When more choice means less freedom: a note on candidate sets and typologies

Max Bane and Jason Riggle

University of Chicago – November 5, 2010

1 A Small Puzzle

This is perhaps the kind of thing that seems obvious in hindsight, but we initially thought that our typology generation algorithm had a bug when it produced the following results:

(1) Using the constraints ONSET, NOCODA, DEP-C, DEP-V, MAX to model the CV-syllable theory of Prince and Smolensky (1993:ch 6) over the 14 forms in the input set $\mathrm{CV}^3 = \bigcup_{i=1}^{3}\{\mathrm{C}, \mathrm{V}\}^i$ (i.e., all strings of C and V up to length three):

   i. the Optimality Theory (OT) typology has 12 languages,

   ii. the Harmonic Grammar (HG) typology has 23 languages (all 12 of the OT patterns plus 11 more patterns with cumulativity),

   iii. *but*, generating an HG typology using OT candidates yields 38 languages.

For almost all inputs in CV^3, the set of contenders (i.e., non-harmonically-bounded candidates) is the same under the ranked constraint interactions of Optimality Theory (OT; Prince and Smolensky 1993) and the weighted interactions of Harmonic Grammar (HG; Legendre et al. 1990). However, two of the input forms, /VVC/ and /CCC/, respectively have 1 and 2 extra contenders under weighting. These

are not crucial to the OT/HG distinction; omitting the tableaux for these inputs has no effect on the size of the HG typology because even without them the 11 languages that HG adds are still represented by combinations of candidates in the other 12 tableaux that can be co-optimal under weighting but cannot be co-optimal under ranking (see Pater et al. (2007) and Tesar (2007) for discussion of this effect).

Thus it came as a surprise that the HG-typology grew by 30% when we retained the tableaux for inputs /VVC/ and /CCC/ but omitted their extra HG candidates—that is, we used the set of OT contenders for these tableaux rather than the HG contenders, effectively removing three contenders from HG.

Since a language corresponds to choosing an optimal output form in each tableau, more candidates yield more ways to choose and thus more logically possible languages. Conversely, fewer candidates allow fewer choices and thus fewer possible languages. In general, the actual typology of HG or OT for a given set of constraints and tableaux will be considerably more nuanced than the set of logically possible candidate combinations, but all else equal, removing candidates removes possible combinations and thus should remove languages. In some cases this is indeed what happens, but in other cases, we find just the opposite. In these other cases all else is clearly not equal and this provides a small puzzle that illuminates the relationship between candidate sets and typology.

A puzzle: Why did the HG typology grow when the HG candidates were omitted?

This puzzle is not particular to the weighting mechanism of HG. We show below that, for the 5 constraints in (1), the pure OT typology can be (spuriously) inflated beyond 12 languages by selectively omitting candidates.

2 Less Choice, More Freedom

For a schematic example that shows how the omission of candidates can cause typological inflation that is couched completely within the ranked constraint interactions allowed by OT, consider the two tableaux in (2). Each candidate is annotated with the ranking conditions under which it is optimal and r, the number of rankings the conditions describe.

(2)

T_1	C1	C2	C3	C4
a.	*	*		*
b.	*		*	*
c.		*	*	

rankings
a wins if C3 $\gg$ {C1 C2 C4}, $r = 6$
b wins if C2 $\gg$ {C1 C3 C4}, $r = 6$
c wins if C1 or C4 $\gg$ {C2 C3}, $r = 12$

T_2	C1	C2	C3	C4
d.		*	*	*
e.	*	*	*	
f.	*			*

rankings
d wins if C1 $\gg$ {C2 C3 C4}, $r = 6$
e wins if C4 $\gg$ {C1 C2 C3}, $r = 6$
f wins if C2 or C3 $\gg$ {C1 C4}, $r = 12$

What is the typology of these two tableaux? Among the nine ways of pairing a candidate from T_1 with a candidate from T_2, five of the pairings are incompatible. The other four are compatible and the rankings that select them are given in (3).

(3)

ad : incompatible	ae : incompatible	af : C3 $\gg$ {C1 C2 C4}
bd : incompatible	be : incompatible	bf : C2 $\gg$ {C1 C3 C4}
cd : C1 $\gg$ {C2 C3 C4}	ce : C4 $\gg$ {C1 C2 C3}	cf : incompatible

The typology of {T_1, T_2} thus consists of four languages $\{af, bf, cd, ce\}$ and each language is selected by 6 of the 24 possible rankings of the constraints.

The reason that the typology grows with the removal of candidates becomes clearer upon considering the way that the ranking conditions associated with the

candidates in T_2 change when f is removed from the candidate set. This is illustrated in (4).

(4)

T_2'	C1	C2	C3	C4
d.		*	*	*
e.	*	*	*	

rankings
d wins if C1 $\gg$ C4, $r = 12$
e wins if C4 $\gg$ C1, $r = 12$

Whereas only f could be paired with candidates a and b before, once it is out of the picture the concomitant simplification in the ranking conditions on candidates d and e doubles the number of rankings—from 6 to 12—that each one is consistent with and thereby allows either one to be paired with both a and b. The upshot of this change is that, even though there are now only six ways to choose one candidate from each tableau (versus nine before), we are free to use any of the combinations because all six are compatible. These are listed in (5).

(5) ad : C3 $\gg$ C1 $\gg$ C4 & C3 $\gg$ C2, $r = 3$ ae : C3 $\gg$ C4 $\gg$ C1 & C3 $\gg$ C2, $r = 3$

bd : C2 $\gg$ C1 $\gg$ C4 & C2 $\gg$ C3, $r = 3$ be : C2 $\gg$ C4 $\gg$ C1 & C2 $\gg$ C3, $r = 3$

cd : C1 $\gg$ {C2, C3, C4}, $r = 6$ ce : C4 $\gg$ {C1, C2, C3}, $r = 6$

What has happened is that languages af and bf have each been split into two languages. Where af was generated by all 6 rankings with C_3 undominated, the new languages ad and ae distinguish $C_1 \gg C_4$ vs. $C_4 \gg C_1$ under the domination of C_3. The same happens to bf with C_2 undominated. This new distinction is 'stable' in the sense that it remains intact if the full version of T_2 is added to the mix (i.e., the typology of {T_1, T_2', T_2} has 6 languages not 4). Tableau T_2' introduces a crucial interaction between C1 and C4 that pervades the typology; whereas C1 and C4 are crucially ranked in 2 of the 4 languages in the {T_1, T_2} typology, they are crucially ranked in all 6 of the languages in the {T_1, T_2', T_2} typology.

3 Adding Tableaux vs. Adding Candidates

Usually, adding comparisons to the data set for a typology problem is done by adding new tableaux. This can introduce new distinctions that refine languages in the typology into new sub-languages but it cannot reduce the size of the typology. Adding a candidate to a tableau that is already in the data set, on the other hand, *can* reduce the size of the typology because doing so can increase the specificity of the ranking conditions on the other candidates and thus can result in there being fewer ways to combine candidates across tableaux. And, conversely, removing a candidate from one of the tableaux in a data set can introduce new distinctions in much the same way as adding tableaux; this is what we saw with T_2'. Removing candidates can also remove distinctions, but this change is not stable because the distinctions can be preserved or reintroduced by other tableaux in the data-set. The moral is that, once they are introduced, distinctions cannot be removed but if a distinction is missed it can always be added by another tableau.

These facts have significant ramifications. Some candidates can be harmlessly omitted because the distinctions that they encode can be gotten from other tableaux, but there are also some candidates whose omission introduces (possibly spurious) typological distinctions that cannot be undone. One way to think of the latter case is that removing a candidate from a tableau instantiates a kind of *covert constraint* which can add distinctions that expand the typology.

Covert constraints that are introduced by candidate omission act somewhat like undominated constraints (or hard constraints) in that they do not increase the number of possible rankings but can increase the number of languages generated

by exposing constraint interactions among 'runner-up' candidates that had been occluded by the triumph of the candidate that has been removed.

For example, the basic-CV typology mentioned in (1) has 12 languages rather than the 9 discussed in (Prince and Smolensky 1993:ch 6) due to action of a hard constraint against clusters that introduces a distinction between MAX and DEP (or PARSE and FILL in Prince & Smolensky's analysis) in terms of how coda clusters are avoided. This additional distinction splits 3 of the 9 languages to produce a typology of 12 languages (see Riggle 2004).

The analogy with hard constraints is not perfect. This can be illustrated for the CV syllable model and a ban on the null parse. (See Orgun and Sprouse (1999) and Wolf and McCarthy (2009) for contrasting views on constraints on null candidates.) For some inputs, a hard constraint will simply omit the null-candidate, but for others the null-candidate will be replaced by one or more novel candidates.

(6)

/CC/	DEPC	DEPV	MAX	*CODA	ONSET
a. CV.CV		**			
b. CVC		*		*	
c. ∅			**		

rankings
*CODA, MAX ≫ DEPV, $r = 40$
DEPV ≫ *CODA ≫ MAX, $r = 20$
DEPV ≫ MAX, $r = 60$

With the addition of a hard constraint against the null candidate, ∅ is replaced by [CV], which was formerly collectively harmonically bounded by [CVC] and ∅.

(7)

/CC/	DEPC	DEPV	MAX	*CODA	ONSET
a. CV.CV		**			
b. CVC		*		*	
c. CV		*	*		

rankings
*CODA, MAX ≫ DEPV, $r = 40$
DEPV, MAX ≫ *CODA, $r = 40$
DEPV, *CODA ≫ MAX, $r = 40$

In cases like this, discarding the null candidate and failing to replace it with a

runner-up can introduce spurious typological distinctions. For example, in the OT analysis with the 5-constraint model described in (1), CV^3 yields 14 tableaux with a total of 55 candidates, and these can be combined into 12 languages. Nine of the tableaux contain null-candidates. If these null-candidates are simply omitted (leaving 46 candidates in the 14 tableaux) the typology expands to 26 languages. On the other hand, if a hard constraint against the null candidate is implemented and the contenders are generated, then there are 58 candidates. This is 3 more than 55 because some of the nulls are replaced by multiple candidates. Furthermore, these 58 candidates can be combined to create 20 languages.

This illustrates that simply omitting a candidate is very different from using a (hard) constraint that bans it. This case also provides another example where fewer candidates yields a larger typology: 46 candidates in the 'censored' model yield 26 languages while 58 candidates in the 'hard constraint' model yield 20 languages.

4 A (Hypothetical) Cautionary Tale

Any new typological distinctions created by removing a candidate from a tableau are stable (i.e., they do not disappear if the full tableau is also included in the data set). This means that the 38 language pseudo-HG-typology obtained by building an HG typology using OT candidates for the inputs in CV^3 is not simply an artifact of having used such a tiny set of input forms. Considering more tableaux with their HG-only candidates omitted can add more spurious distinctions to the typology but cannot undo the distinctions already present. This is illustrated in Table 1 where the HG_{OT} column indicates the size of the pseudo-HG-typology that results

from constructing an HG typology using only OT contenders for input forms up to length n drawn from $\mathrm{CV}^n = \bigcup_{i=1}^{n} \{\mathrm{C}, \mathrm{V}\}^i$ for n ranging from 2 to 5.

inputs	OT_{OT}	HG_{HG}	HG_{OT}
CV^2	11	20	20
CV^3	12	23	38
CV^4	12	23	58
CV^5	12	23	109

Table 1: Size of OT and HG typologies over OT and HG contenders.

Note that for CV^2 the number of patterns in the HG_{HG} typology is the same as that of the HG_{OT} typology because the OT and HG contenders are identical for CV^2.

Regardless of whether constraints are ranked or weighted, omitting a single candidate from a single tableau can introduce spurious typological distinctions that cannot be undone by the addition of more tableaux. What this tells us is that it is important to include *all* the relevant candidates in tableaux lest we overestimate the diversity of the typology. This argues strongly for algorithmic generation of tableaux. Consider the pair of tableaux in (8).

(8)

/VC/	DEPC	DEPV	MAX	*CODA	ONSET	
a. CVC	*			*		removal ↦ 12 lgs.
b. V			*		*	removal ↦ 11 lgs.
c. ∅			**			removal ↦ 12 lgs.
d. CV		*	*			removal ↦ 12 lgs.
e. CV.CV	*	*				removal ↦ 13 lgs.
f. VC				*	*	removal ↦ 12 lgs.
g. V.CV		*			*	removal ↦ 13 lgs.
h. CV	*		*			removal ↦ 11 lgs.

/CCVC/	DEPC	DEPV	MAX	*CODA	ONSET	
i. CVC			*	*		removal ↦ 11 lgs.
j. CV.CV.CV		**				removal ↦ 14 lgs.
k. CV			**			removal ↦ 14 lgs.
l. CV.CVC		*		*		removal ↦ 11 lgs.

These tableaux contain complete sets of contenders under the assumptions that clusters are banned and insertion/deletion is the only way to change candidates (i.e., any changes must be penalized by explicitly included faithfulness constraints).

There are twelve ways to choose pairs of candidates from these tableaux that are co-optimal in OT. Each choice corresponds to one of the languages in the full OT typology. In (8), each candidate is annotated with the number of languages in the resulting OT typology (for these two tableaux) if that candidate were to be omitted. Omitting candidates a, c, d, or f keeps typology at 12 languages, the omission of b, h, i, or l reduces the size of the typology (though these distinctions may be recovered with other tableaux), and omitting e, g, j, or k introduces new distinctions that increase the size of the typology. Thus, a third of the candidates in this example can imply false distinctions between rankings if they are omitted.

5 Algorithms and Typologies

Algorithms for generating complete sets of contenders within the framework of OT are provided by Tesar (1995) for syllable structures, by Prince (2010) for metrical structures, and by Riggle (2004) for the general case of structures evaluated by grammars in which the constraints are *rational* (i.e., expressible as finite automata). Riggle (2009) gives an improved algorithm whose complexity is shown to be linear

in the length of input forms, modulo the number of resulting contenders (which can, of course, be quite large). Moreover, this method is expressed in the semiring parsing framework (see Kuich and Salomaa 1986, Goodman 1998, Mohri 2002), and thus extends transparently to cases with context-free constraints. Bane and Riggle (2009) discuss the extension of this algorithm to weighted constraints and algorithms for building typologies with both ranked and weighted constraints.

The candidates and typolgies discussed here were generated with the software package PYPHON (Bane and Riggle 2010) available at: `code.google.com/p/clml/`. PYPHON implements the contender generation algorithm of Riggle (2009) with the extension to HG from Bane and Riggle (2009) and provides a syntax for expressing constraints in terms of simple regular expressions which can be used to generate contenders and construct typologies for OT/HG.

PYPHON shares many features with other software packages for constraint-based grammars such as OTSOFT (Hayes et al. 2003), OT-HELP (Becker et al. 2007) and PRAAT (Boersma and Weenink 2007); it can generate typologies from tableaux of candidates and can help to avoid errors by automating basic calculations of ranking arguments. The feature that sets it apart is the automatic generation of complete sets of candidates, which seems especially important given the crucial role that candidate sets play in the analysis of predicted typologies.

6 Conclusions

We have shown here that arguments about merits of alternative (sets of) constraints and/or modes of constraint interaction based on typological predictions are highly

contingent on what candidates are considered. Specifically, with anything other than complete sets of contenders, the absence of the omitted candidates can imply spurious typological distinctions that are not justified by any constraint explicitly included in the models under consideration.

Given the inherent difficulty of hand-generating complete sets of contenders, the importance of methods for systematically generating candidates is paramount. The typological analyses of stress systems given in Gordon (2002) and Tesar (2004) are instructive and suggest (parts of) a 'gold standard' for analytical clarity. In these cases, it has been possible for other researchers to exactly replicate typologies of hundreds and thousands of predicted languages because clear definitions of the constraints *and* clear definitions of the candidate sets have been provided. This was facilitated by the relatively tiny alphabet of symbols needed to broadly represent stress systems.

More analyses that show this level of specificity and/or more general tools for generating candidates (such as, e.g., PYPHON) are clearly needed if typological predictions are to play a significant role in shaping phonological models. This need is more pressing than one might think, because, as we have shown here, omitting a single candidate in a single tableau can drastically alter a factorial typology.

References

Max Bane and Jason Riggle. pyphon 1.0. Software package, 2010. URL http://code.google.com/p/clml/.

Maximilian Bane and Jason Riggle. The typological consequences of weighted constraints. In *Proceedings of the annual meeting fo the Chicago Linguistics Society 45*, 2009.

Michael Becker, Joe Pater, and Christopher Potts. Ot-help 1.2 software package, university of massachusetts, amherst, 2007.

Paul Boersma and David Weenink. Praat. *Software Package*, 2007. URL http://www.praat.org/.

J. Goodman. *Parsing Inside-Out*. PhD thesis, Harvard University, Harvard, 1998.

Matthew Gordon. A factorial typology of quantity-insensitive stress. *Natural Language and Linguistic Theory*, 20(3):491–552, 2002.

Bruce Hayes, Bruce Tesar, and Kie Zuraw. OTSoft 2.3 software package: /www.linguistics.ucla.edu/people/hayes/otsoft/, 2003. URL http://www.linguistics.ucla.edu/people/hayes/otsoft/.

Werner Kuich and Arto Salomaa. *Semirings, automata, languages*. Springer-Verlag, 1986.

Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Harmonic grammar—a formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 388–395. Lawrence Erlbaum Associates, 1990.

Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

Cemil Orhan Orgun and Ronald Sprouse. From mparse to control: deriving ungrammaticality. *Phonology*, 16:191–224, 1999.

Joe Pater, Christopher Potts, and Rajesh Bhatt. Harmonic grammar with linear programming. Ms., October 2007.

Alan Prince. Counting parses. Ms. ROA 1097-0810, 2010.

Alan Prince and Paul Smolensky. Optimality Theory: Constraint Interaction in Generative Grammar. Technical Report RuCCS Technical Report 2, Center for Cognitive Science, Rutgers University, Piscataway, 1993.

Jason Riggle. Generating contenders. ROA 1044-0809, 2009.

Jason Riggle. *Generation, Recognition, and Learning in Finite State Optimality Theory*. PhD thesis, University of California, Los Angeles, 2004.

Bruce Tesar. A comparison of lexicographic and linear numeric optimization using violation difference ratios. Ms. ROA 939-1207, 2007.

Bruce Tesar. *Computational Optimality Theory*. PhD thesis, University of Colorado, 1995. URL citeseer.ist.psu.edu/article/tesar95computational.html.

Bruce Tesar. Using inconsistency detection to overcome structural ambiguity. *Linguistic Inquiry*, 35(2):219 – 253, 2004.

Matthew Wolf and John J. McCarthy. Less than zero: Correspondence and the null output. In Curt Rice, editor, *Modeling ungrammaticality in Optimality Theory*. London: Equinox., 2009.