are taken from Chs. 1 and 2 of Coetzee (2004); the inadequacies are mine.

We can model how well a particular candidate k satisfies the constraints by the following function f:

(1) $f(k) = w_1 * c_1(k) + \ldots + w_n * c_n(k)$

(to be consistent with how things are done in OT, I assume that if $c(k_1) < c(k_2)$, then k_1 satisfies c to a greater extent than k_2 does.)

The optimization problem now reduced to determining the $k \in K$ (or the subset of K) such that $\forall k'[k' \neq k \rightarrow f(k) < f(k')]$.

We could think of the ordering of the weights as giving us something like the ranking in OT.

Consider the following constraints with the suggested weights:
i. ***A**: must not contain an **a**, weight 5
ii. ***B**: must not contain a **b**, weight 1

Case 1:
Let $K = \{$**a, b**$\}$.
Then f(**a**) = 5 and f(**b**) = 1
Hence **b** is the optimal candidate.

Case 2:
Let $K = \{$**a, bbbbbb**$\}$.
Then f(**a**) = 5 and f(**bbbbbb**) = 6
Hence **a** is the optimal candidate.

What we see here is that a ‘lower ranked’ constraint can overturn the judgement of a ‘higher ranked’ constraint. This is quite different from what happens in OT.

In OT, a lower ranked constraint can never overturn the judgement of a higher ranked constraint. This property of an OT grammar is referred to as *strictness of domination* (cf. McCarthy (2002) and references therein), and restricts unbounded constraint interaction - what we could call ‘bargaining’ between groups of constraints.

In an OT grammar which ranked ***A** above ***B**, **a** would not be optimal in either grammar.

We can thus think of OT as a constraint optimization system where if c_1 is ranked above c_2, then w_1, the weight associated with c_1 is incommensurably larger than w_2, the weight associated with c_2.

1.3 What we want from an OT Grammar

Any OT grammar must be able to give us the set of most optimal candidates given a set of candidates and a constraint hierarchy.

In addition, there are certain other decisions one can make concerning what we get from an OT grammar.

1.3.1 Bipartition or a Ranking

Given a set of candidates $K = \{k_1, \ldots, k_n\}$, an OT grammar could return us:

(i) a bipartition of K into a set of winners and a set of losers:

$$K = \text{Winners} \cup \text{Losers}$$

The winners are all equally optimal and more optimal than the losers. No information is represented concerning further ranking among the losers.

(ii) a rank-ordering of K:

$$\Lambda_1 \succ \ldots \succ \Lambda_n$$
$$\text{s.t. } \{\Lambda_1, \ldots, \Lambda_n\} \text{ is a partition of } K.$$

• Classic OT: the grammar yields a bipartition of K.

• Coetzee (2004): the grammar yields a rank-ordering of K.[2]

1.3.2 What is Compared

Let K be the set of all possible candidates (independent of input).

We can distinguish between two kinds of subsets of K:

- Generated Comparison Sets: Λ is a generated comparison set if Λ is a subset of K and there is an input ι such that $GEN(\iota) = \Lambda$.

[2]Coetzee (2004) presents an analysis of frequency effects in intra-contextual variation which relies on the availability of such information. See Chs. 3-5.

- Non-Generated Comparison Sets: Λ is a subset of K and is not a Generated Comparison Set.

Two Options:

- Classic OT: the grammar can only apply to Generated Comparison Sets
- Coetzee (2004): the grammar can apply to arbitrary subsets of K[3]

2 Two Formalizations

We have seen two different conceptions of what one might expect an OT grammar to provide. These leads to two quite different formalizations.

2.1 A Classical OT formalization

• Samek-Lodovici and Prince (1999) and Prince (2002)

• Constraints are modelled as functions from sets of candidates to sets of candidates.[4]

(2) If $c \in C$, then $c : \wp(K) \rightarrow \wp(K)$

Suppose the constraint c applies to $\Lambda \subseteq K$.
Then $c(\Lambda) = \Lambda_1$ s.t.
(i) $\Lambda_1 \subseteq \mathit{\Lambda}$
(ii) the elements of Λ_1 are equally optimal w.r.t. c
(iii) the elements of Λ_1 are more optimal w.r.t. c than the elements of $\Lambda - \Lambda_1$.

[3]Coetzee (2004) motivates this decision on the basis of his analyses of frequency effects in intercontextual variation (Chs. 3-5) and well-formedness judgements and lexical decision tasks with Hebrew non-words (Ch. 6) that rely on the ability to compare candidates with different inputs.

[4]Note that this modelling of constraints combines rating of individual candidates with ranking of the candidates with respect to each other.

• The EVAL function which yields the set of optimal candidates is likewise a function from sets of candidates to sets of candidates. It is the composition of the constraint hierarchy.

(3) Let the constraint hierarchy $C = |c_1 \gg c_2 \gg \ldots \gg c_n|$.

Then EVAL: $\wp(K) \rightarrow \wp(K)$, where

$$\text{EVAL} = (c_n \circ \ldots c_2 \circ c_1)$$

i.e. for a set of candidates $\Lambda \subseteq K$,

$$\begin{aligned} \text{EVAL}(\Lambda) &= (c_n \circ \ldots c_2 \circ c_1)(\mathit{\Lambda}) \\ &= c_n(\ldots(c_2(c_1(\mathit{\Lambda})))) \end{aligned}$$

2.1.1 A Test Case

(4) $\Lambda = \{k_1, k_2, k_3, k_4, k_5\}$ evaluated by $|c_1 \gg c_2 \gg c_3|$

	c$_1$	c$_2$	c$_3$
k$_1$	*	*	
k$_2$	**	*	*
k$_3$	*	*	**
k$_4$	*	**	
k$_5$	**		**

(5) a. $\Lambda_1 = c_1(\mathit{\Lambda}) = \{k_1, k_3, k_4\}$
b. $\Lambda_2 = c_2(\mathit{\Lambda}_1) = \{k_1, k_3\}$
c. $\Lambda_3 = c_3(\mathit{\Lambda}_2) = \{k_1\}$
$\Lambda_3 = (c_3 \circ c_2 \circ c_1)(\mathit{\Lambda}) = c_3(c_2(c_1(\mathit{\Lambda})))$

2.1.2 Bipartition vs. Rank-Ordering

• The system presented above yields a bipartition of the candidate set into the set of winners and the set of losers.

• The exposition above suggests a picture where c_2 and c_3 don't even apply to all the candidates. This is however an illusion due to the serial presentation adopted above.

• Unless we assume a derivational implementation of OT where one constraint applies after another, something I take not to be in the spirit of OT, what applies to the candidate set Λ is the EVAL function. How the EVAL function is composed is invisible from the outside - that information is lost. What we have is a complex function.

• Now in order to determine the most optimal members, EVAL has to examine the entire candidate set. There is no other way. And once we have a way to determine the most optimal members of a set, we also have a way to impose a full rank-ordering on the set. [5]

(6) Let $\Lambda_1 = \text{EVAL}(\Lambda)$
  a. $\Lambda_2 = \text{EVAL}(\Lambda - \Lambda_1)$
  b. $\Lambda_3 = \text{EVAL}(\Lambda - \Lambda_2 - \mathit{\Lambda}_1)$
  c. $\ldots$
  d. $\Lambda_n = \text{EVAL}(\Lambda - \Lambda_{n-1} \ldots - \mathit{\Lambda}_1)$, $\Lambda_n = \mathit{\Lambda} - \mathit{\Lambda}_1 - \ldots - \mathit{\Lambda}_{n-1}$
  Then: $\Lambda_1 \succ \mathit{\Lambda}_2 \succ \ldots \succ \mathit{\Lambda}_n$

• But of course, in order to extract the full rank-ordering from this conception of EVAL, we have to assume that speakers are able to manipulate candidate sets in the manner indicated in (6). This is far from obvious and we will return to this point in the next section.

Conclusion: If speakers are able to manipulate candidate sets directly, then Samek-Lodovici and Prince (1999) / Prince (2002)'s formalization also induces a full rank-ordering on the candidate set (despite not being designed to do so).[6]

[5]It is not clear to me whether speakers even have direct access to the individual constraints or only to EVAL, the composition of the constraint hierarchy. I assume that speakers cannot directly access individual constraints. If they could directly access individual constraints, then what I am suggesting here would lead to an explosion in the amount of ranking information available (A. Coetzee p.c.).

[6]This is in the spirit of Coetzee's point that the theory as it stands makes finer distinctions available than utilized by most OT practitioners (pg. 32). The conditional in the conclusion could potentially undercut Coetzee's conclusion though.

2.1.3 What is Ordered

• As it stands, Samek-Lodovici and Prince (1999) / Prince (2002)'s EVAL function will produce the optimal set of candidates out of any set of candidates given to it -be it a Generated Comparison Set or a Non-Generated Comparison Set.

• It will not produce a full rank-ordering of the candidate set, but will otherwise be able to compare unrelated candidates.

• If we assume that speakers do not have direct access to candidate sets, their direct access being limited to inputs, then an alternative becomes available:

(7) Making EVAL a function from inputs to sets of optimal candidates:

Let the constraint hierarchy $C = |c_1 \gg c_2 \gg \ldots \gg c_n|$ and GEN: INPUTS $\rightarrow \wp(K)$, where INPUTS is the set of inputs and K the set of candidates.

Then EVAL: INPUTS $\rightarrow \wp(K)$, where

$$\text{EVAL} = (c_n \circ \ldots c_2 \circ c_1 \circ \text{GEN})$$

i.e. for an input $\iota \in$ INPUTS,

$$\begin{aligned} \text{EVAL}(\iota) &= (c_n \circ \ldots c_2 \circ c_1 \circ \text{GEN})(\iota) \\ &= c_n(\ldots(c_2(c_1(\text{GEN}(\iota))))) \end{aligned}$$

• With this reformulation, EVAL will only be able to compare Generated Comparison Sets. Speakers will not have direct access to the function that would apply to arbitrary subsets of K.

2.2 A Rank-Ordering Model of EVAL

• Ch. 2 of Coetzee (2004)

• We saw that the Samek-Lodovici and Prince (1999) / Prince (2002) model of EVAL does not (or at least not directly) yield a complete rank-ordering on the candidate set. Coetzee (2004) formalization is designed to directly yield such an ordering.[7]

[7]Moreton (1999)'s model, though not designed to impose a rank-ordering on the candidate set, does so quite well, and is, I think, ultimately equivalent and perhaps more direct than Coetzee's formalization. But Coetzee's formalization hews closer to the step-by-step way we think of

2.2.1 Formalizing Constraints

• Constraints are formalized as functions from candidates to natural numbers. They count stars.

(8) Let C be the set of constraints and K be the set of candidates. Then for all $c \in C$:

$c : K \rightarrow N$ s.t.

$\forall k \in K.c(k) =$ the number of violations ok c in k.

Applying (8):

(9) $\Lambda = \{k_1, k_2, k_3, k_4, k_5\}$ evaluated by $|c_1 \gg c_2 \gg c_3|$

	c_1	c_2	c_3
k_1	*	**	
k_2	**	*	*
k_3		*	**
k_4	*	**	
k_5	**		**

2.2.2 Ordering the Candidate Set Constraint by Constraint

(10) The relation $\approx_c$ on K.

Let $\Lambda \subseteq K$ be the candidate set to be evaluated, and C the set of constraints. Then for all $k_1, k_2 \in \Lambda$ and for all $c \in C$, let:

$k_1 \approx_c k_2$ iff $c(k_1) = c(k_2)$.

• For any constraint $c \in C$, and any subset $\Lambda \subseteq K$, $\approx_c$ is an equivalence relation on Λ.

constraints as applying.

• Since it is an equivalence relation on Λ, it induces a partition on Λ.

(11) a. Equivalence classes on $\Lambda \subseteq K$ in terms of $\approx_c$:

$$\forall k_1 \in \Lambda, [\![k_1]\!]_c := \{k_2 \in \Lambda | k_1 \approx_c k_2\}$$

b. Quotient set on Λ modulo $\approx_c$:

$$\Lambda/c := \{[\![k]\!]_c | k \in \Lambda\}$$

c. Λ/c is a partition of Λ.

• Now that we have a partition, we can define an order on the sets of candidates that constitute this partition. With candidates we couldn't do this because two candidates could be equally optimal w.r.t. a given constraint. That is no longer possible.

(12) The ordering relation $\leq_c$ on the set K/C:

$$\forall c \in C.\ \forall A, B \in \Lambda/c:$$
$$A \leq_c B \text{ iff } \exists k_1 \in A, k_2 \in B. c(k_1) \leq c(k_2)$$

• Two important properties of $\leq_c$

(13) a. $\leq_c$ defines a chain.[8]
(this means that c orders any two members of Λ/c.)

b. The chain defined by $\leq_c$ on Λ always has a minimum.

2.2.3 Combining Multiple Constraints

• So far we have been working with single constraints, we need a way to combine multiple constraints.[9]

[8] A chain is defined as follows:

i. Let P be an ordered set. Then P is a chain iff for all $x, y \in P$, either $x \leq y$ or $y \leq x$.

[9] Since Moreton (1999) applies all the constraints at once to the candidate set, out of the operations we discuss here, he only needs to do lexicographic ordering.

• The steps involved:

(14) Let $C = |c_1 \gg \ldots \gg c_n|$ be the constraint hierarchy and let $\Lambda \subseteq K$ be the candidate set.

a. The Cartesian Product $\Lambda/C_\times$:
$$\Lambda/C_\times = \mathit{\Lambda}/c_1 \times \ldots \times \mathit{\Lambda}/c_n$$

b. Impose lexicographic order on $\Lambda/C_\times$ using the various $\leq_{c_i}$'s appropriately, thus creating $\leq_\times$, a chain with a minimum on $\Lambda/C_\times$.

c. Simplify $\langle \Lambda/C_\times, \leq_\times \rangle$ to the final ordering $\langle \Lambda/C_{Com}, \leq_{Com} \rangle$ using n-ary set intersection on each of the members of $\Lambda/C_\times$.

• Some properties of Λ/C_{Com} and $\leq_{Com}$

(15) The ordering $\leq_{Com}$ abides by the Strictness of Domination Principle
i.e. let $\Lambda_1, \mathit{\Lambda}_2 \in \mathit{\Lambda}/C_{Com}$, with $k_1 \in \mathit{\Lambda}_1$ and $k_2 \in \mathit{\Lambda}_2$. Then:
$\Lambda_1 <_{Com} \mathit{\Lambda}_2$ iff
the highest rank constraint that distinguishes between k_1 and k_2 prefers k_1 over k_2.

(16) a. $\leq_{Com}$ defines a chain.

b. Λ/C_{Com} is a partition on Λ
Implication: any two candidates can be ordered with respect to each other. There is no indeterminacy in the output.

c. The chain defined by $\leq_{Com}$ always has a minimum.

Implication: there is always a unique entry point to the rank ordering of a potentially unbounded list that is taken here to be the output of the OT grammar. Further for any input, there is guaranteed to be an output. The grammar will never throw its hands up in the air and say 'I give up.'

• Indistinguishable Candidates: there is an intuition that certain candidates are indistinguishable from the perspective of an OT grammar. This can be stated formally as follows:

(17) Two equivalent characterizations of *indistinguishable*:

a. Through Λ/C_{Com}:

Two candidates $k_1, k_2 \in \mathit{\Lambda}$ are indistinguishable for a particular OT grammar iff

$$\exists \Lambda' \in \Lambda/C_{Com}.[k_1 \in \mathit{\Lambda}' \wedge k_2 \in \mathit{\Lambda}']$$

b. Through the equivalence relation $\approx_{Com}$ on Λ

(assuming constraint hierarchy $C = |c_1 \gg \ldots \gg c_n|$, and candidate set $\Lambda \subseteq K$):

$$\approx_{Com} = \{(k_1, k_2) : k_1, k_2 \in \mathit{\Lambda} \wedge \forall i \leq n[c_i(k_1) = c_i(k_2)]\}$$

Two candidates $k_1, k_2 \in \mathit{\Lambda}$ are indistinguishable for a particular OT grammar iff

$$k_1 \approx_{Com} k_2.$$

It should be noted that even though the above result is couched in terms of a particular OT grammar (i.e. a particular constraint hierarchy and a particular set of candidates), it is completely general.

Assuming that the set of constraints is universal, it follows that if two candidates are indistinguishable as defined above, they will be indistinguishable for every grammar in the factorial typology associated with the universal constraint set.

Indistinguishability modulo a particular grammar vs. Absolute Indistinguishability

3 Summing Up

If we assume that speakers have direct access to candidate sets, and can manipulate these candidate sets, the following results follow both under Samek-Lodovici and Prince (1999) / Prince (2002)'s formalization and Coetzee (2004)'s formalization:

(18) a. Rank-Ordering:

The OT Grammar makes available a full rank-ordering, and not just a bipartition of the candidate set into a set of winners and losers.

b. Comparison of Arbitrary Candidate Sets:

The OT Grammar makes it possible to impose an ordering on any arbitrary set of candidates, and not just Generated Candidate Sets.

If, however, speakers only have access to inputs and cannot manipulate candidate sets directly, then these conclusions do not follow.

References

Coetzee, A. (2004) *What it Means to be a loser: non-optimal candidates in Optimality Theory*, Doctoral dissertation, University of Massachusetts-Amherst, Amherst, Massachusetts. available at ROA.

McCarthy, J. J. (2002) *A Thematic Guide to Optimality Theory*, Cambridge University Press, Cambridge.

Moreton, E. (1999) "Non-Computable Functions in Optimality Theory," university of Massachusetts at Amherst ms.

Prince, A. (2002) "Entailed Ranking Arguments," rutgers University ms.

Samek-Lodovici, V., and A. Prince (1999) "Optima," university College London and Rutgers University ms.