Formal and Computational Approaches to Phonology

Tuesday: Optimality Theory

Julian Bradfield and James Kirby

University of Edinburgh

Optimality Theory

was introduced by Prince and Smolensky in 1993, following some years of development of precursor theories (Harmonic Grammar). It rapidly became the dominant paradigm.

OT abandons the generative model (whether traditional or autosegmental) entirely, and replaces formal language theory by constraint solving.

OT in a nutshell

A grammar is a set of *constraints* which say how well an input string (typically in *SPE*-like representation) matches an output string. This may include constraints solely on the output string.

OT in a nutshell

A grammar is a set of *constraints* which say how well an input string (typically in *SPE*-like representation) matches an output string. This may include constraints solely on the output string.

The act of phonological processing compares, in parallel, all possible output strings against the input string, and selects the 'best' match, that is the one that satisfies the most important constraints.

OT in a nutshell

A grammar is a set of *constraints* which say how well an input string (typically in *SPE*-like representation) matches an output string. This may include constraints solely on the output string.

The act of phonological processing compares, in parallel, all possible output strings against the input string, and selects the 'best' match, that is the one that satisfies the most important constraints.

At this point, you should be very worried . . .

OT in a nutshell

A grammar is a set of *constraints* which say how well an input string (typically in *SPE*-like representation) matches an output string. This may include constraints solely on the output string.

The act of phonological processing compares, in parallel, all possible output strings against the input string, and selects the 'best' match, that is the one that satisfies the most important constraints.

At this point, you should be very worried . . .

The set of constraints is posited to be universal, and languages differ only in the priorities they assign to constraints.

OT by example

The mechanism of OT is agnostic about the form of input and output; it is applied to syntax, morphology, prosody and segmental phonology. We'll stick to (mostly segmental) phonology.

OT by example

The mechanism of OT is agnostic about the form of input and output; it is applied to syntax, morphology, prosody and segmental phonology. We'll stick to (mostly segmental) phonology.

In phonological OT, input strings and output strings are typically feature bundles as in *SPE*, abbreviated by the usual phonetic symbols. For example, the English word 'income' might appear as the input string /ɪnkʌm/.

OT by example

The mechanism of OT is agnostic about the form of input and output; it is applied to syntax, morphology, prosody and segmental phonology. We'll stick to (mostly segmental) phonology.

In phonological OT, input strings and output strings are typically feature bundles as in *SPE*, abbreviated by the usual phonetic symbols. For example, the English word 'income' might appear as the input string /ɪnkʌm/.

The candidate output strings are all logically possible sequences of English phonemes (or phones, depending on your point of view). How do we select the right one (/ɪŋkʌm/)?

OT by example

The mechanism of OT is agnostic about the form of input and output; it is applied to syntax, morphology, prosody and segmental phonology. We'll stick to (mostly segmental) phonology.

In phonological OT, input strings and output strings are typically feature bundles as in *SPE*, abbreviated by the usual phonetic symbols. For example, the English word 'income' might appear as the input string /ɪnkʌm/.

The candidate output strings are all logically possible sequences of English phonemes (or phones, depending on your point of view). How do we select the right one (/ɪŋkʌm/)?

OT constraints fall into two main classes: *faithfulness* and *markedness*. Faithfulness says: 'the output should be the same as the input'. Markedness says: 'but other considerations apply'.

Faithfulness

In general, there are faithfulness constraints that say:

- MaxIO: every segment in the input matches one in the output

Faithfulness

In general, there are faithfulness constraints that say:

- MaxIO: every segment in the input matches one in the output
- DepIO: every segment in the output matches one in the input

Faithfulness

In general, there are faithfulness constraints that say:

- MaxIO: every segment in the input matches one in the output
- DepIO: every segment in the output matches one in the input
- IdentIO(F) matching segments have the same value for feature F.

Faithfulness

In general, there are faithfulness constraints that say:

- MaxIO: every segment in the input matches one in the output
- DepIO: every segment in the output matches one in the input
- IdentIO(F) matching segments have the same value for feature F.

If these were all, the best output is 'obviously' /ɪnkʌm/.

Markedness

Simplifying somewhat, the constraint that effects nasal assimilation is:

- *HetNasStop: heterorganic nasal–stop clusters are prohibited. That is, in any C_1C_2 cluster in the output, where C_1 is [+nasal,−cont] and C_2 is [−nasal,−cont], C_1 and C_2 must agree on the place features [ant,cor].

Markedness

Simplifying somewhat, the constraint that effects nasal assimilation is:

- *HetNasStop: heterorganic nasal–stop clusters are prohibited. That is, in any C_1C_2 cluster in the output, where C_1 is [+nasal,−cont] and C_2 is [−nasal,−cont], C_1 and C_2 must agree on the place features [ant,cor].

If this constraint is less important than all the faithfulness constraints, it makes no difference: but if we rank it above the place faithfulness constraints:

*HetNasStop $\gg$ IdentIO(ant,cor)

then either /ɪntʌm/ or /ɪŋkʌm/ will be the best candidate. Now what?

Constraint proliferation

Now we have to explain why the nasal assimilates rather than the stop. Two most obvious possibilities . . .

1. We could break apart the faithfulness constraints, and say that place faithfulness is more important for non-nasals than for nasals:
   $\textsc{IdentIO}(\text{ant, cor} \mid -\text{nas}) \gg {}^*\textsc{HetNasStop} \gg \textsc{IdentIO}(\text{ant, cor} \mid +\text{nas})$

Constraint proliferation

Now we have to explain why the nasal assimilates rather than the stop. Two most obvious possibilities . . .

1. We could break apart the faithfulness constraints, and say that place faithfulness is more important for non-nasals than for nasals:
   IDENTIO(ant, cor | −nas) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | +nas)
2. Or maybe it's that faithfulness for onset consonants is more important than for coda consonants:
   IDENTIO(ant, cor | onset) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | coda)

Constraint proliferation

Now we have to explain why the nasal assimilates rather than the stop. Two most obvious possibilities . . .

1. We could break apart the faithfulness constraints, and say that place faithfulness is more important for non-nasals than for nasals:
   IDENTIO(ant, cor | −nas) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | +nas)
2. Or maybe it's that faithfulness for onset consonants is more important than for coda consonants:
   IDENTIO(ant, cor | onset) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | coda)

(2) sounds plausible: the onset position is more salient. But now we have constraints referring to 'onset' and 'coda'. How do we know the syllable structure of /ɪnkʌm/? By checking constraints about syllables! That gets done in parallel with everything else.

Constraint proliferation

Now we have to explain why the nasal assimilates rather than the stop. Two most obvious possibilities . . .

1. We could break apart the faithfulness constraints, and say that place faithfulness is more important for non-nasals than for nasals:
   IDENTIO(ant, cor | −nas) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | +nas)
2. Or maybe it's that faithfulness for onset consonants is more important than for coda consonants:
   IDENTIO(ant, cor | onset) ≫ *HETNASSTOP ≫
   IDENTIO(ant, cor | coda)

(2) sounds plausible: the onset position is more salient. But now we have constraints referring to 'onset' and 'coda'. How do we know the syllable structure of /ɪnkʌm/? By checking constraints about syllables! That gets done in parallel with everything else. What about 'ink', 'succinct'?

Compare with *SPE*

In traditional generative phonology, we would write (simplified) nasal assimilation

$$[+\text{nasal}] \rightarrow \begin{bmatrix} \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix} / \text{\_\_} \begin{bmatrix} -\text{cont} \\ -\text{nasal} \\ \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix}$$

Compare with *SPE*

In traditional generative phonology, we would write (simplified) nasal assimilation

$$[+\text{nasal}] \rightarrow \begin{bmatrix} \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix} / \text{\_\_} \begin{bmatrix} -\text{cont} \\ -\text{nasal} \\ \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix}$$

Class Discussion! Compare and contrast the phonological understanding the two approaches seem to give. From what we know of OT so far, how hard is it to compute? (*SPE* has a hundred or so rules, expanding to a few thousand once schemes are unpacked; an OT equivalent (never been done) would have thousands or tens of thousands of constraints.)

Compare with *SPE*

In traditional generative phonology, we would write (simplified) nasal assimilation

$$[+\text{nasal}] \rightarrow \begin{bmatrix} \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix} / \text{\_\_} \begin{bmatrix} -\text{cont} \\ -\text{nasal} \\ \alpha\text{cor} \\ \beta\text{ant} \end{bmatrix}$$

Class Discussion! Compare and contrast the phonological understanding the two approaches seem to give. From what we know of OT so far, how hard is it to compute? (*SPE* has a hundred or so rules, expanding to a few thousand once schemes are unpacked; an OT equivalent (never been done) would have thousands or tens of thousands of constraints.)

Exercise: One of the simplifications was that this story only really applies to /n/. English deals with /m/-stop clusters by epenthesis: e.g. empty (orig. emti), umpteen, bum(p)kin. Refine our constraints to do this.

Constraint interaction

The power of OT comes from the interaction of differently ranked constraints. The difficulty of OT . . .

Constraint interaction

The power of OT comes from the interaction of differently ranked constraints. The difficulty of OT . . .

Problems in analysing the complexity of OT:

1. that infinite number of candidate outputs!
2. what exactly are we allowed to say in constraints?
3. what exactly does being the best candidate mean?

Constraint interaction

The power of OT comes from the interaction of differently ranked constraints. The difficulty of OT . . .

Problems in analysing the complexity of OT:

1. that infinite number of candidate outputs!
2. what exactly are we allowed to say in constraints?
3. what exactly does being the best candidate mean?

(3) is addressed with formal precision by Prince and Smolensky; (1) and (2) are not.

(3) makes OT non-finite-state, in theory: evaluation counts not only *which* constraints are violated, but *how many times* they are violated.

Regular OT

T. Mark Ellison proposed regular OT:

- the candidates are a regular set (e.g. the set of all strings!)
- constraints are finite transducers (from input/output pairs to a unary number)

Then apply standard automata-theoretic techniques, plus some cleverness, to find the best candidate.

Regular OT

T. Mark Ellison proposed regular OT:

- the candidates are a regular set (e.g. the set of all strings!)
- constraints are finite transducers (from input/output pairs to a unary number)

Then apply standard automata-theoretic techniques, plus some cleverness, to find the best candidate.

Good news: time is linear in the size of the input string.

Regular OT

T. Mark Ellison proposed regular OT:

- the candidates are a regular set (e.g. the set of all strings!)
- constraints are finite transducers (from input/output pairs to a unary number)

Then apply standard automata-theoretic techniques, plus some cleverness, to find the best candidate.

Good news: time is linear in the size of the input string.

Bad news: it's linear(ish) in the size of the constraint automaton, which is a product (approximately) of the individual constraint automata.

Primitive OT

Jason Eisner proposed Primitive OT (OTP):

Primitive OT

Jason Eisner proposed Primitive OT (OTP):

- uses an autosegmental representation – but simpler than standard.

Primitive OT

Jason Eisner proposed Primitive OT (OTP):

- uses an autosegmental representation – but simpler than standard.
- Candidates are linearizations of the autosegmental input, freely decorated with output-only stuff.

Primitive OT

Jason Eisner proposed Primitive OT (OTP):

- uses an autosegmental representation – but simpler than standard.
- Candidates are linearizations of the autosegmental input, freely decorated with output-only stuff.
- Constraints mention only 'implication' and 'clash' of autosegmental elements: α overlaps with β in the linearization, or α must not overlap with β in the linearization.

This is more restricted than OT in practice, but can be argued to be adequate.

Computation is done similarly to Ellison, with some improvements.

Primitive OT

Jason Eisner proposed Primitive OT (OTP):

- uses an autosegmental representation – but simpler than standard.
- Candidates are linearizations of the autosegmental input, freely decorated with output-only stuff.
- Constraints mention only 'implication' and 'clash' of autosegmental elements: α overlaps with β in the linearization, or α must not overlap with β in the linearization.

This is more restricted than OT in practice, but can be argued to be adequate.

Computation is done similarly to Ellison, with some improvements.

Again, linear in size of input, but apparently exponential (NP-hard) in the size of the grammar.

‘The insufficiency of pencil and paper linguistics’

Lauri Karttunen gave a lovely demonstration of why even theoretical phonologists should implement, especially in OT . . .

Prosody – syllabification and stress – was the original OT application (also of autosegmental theories). Finnish stress is a popular example: traditionally,

- Trochaic stress: main stress on first syllable, secondary stresses on alternate syllables thereafter (except last). ***pää****tös****kon****ser****tis****ta*

‘The insufficiency of pencil and paper linguistics’

Lauri Karttunen gave a lovely demonstration of why even theoretical phonologists should implement, especially in OT . . .

Prosody – syllabification and stress – was the original OT application (also of autosegmental theories). Finnish stress is a popular example: traditionally,

- Trochaic stress: main stress on first syllable, secondary stresses on alternate syllables thereafter (except last). ***pää****tös****kon****ser****tis****ta***
- But secondary stress skips a light syllable followed by a non-final heavy one: E.g. ***ra****kas*ta***jat****ta****re****na****ko***

‘The insufficiency of pencil and paper linguistics’

Lauri Karttunen gave a lovely demonstration of why even theoretical phonologists should implement, especially in OT . . .

Prosody – syllabification and stress – was the original OT application (also of autosegmental theories). Finnish stress is a popular example: traditionally,

- Trochaic stress: main stress on first syllable, secondary stresses on alternate syllables thereafter (except last). ***pää****tös****kon****ser****tis****ta***
- But secondary stress skips a light syllable followed by a non-final heavy one: E.g. ***ra****kas****jat****tare****nako***
- (Other refinements ignored here.)

Elenbaas in her thesis developed an OT theory of prosody, with Finnish as the running example. Kiparsky took it up and developed it further.

Elenbaas in her thesis developed an OT theory of prosody, with Finnish as the running example. Kiparsky took it up and developed it further.

They require nine constraints, and they showed that only one ranking does the job for Finnish.

Elenbaas in her thesis developed an OT theory of prosody, with Finnish as the running example. Kiparsky took it up and developed it further.

They require nine constraints, and they showed that only one ranking does the job for Finnish.

Karttunen implemented it, and showed that that ranking does not in fact do the job: for example, it gives ***ka**las**te**l**e**minen* instead of ***ka**las**te**l**mi**nen*

His implementation is a finite state approximation to real OT (limiting the maximum number of constraint violations), with various techniques to prune candidate generation efficiently.

Is OT natural?

The first motivating example in Prince and Smolensky 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} $\rightarrow$ txz.nakk^{w}

tftkt $\rightarrow$ 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

D&E *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 $s_j^{\mathsf{C}} s_{j+1}^{\mathsf{V}}$
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)

PrS93, 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)

PrS93, 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.
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/□, *M/a
≫ *M/□ ≫ *M/i ≫ . . . ≫ *M/v ≫ . . . ≫ *M/t
≫ −Cod, *P/t . . . ≫ . . . *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/□, *M/a
≫ *M/□ ≫ *M/i ≫ . . . ≫ *M/v ≫ . . . ≫ *M/t
≫ −Cod, *P/t . . . ≫ . . . *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
≫ *M/□ ≫ *M/i ≫ . . . ≫ *M/v ≫ . . . ≫ *M/t
≫ −Cod, *P/t . . . ≫ . . . *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 $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, 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'
3. $x^- y^- \rightarrow x^{\mathsf{C}} y^{\mathsf{V}}$ $(x \neq \#)$ '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 $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, apply (repeating each from left before moving on):

1. $x^- y^- z^- \to x^{\mathsf{C}} y^{\mathsf{V}} z^-$ if $x \prec y \succ z$
   'a sonority peak must be a syllable peak'
2. $x^- y^- z^{\mathsf{C}} \to x^{\mathsf{C}} y^{\mathsf{V}} z^{\mathsf{C}}$ if $x \prec y$
   'a pre-consonantal sonority rise makes a syllable'
3. $x^- y^- \to x^{\mathsf{C}} y^{\mathsf{V}}$ $(x \neq \#)$ 'fill up the rest with CV'
4. $x^- \to x^{\mathsf{C}}$ '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 $\#^- s_1^- s_2^- \ldots s_n^- \#^-$, apply (repeating each from left before moving on):

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

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

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 ...