Toward Computing Large Factorial Typologies in Your Lifetime

Jonathon E. Cihlar
University of Chicago
Paper only

Though factorial typologies serve as powerful tools to reveal the full set of predicted grammars of a constraint set, these typologies present a challenge inherent in their structure. That challenge is that the numbers of grammars predicted is of order n! (n equals the number of constraints), which is considered to be intractable. Highly successful approaches to this problem have come from research on OT learning. Namely, the Constraint Demotion (CD) algorithm (Tesar and Smolensky 2000) can produce factorial typologies in worst case time n^4. As more constraints are added to the universal inventory, furthering tractable approaches to the problem becomes critical.

The results presented here detail a new algorithm, Automatic Grammar Deduction (AGD), which computes a factorial typology from a finite set of candidates already assessed for violations with respect to finite constraint set. First, the algorithm produces a tableau inspired by the comparative tableaux of Prince (2002). To produce the initial tableau, AGD looks at each constraint column; for each unique, low-violating candidate found in a column, a W is assigned (winner). The remaining candidates are assigned Ls (non-winner) in that column. If more than one low-violator is found, those candidates are assigned Ts (tie); the remaining candidates receive Ls. The stripped-down mini-example illustrates this procedure for the Tagalog imperfective aspect (reduplication) of the actor voice (-um- infixation) for the verb 'to graduate'.

Mini-Example: *Tagalog Reduplication and Infixation*

/RED+-um-+gradwet/	MAX_{BR}	ALIGN-L	NO-CODA
a. gu.ma.grad.wet	$L_{(5)}$	$L_{(1)}$	$\mathbf{T_{(2)}}$
b. gru.ma.grad.wet	$\mathbf{W_{(4)}}$	$L_{(2)}$	$\mathbf{T_{(2)}}$
c. um.ga.grad.wet	$L_{(5)}$	$\mathbf{W_{(0)}}$	$L_{(3)}$

When a constraint yields a winner, the procedure concludes that all permutations with that constraint top-ranked yield the same output. The total number of grammars automatically deduced is w(n-1)!, where w is the number of winners discovered. The remaining grammars must be computed by brute force.

Two winners have been discovered in the mini-example. Thus, 2(3-1)! = 4 of the 6 logical grammars can be predicated automatically. The remaining 2 permutations must be computed manually. These results are summarized below.

Ranking	Output	Grammars	Deduction
MAX_{BR} » ALIGN-L, NO-CODA	b	2	Automatic
ALIGN-L » MAX_{BR}, NO-CODA	c	2	Automatic
NO-CODA » MAX_{BR} » ALIGN-L	c	1	Brute
NO-CODA » ALIGN-L » MAX_{BR}	b	1	Brute

For AGD, the best case scenario (though improbable) entails finding winners for all constraints; the complexity of this scenario is constant for all n. The worst case complexity (shown to be improbable) is the brute case scenario and grows at a factorial rate. Comparing AGD to learning algorithms solely on worst-case complexity is shown to be not fruitful because the basis to compute complexity is different. The average case is shown to vary with the number of winners produced; a lower bound is required for the procedure to be tractable. The results show that AGD is a promising way to approach the typology issue.

Selected References

Tesar, Bruce, and Paul Smolensky. (2000). Learnability in Optimality Theory. Cambridge: MIT Press.
Prince, Alan S. (2002). Arguing Optimality. In Coetzee, Andries, Angela Carpenter and Paul de Lacy (eds.), Papers in Optimality Theory II. Amherst, Massachusetts.