Knotty Ties

John McCarthy
5/3/2009

1. Introduction

Joe's email raised the question of how to handle ties in HS and its implementation. Joe described two possible ways of handling ties and asked whether there are empirical differences between them.

Joe's question led me in various directions. I report the results of my journeys here.

2. Types of ties

In a tie, two or more candidates from the same input are optimal under the same ranking. These candidates must have identical constraint violation vectors.

It is useful to distinguish two kinds of ties according to their causes:

> In *ties of neglect*, the analyst has failed to include a constraint (or constraints) that would select a single optimal candidate. The tie-breaking constraint exists or, at least, could be posited without causing any typological problems. Ties of neglect are uninteresting theoretically, but they are important for the implementation.
>
> In *ties of principle*, no tie-breaking constraint exists. Putative constraints that would break the tie are at best ad hoc and at worst cause typological problems. These ties are interesting theoretically. How important they are for the implementation depends on how common they are.

3. Ties in classic OT

Ties of neglect occur all the time when doing analysis in OT. They're a nuisance, but they don't present any sort of deep problem.

Most analysts would probably assume that ties of principle never occur in classic OT. This is an article of faith, since nothing in the theory ensures it.

4. Handling ties in the HS implementation

Joe's email raises the question of how the HS implementation should handle ties arising at intermediate steps of the derivation. There are various ways that the implementation could handle them:

(i) When a tie is discovered, report it to the user and refuse to continue, forcing the user to correct it by adding another constraint.

(ii) Arbitrarily choose one of the tied candidates as the unique winner and discard the rest. Inform the user that this has occurred.

(iii) Bifurcate the derivation. Each tied winner continues along its own derivational path. There is no competition between candidates derived from different winners. (This is what I've always assumed.)

(iv) All tied winners are input to Gen and the resulting candidates compete against one another. (This is analogous to the competition in systems of pre-Wolfian allomorphy. It was new to me in Joe's email.)

For ties of neglect, (i) would probably be the best strategy, since any of the other methods could lead to error (because the tied candidates aren't really tied). But this strategy won't work for ties of principle, and ties of principle are not uncommon in HS.

5. Ties of principle in HS

Ties of principle arise in HS whenever there are two or more loci that could undergo a process and no constraint decides which should change first. For example, in a language where /k/ palatalizes before /i/, which /k/ should palatalize first in /takikita/? No known constraint favors either of [tak^{j}ikita] and [takik^{j}ita] over the other. Making the user invent a constraint to resolve this tie is not a good implementational choice. And since the implementation, no matter how sophisticated it is, has no way of knowing whether it's dealing with a tie of neglect or a tie of principle, we can't use strategy (i), period.

Interestingly, strategies (ii), (iii), and (iv) will all work equally well for this example: all will end up with the correct output form, [tak^{j}ik^{j}ita]. In this case, the variation in the winner at an intermediate stage converges on a single final output, like so:[1]

(1)

takikita

tak^{j}ikita takik^{j}ita

tak^{j}ik^{j}ita

Under strategy (ii), picking either intermediate winner will end up with the correct final output. Under strategy (iii), the bifurcated derivations will end up with the same ultimate output. Under strategy (iv), the final output will be included in the candidate set from both winners.

Strategy (ii) would serve the user poorly in a related situation, however. Suppose an OCP-type constraint on [k^j] dominates the markedness constraint that favors velar palatalization. In the HS implementation, we would like readers to be aware that the grammar is giving equally harmonic forms as the ultimate output, since that is very likely a problem that would need to be corrected. Strategy (ii) warns users of the intermediate tie, but it doesn't impress users with the potential consequences of that intermediate tie.

[1] It's no accident that [tak^{j}ik^{j}ita] harmonically bounds [tak^{j}ikita] and [takik^{j}ita] in classic OT.

This leaves strategies (iii) and (iv) as potentially viable candidates for how to implement HS. These aren't just strategies; they're claims about how HS works as a theory. I'll show that each has a liability on the theoretical side.

6. Bifurcating derviations (strategy (iii))

I'll show that this strategy can produce an unlikely typological result: a language where stress on any syllable is just as harmonic as stress on any other syllable.

Assumptions:

Syllabification is done serially, one CV core syllable at a time.

Stress is assigned to syllables. Thus, stress can't be assigned until at least one syllable has been built.

Constraints:

LEX≈PR — every lexical word is a prosodic word

HEAD(PrWd) — every prosodic word contains a head foot.

PARSE-SEG — every segment belongs to some syllable.

Ranking:

LEX≈PR >> HEAD(PrWd) >> PARSE-SEG

Derivation(s):

Step 1

baba	LEX≈PR	HEAD(PrWd)	PARSE-SEG
a. → $[baba]_{PrWd}$		1	4
b. baba	1 W	L	4

Step 2 — divergence

$[baba]_{PrWd}$	LEX≈PR	HEAD(PrWd)	PARSE-SEG
a. → $[(ba)_{\sigma}ba]_{PrWd}$		1	2
b. → $[ba(ba)_{\sigma}]_{PrWd}$		1	2
c. $[baba]_{PrWd}$		1	4 W

Step 3 — if $[(\text{ba})_\sigma\text{ba}]_{\text{PrWd}}$ wins at step 2

	$[(\text{ba})_\sigma\text{ba}]_{\text{PrWd}}$	LEX≈PR	HEAD(PrWd)	PARSE-SEG
a. →	$[(\text{bá})_\sigma\text{ba}]_{\text{PrWd}}$			2
b.	$[(\text{ba})_\sigma\text{ba}]_{\text{PrWd}}$		1 W	2
c.	$[(\text{ba})_\sigma(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	L

Step 3 — if $[\text{ba}(\text{ba})_\sigma]_{\text{PrWd}}$ wins at step 2

	$[\text{ba}(\text{ba})_\sigma]_{\text{PrWd}}$	LEX≈PR	HEAD(PrWd)	PARSE-SEG
a. →	$[\text{ba}(\text{bá})_\sigma]_{\text{PrWd}}$			2
b.	$[\text{ba}(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	2
c.	$[(\text{ba})_\sigma(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	L

At this point, we've lost control of the stress pattern. We're on *n* divergent derivational paths, one for each of the *n* syllables in a word. Stress on any syllable is just as good as stress on any other syllable. As Joe noted in his email, OT-CC is equipped to handle competing derivations, but HS isn't. Of course, this sort of anything-goes variation isn't possible in classic OT.

Is the tie at step 2 one of neglect or principle? I think it's a tie of principle. The segment-counting syllable alignment constraints in Mester and Padgett (1994) could settle the tie, but I don't know of any good evidence for these constraints. (The evidence that Mester and Padgett cited can be analyzed in other ways.)

7. Remark on the previous example

The example of a divergent derivation in the previous section is trickier than it might seem. Schematically, it works like this:

> Process A (syllabification) can apply at several loci. This is the source of the divergent derivations.
>
> Process B (assignment of head foot) is fed by process A. Process B applies as soon as process A has applied once. This solidifies the divergence.
>
> Process A subsequently applies in other loci, but process B is somehow prohibited from applying again, so the derivations continue to diverge.

This situation is not easy to contrive. To get B to apply before all of the subsequent applications of A, the process-B-triggering markedness constraint MB has to dominate the process-A-triggering markedness constraint MA. But usually in HS, if A feeds B, then MA has to dominate MB (e.g., Arabic <ktub, uktub, ʔuktub> requires that *COMPLEX-ONSET dominate ONSET). That's because feeding relations usually arise when satisfying MA creates a violation of MB. My contrived example gets around this intrinsic limitation because syllabification is in an inherent (immutable) feeding relationship with foot parsing. Iterative syllabification seems to be a crucial element of this example.

8. Variants as competing inputs (strategy (iv))

In this view, the tie at step 2 above would not cause bifurcation of the derivation. Instead, there would be a single step 3 in which all of the candidates in the two steps 3 above compete with one another:

Step 3

$[(\text{ba})_\sigma\text{ba}]_{\text{PrWd}}$ $[\text{ba}(\text{ba})_\sigma]_{\text{PrWd}}$	LEX≈PR	HEAD(PrWd)	PARSE-SEG
a. → $[(\text{bá})_\sigma\text{ba}]_{\text{PrWd}}$			2
b. → $[\text{ba}(\text{bá})_\sigma]_{\text{PrWd}}$			2
c. $[(\text{ba})_\sigma\text{ba}]_{\text{PrWd}}$		1 W	2
c. $[\text{ba}(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	2
d. $[(\text{ba})_\sigma(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	L
d. $[(\text{ba})_\sigma(\text{ba})_\sigma]_{\text{PrWd}}$		1 W	L

Eventually, this tie will be resolved in the usual way that OT resolves ties — e.g., another syllable will be formed and adjoined to the foot, and TROCHEE vs. IAMB will decide. In other words, the variation will converge.

There is a way of getting this tie to produce a bad typological result, however. As we'll see, this strategy reintroduces the positional faithflness problem that Jesney (2009) proposed to solve with HS.

Input /bidɛ/

Step 1 parses it as a PrWd, as above.

Step 2

$[\text{bid}\varepsilon]_{\text{PrWd}}$	LEX≈PR	ID(ATR)/ˈσ	HEAD(PrWd)	PARSE-SEG	*[-ATR]
a. → $[(\text{bi})_\sigma\text{d}\varepsilon]_{\text{PrWd}}$				2	1
b. → $[\text{bi}(\text{d}\varepsilon)_\sigma]_{\text{PrWd}}$				2	1
c. $[\text{bid}\varepsilon]_{\text{PrWd}}$				4 W	1
d. $[\text{bide}]_{\text{PrWd}}$				4 W	L

Step 3 — like previous examples with *n* equiharmonic stress loci

[(bi)$_\sigma$dɛ]$_{PrWd}$ [bi(dɛ)$_\sigma$]$_{PrWd}$	LEX≈PR	ID(ATR)/ˈσ	HEAD(PrWd)	*[-ATR]	PARSE-SEG
a. → [(bí)$_\sigma$dɛ]$_{PrWd}$				1	2
b. → [bi(dɛ́)$_\sigma$]$_{PrWd}$				1	2
c. [(bi)$_\sigma$dɛ]$_{PrWd}$			1 W	1	2
d. [bi(dɛ)$_\sigma$]$_{PrWd}$			1 W	1	2
e. [(bi)$_\sigma$de]$_{PrWd}$			1 W	L	2
f. [bi(de)$_\sigma$]$_{PrWd}$			1 W	L	2
g. [(bi)$_\sigma$(dɛ)$_\sigma$]$_{PrWd}$			1 W	1	L

Step 4 — two inputs, but only one winner.

[(bí)$_\sigma$dɛ]$_{PrWd}$ [bi(dɛ́)$_\sigma$]$_{PrWd}$	LEX≈PR	ID(ATR)/ˈσ	HEAD(PrWd)	*[-ATR]	PARSE-SEG
a. → [(bí)$_\sigma$de]$_{PrWd}$					2
b. [(bí)$_\sigma$dɛ]$_{PrWd}$				1 W	2
c. [bi(dɛ́)$_\sigma$]$_{PrWd}$				1 W	2
d. [bi(dé)$_\sigma$]$_{PrWd}$		1 W			2
f. [(bí)$_\sigma$(dɛ)$_\sigma$]$_{PrWd}$				1 W	L
g. [(bi)$_\sigma$(dɛ́)$_\sigma$]$_{PrWd}$				1 W	L

At this point, I believe we're screwed. We've let ID(ATR)/ˈσ and *[-ATR] decide the placement of stress. If the input were instead [bɛdi], then stress would end up on the final syllable. This is exactly the unattested and implausible language that Jesney sought to eliminate.

From Jesney — bad typological prediction of positional faithfulness with parallel OT:

(6) a. *With the second input vowel [-ATR], output [+ATR] only and a **trochaic** parse*

	/bidɛ/	IDENT[±ATR]/ˈσ	*[-ATR]	IDENT[±ATR]	TROCH	IAMB
i.	(ˈ**bi**.dɛ)		*!			*
ii.	☞ (ˈ**bi**.de)			*		*
iii.	(bi.ˈ**dɛ**)		*!		*	
iv.	(bi.ˈ**de**)	*!		*	*	

b. *With the initial input vowel [-ATR], output [+ATR] only and an **iambic** parse*

	/bɛdi/	IDENT[±ATR]/ˈσ	*[-ATR]	IDENT[±ATR]	TROCH	IAMB
i.	(ˈ**bɛ**.di)		*!			*
ii.	(ˈ**be**.di)	*!		*		*
iii.	(bɛ.ˈ**di**)		*!		*	
iv.	☞ (be.ˈ**di**)			*	*	

From Jesney: good typological prediction with HS

(12) a. *With the second input vowel [-ATR], output [+ATR] only and a* ***<u>trochaic</u>*** *parse*

/bidɛ/	Ident[±ATR]/ˈσ	*[-ATR]	Ident[±ATR]	Troch	Iamb
☞ (ˈbi.dɛ)		*			*
(bi.ˈdɛ)		*		*!	
(ˈ<u>**bi**</u>.dɛ)	Ident[±ATR]/ˈσ	*[-ATR]	Ident[±ATR]	Troch	Iamb
(ˈ<u>**bi**</u>.dɛ)		*!			*
☞ (ˈ<u>**bi**</u>.de)			*		*
(<u>**bi**</u>.ˈdɛ)		*!		*	
(<u>**bi**</u>.ˈde)			*	*!	
(ˈ<u>**bi**</u>.de)	Ident[±ATR]/ˈσ	*[-ATR]	Ident[±ATR]	Troch	Iamb
☞ [(ˈ<u>**bi**</u>.de)]					*
(<u>**bi**</u>.ˈde)				*!	

Output: [(ˈbi.de)] **Winning derivation: <bidɛ, (ˈbi.dɛ), (ˈbi.de)>**

b. *With the first input vowel [-ATR], a [-ATR] output vowel and a* ***<u>trochaic</u>*** *parse*

/bɛdi/	Ident[±ATR]/ˈσ	*[-ATR]	Ident[±ATR]	Troch	Iamb
☞ (ˈbɛ.di)		*			*
(bɛ.ˈdi)		*		*!	
(ˈ<u>**bɛ**</u>.di)	Ident[±ATR]/ˈσ	*[-ATR]	Ident[±ATR]	Troch	Iamb
☞ [(ˈ<u>**bɛ**</u>.di)]		*			*
(ˈ<u>**be**</u>.di)	*!		*		*
(<u>**bɛ**</u>.ˈdi)		*		*!	
(<u>**be**</u>.ˈdi)	*!		*	*	

Output: [(ˈbɛ.di)] **Winning derivation: <bɛdi, (ˈbɛ.di)>**

9. Conclusions

How should HS the theory deal with ties? How should the implementation deal with ties?

10. References

Jesney, Karen (2009). Positional faithfulness, non-locality, and the Harmonic Serialism solution. *Proceedings of NELS 39.* [Available on Rutgers Optimality Archive, ROA-1018.]

Mester, Armin & Jaye Padgett (1994). Directional syllabification in Generalized Alignment. In Jason Merchant, Jaye Padgett & Rachel Walker (eds.) *Phonology at Santa Cruz 3*. Santa Cruz, CA: University of California at Santa Cruz. 79-85. [Available on Rutgers Optimality Archive, ROA-1.]

Ad hoc solution: All processes that meet this condition are directional. In determining where to apply process A, where no constraint favors a particular locus for A, GEN scans the input from left to right (say) until it finds a place to apply A. So syllabification will be directional in GEN if there are no constraints on direction of syllabification. Footing will be non-directional in GEN because there are constraints on direction of footing.

Another possible solution is to assume that GEN perseverates. The last optimal operation is the one that's tried first. On this view, the relation between Gen and Eval is dynamic: Gen offers Eval things in some (partial) order until something changes.