OT or not OT – is that a question?

Julian Bradfield
University of Edinburgh

Why (might people) do OT?

- Wide ranging framework.
- Claims to universality.
- Simple basic mechanism.
- Sociology.

Why (might people) not do OT?

- Framework underspecified.
- ‘Universality’ isn’t.
- Complex basic mechanism.
- Sociology.

Framework

- + Solving ranked constraints to extract a ‘best’ member of a set of candidates is general enough to apply to most linguistic domains – or anything else.
- – It’s almost meaningless until you know what ‘constraints’ (Con) are, and how candidates are produced (Gen).

How is OT pinned down?

In theory: OT soon formalized (Eisner (1997), Frank and Satta (1998), et al.) in various mathematically nice ways.
Consequence: (a pinned and cut down version of) OT is equivalent to finite-state transducers, so basically the same as those re-writing rules used in the *practice* of generative phonology.

How is OT pinned down?

In theory: OT soon formalized (Eisner (1997), Frank and Satta (1998), et al.) in various mathematically nice ways.
Consequence: (a pinned and cut down version of) OT is equivalent to finite-state transducers, so basically the same as those re-writing rules used in the *practice* of generative phonology.

In practice: People write arbitrary stuff as constraints, whatever suits the problem at hand. As for GEN . . .

Simple Basic Mechanism

Recap the basic mechanism:

- GEN produces a set of candidate output strings from an input representation
- Check these strings against set of constraints (CON), linearly (partially, probabilistically . . .) ordered in priority.
- The answer is the string(s) whose first failed constraint is lowest.

Simple to describe, apparently simple to apply . . .

. . . is complex

Filling in the dots:

- GEN is a function taking each of the (∞-ly many) input representations to a (maybe ∞) set of candidates.

. . . is complex

Filling in the dots:

- Gen is a function taking each of the (∞-ly many) input representations to a (maybe ∞) set of candidates.
- Defining the way ranked constraints combine is non-trivial.

. . . is complex

Filling in the dots:

- GEN is a function taking each of the (∞-ly many) input representations to a (maybe ∞) set of candidates.
- Defining the way ranked constraints combine is non-trivial.
- Finding the answer(s) is a hard problem, even computationally.

. . . is complex

Filling in the dots:

- GEN is a function taking each of the (∞-ly many) input representations to a (maybe ∞) set of candidates.
- Defining the way ranked constraints combine is non-trivial.
- Finding the answer(s) is a hard problem, even computationally.

OT is hard in theory: reasonable formalizations of OT are NP-complete – and anyway, everybody knows that constraint solving is hard.

It's also hard to understand in practice . . .

How easy it is to be wrong:

Karttunen (2006)
“The insufficiency of pencil-and-paper linguistics”

Elenbaas (1999) and Kiparsky (2003) develop:

- OT theory of prosody, in ptic for stress patterns of Finnish.
- Target: trochaic stress, modified by ‘non-initial light syllable followed by heavy becomes extra-metrical’. E.g. ***ra***_kas(ta)_***jat***_ta_***re***_nako_
- Use nine constraints: only one ranking works.

How easy it is to be wrong:

Karttunen (2006)
“The insufficiency of pencil-and-paper linguistics”
Elenbaas (1999) and Kiparsky (2003) develop:

- OT theory of prosody, in ptic for stress patterns of Finnish.
- Target: trochaic stress, modified by ‘non-initial light syllable followed by heavy becomes extra-metrical’. E.g. ***ra**kas(ta)**jat**ta**re**nako*
- Use nine constraints: only one ranking works.
- Ka. shows: it doesn’t, on ****ka**las(te)**le**mi(nen)* and many other longer words.

What is a grammar, anyway?

Where do you stand between:

- A grammar is a *descriptively adequate* model of certain empirical data.
- A grammar is an *intensionally correct* account of how language actually works.

Back to basics

The first motivating example in Prince and Smolensky (1993) is syllabification in Imdlawn Tashlhiyt Berber (Dell and Elmedlaoui 1985). Why does this motivate OT?

(Note that discrete prosody is a good place to do OT, because the meaning of GEN is pretty obvious . . .)

ITB syllabification – overview

ITB has a simple CV(C) syllable structure – but any sound can be a 'vowel'.

How are words syllabified? "Simple".

The most sonorous sounds (vowel or consonant) form the nuclei. E.g.

txznakk^w $\longrightarrow$ txz.nakk^w

tftkt $\longrightarrow$ tf.tkt

'Most sonorous' is defined in the familiar way (low vowel, high vowel, liquid, nasal, vcd fric, vcl fric, vcd stop, vcl stop).

ITB syllabification – original account

Dell and Elmedlaoui (1985) *describe* it in terms of two constraints (see later).

They *do* it by an algorithm to do 'core syllabification'.

The algorithm refers explicitly to the levels of the sonority hierarchy:

Let T_i, for $i = 1..8$, be the set of segments at level i of the (descending) sonority hierarchy. T_i was given as a feature matrix. $T_1 = \{a\}$, $T_2 = \{i, u\}$, $T_3 = \{l, r\}$, etc.

The DEA

Input: an array $\#s_1 s_2 \ldots s_n$ of segments.

Output: the array with each segment tagged s^{C} or s^{V} if it's onset or nucleus, or s^{-} otherwise.

Algorithm:

```
tag every segment with $^{-}$
**for** $i = 1..8$ **do**
   **for** $j = 0..n-1$ **do**
      **if** $s_j^{-} s_{j+1}^{-}$ and $s_{j+1} \in T_i$ **then**
         tag as $s_j^{\mathrm{C}} s_{j+1}^{\mathrm{V}}$
do some patch-up for codas etc.
```

i.e. "find the most sonorous CV syllables (from the left), then the next most, and so on".

ITB syllabification – the OT account (1)

Prince and Smolensky (1993), chap. 2, adapted for the same notation:

There are two constraints in Con.

Ons: every non-initial syllable must have an onset (i.e. $s^{\mathsf{V}}_{j+1} \Rightarrow s^{\mathsf{C}}_{j}$ for $j > 0$)

HNuc: If x is more sonorous than y, x makes a better nucleus than y.

ITB syllabification – the OT account (1)

Prince and Smolensky (1993), chap. 2, adapted for the same notation:

There are two constraints in Con.

Ons: every non-initial syllable must have an onset (i.e. $s_{j+1}^{\mathrm{V}} \Rightarrow s_j^{\mathrm{C}}$ for $j > 0$)

Hnuc: If x is more sonorous than y, x makes a better nucleus than y.

Implemented as

A nucleus s^{V} at level i on the sonority scale scores i violations of Hnuc.

For ITB, Ons $\gg$ Hnuc.

Gen generates all values of s that are possible syllabifications. (I.e. every onset is followed by a nucleus.)

‘Why OT is better’

In (fair, I hope) précis:

- The algorithm is aiming to do ‘harmonic evaluation’, i.e. find the parse with the most harmonic (most sonorous) syllables.
- But it has this artifice of looping over the eight feature matrices describing the sonority hierarchy.
- And it can be seen a bunch of traditional re-write rules, with the harmonic evaluation hard-wired into the order of the rules.
- Whereas in OT, harmonic evaluation is the primitive, and is out front, and
- The harmony is given by the interactions of the simple local constraints.

ITB syllabification – the OT account (2)

In chap. 8, the OT account that *really* (or not) does the same as the algorithm: CON is:

ONS, PARSE, $*P/\square$, $*M/a$
$\gg *M/\square \gg *M/i \gg \ldots \gg *M/v \gg \ldots \gg *M/t$
$\gg -$COD, $*P/t \ldots \gg \ldots *P/a$

ITB syllabification – the OT account (2)

In chap. 8, the OT account that *really* (or not) does the same as the algorithm: Con is:

Ons, Parse, $*P/\Box$, $*M/a$
$\gg *M/\Box \gg *M/i \gg \ldots \gg *M/v \gg \ldots \gg *M/t$
$\gg -$Cod, $*P/t \ldots \gg \ldots *P/a$

And there are non-trivial definitions embedded in the way harmonic evaluation *really* works . . . to which correctness is very sensitive.

ITB syllabification – the OT account (2)

In chap. 8, the OT account that *really* (or not) does the same as the algorithm: Con is:

Ons, Parse, *P/□, *M/a
$\gg$ *M/□ $\gg$ *M/i $\gg \ldots \gg$ *M/v $\gg \ldots \gg$ *M/t
$\gg -$Cod, *P/t $\ldots \gg \ldots$ *P/a

And there are non-trivial definitions embedded in the way harmonic evaluation *really* works . . . to which correctness is very sensitive.

The sonority hierarchy is hard-coded again, once forwards and once backwards.

Lessons for a pragmatic phonologist?

- *Locally*, constraints may well be easier to comprehend than re-write rules. The *local* interaction of two constraints is also easy.
- *Globally*, writing down a correct OT grammar is, um, challenging.
- Constraints are not very compositional.

Do we need OT to make use of harmony?

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB . . .

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, apply (repeating each from left before moving on):

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB ...

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \rightarrow x^{\mathsf{C}} y^{\mathsf{V}} z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB ...

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \rightarrow x^{\mathsf{C}} y^{\mathsf{V}} z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'
2. $x^- y^- z^{\mathsf{C}} \rightarrow x^{\mathsf{C}} y^{\mathsf{V}} z^{\mathsf{C}}$ if $x \prec y$
   'a pre-consonantal sonority rise makes a syllable'

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB ...

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \dots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \rightarrow x^C y^V z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'
2. $x^- y^- z^C \rightarrow x^C y^V z^C$ if $x \prec y$
   'a pre-consonantal sonority rise makes a syllable'
3. $x^- y^- \rightarrow x^C y^V$ $(x \neq \#)$ 'fill up the rest with CV'

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB . . .

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \dots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \rightarrow x^{\mathrm{C}} y^{\mathrm{V}} z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'
2. $x^- y^- z^{\mathrm{C}} \rightarrow x^{\mathrm{C}} y^{\mathrm{V}} z^{\mathrm{C}}$ if $x \prec y$
   'a pre-consonantal sonority rise makes a syllable'
3. $x^- y^- \rightarrow x^{\mathrm{C}} y^{\mathrm{V}}$ $(x \neq \#)$ 'fill up the rest with CV'
4. $x^- \rightarrow x^{\mathrm{C}}$ 'anything left over is a consonant (onset initially, coda otherwise)'

Forward to the past

If you don't like doubly looped algorithms with hard-wired sonority scales, how about the following account of ITB . . .

Let $x \succ y$ mean x is more sonorous than y.

Given the input string $\#^- s_1^- s_2^- \dots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \rightarrow x^C y^V z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'
2. $x^- y^- z^C \rightarrow x^C y^V z^C$ if $x \prec y$
   'a pre-consonantal sonority rise makes a syllable'
3. $x^- y^- \rightarrow x^C y^V$ $(x \neq \#)$ 'fill up the rest with CV'
4. $x^- \rightarrow x^C$ 'anything left over is a consonant (onset initially, coda otherwise)'

E.g. txznakk^w $\xrightarrow{1}$ txz.nakk^w $\xrightarrow{2}$ t.xz.nakk^w $\xrightarrow{3}$ t.xz.na.kk^w $\xrightarrow{4}$ .txz.na.kk^w

Re-write rules with more notational freedom

How nice is the just given account?

– Three rules (plus codas) rather than two constraints (plus codas)

– With some notation

+ but no external baggage, and

+ no hard-wired sonority hierarchy, only *comparison*

+ that is strictly *local*

+ with deterministic generation of the answer.

+ Also can be done on-line with *bounded look-ahead* (never need to look beyond the next sonority peak, so no more than 7 segments ahead, usually fewer).

Exploring changes

Neither the DEA, nor PrS's OT version†, nor mine quite accords with reality. For example:

	DEA	PrS	here
bddl	*.bd.dl	.bd.dl	.bd.dl
raymmɣi	.ra.ymm.ɣi	? .ra.ymm.ɣi *.ray.mm.ɣi	.ra.ymm.ɣi
!itbdrin	*.i.tbd.rin	? *.i.tbd.rin .it.bd.rin	*.i.tbd.rin

†PrS's OT account is not the same as the (modified) DEA they present

Summary

You don't have to do OT to exploit harmony.

'Choosing the right notation is half the battle'.

Is OT always the right notation?

Maybe you sometimes get more insight from something simpler . . .

Dell, F. and M. Elmedlaoui (1985).
Syllabic consonants and syllabification in Imdlawn Tashlhiyt Berber.
J. African Languages and Linguistics 7, 105–130.

Eisner, J. (1997).
Efficient generation in primitive Optimality Theory.
In *Proc. 35th Annual Meeting of the ACL*, pp. 313–320.

Elenbaas, N. (1999).
A unified account of binary and ternary stress.
Ph. D. thesis, University of Utrecht.

Frank, R. and G. Satta (1998).
Optimality theory and the generative complexity of constraint violability.
Computational Linguistics 24(2), 307–316.

Karttunen, L. (2006).
The insufficiency of paper-and-pencil linguistics: the case of finnish prosody.

In M. Butt, M. Dalrymple, and T. H. King (Eds.), *In Intelligent Linguistic Architectures: Variations on themes by Ronald M. Kaplan*, pp. 287–300. CSLI Publications.

Kiparsky, P. (2003).
Finnish noun inflection.
In D. Nelson and S. Manninen (Eds.), *Generative Approaches to Finnic Linguistics*. CSLI.

Prince, A. and P. Smolensky (1993).
Optimality theory: Constraint interaction in generative grammar.
Technical Report 2, Rutgers University Center for Cognitive Science.