

What is OT?

What is OT?

Wir kannten nicht sein unerhörtes Haupt
darin die Augenäpfel reiften. Aber ...

What is OT?

Neo: I know kung fu.

Morpheus: [*eyeing him, hand on chin*] Show me.

What is OT?

A concise overview

Alan Prince

Last mod. July 2, 2016

Acknowledgements

Thanks to Paula Houghton for originally suggesting that I pull this material out from the initial segments of various talks given at m@90, OCP, FLYM, UCSC, USCD, Stanford, and Rutgers over the last couple of years. Comments and responses from Houghton, Natalie DelBusso, Birgit Alber, and Naz Merchant have deepened my understanding of the issues and greatly improved the presentation.

Tableaux and diagrams constructed and calculations conducted in OTWorkplace (Prince, Tesar, and Merchant 2007-2015).

January 31, 2016

Goals

The aim is to provide what Leonard Susskind in another domain has ambitiously described as ‘the theoretical minimum’ — here, for understanding and working in or on OT. The conceptual perspective is that of the original work in the area, as further developed and clarified in the 21st century.

Presentation is relatively concise, in the interests of constructing an unobstructed view of OT as a theory of choice and the way that its premises guide analysis.

What is OT?

A concise overview

Alan Prince

- The System
- The Objects of OT
- Optimality
- Analysis

- **The System**
- The Objects of OT
- Optimality
- Analysis

Working within the System

OT is defined within a ***system***:

- a fully-specified formal entity
about which true and false things may be said.
- not chunks raggedly extracted from a looming, unknown whole (“Human Language”),
about which hedged, vague things may be said.

What is True and What is Not

In approaching reality, we proceed from system to system.

- So we always know what we're talking about.
- Or stand a chance of figuring it out.

A given *system* may be studied to shed light on the theory.

- Not because it tightly models some presumed facts.

Overfitting is the enemy of understanding.

Why the System

The need devolves from the definition of *optimality*.

An optimal form is better than all distinct candidates. *!!!*

- The set of competitors must be defined.

Comparison polls the judgment of all constraints. *!!!*

- The entire constraint set must be specified.

Outside the bounds of the *system*, *optimality* loses meaning.

$$S = \langle \text{GEN}_S, \text{CON}_S \rangle$$

GEN_S defines the candidates and candidate sets of S .

A definition need not be a procedure:

its role is to delimit, unambiguously.

CON_S defines the constraints of S .

A constraint is a function from candidates to
the non-negative integers $0, 1, \dots$

It associates each *candidate* from GEN_S with a penalty.

GEN and CON from Prince & Smolensky 1993/2004.

Role of the System

Analysis. Every *OT analysis* occurs within a system S , and rests on the definitions of GEN_S and CON_S . Because an 'analysis' is a grammar admitted by S .

No Exit. This is true whether or not these are defined explicitly. Because the force of logic is inescapable.

Same, same. Much the same will be true, in reality, of rational discourse within other theories when clearly defined.
- OT, taken seriously, allows very little wiggle room.

- The System
- **The Objects of OT**
- Optimality
- Analysis

The Objects of OT

1. Language

The *candidates* **optimal** under a given constraint hierarchy.

2. Grammar

The *set of all hierarchies* that yield the same language.

3. Typology

Extensional. The *languages* of S.

Intensional. The *grammars* of S.

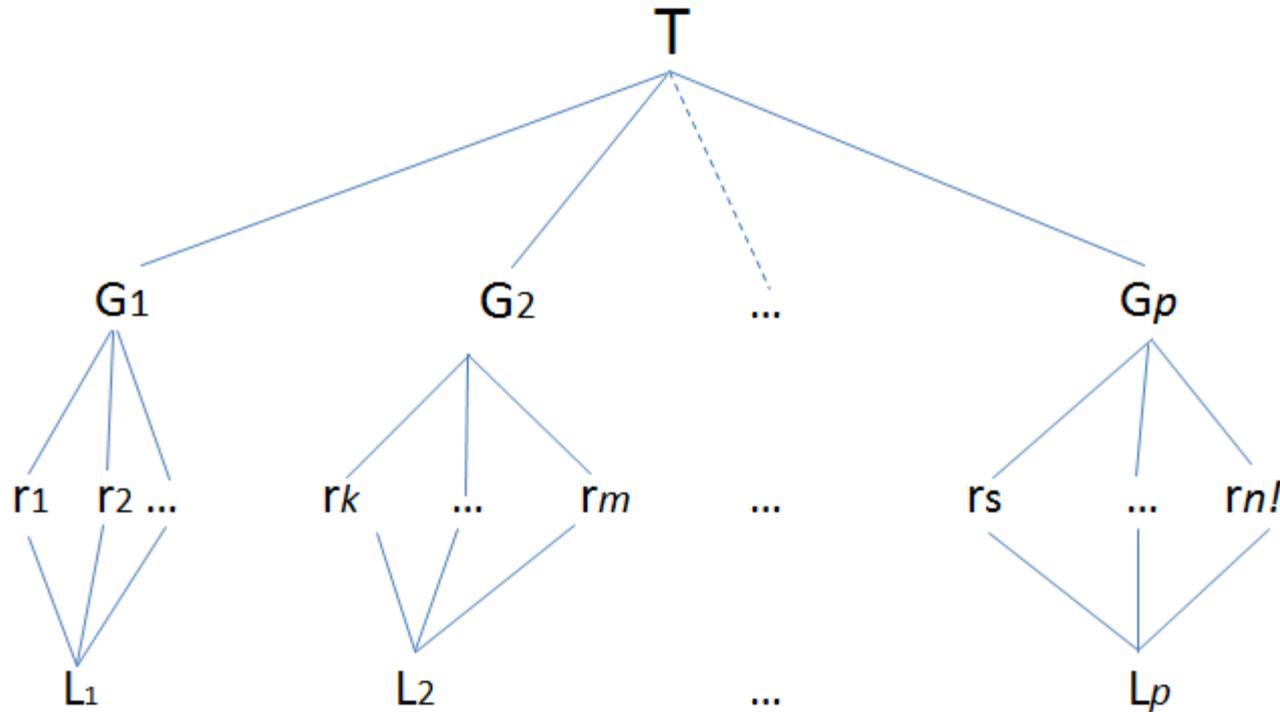
To understand S,

we must understand the typology of S.

To understand the typology of S,

we must understand its languages and their grammars.

The Objects of OT



A typology consists of grammars;
a grammar, of hierarchies that all deliver the same optima.

A Prosodic *System*

nGX

GEN_{nGX}: defining Candidate, Candidate set, Input, Output.

Candidate: A pair (Input, Output).

Candidate set (cset): All such pairs with same Input.

Input: A string of syllables, taken as atomic units.

Output for a given input: All parses of the input into a single Prosodic Word; no deletions or insertions.

Analyzed in Alber & Prince, 2015, in prep., and Alber, DelBusso, & Prince (to appear).

GEN_{nGX}: the structures

Categories

A **PrWd** consists of feet and syllables.

A **Foot** consists of syllables.

A **Syllable** is atomic.

Arrangements

- Constituents neither overlap nor recurse.
- A Foot may contain 1 or 2 syllables
- Every Foot has a unique *head* syllable
- Syllables may be parsed into Feet or not, freely.
- Every Prosodic Word contains at least one foot.

Spelling GEN_{nGX}

nGX assumes a familiar constituent structure for prosody.

For convenience, we allude to its elements as follows:

- X head of foot
- u nonhead of foot
- o syllable not parsed into foot
- edge of prosodic unit

Thus:

- Xu- trochaic foot: *head initial*
- uX- iambic foot: *head final*
- X- monosyllabic foot: *head initial and head final*
- o- unfooted syllable

CON_{nGX}

Name	Def.	Verbose: For each candidate, the constraint returns <i>the number of matches in Output</i> to:
1. Parse-s	*o	a syllable not belonging to a foot.
2. lamb	*-X	a head-initial foot: -Xu- , -X-
3. Troch	*X-	a head-final foot: -uX- , -X-
4. AFL	*(σ ,ft): σ ...ft	each pair (syll , foot), where σ precedes ft.
5. AFR	*(σ ,ft): ft... σ	each pair (syll , foot) , where foot precedes σ .

-
- *P takes a candidate as argument, returns the number of matches in it to pattern P.
 - Defns. of AFL (“All Feet Left”), AFR (“All Feet Right”) simplified from Hyde 2012.
 - Defns. of lamb, Troch are ‘new’ in that they penalize, rather than accept, -X-.

nGX Sampled

Violation Tableaux — VTs.

2s and 3s csets complete:
all admitted candidates.

Plus beginning of 4s
continuing downward...
to list the 32 more 4s cand.

input	output	opt	P-s	lamb	Troch	AFL	AFR
oo	-uX-		0	0	1	0	0
	-o-X-		1	1	1	1	0
	-Xu-		0	1	0	0	0
	-X-o-		1	1	1	0	1
	-X-X-		0	2	2	1	1
ooo	-o-uX-		1	0	1	1	0
	-o-o-X-		2	1	1	2	0
	-uX-o-		1	0	1	0	1
	-uX-X-		0	1	2	2	1
	-o-Xu-		1	1	0	1	0
	-o-X-o-		2	1	1	1	1
	-o-X-X-		1	2	2	3	1
	-Xu-o-		1	1	0	0	1
	-Xu-X-		0	2	1	2	1
	-X-o-o-		2	1	1	0	2
	-X-uX-		0	1	2	1	2
	-X-o-X-		1	2	2	2	2
	-X-Xu-		0	2	1	1	2
	-X-X-o-		1	2	2	1	3
	-X-X-X-		0	3	3	3	3
oooo	-o-o-uX-		2	0	1	2	0
	-o-o-o-X-		3	1	1	3	0
	-o-uX-o-		2	0	1	1	1
	-o-uX-X-		1	1	2	4	1
	-o-o-Xu-		2	1	0	2	0
	-o-o-X-o-		3	1	1	2	1
	-o-o-X-X-		2	2	2	5	1
	-uX-o-o-		2	0	1	0	2
	-uX-uX-		0	0	2	2	2
	-uX-o-X-		1	1	2	3	2
	-uX-Xu-		0	1	1	2	2

nGX is so named because

- *n*ew defn. of lamb/Troch used
- **G**eneralized Alignment positions feet
- **X** occurs at least once in every word

What is nGX about?

nGX deals with aspects of stress prosody that depend on

- grouping into feet
- headedness of feet
- foot population of word

nGX abstracts away from

- distinctions in prominence among foot heads
- effects of the internal composition of syllables

The excluded traits often function independently of those represented, making this a mild abstraction.

-
- Mildness I: *Many patterns arrange feet without reference to main stress.*
 - Mildness II: *Every QS system has a QI system inside it.*
 - Far more drastic abstractions are undoubtedly necessary for theoretical advance.

- The System
- The Objects of OT
- **Optimality**
- Analysis

Optimality by Mass Filtration

Take the Best, Ignore the Rest /verbose=on

1. Filtration **by a single constraint C** of a set of candidates K.
C[K] is the best of K as judged by C: smallest penalty.
2. Filtration **by a hierarchy of constraints H** = $C_1 \gg C_2 \gg \dots \gg C_n$.
 - Take the best on $C_1 = C_1[K]$ — *the best*
 - Filter this with C_2 to get $C_2[C_1[K]]$ — *the best of the best*
 - Iterate until you've gone through all of H. — *...of the best*

3. Optimality

A candidate is *optimal* iff it's among the result of filtering by all.

A 'hierarchy' or 'ranking' is a linear order on the constraint set CON_S .

Optimality by Mass Filtration

Take the Best, Ignore the Rest /verbose=off

1. Filtration of a set of candidates K by a single constraint C .

$C[K] = \{q \in K \mid C(q) \text{ is } \mathbf{minimal}: \text{ for all } z \in K, C(q) \leq C(z)\}$

- $C(q)$ is the integer value assigned by C to q .
- $C[K]$ is the *best* of K , as judged by C .

2. Filtration by a hierarchy $H = C_1 P$ (P a sequence of constraints)

$H[K] = C_1 P[K] = P[C_1[K]]$ (recursive formulation)

3. **Optimality.** For $H = C_1 \gg C_2 \gg \dots \gg C_n$, $C_k \in \text{CON}_S$

$q \in K$ is **optimal on H** iff $q \in H[K]$.

VT with columns in ranking order

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-		0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

A Filtration

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-		0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

A Filtration

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-		0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

A Filtration

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-		0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

A Filtration

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-		0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

A Filtration

nGX.5s				*o	*-X	*X-	func:AFL	func:AFR
Cand#	input	output	opt	1:P-s	2:lamb	3:Troch	4:AFL	5:AFR
1	5s	-X-uX-uX-	OPT!	0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

What we have got

Why are we sure that we've got the *optimum*?

Shouldn't we filter the entire set of 119 five-syllable forms?

- To be *optimal* is to be the best of *all*.

Here we filter only those 12 forms that are *possible optima*.

Some ranking exists that makes each of them optimal.

But the other 107 are never optimal:

they lose to something else under every ranking.

They are said to be *harmonically bounded*.

They never affect the choice of optimum

nor the rankings that choose it.

The Selection Problem, Solved

The filtration view works splendidly for us if we have the hierarchy H in hand, as in the case just reviewed.

Having fixed GEN_S , CON_S , and H , we can find the optima of H .

This is the *selection problem*: what does H select from K ?

We can solve it because we know

- All the candidates and what they compete with
- All the constraints
- The definition of optimality
- The particular hierarchy we want to filter with

The Other Way Round

But the usual problem taken on by the analyst is quite different.

With observations, generalizations and representations in mind,
a set of desired *optima* is identified.

We know GEN_S and CON_S and we want certain optima to emerge.

- The ranking problem:

What *hierarchies* select the desired optima?

The Ranking Problem, Solved

The ranking problem finds its resolution in the following fact:

Veridical ranking information can be obtained
by comparing 2 candidates, over the entirety of CON_S ,
with one asserted to be optimal.

This emerges from a deep feature of OT:
the independence of irrelevant alternatives.

Choice between q and z depends only on q and z,
not on how any other choices are made.
But knowledge of all constraints is still required.

Optimality by Pairwise Comparison

/verbose=on

1. **Better than** on a constraint

q is *better than* z on C

iff C assigns a smaller penalty to q than to z .

2. **Better than** on a hierarchy H

q is *better than* z on H

iff q is *better than* z on

the highest ranked *constraint* in H

that assigns different penalties to q and z .

3. **Optimal**

q is optimal in K on H

iff q is *better on* H than *all* (violation-distinct) competitors.

Optimality by Pairwise Comparison

/verbose=off

1. **Better than** on a constraint C. For candidates q,z
q is *better than* z on C iff $C(q) < C(z)$.
2. **Better than** on a hierarchy $H = CP$
q is *better than* z on $H = CP$ iff
 - q is better than z on C, or
 - if $C(q) = C(z)$ (i.e. z is *not* better than q on C)then q is better than z on the hierarchy P.
3. **Optimal** in K on H
q is optimal in K on H
iff q better than z on H for *all* (distinct) z in K.

Nothing new under the Sun

The definition of optimality from pairwise comparison is equivalent to the mass filtration definition.

The same candidates are optimal under both.

The two ways of thinking comport with different situations.

Outside of doctrine-driven discourse, it's good not bad to command several ways of thinking about things.

Comparing a Pair

Suppose we want \mathbf{q} better than \mathbf{z} .

- *Which hierarchies make this happen?*

We construct the **Elementary Ranking Condition** $\mathbf{q} \sim \mathbf{z}$,
collecting the judgment of *every* constraint $\mathbf{C} \in \text{CON}_S$
on the relation between \mathbf{q} and \mathbf{z} .

From \mathbf{C} , we build a new kind of function \mathbf{dC} ,
which returns the effective *difference* between \mathbf{q} and \mathbf{z} .

Win, Lose, or Draw

Let us define dC , a function of pairs, from $C \in \text{CON}_S$.

1) $dC(q,z) = W$ when $C\{q,z\} = \{q\}$, *i.e.* $C(q) < C(z)$
'q is better than z on C'

2) $dC(q,z) = e$ when $C\{q,z\} = \{q,z\}$, *i.e.* $C(q) = C(z)$
'q is the same as z on C'

3) $dC(q,z) = L$ when $C\{q,z\} = \{z\}$, *i.e.* $C(z) < C(q)$
'z is better than q on C'

By the Bootstraps

C and dC are different functions.

$$C: \text{CAND}_S \rightarrow \{0, 1, 2, \dots\}$$

$$dC: \text{CAND}_S \times \text{CAND}_S \rightarrow \{W, L, e\}$$

dC is *derived from* $C \in \text{CON}_S$.

CAND_S = candidates of S.

- C maps from candidates to the non-negative integers.
- dC maps from pairs of candidates to a 3-element set.

They provide different information.

They do *not* display the same information in a different ‘format’.

We observe a notational distinction here for purposes of clarity.

Though not customary, this disables the murky practice of displaying both kinds of information in the same tableau.

What the ERC says

The ERC $\mathbf{q} \sim \mathbf{z}$ is the collection of evaluations $\mathbf{dC}(\mathbf{q}, \mathbf{z})$ for all C in CON_S .

Conventionally presented as list ('vector') with some arbitrary order on the constraints.

ERC	dIamb	dAFR	dTroch	dP-s	dAFL
$\mathbf{q} \sim \mathbf{z}$	e	W	W	L	e

The ERC vector $\mathbf{q} \sim \mathbf{z}$ tells us this:

**In every H in which \mathbf{q} is *better than* \mathbf{z}
either **AFR (W)** or **Troch (W)** dominates **P-s (L)**.**

Why?

If P-s were encountered before AFR and Troch in any filtration,
 \mathbf{z} would be chosen from $\{\mathbf{q}, \mathbf{z}\}$, and thus be better than \mathbf{q} .

Obtaining \mathbf{q}

Suppose we want \mathbf{q} *better than* \mathbf{z} .

What hierarchies \mathbf{H} make this happen?

- We examine the **ERC** $\mathbf{q} \sim \mathbf{z}$ to find out.

Elementary Ranking Condition. Some W dominates all L 's.

This is what we learn from comparing with a desired optimum.

Grammar. A *grammar* is a set of ERCs that collectively tells us how to make \mathbf{q} better than **all** violation-distinct competitors \mathbf{z} .

The **ERC** is the essential quantum of ranking information from which grammars are built.

Winning is Everything

nGX.5s								
Cand#	input	output	opt	P-s	lamb	Troch	AFL	AFR
1	5s	-X-uX-uX-	WIN!	0	1	3	4	6
2		-uX-uX-X-		0	1	3	6	4
3		-X-Xu-Xu-		0	3	1	4	6
4		-Xu-Xu-X-		0	3	1	6	4
5		-uX-uX-o-		1	0	2	2	4
6		-o-uX-uX-		1	0	2	4	2
7		-Xu-Xu-o-		1	2	0	2	4
8		-o-Xu-Xu-		1	2	0	4	2
9		-uX-o-o-o-		3	0	1	0	3
10		-o-o-o-uX-		3	0	1	3	0
11		-Xu-o-o-o-		3	1	0	0	3
12		-o-o-o-Xu-		3	1	0	3	0

Divide and Conquer

nGX.5s								
Erc	Input	Winner	Loser	dP-s	dlamb	dAFL	dTroch	dAFR
1~8	5s	-X-uX-uX-	-o-Xu-Xu-	W	W		L	L
1~7	5s	-X-uX-uX-	-Xu-Xu-o-	W	W	L	L	L
1~11	5s	-X-uX-uX-	-Xu-o-o-o-	W		L	L	L
1~12	5s	-X-uX-uX-	-o-o-o-Xu-	W		L	L	L
1~6	5s	-X-uX-uX-	-o-uX-uX-	W	L		L	L
1~5	5s	-X-uX-uX-	-uX-uX-o-	W	L	L	L	L
1~9	5s	-X-uX-uX-	-uX-o-o-o-	W	L	L	L	L
1~10	5s	-X-uX-uX-	-o-o-o-uX-	W	L	L	L	L
1~4	5s	-X-uX-uX-	-Xu-Xu-X-		W	W	L	L
1~3	5s	-X-uX-uX-	-X-Xu-Xu-		W		L	
1~2	5s	-X-uX-uX-	-uX-uX-X-			W		L

P-s \gg {lamb, Troch, AFL, AFR}

nGX.5s								
Erc	Input	Winner	Loser	dP-s	dlamb	dAFL	dTroch	dAFR
1~8	5s	-X-uX-uX-	-o-Xu-Xu-	W	W		L	L
1~7	5s	-X-uX-uX-	-Xu-Xu-o-	W	W	L	L	L
1~11	5s	-X-uX-uX-	-Xu-o-o-o-	W		L	L	L
1~12	5s	-X-uX-uX-	-o-o-o-Xu-	W		L	L	L
1~6	5s	-X-uX-uX-	-o-uX-uX-	W	L		L	L
1~5	5s	-X-uX-uX-	-uX-uX-o-	W	L	L	L	L
1~9	5s	-X-uX-uX-	-uX-o-o-o-	W	L	L	L	L
1~10	5s	-X-uX-uX-	-o-o-o-uX-	W	L	L	L	L
1~4	5s	-X-uX-uX-	-Xu-Xu-X-		W	W	L	L
1~3	5s	-X-uX-uX-	-X-Xu-Xu-		W		L	
1~2	5s	-X-uX-uX-	-uX-uX-X-			W		L

lamb ≫ Troch

nGX.5s								
Erc	Input	Winner	Loser	dP-s	dlamb	dAFL	dTroch	dAFR
1~8	5s	-X-uX-uX-	-o-Xu-Xu-	W	W		L	L
1~7	5s	-X-uX-uX-	-Xu-Xu-o-	W	W	L	L	L
1~11	5s	-X-uX-uX-	-Xu-o-o-o-	W		L	L	L
1~12	5s	-X-uX-uX-	-o-o-o-Xu-	W		L	L	L
1~6	5s	-X-uX-uX-	-o-uX-uX-	W	L		L	L
1~5	5s	-X-uX-uX-	-uX-uX-o-	W	L	L	L	L
1~9	5s	-X-uX-uX-	-uX-o-o-o-	W	L	L	L	L
1~10	5s	-X-uX-uX-	-o-o-o-uX-	W	L	L	L	L
1~4	5s	-X-uX-uX-	-Xu-Xu-X-		W	W	L	L
1~3	5s	-X-uX-uX-	-X-Xu-Xu-		W		L	
1~2	5s	-X-uX-uX-	-uX-uX-X-			W		L

AFL \gg AFR

nGX.5s								
Erc	Input	Winner	Loser	dP-s	dlamb	dAFL	dTroch	dAFR
1~8	5s	-X-uX-uX-	-o-Xu-Xu-	W	W		L	L
1~7	5s	-X-uX-uX-	-Xu-Xu-o-	W	W	L	L	L
1~11	5s	-X-uX-uX-	-Xu-o-o-o-	W		L	L	L
1~12	5s	-X-uX-uX-	-o-o-o-Xu-	W		L	L	L
1~6	5s	-X-uX-uX-	-o-uX-uX-	W	L		L	L
1~5	5s	-X-uX-uX-	-uX-uX-o-	W	L	L	L	L
1~9	5s	-X-uX-uX-	-uX-o-o-o-	W	L	L	L	L
1~10	5s	-X-uX-uX-	-o-o-o-uX-	W	L	L	L	L
1~4	5s	-X-uX-uX-	-Xu-Xu-X-		W	W	L	L
1~3	5s	-X-uX-uX-	-X-Xu-Xu-		W		L	
1~2	5s	-X-uX-uX-	-uX-uX-X-			W		L

All that it takes

nGX.5s								
Erc	Input	Winner	Loser	dP-s	dlamb	dAFL	dTroch	dAFR
1~8	5s	-X-uX-uX-	-o-Xu-Xu-	W	W		L	L
1~7	5s	-X-uX-uX-	-Xu-Xu-o-	W	W	L	L	L
1~11	5s	-X-uX-uX-	-Xu-o-o-o-	W		L	L	L
1~12	5s	-X-uX-uX-	-o-o-o-Xu-	W		L	L	L
1~6	5s	-X-uX-uX-	-o-uX-uX-	W	L		L	L
1~5	5s	-X-uX-uX-	-uX-uX-o-	W	L	L	L	L
1~9	5s	-X-uX-uX-	-uX-o-o-o-	W	L	L	L	L
1~10	5s	-X-uX-uX-	-o-o-o-uX-	W	L	L	L	L
1~4	5s	-X-uX-uX-	-Xu-Xu-X-		W	W	L	L
1~3	5s	-X-uX-uX-	-X-Xu-Xu-		W		L	
1~2	5s	-X-uX-uX-	-uX-uX-X-			W		L

Follow the Logic

An ERC is a logical expression:

Some W dominates all L 's.

W_j or ...or $W_m \gg L_k$ and ... and L_n

A logic of entailment is associated with expressions of this type.

By this logic, we can eliminate all redundancy from an ERC set.

We may even be able to simplify it beyond what we are given.

This can be done algorithmically.

Follow the Propositional Logic

ERC 1~10 [WLLLL] says:

(1) **P-s** dominates all of {lamb, Troch, AFL, AFR}

ERC 1~7 [WWLLL] says:

(2) **P-s or lamb** dominates all of {Troch, AFL, AFR}

BUT

*If (1) 'P-s dominates everything else' is true,
then (2) 'P-s or lamb dominates various other constraints' is also true.*

Statement (2) is redundant and adds nothing to (1).

We may *omit it* from a defining collection of ERCs: a grammar.

Follow the ERC Logic

To reduce an ERC set to its essentials, we must also take account of the order properties of constraint hierarchies.

‘ \gg ’ is a strict order: asymmetric, irreflexive, transitive.

This can be carried out by simple operations on W,L,e values, with no reference to \wedge, \vee, \gg and their formal properties.

The useful ERC logic operation of combination is *fusion*. It gives us

$$[eWeLe] \circ [eeWeL] = eWWLL$$

allowing us to eliminate $1\sim 4$ in favor of $\{1\sim 2, 1\sim 3\}$.

Fusion of these ERCs encapsulates in one stroke the following argument: in any given hierarchy, one of the L's must dominate the other (strictness). One of the W's must dominate it (from $1\sim 2$ and $1\sim 3$). Therefore, one of the W's dominates both L's (transitivity).

A grammar

The ERC expresses the *kind* of ranking relations that data yields.

It can be used to represent those relations *per se*, regardless of where they come from.

This ERC set represents our findings:

P-s	lamb	AFL	Troch	AFR
W	L	L	L	L
	W		L	
		W		L

This is the MIB, the 'Most Informative Basis', of the grammar. See Brasoveanu & Prince 2005/11.

A grammar

Further calculations can be made.

If we remove all L's derivable from other ERCs in the set,
those that follow from the transitivity of ranking,

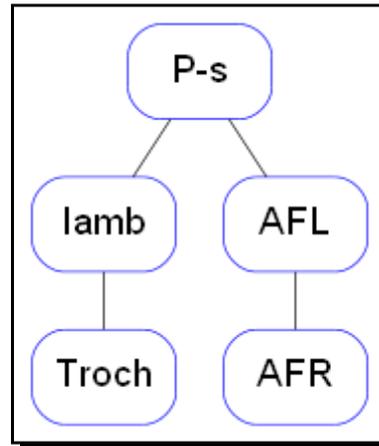
We arrive at a sparser representation:

P-s	lamb	AFL	Troch	AFR
W	L	L		
	W		L	
		W		L

This is the SKB, the 'Skeletal Basis', of the grammar. See Brasoveanu & Prince 2005/11.

A picture

In some cases, a clean picture can be made from the sparse set:



Such a *Hasse Diagram* is available when there is one W per ERC.

In the general case, with ERCs containing multiple W 's, there is no Hasse diagram. Graphical representation loses perspicuity, employing either a swirl of arcs indexed by ERC, or multiple diagrams.

And unlike ERC sets, pictures cannot be analyzed by calculation.

- The System
- The Objects of OT
- Optimality
- **Analysis**

The System

OT analysis takes place within a *system*:

- About a system, we can say things demonstrably true or false.

Optimality is meaningful within a system $\langle \text{GEN}_S, \text{CON}_S \rangle$.

An optimal form is better than all distinct candidates. **A!!!**

- No claim of optimality can be validated or refuted without knowledge of the entire candidate set.
- GEN_S must be specified.

The ERC polls the judgments of all constraints. **A!!!**

- No claim of betterness can be validated or refuted without knowledge of the entire constraint set.
- CON_S must be specified.

The Objects of OT

Given an OT system $S = \langle \text{GEN}_S, \text{CON}_S \rangle$,
we obtain three fundamental objects.

1. Language

2. Grammar

3. Typology

The Objects of OT

1. Language

The *candidates* from each candidate set that are **optimal** under some given hierarchy, a total order on CON_S .

Extensional: a set of linguistic structures and mappings.

2. Grammar

The set of all hierarchies that yield the same language.

Intensional: a description of the extensional language.

A grammar may be given by a set of **ERCs**

that exactly delimits the set of hierarchies yielding the lg.

Intensional: a description of the extensional language.

The Objects of OT

3. Typology of S

Extensional. The *languages* of S.

Intensional. The *grammars* of S.

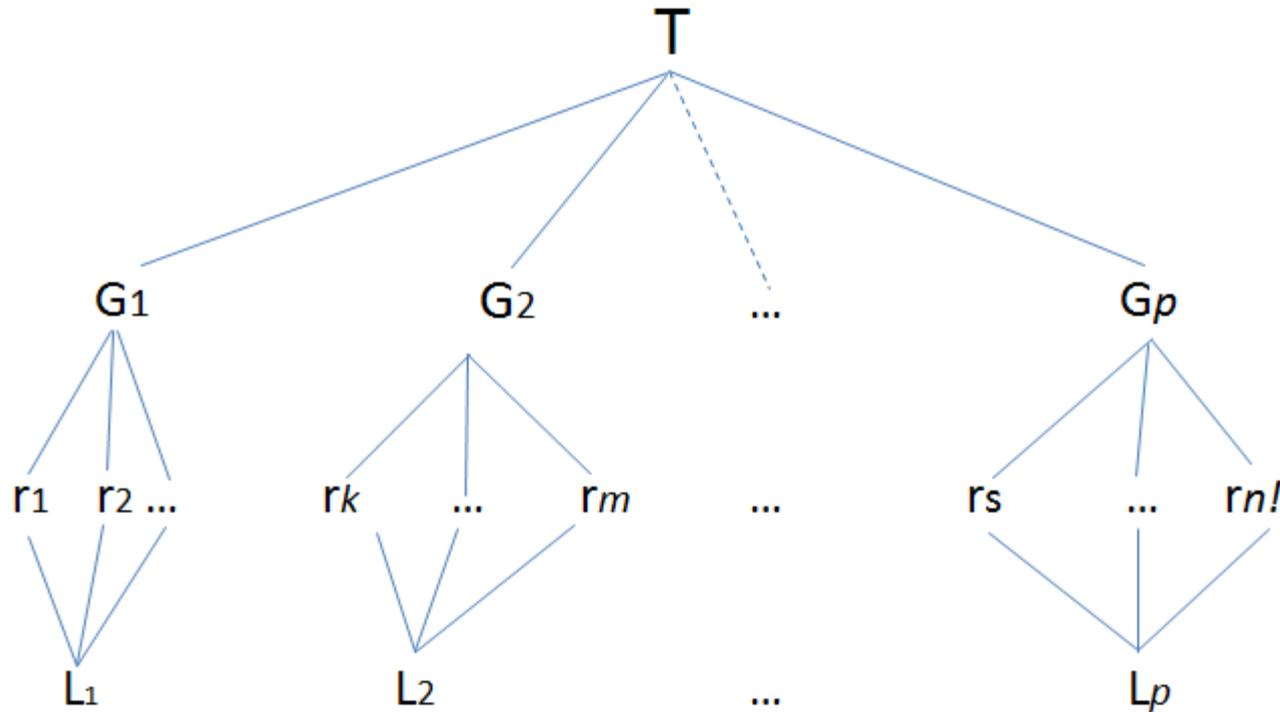
To understand S, we must understand the typology of S.

To understand the typology of S, we must understand its languages and their grammars.

Once we have $\langle \text{GEN}_S, \text{CON}_S \rangle$, the languages & grammars are fixed.

- We must figure out what they are: *analysis*.

The Objects of OT



A typology consists of grammars;
a grammar, of hierarchies that all deliver the same language.

OT analysis: Getting Grammars

1. *OT analysis* therefore aims to arrive at **the grammars** that the system *S* provides for the data under consideration.
2. We seek the grammar(s) delivering *optima* that accord with the data.
3. We want the grammar(s) in the form that grammars are defined: as a set of ERCs.
4. We are not primarily or directly engaged in ‘ranking’ the constraints.
5. We are ‘arguing optimality’.
 - The ERCs we obtain from optima delimit the set of rankings that constitutes the grammar.

OT analysis: from S to T

1. **After such knowledge**, what forgiveness?

Once you have fixed S, you have no further choices.

2. **One among many**. The analysis of even one language requires knowledge of the typology of S.

3. **The typology of S** contains the information about which languages and grammars admitted by the system S comport with (more-or-less comport with) the data under analysis.

4. **The analytical situation**. You have the data on one side and the assumptions of S on the other.

Data, meet Theory

1. To justify the claim that the data has *exactly this analysis* under S, you must show that *it has no others* under S.
2. We do not *observe* feet, features, tones, phrases, phases, nodes, labels, links, operators, ..., the substance of linguistic representation.
We posit them in S.
How many arrangements yield the same observables?
We may like one, but what does the theory say?
3. Only when we know the *typology* do obtain a complete, guess-less view of how S characterizes the data.

OT analysis: Getting Typologies

1. Having specified the system $S = \langle \text{GEN}_S, \text{CON}_S \rangle$, we have brought its typology into logical existence. But we do not know what it is.
2. To obtain it, we must assemble a collection of candidate sets (csets), in accordance with GEN_S , sufficient to distinguish all the grammars of T.
3. This sample of csets, if sufficient, is a *universal support* for T.
4. A *support* for a grammar is a collection of csets sufficient to yield the entire grammar.
5. A *universal support* is so termed because, when all choices of possible optima are made, one from each cset, we get all the grammars of T.

See Alber, DelBusso, and Prince (to appear) and Appendix for the universal supports of nGX.

OT analysis: Getting Typologies

1. **About Optima.** The individual csets must contain every candidate that is *possibly optimal* --- optimal under some ranking.

- The csets must be *optimum-complete*.
- Because to be optimal is to beat all.

2. **The Big Easy.** Finding a universal support, and establishing its universality, need not be trivial.

- But it is essential.

Without it, we are open to speculation and self-delusion.

- With it, we can begin to see where the theory leads.

What is OT?

The System. $S = \langle \text{GEN}_S, \text{CON}_S \rangle$.

The Objects. Language, grammar, typology.

Optimality. Better on a constraint,
Better on a hierarchy of constraints,
Best.

Analysis. $S \Rightarrow T \Rightarrow G$, an analysis if it exists under S .

Some Downloadable Resources

ERCs, Ranking, OT:

RCD – The Movie. Includes notes on OT fundamentals.

Brasoveanu & Prince. Includes introductory description of ERC logic.

Prince 2002. Develops and explores ERC logic.

Software based on the concepts discussed here:

OTWorkplace. Excel-based calculation of the objects of OT.

nGX – analysis & universal supports:

Alber, DelBusso, and Prince. To appear.

See References for links.

References

- Alber, B. and A. Prince. 2015, in prep. *Typologies*. Ms. U. Verona and Rutgers U.
- Alber, B., N. DelBusso, A. Prince. 2015 & To appear. From Intensional Properties to Universal Support. *Language: Phonological Analysis*. [ROA-1235](#).
- Brasoveanu, A. and A. Prince. 2005/11. Ranking and Necessity. *NLLT* 29:3-70. ROA-794.
- Hyde, B. 2012. Alignment Constraints. *NLLT* 30: 789-836.
- Merchant, N. and A. Prince. 2016. *The Mother of All Tableaux*. Ms. Eckerd College and Rutgers University.
- Prince, A., B. Tesar, and N. Merchant. 2007-2015. [OTWorkplace](#). [Updates & info](#).
- Prince, A. 2002. *Entailed Ranking Arguments*. [ROA-500](#).
- Prince, A. 2009. RCD – The Movie. [ROA-1057](#)
- Prince, A. 2013. Metrical Theory as a Portal on Theory. [YouTube](#).
- Prince, A. and P. Smolensky. 1993/2004. *Optimality Theory*. Blackwell. [ROA-537](#).
- Samek-Lodovici, V. and A. Prince. 1999. Optima. [ROA-363](#).
- Susskind, L. [The theoretical minimum](#).

Appendix: Universal Supports for nGX

Alber, DelBusso, and Prince (to appear) prove that the minimal universal supports for nGX are exactly the following:

- The 3s cset + a cset of any even length greater than 2s.
- Any cset with candidates of an odd length greater than 3s.

‘Minimal’ means that you can’t take away a cset and still have a universal support.

Minimal universal supports for nGX with the shortest candidates:

- The 3 and 4 syllable csets
- The 5 syllable cset

These are shown on the following slides. Only possible optima are included.

A Universal Support for nGX

nGX	output	opt	P-s	lamb	Troch	AFL	AFR
3s	-o-uX-		1	0	1	1	0
	-uX-o-		1	0	1	0	1
	-uX-X-		0	1	2	2	1
	-o-Xu-		1	1	0	1	0
	-Xu-o-		1	1	0	0	1
	-Xu-X-		0	2	1	2	1
	-X-uX-		0	1	2	1	2
	-X-Xu-		0	2	1	1	2
4s	-o-o-uX-		2	0	1	2	0
	-o-o-Xu-		2	1	0	2	0
	-uX-o-o-		2	0	1	0	2
	-uX-uX-		0	0	2	2	2
	-Xu-o-o-		2	1	0	0	2
	-Xu-Xu-		0	2	0	2	2

A Universal Support for nGX

nGX	output	opt	P-s	lamb	Troch	AFL	AFR
5s	-o-o-o-uX-		3	0	1	3	0
	-o-o-o-Xu-		3	1	0	3	0
	-o-uX-uX-		1	0	2	4	2
	-uX-o-o-o-		3	0	1	0	3
	-uX-uX-o-		1	0	2	2	4
	-uX-uX-X-		0	1	3	6	4
	-o-Xu-Xu-		1	2	0	4	2
	-Xu-o-o-o-		3	1	0	0	3
	-Xu-Xu-o-		1	2	0	2	4
	-Xu-Xu-X-		0	3	1	6	4
	-X-uX-uX-		0	1	3	4	6
	-X-Xu-Xu-		0	3	1	4	6



Du mußt
dein Leben
ändern.

Mural by
Jonathan
Horowitz,
Highland
Park, NJ.
[jonathanjacob
horowitz.com](http://jonathanjacobhorowitz.com)