Ms., UMass Amherst

# Harmonic grammar with linear programming*

Joe Pater, Christopher Potts, and Rajesh Bhatt
UMass Amherst

October 8, 2006

**Abstract**

We show that Harmonic Grammars (HGs) translate into linear systems and are thus solvable using the simplex algorithm, an efficient, widely-deployed optimization algorithm that is guaranteed to deliver the optimal solution if there is one and to detect when no solution exists. Our associated software package *HaLP* provides a practical tool for studying even large and complex HGs. We provide an initial comparison of HG with standard Optimality Theory and with the enriched version allowing local constraint conjunction. This comparison shows that HG has considerable potential as a framework for the study of typology. The availability of HaLP can facilitate the future evaluation of that potential.

## 1 Introduction

Harmonic Grammar (HG; Legendre et al. 1990a,b; Smolensky and Legendre 2006) advances the hypotheses in (1) and (2).

(1) A grammar selects the most harmonic output for an input from a set of candidate outputs, where harmony is defined in terms of satisfaction of a set of *weighted* constraints.

[*] Our thanks to Ash Asudeh, Michael Becker, Tim Beechey, Maitine Bergonioux, Paul Boersma, John Colby, Edward Flemming, John Goldmsith, Karen Jesney, René Kager, John McCarthy, Andrew McKenzie, Kathryn Pruitt, Jason Riggle, and Paul Smolensky for very helpful discussion. Thanks also to the participants in Ling 730, UMass Amherst, Fall 2005, especially Kathryn Flack and Shigeto Kawahara for their initial insight about weighting conditions.

(2) The weighting of the constraint set is established through training of a connectionist network.

Our focus is (1), which is defined more precisely in (5) below. It has received little attention in linguistics when compared with (3), the closely related hypothesis about constraint interaction in Optimality Theory (OT; Prince and Smolensky 1993/2004).

(3) A grammar selects the most harmonic output for an input from a set of candidate outputs, where harmony is defined in terms of satisfaction of a set of *ranked* constraints.

The criteria for evaluating (3) are the usual ones from the generative tradition: one measures both (i) the extent to which theories adopting this hypothesis allow complex language-specific patterns to be reduced to the interaction of general principles, and (ii) the extent to which such theories succeed in generating all and only the linguistic patterns observed cross-linguistically. A large segment of the linguistic community has been evaluating (3) in these terms for well over a decade now.

Our primary interest is in applying the same criteria to the evaluation of hypothesis (1). This is of course a project that goes far beyond the scope of a single paper. But we can greatly facilitate future progress on the topic, by removing one of the main obstacles to fair evaluation: finding appropriate weightings (weightings that properly favor the grammatical forms) is difficult and time-consuming if one is forced to rely on hand calculations. Determining whether there are such weightings is even more challenging.

We show that it need not be this way, and this brings us to the central formal result of the present paper:

(4) Harmonic grammars translate into linear systems.

This is an important mathematical fact, and we review the translation process in detail in section 4.1. It has a significant practical corollary: linear systems, among the best understood in computer science, are solvable using the *simplex algorithm*, a widely-deployed, efficient optimization algorithm that has found applications in fields as far-ranging as troop movement and computer networks.[1]

Previous methods of establishing linguistic constraint weightings include HG's connectionism, the closely related technique of Maximum Entropy learning (Goldwater and Johnson, 2003; Jäger, to appear; Wilson, to appear; Hayes and Wilson, 2006), and the somewhat more distantly related Gradual Learning Algorithm (Boersma 1998; Hayes and

[1] Keller's (2006) Linear Optimality Theory might be seen as a precedent for our approach. However, his goals are different (modeling of gradient well-formedness judgments) as are his methods (solving of linear equations with judgment scores as given values).

Boersma 2001; see Pater to appear and Jesney et al. in prep for HG applications). The main advantage of our approach derives from the fundamental theorem of linear programming: the simplex algorithm is guaranteed to deliver, for any linear system, an optimal solution if there is one, else a verdict of 'infeasible' (if there are no solutions) or 'unbounded' (if, for any solution, we can always find a more optimal one). We formulate our reduction in such a way that unboundedness is not an issue, so we are left with two possible outcomes for a linguistic system that has been processed by the simplex algorithm: we are presented with an optimal weighting, or we are informed that the system is infeasible, i.e., has no consistent weightings. As we discuss below, the property of infeasibility detection is crucial to assessing the typological predictions of a constraint set. This property is not shared by connectionist and other probabilistic learners.

Our results are parallel to those produced by OT's Constraint Demotion Algorithm (Tesar and Smolensky, 1998), which finds a ranking if there is one that fits the data, else it indicates that none exists. It is therefore a useful tool not only for evaluating (1) directly but also for determining how it compares with (3).

Our translation (4) and subsequent application of the simplex algorithm form the basis of the *HaLP* software package (Harmonic Grammar with Linear Programming; Potts et al. 2006), which allows linguists to quickly and easily find a weighting suitable for their data or else learn that none exists. The input for this program is the same as is used in OT software programs like OT-Soft (Hayes et al., 2003) and Praat (Boersma and Weenink, 2006): a set of tableaux containing constraints, candidates, and violation patterns, as well as a list of the optimal candidates for each tableau. As such, it is straightforward to determine the differences between weighting and ranking, and these results can build on research already conducted in the OT framework.

The paper is structured as follows. In section 2, we compare OT and HG and introduce the method of HG analysis that underlies our linear-programming approach. This method, inspired by the discussion of HG in Prince 2002a and Legendre et al. 2006b, as well as by much work in OT (especially Prince 2002b), involves the logical analysis of weighting conditions. Section 3 defines the optimization problem we focus on. In section 4, we cover the translation to linear systems and our application of the simplex algorithm, thereby showing that weighting systems are highly tractable. Section 5 illustrates how the simplex-based approach can be used in the analysis of a linguistic pattern (Meccan Arabic voicing assimilation; McCarthy 2003a) and in the generation of typological predictions. This section also continues the comparison with OT, including a discussion of the relation of HG to OT enriched with local constraint conjunction (Smolensky, 2006). We close by explicitly connecting the fundamental theorem of linear programming with the questions linguists are likely to ask about their grammars.

## 2 Ranking and weighting

The central premise of Optimality Theory (OT: Prince and Smolensky 1993/2004) is that the grammar of a language consists of a set of ranked constraints. Ranking negotiates conflict between constraints; where two constraints differ in their assessment of output candidates, the higher ranked constraint determines which one is chosen. The premise that constraints are conflicting and ranked has two useful consequences. First, it allows relatively general constraints to produce intricate language-specific patterns. This results in attractive analyses of individual languages. Second, it allows a set of constraints to produce a relatively large number of distinct languages. This results in interesting predictions about language typology.

Ranking is an alternative to representing the relative strength of constraints in numeric terms, an approach taken most notably in OT's immediate predecessor HG. Here we assume the version of HG suggested by Prince and Smolensky (1993/2004:236): constraint violations are multiplied by positive valued weights,[2] and the optimal candidate has the lowest summed weighted violation score (see also Prince 2002a; Keller 2006). Though this is different from standard HG, we refer to it as such because it retains many of its core notions. We define the optimal candidate as in (5).

(5) Harmonic grammar

A candidate $\pi = \langle \iota, \omega \rangle$ is *optimal* for constraint set $C = \{c_1 \ldots c_n\}$, candidate set $\Pi$, and weighting $w = w_1 \ldots w_n$ iff, for every candidate in $\Pi$ of the form $\pi' = \langle \iota, \omega' \rangle$ where $\omega \neq \omega'$, it is true that

$$\sum_{j=1}^{n} w(c_j(\pi)) < \sum_{j=1}^{n} w(c_j(\pi'))$$

Each triple $(C, \Pi, w)$ determines a set $O \subseteq \Pi$ of optimal candidates. These are the predicted phonological forms of the language in question. This definition allows for only a single optimal output for an input. We leave it as an open question whether this definition should be altered to allow for tied optima, to account for free variation or other cases of optionality. (Section 4.1.1 contains a brief discussion of how to obtain the requisite linear systems for such a revised definition.) Other approaches to phonological variation compatible with HG that do not require ties are presented in Boersma (1998) and Hayes and Boersma (2001), as well as in Goldwater and Johnson (2003) and Jäger (to appear). One

[2] If the weights of individual constraints are allowed to have both negative and positive values, then the set of possible languages expands considerably, since the effect is similar to allowing the negation of those constraints (cf. Alderete 2001).

major advantage of working with (5) is that is that it fits with the empirical domain of most typological research, which abstracts away from variation.[3]

In the abstract example in (6), the fact that Constraint 1 has a higher weight than Constraint 2 results in Output$_1$ being chosen as optimal.

(6) A weighted constraint tableau

| *Weight* | 2 | 1 | $\sum$ |
|---:|:---:|:---:|:---:|
| Input | Constraint 1 | Constraint 2 | |
| ☞ Output$_1$ | | * | 1 |
| Output$_2$ | * | | 2 |

Here and throughout, we indicate the weight of a constraint in the top row. The rightmost column, labeled $\sum$, gives the weighted violation count for each candidate.

Ranking differs from weighting in that, for ranking, the degree of violation of lower ranked constraints is irrelevant, a property termed *strict domination* by Prince and Smolensky (1993/2004). If tableau (6) represented a ranking, Output$_1$ would be chosen because of its satisfaction of Constraint 1, regardless of how many violations it incurred on Constraint 2 or any other constraint ranked beneath Constraint 1. This is not true of systems with weighted constraints, as demonstrated in (7), which adds two further violations to Output$_1$ on Constraint 2. Here and in the rest of the paper, we give constraint violations numerically so as to transparently represent the mathematics in HG.

(7) Weighting ≠ Ranking

| *Weight* | 2 | 1 | $\sum$ |
|---:|:---:|:---:|:---:|
| Input | Constraint 1 | Constraint 2 | |
| Output$_1$ | 0 | 3 | 3 |
| ☞ Output$_2$ | 1 | 0 | 2 |

These further violations render Output$_2$ optimal in this harmonic grammar. If the same violation profile were evaluated in OT with Constraint 1 ranked above Constraint 2, Output 1 would continue to be optimal; the additive or *gang* effects of violation of the type in (7) are impossible in OT.

A theory of grammar that used weighted rather than ranked constraints would have the same attractive properties as OT: it would allow the reduction of complex patterns to the interaction of general constraints, and it would make testable non-trivial predictions about

[3] Our thanks to René Kager and John McCarthy for challenging questions relating to this point.

typology. Furthermore, such a theory would avoid the difficult question of how to appropriately transform a ranking into a numeric weighting for mathematical implementation (Prince and Smolensky 1993/2004:236; Prince 2002a; Legendre et al. 2006b). What is gained, then, by the move to ranking? Prince and Smolensky (1997:1604) claim that:

> In a variety of clear cases where there is a strength asymmetry between two conflicting constraints, no amount of success on the weaker constraint can compensate for failure on the stronger one.

However, it is more difficult than Prince and Smolensky imply to establish the necessity of strict domination. Their example involves the interaction of NoCoda and Parse: NoCoda bans codas from output representations, and Parse demands that input segments be parsed into output syllable structure. Prince and Smolensky (1997:1606) state that:

> No matter how many consonant clusters appear in an input, and no matter how many consonants appear in any cluster, [the grammar with NoCoda ≫ Parse] . . . will demand that they all be simplified by deletion (violating Parse as much as is required to eliminate the occasion for syllable codas), and [the grammar with NoCoda ≫ Parse] . . . will demand that they all be syllabified (violating NoCoda as much as is necessary). No amount of failure on the violated constraints is rejected as excessive, as long as failure serves the cause of obtaining success on the dominating constraint.

One problem is that in many, if not all, cases of NoCoda and Parse interaction this is just as true of any HG weighting as it is of any OT ranking (see Prince 2002a). For example, HG is equally insensitive to the number of consonant clusters in the input: if the weighting value of NoCoda is greater than Max (McCarthy and Prince's (1999) replacement for Parse), all potential codas will be deleted, and if the weighting value of Max is greater than NoCoda, they will all surface faithfully. One might imagine that the additive constraint interaction of HG could produce a system in which one coda is tolerated, but a second potential coda is deleted. That this is impossible can be demonstrated by determining the weighting conditions needed to produce each outcome. To make deletion of one of two potential codas optimal, as in (8a), NoCoda must have a weight greater than Max. To make preservation of a single potential coda optimal, as in (8b), Max must have a greater weight than NoCoda.

(8) a.

| /bantan/ | NoCoda | Max |
|---|---|---|
| ban.tan | 2 | 0 |
| ☞ ba.tan | 1 | 1 |

b.

| /bantan/ | NoCoda | Max |
|---|---|---|
| ☞ ba.tan | 1 | 0 |
| ba.ta | 0 | 1 |

The contradictory weighting conditions for (8a) and (8b) can be represented as in (9a) and (9b) respectively. Because they are contradictory, no weighting can simultaneously satisfy both, and no harmonic grammar can produce both outcomes in (8).

(9) a. $w$(NoCoda) > $w$(Max)

b. $w$(Max) > $w$(NoCoda)

Weighting conditions like those in (9) are HG's equivalent of OT's Elementary Ranking Conditions (Prince, 2002b,c): statements that must be true if the optimal candidate (a 'Winner' in Prince's terminology) is to be preferred over one of its competitors (a 'Loser'). Just as the Constraint Demotion Algorithm finds a ranking to satisfy the Elementary Ranking Conditions, so too does our application of the simplex find a weighting that satisfies the weighting conditions. And just as the Constraint Demotion Algorithm detects the inconsistency in this case, our application of the simplex returns 'infeasible'.

There are two morals to be gleaned from this simple example. The first is that HG, as we are using it, is an optimization system like OT. A grammar does not impose a single numerical cut-off on well-formedness, but instead chooses the best outcome for each input. A numerical cut-off might be used to rule out /bantan/ → [bantan] but not /batan/ → [batan] (with for example, $w$(NoCoda) $=$ 2, and the cut-off above 2 and below 4). However, when optimizing, the relative score of the candidate outputs for different inputs has no effect on whether each is chosen as optimal. The scores of outputs are also used in HG to model gradient acceptability judgments (e.g., Legendre et al. 1990b, 2006a; Keller 2006; see also Hayes and Wilson 2006). In this domain, relative score does affect relative acceptability, which predicts that two markedness violations (like two codas) could be judged as worse than one. Putting this prediction together with the limited effect of cumulativity in optimization (see further section 5) leads to the prediction that cumulativity will be more evident in acceptability judgments than in input–output mappings (e.g., phonological alternations). This prediction appears to be supported, at least in phonology (see Pater and Coetzee 2005:§4.4).

The second moral is that where violations trade off one-to-one between candidates, weighting and ranking are indistinguishable (Prince, 2002a). There is a one-to-one tradeoff in the hypothetical case in (8) because the decision about whether to parse each potential coda is an independent one. This may well be true of all cases involving just these two constraints.[4]

In (7), we presented an abstract example of a violation profile that can yield a difference between weighting and ranking. In that case, there is an asymmetric trade-off in constraint violations between candidates: three violations of Constraint 2 can be traded for one violation of Constraint 1. An asymmetric trade-off is a necessary, but not sufficient, condition for distinguishing OT from HG (cf. McCarthy and Prince 1993:76, 97; Prince and Smolensky 1993/2004:144, 148). To see this, imagine that (7) represented the whole set of constraints, inputs, and candidates. Both HG and OT would generate two languages: the one in which Candidate 1 is optimal, and the one in which Candidate 2 is.

To demonstrate a difference between HG and OT, we must consider the results for multiple inputs and show that there is a weighting that chooses a set of optima that is not chosen by any ranking.[5] For our abstract example, if we considered (6) and (7) to represent separate input-output mappings in one language , we would have exactly such a case. The weighting provided chooses Candidate 1 for (6), and Candidate 2 for (7), but no ranking chooses those two candidates at once.

Legendre et al. (2006b) use a real example of this type to argue that HG produces implausible typological predictions. The instantiation of "Constraint 2" is a gradient Alignment constraint that requires stress to appear on the rightmost syllable of the word and whose degree of violation depends on the distance between the main stress and the right edge. The conflicting constraint is Weight-to-Stress, which demands that heavy syllables be stressed.

(10) a. Align-Head-R: Assess one violation mark for every syllable intervening between the main stress and the right edge of the word.

b. Weight-to-Stress: Assess one violation mark for every unstressed heavy syllable.

If there is only one stress per word, then these constraints will come into conflict every time a word ends in a light syllable. Differences between HG and OT can arise any time more than one light syllable intervenes between the rightmost heavy syllable and the right edge. As the tableaux in (11) show, an HG weighting can place stress on the rightmost

[4] Suppose we define NoCoda so that a coda cluster violates it just once. If $2w(\text{Parse}) > w(\text{NoCoda})$, then a pair of consonants will surface faithfully. This weighting is consistent with $w(\text{NoCoda}) > w(\text{Parse})$, which leads to deletion of a single consonant. This unwelcome result can be avoided by defining NoCoda such that every segment in coda position incurs one violation.

[5] It is also possible to get a difference in the predicted results for a single input. As Prince (2002a) points out, a candidate that is collectively bounded in OT may emerge as optimal in HG, although simple harmonic bounding of one candidate by another is preserved between OT and HG. We know of one published typological prediction that depends on collective bounding (McCarthy 2002:204), but because of the pattern of violations, this result is preserved under weighting.

heavy syllable when it is $n$ syllables away from the right edge, and on the final syllable when the rightmost syllable is $n + 1$ syllables away.

(11) a.

| *Weight* | 3.5 | 1 | ∑ |
|---:|:---:|:---:|:---:|
| /bantanama/ | Weight-to-Stress | Align-Head-R | |
| ban.ta.na.má | 1 | 0 | 3.5 |
| ☞ bán.ta.na.ma | 0 | 3 | 3 |

b.

| *Weight* | 3.5 | 1 | ∑ |
|---:|:---:|:---:|:---:|
| /bantanavama/ | Weight-to-Stress | Align-Head-R | |
| ☞ ban.ta.na.va.má | 1 | 0 | 3.5 |
| bán.ta.na.va.ma | 0 | 4 | 4 |

In this example $n = 3$, and further weightings will create the same effect for $n = 4, 5, 6$, and so on. Stress systems are extremely well-studied typologically, and while many examples of three-syllable windows have been found (i.e., $n = 2$), as far as we know, not a single example of a stress window of any larger size exists.

Legendre et al. (2006b) take this sort of unattested additive effect to be fatal for HG as a theory of typology and therefore as a theory of grammar. However, just as in OT, the predictions of HG depend on the contents of the constraint set, and there are independent reasons to reject gradient alignment constraints like Align-Head-R. Gradient alignment produces other kinds of unattested stress systems, even in OT (Eisner, 1998; Kager, 2001; McCarthy, 2003b). McCarthy (2003b) proposes a theory of OT constraints that admits only categorical evaluation, that is, constraints that assign one and only one violation mark for each locus of violation. McCarthy also draws on proposals from Bakovic (2004) and Kager (2001) to provide reanalyses of the stress systems previously dealt with in terms of gradient alignment constraints.

If we eliminate gradient alignment constraints from OT, then we also eliminate the only published example of a difference in the typological predictions between OT and HG.[6] This underlines the fact that the HG is understudied as a theory of typology. Why the relative neglect? One possibility is that researchers may have assumed that it was clearly too powerful (following Prince and Smolensky 1993/2004:233). If so, this assumption bears reexamination, as we have argued here, and continue to argue in section

[6] Smolensky et al. (2006:455) briefly discuss an unpublished example from stress typology that led to the adoption of strict domination. Prince (2002a) constructs a hypothetical pattern generated by HG but not by OT: a single consonant that cannot be syllabified is deleted ($w$(Dep) > $w$(Max)), while a pair of consonants is saved by an intervening epenthetic vowel (2$w$(Max) > $w$(Dep)). Prince does not comment on the typological status of this pattern. It appears to be an open question whether single underlying consonants and consonant strings are always treated identically when they lead to a repair.

5. Another possibility is that it is simply easier to construct a phonological analysis of a language with ranked than with weighted constraints and to assess the typological predictions of ranked than weighted constraints. In what follows, we introduce an application of linear programming that overcomes these potential drawbacks to HG. We hope it will contribute to a more thorough assessment of the possibilities afforded for linguistic analysis by weighted constraint.

# 3 Formulating the problem

We concentrate on the optimization problem in (12).

(12) Let $\Pi$ be a candidate set, and let $O \subseteq \Pi$ be the set of grammatical forms. Let $C$ be a constraint set. Is there a weighting of the constraints in $C$ that defines all and only the forms in $O$ as optimal (definition (5))? If so, what does such a weighting look like?

There is an analogue of this problem in ranking systems (Prince and Smolensky, 1993/2004; Tesar and Smolensky, 1998; Prince, 2002c): given a set of grammatical forms and a set of constraints $C$, is there a ranking of the constraints in $C$ that determines all and only the grammatical forms to be optimal?

For weighting systems, it is typically fairly easy to answer the question for small systems like (13).

(13)

| Input | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|
| Winner | 4 | 0 | 4 |
| Loser | 0 | 2 | 0 |

Here, we can fairly easily reason to the conclusion that a weighting $\langle 1, 4.1, 1\rangle$ suffices. That is, the weights for $c_1$ and $c_3$ are both 1, and the weight for $c_2$ is 4.1. This gives $\langle I, W\rangle$ a total weighted violation count of 8 and $\langle I, L\rangle$ a total weighted violation count of 8.2. And it is easy to see furthermore that many other weightings work as well.

Similarly, it was fairly easy to see that (8) above lacks a consistent weighting, especially once it had been reduced to the statements in (9). But it quickly becomes challenging to reason this way. Even for only modestly-sized systems like (14) and (15), it is extremely difficult to reason to an answer to question (12).[7]

[7] We are grateful to Karen Jesney for the examples in (14) and (15). Example (14) has consistent weighting $3, 3, 5, 4, 1, 1$, and example (15) is inconsistent.

(14)

| Input$_1$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ |
|---|---|---|---|---|---|---|
| Winner$_1$ | 0 | 1 | 0 | 1 | 0 | 0 |
| Loser$_1$ | 1 | 0 | 1 | 0 | 0 | 0 |
| Input$_2$ | | | | | | |
| Winner$_2$ | 0 | 0 | 1 | 0 | 1 | 0 |
| Loser$_2$ | 0 | 1 | 0 | 1 | 0 | 0 |
| Input$_3$ | | | | | | |
| Winner$_3$ | 0 | 0 | 0 | 1 | 0 | 1 |
| Loser$_3$ | 0 | 0 | 1 | 0 | 1 | 0 |
| Input$_4$ | | | | | | |
| Winner$_4$ | 1 | 0 | 0 | 0 | 1 | 0 |
| Loser$_4$ | 0 | 0 | 0 | 1 | 0 | 1 |

(15)

| Input$_1$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ |
|---|---|---|---|---|---|
| Winner$_1$ | 0 | 1 | 0 | 1 | 0 |
| Loser$_1$ | 1 | 0 | 1 | 0 | 0 |
| Input$_2$ | | | | | |
| Winner$_2$ | 0 | 0 | 1 | 0 | 1 |
| Loser$_2$ | 0 | 1 | 0 | 1 | 0 |
| Input$_3$ | | | | | |
| Winner$_3$ | 1 | 0 | 0 | 1 | 0 |
| Loser$_3$ | 0 | 0 | 1 | 0 | 1 |
| Input$_4$ | | | | | |
| Winner$_4$ | 0 | 1 | 0 | 0 | 1 |
| Loser$_4$ | 1 | 0 | 0 | 1 | 0 |
| Input$_5$ | | | | | |
| Winner$_5$ | 1 | 0 | 1 | 0 | 0 |
| Loser$_5$ | 0 | 1 | 0 | 0 | 1 |

This is where the simplex method becomes so valuable. It can answer question (12) quickly for even very large and complex systems.

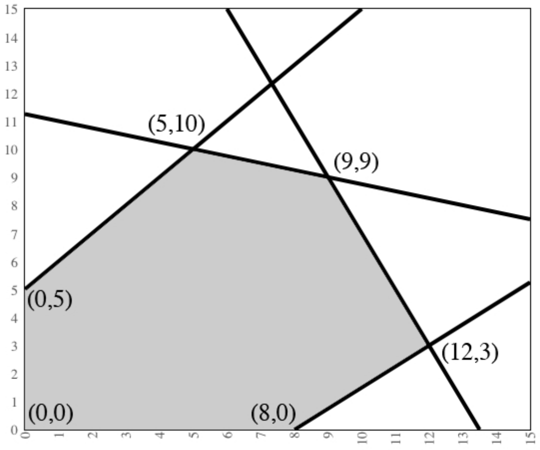
# 4 Solving linguistic systems with the simplex

The simplex algorithm is a cornerstone of the field of operations research, a branch of applied mathematics that focuses on optimization problems. The algorithm was developed in 1947 by the American mathematician George Dantzig. It was initially used by the military, for problems concerning troop movements, production costs, and the like. Though it has a nonpolynominal running time, it is known to be extremely efficient in practice, often besting its theoretically more efficient competitors (Cormen et al. 2001:820–821, Chvátal 1983:§4). Its inventor once wrote, “The tremendous power of the simplex method is a constant surprise to me” (Dantzig, 1981/1982).

When discussing the simplex algorithm, it is useful to start by considering two- and three-dimensional systems, where one can take a geometric perspective. It is then generally not too large an intellectual hurdle to think about working with it in higher dimensions. Consider, for instance, the small system in (16), which is adapted from Sedgewick 1992:609.

(16)

$$\begin{array}{rr} \text{minimize} & -1x - 1y \\ \text{subject to} & 1x - 1y \geqslant -5 \\ & -1x - 4y \geqslant -45 \\ & -2x - 1y \geqslant -27 \\ & -2x + 4y \geqslant -24 \\ & x, y \geqslant 0 \end{array}$$



The first line is the *objective function*. It is the function we are trying to optimize. Here, we are dealing with a minimization problem (the simplex also solves maximization problems). Our goal is to minimize the value of the objective function, but subject to the conditions that follow. The *feasible region*, the space of solutions that make all the linear inequalities true, is shaded. It contains an infinite number of solutions. But the major lesson of the simplex is that, when optimizing, we can ignore the vast majority of these solutions. We care only about solutions that lie along the outer edge of the feasible region. And, in fact, the only points of interest are the vertices, which we have labeled with their values. If we try out the labeled feasible solutions, we find that $(9, 9)$ is the one that does best by the objective function: it brings us to $-18$ for the objective function, a lower value that its nearest competitor $(5, 10)$, which yields an objective function value of $-15$.

The simplex algorithm solves linear systems by pivoting around the edge of the feasible region until it hits the optimal value, at which point it stops. (Section 4.3 describes the procedure in more detail.) As far as the algorithm is concerned, the feasible region can be of arbitrarily high finite dimension. (It is common for industrial applications of the simplex to involve thousands of variables.)

So if we can transform our linguistic constraint systems into linear systems of the sort in (16), then the simplex algorithm will find the optimal solution for us. Thus, the task of the next section is the formulation of this translation procedure. Once that is done, we need only run the simplex.

## 4.1 From linguistic systems to linear systems

In this section, we translate violation profiles of winner–loser pairs into linear systems. The discussion proceeds by way of example. We employ a system with three constraints, just one input structure $O$, and two candidate output structures, $W$ and $L$, where $W$ is the grammatical form that we wish to judge optimal according to the definition in (5).

(17) a. Constraints: $\{c_1, c_2, c_3\}$

b. Candidates: $\{\langle O, W\rangle, \langle O, L\rangle\}$

c. Winners: $\{\langle O, W\rangle\}$

d. Violation patterns:

| | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|
| $\langle I, W\rangle$ | 4 | 0 | 4 |
| $\langle I, L\rangle$ | 0 | 2 | 0 |

Our optimization task: find the minimal weighting $w$ that favors $\langle I, W\rangle$, if there is such a $w$; if there isn't, return 'infeasible'.

### 4.1.1 Equations in the linear system

We first convert the weighting conditions into linear inequalities. For each winner/loser pair, we want an inequality that guarantees that the winner has a lower weighted violation total than the loser:

(18) $4w_1 + 0w_2 + 4w_3 < 0w_1 + 2w_2 + 0w_3$

We then initially solve this inequality for 0, by converting (18) into the equivalent (19).

(19) $-4w_1 + 2w_2 - 4w_3 > 0$

This properly expresses what we want, namely, that the $W$ output is favored by the weighting over the $L$ output. But we are not quite ready yet. The problem is that this is a strict inequality, and it is impossible to optimize a strict inequality. To see this, consider what such a question would look like in its simplest form (Chvátal 1983:43):

(20) Minimize $x$ subject only to $x > 0$.

Of course, no matter what number we pick, there will always be another positive number that is closer to 0; there is no optimal solution. It is for this fundamental reason that optimization problems are given with nonstrict inequalities. So, if we are to solve by optimization, the best we can do is $\geqslant$. This means starting over and working as follows:

(21)
$$\begin{aligned} 4w_1 \ + \ 0w_2 \ + \ 4w_3 \ &\leqslant \ 0w_1 \ + \ 2w_2 \ + \ 0w_3 \ \Longrightarrow \\ -4w_1 \ + \ 2w_2 \ - \ 4w_3 \ &\geqslant \ 0 \end{aligned}$$

But this doesn't capture what we want. It allows weightings that assign the hopeful winner weighted violation totals that are equal to the weighted violation totals of some losing candidates. We want the winner to be strictly better, though.

For this reason, we need to solve for a special constant. It can be arbitrarily small, as long as it is above 0. It allows us to have regular inequalities without compromising our goal of having the winner win (not tie). So the final form our linear inequality is in fact (22).

(22) $-4w_1 + 2w_2 - 4w_3 \geqslant a$ where $a > 0$

Because our feasible regions are, when nonempty, open at the top, we do not render any otherwise feasible systems infeasible with this move. In our computational implementation, we set $a$ to 1 for convenience.

One repeats the above process for every winner–loser combination in one's tableaux. It is the heart of the translation procedure.

We use $\geqslant$ inequalities throughout, but this is not an intrinsic feature of our approach to finding weights. Linear systems can also contain $\leqslant$ statements and equalities. In general, we achieve this flexibility by converting everything into a standard format. For instance, in its current implementation, HaLP actually processes $\leqslant$ statements, so (22) is first multiplied through by $-1$ to obtain the equivalent (23).

(23) $4w_1 - 2w_2 + 4w_3 \leqslant -a$ where $a > 0$

This flexibility might prove useful if one wanted to allow individual inputs to have multiple optimal output forms, as we discussed briefly below (5). Suppose we want two distinct competing forms $A$ and $B$ to emerge as winners together. Let $v_1^A \ldots v_n^A$ and $v_1^B \ldots v_n^B$ be their respective violation profiles. Then we add condition (24) to ensure that they end up with the same weighted violation counts for any feasible weighting $w$, and we convert it into a pair of inequalities, as indicated.

(24) $wv_1^A + \ldots + wv_n^A = wv_1^B \ldots + wv_n^B \Longrightarrow$

$$\begin{tikzcd}[column sep=tiny]
 & w(v_1^A - v_1^B) + \ldots + w(v_n^A - v_n^B) = 0 \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
w(v_1^A - v_1^B) + \ldots + w(v_n^A - v_n^B) \leqslant 0 & & w(v_1^A - v_1^B) + \ldots + w(v_n^A - v_n^B) \geqslant 0 \Longrightarrow \\
 & & -w(v_1^A - v_1^B) - \ldots - w(v_n^A - v_n^B) \leqslant 0
\end{tikzcd}$$

Then we add conditions ensuring that each of these winners bests all competitors, as described above.

### 4.1.2 Blocking zero weights

The next substantive question we address is whether to allow 0 weights. A weighting of 0 is equivalent to canceling out violation marks. To prevent such cancellation, we impose additional conditions, over and above those given to us directly by the weighting conditions: for each constraint $c_i$, we add the inequality $w_i \geqslant 1$. Thus, we continue building the system in (22) as follows:

(25)
$$\begin{array}{rcrcrcl} -4w_1 & + & 2w_2 & - & 4w_3 & \geqslant & a \\ 1w_1 & & & & & \geqslant & b \\ & & 1w_2 & & & \geqslant & b \\ & & & & 1w_3 & \geqslant & b \\ & & & & a, b & > & 0 \end{array}$$

As before, we just need to solve for some positive constant $b$, which can be the same as $a$ or distinct from it. Once again, because our nonempty feasible regions are open at the top, excluding this subregion does not yield spurious verdicts of infeasibility.

Linear systems come with *nonnegativity conditions* on all variables, as in (26).

(26) all $w_i \geqslant 0$

See, for example, the final inequality in (16). We need not impose these directly on our basic weights, since each of them must be above the designated special positive constant $b$. But we include such conditions anyway because, as we discuss in section 4.3.1, the simplex solver introduces new *slack* weights as it runs, and each of these must be constrained as in (26).

It is worth exploring briefly what happens if we remove the extra non-0 restrictions. In such systems, some constraint violations can be canceled out when weighted, via multiplication by 0. This cancellation will occur when a given constraint is inactive for the data set in question, i.e., when it is not required in order to achieve the intended result. For example, our current reduction process returns $\langle 1, 3, 1\rangle$ as the minimal solution for the small system in (27) (assuming that we set the righthand sides of all the equations to 1).

(27)

| Input | $c_1$ | $c_2$ | $c_3$ |
|---:|:---:|:---:|:---:|
| Winner | 1 | 0 | 1 |
| Loser | 0 | 1 | 0 |

However, if we do not ensure that all weights are at or above 1, then the minimal solution for this is $\langle 0, 1, 0\rangle$. The winner's violations disappear, and it is revealed that only $c_2$ really matters in deciding things in favor of the Winner candidate. This kind of insight into the relationship between the intended optima and the constraint set might might be useful for finding the smallest constraint set that can solve the problem.

On the other hand, setting 1 as the minimum value for all constraints has useful consequences for understanding the nature of weighted constraint interaction. With this restriction, additive interactions between constraints are often maximized, as in the analysis of Meccan Arabic in section 5. This can yield insights into the differences between HG and OT.

### 4.1.3 The objective function

Our solution to (13) above probably made it apparent that there are infinitely many other feasible solutions: $\langle 1, 4.1, 1\rangle$ is the solution we used, but of course $\langle 2, 5, 2\rangle$ is also feasible, as is $\langle 200, 387400, 401\rangle$. There is no maximum weighting; if we think geometrically, we can see that the feasible region is "open at the top". This more or less demands that we define our systems as minimization tasks, by specifying that the goal is to minimize the objective function:

(28) minimize $1w_1 + 1w_2 + 1w_3$

Of course, there is a sense in which $\langle 1, 4.1, 1\rangle$ is not our minimal solution to (13) either. $\langle 1, 4.001, 1\rangle$ would also do the job, for example. The simplex itself takes us part of the way toward a solution to this. In section 4.1.1, we pointed out that all the inequalities are non-strict inequalities. These furnish a well-defined minimal value. For us, that minimal value is given by a constant (*a* and *b* on the righthand sides in (25)). An absolutely optimal weighting would be one in which all the inequalities were equal to these constants. Typically, we do not attain this, but it indicates that our minimization problem is well defined.

Given that all the coefficients are 1, we can define a weighting as minimal iff the sum of its weights is at least as low as the sum of the weights of any other feasible solution, as in (5). Translated back into the linguistic systems, this means that the sum of the weights will be minimal, subject to the restrictions that every loser's score will be at least one greater than the winner, and every weight will be at least 1.

### 4.1.4 The final form of the system

We can now present the complete system in its final form:

(29)
$$
\begin{array}{rrcrcrcl}
\text{minimize} & 1w_1 & + & 1w_2 & + & 1w_3 & & \\
\\
\text{subject to} & -4w_1 & + & 2w_2 & + & -4w_3 & \geqslant & a \\
 & 1w_1 & & & & & \geqslant & b \\
 & & & 1w_2 & & & \geqslant & b \\
 & & & & & 1w_3 & \geqslant & b \\
 & & & & & a,b & > & 0 \\
 & & & & & \text{all } w_i & \geqslant & 0
\end{array}
$$

This system is a suitable input for the simplex algorithm. In some implementations, it is necessary to change the system slightly, so that all the linear inequalities are in $\leqslant$ form. As we noted in section 4.1.1, one obtains such (equivalent) conditions from the above simply by flipping the inequality sign around and multiplying all the coefficients as well as the righthand side constant by $-1$. Similarly, some implementations demand that the problem be a maximization problem. To ready the system for such an algorithm, we multiple all the coefficients in the objective function by $-1$. Cormen et al. (2001:§29) includes an extended discussion of these and other conversions.

Before proceeding, we should offer a word of warning about terminology. In operations research, the inequalities are called ‘constraints’, since they constrain the values that the variables can take. Throughout this paper, we are careful to use ‘constraint’ in its linguistic sense. We consistently refer to the inequalities obtained via the methods of section 4.1.1 as ‘weighting conditions’.

## 4.2 The usual situation

We now return briefly to the geometric perspective to summarize the above discussion. The linguistic system is given in (30), its reduction is given in (31), and its geometric interpretation is depicted in (32). (To provide a concrete solution and a visualization, we’ve set all the righthand sides to 1 — recall that this just stands in for any positive value.)

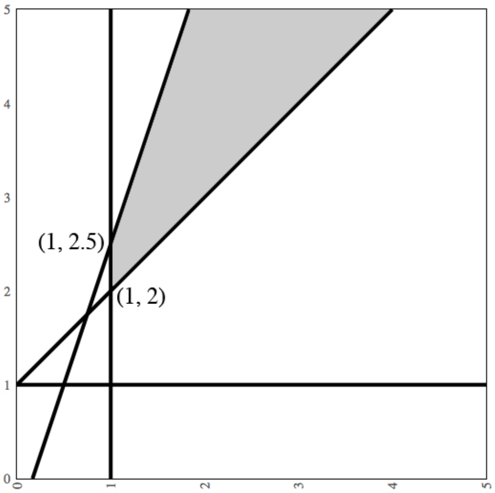
(30)

| Input$_1$ | $c_1$ | $c_2$ |
|---|---|---|
| Winner$_1$ | 0 | 2 |
| Loser$_1$ | 6 | 0 |
| Input$_2$ | | |
| Winner$_2$ | 1 | 0 |
| Loser$_2$ | 0 | 1 |

(31)

$$
\begin{array}{rrcrcl}
\text{minimize} & 1w_1 & + & 1w_2 & & \\
\text{subject to} & 6w_1 & - & 2w_2 & \geqslant & 1 \\
 & -1w_1 & + & 1w_2 & \geqslant & 1 \\
 & 1w_1 & & & \geqslant & 1 \\
 & & & 1w_2 & \geqslant & 1 \\
 & & & \text{all } w_i & \geqslant & 0
\end{array}
$$

(32)



There are two salient features of systems like this when it comes to applying the simplex algorithm.

First, the solution that sets both the weights to 0 is not feasible — it is outside of the feasible (shaded) region. The simplex algorithm always starts from an all-0s initial solution, so this will force us to construct a special *auxiliary program* to move from the all-0s solution to a feasible one (section 4.3.3).

Second, the feasible region is unbounded: open at the top. This means that there is no well-defined sense of maximal when it come to feasible solutions; if we ask what the maximal solution is, the simplex will reply with ‘unbounded’. As discussed above, we solve this problem by defining these as minimization problems. This is arguably natural anyway, as it means that we get a look at the minimal weighting contrasts that must exist between our constraints to obtain the result we are after.

If a system has no solution, the simplex detects this. We discuss how this happens in the next section, in which we briefly review the workings of the simplex algorithm, which takes over from the point we reach in (31).

## 4.3 The (two-phase) simplex algorithm

We will not review the simplex algorithm itself in detail here, for two reasons. First, most textbooks on optimization and constraint logics contain descriptions of how the algorithm works. We recommend Chvátal 1983 and Cormen et al. 2001:§29 in particular. Second, our approach is not intrinsically tied to the simplex algorithm, but rather only to linear optimization in general. The simplex is the solver we use in Potts et al. 2006, but others might actually perform better for this class of problems. Murty 1995 is an excellent resource for those wishing to explore alternatives.

But the simplex brings to the fore a few important features of linguistic theories based in constraint-weighting, so it is worth going over the algorithm's basic logic.

### 4.3.1 Slack forms

Systems like (31) are not quite the forms that the simplex processes. There is one additional algebraic conversion required: the introduction of *slack weights*, which effectively measure the amount by which we can adjust the value of a given weight without violating any of the constraints. We illustrate the process of slack conversion in (33).

(33)
$$\begin{array}{rcl} & & -4w_1 + 1w_2 \geqslant -8 \Longrightarrow \\ w_3 & = & 8 - 4w_1 + 1w_2 \end{array}$$

Slack conversion introduces a new weight, one that does not occur anywhere else in the system. This is the slack weight. We solve for it. Here is a complete slack conversion for the small system from Cormen et al. (2001:773):

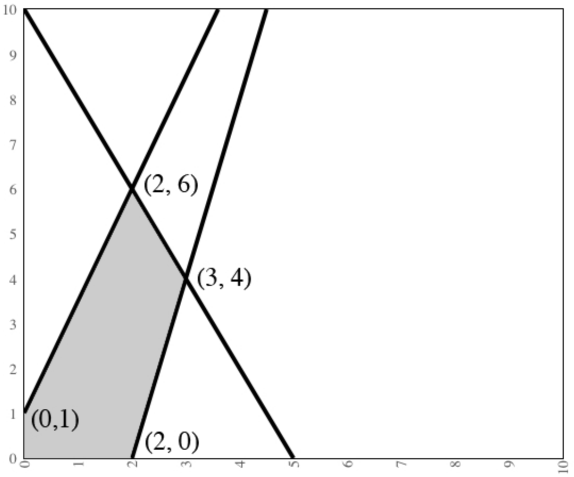
(34)
$$\begin{array}{rl} \text{minimize} & -1w_1 - 1w_2 \\ \text{subject to} & -4w_1 + 1w_2 \geqslant -8 \\ & -2w_1 - 1w_2 \geqslant -10 \\ & 5w_1 - 2w_2 \geqslant -2 \\ & \text{all } w_i \geqslant 0 \end{array} \Longrightarrow \begin{array}{rl} \text{minimize} & -1w_1 - 1w_2 \\ \text{subject to} & w_3 = 8 - 4w_1 + 1w_2 \\ & w_4 = 10 - 2w_1 - 1w_2 \\ & w_5 = 2 + 5w_1 - 2w_2 \\ & \text{all } w_i \geqslant 0 \end{array}$$

This new form of the problem affords us a new way to think about finding viable weightings. Consider the slack form in (34), and suppose we set the initial values of $w_1$ and $w_2$ to 0. If we do this, then the slack variables $w_3$, $w_4$, and $w_5$ take on the values 8, 10, and 2, respectively. The objective value is 0.

But we can do better than this. All our weights are, as always, constrained to be 0 or greater (the nonnnegativity conditions), so we can think about adjusting the values for our variables so that they decrease the objective function without violating the nonnegativity conditions. For example, if we set $w_2$ to 1 and keep $w_1$ at 0, then we have a new feasible solution in which the objective value is $-1$, an improvement over the initial solution. Notice that the final equation sets $w_5$ to 0 for this solution, the lowest it can go. We have taken in all the slack on $w_2$.

And the process can continue. If we bring the value of $w_1$ to 2 and $w_2$ to 6, then our objective value is $-8$, a substantial improvement. This is in fact the optimal solution for this system, as we can see from its graph:

(35)



It's the job of the next section to systematize this process of trying different weights and also to give us a way to determine whether the current solution is the optimal one.

### 4.3.2 Pivoting and optima detection

As we mentioned at the start of section 4, the simplex does not make a brute-force run-through of the entire feasible region. Like most efficient constraint solvers, it uses the inequalities themselves to find the optimal solution, by rewriting them into new but equivalent forms that encode different (increasingly optimal) solutions. For the simplex, the operation that performs these transformation is called *pivoting*. It is closely related to the *Gaussian elimination method* for solving systems of linear inequalities.

The pivoting operation is somewhat complex (Cormen et al. 2001:795), so we do not review its technical details. Suffice it to say that it involves a process of rewriting the input linear system into another, equivalent linear system in which a new solution is more apparent. This rewriting processes takes us to solutions that are vertices of the feasible region. The simplex algorithm continually pivots in this way, a series of "successive improvements" (Chvátal 1983:14), until an optimal solution is found. The result is a sequence of linear systems, each equivalent to the previous one, but each characterizing a different vertex. Here, for instance, is the system that characterizes the $(2, 6)$ point in (35):

(36)
$$\begin{array}{rcccccccc}
\text{minimize} & & & & & .78w_4 & + & .11w_5 \\
\text{subject to} & w_1 & = & 2 & - & .22w_4 & + & .11w_5 \\
 & w_2 & = & 6 & - & .56w_4 & - & .22w_5 \\
 & w_3 & = & 6 & + & .33w_4 & - & .67w_5
\end{array}$$

The slack weights have swapped places with the weights we care about. We set the slack weights $w_4$ and $w_5$ to 0, and then we can read off the solution $w_1 = 2$ and $w_2 = 6$.

We can see that this system is optimal by looking to the objective function. All its values are positive. This indicates that there is no more slack left in any of the variables: increasing any of them will increase the objective value. Thus, the algorithm terminates here.[8] (A fuller explanation for why an objective function with this shape signals optimality requires the concept of *duality*, which would take us much too far afield; see Cormen et al. 2001:§4.)

### 4.3.3 Aux programs and infeasibility detection

For all the systems arrived at via the conversion method of section 4.1, the strategy of setting all the weights to 0 for the initial solution fails, since that solution is not feasible; see the graph in (32). For this reason, we always use the *two-phase simplex* to solve our systems. In the two-phase simplex, one constructs from the initial system a special auxiliary system for which the all-0s solution is feasible and uses this system to move into the feasible region of the initial problem (ending phase one). In (32), this takes us from the origin of the graph to the point $(1, 2)$, which is a feasible solution. (In fact, it is the optimal solution.)

The auxiliary program also provides us with a means for detecting infeasibility. One of the central pieces of this auxiliary program is a new artificial weight, $w_0$. After we have solved the auxiliary program, we check the value of this variable. If its value is 0, then we can safely remove it and, after a few additional adjustments, we have a feasible solution to the original problem. If its value is not 0, however, then it is crucial to our finding a solution in the first place, thereby indicating that the initial problem has no solutions. This is the source of the verdict of ‘infeasible’ — the linguist’s cue that the grammar cannot deliver the desired set of optimal candidates.

## 4.4 Typology calculations

OT provides a successful theory of linguistic typology (factorial typology), and this has been a key component of its success. We would like to stress, therefore, that weighting systems make available their own theory of typology. The simplex-based approach brings us much of the way towards implementing it.

The first notion we require is that of *typological space*:

[8] The algorithm does not guarantee a unique optimal solution. Many systems have multiple optima. There are interesting mathematical issues surrounding this point, but they are not pressing for us, as far as we can tell, since we are generally only asking whether there is a feasible solution or not.

(37) Typological space

Let $\Pi$ be a countable set of candidates. The *typological space* for $\Pi$ is the set of sets $V$ such that $v \in V$ iff $v \subseteq \Pi$ and $v$ contains exactly one winning candidate $\langle I, O \rangle$ for every input $I$.

When $\Pi$ is infinite, the typological space for $\Pi$ is uncountably infinite. But typological spaces are merely our starting point. As in OT, our notion of typology is dependent largely on the constraints. We are interested only in those subsets of the typological space that have consistent weightings given the current constraint set:

(38) Typological predictions

The *typological predictions* for a candidate set $\Pi$ and constraint set $C$ are given by the subset $O$ of the typological space $V$ such that $o \in O$ iff there is a weighting $w$ for $C$ that defines all and only the candidates in $o$ as optimal.

We can think of the set of sets $V$ as the space of potential languages. $o \in V$ is a typological prediction if there is a weighting that characterizes it. If $o$ has no consistent weighting (if the linear system it determines is infeasible), then $o$ is predicted to be impossible.

When doing computational work and empirical typological research, we work with finite candidate sets. This ensures that the typological space and, in turn, the typological predictions, are finite. In the simplest approach, we convert each subset of $V$ to a linear system and then solve that system using the simplex. This is computationally expensive, but it defines the problem in a certain abstract sense. One can then work to formulate more efficient methods for reducing the space. For instance, if two candidates cannot be optimal together in the system one is looking at, then neither can any system containing them. This fact can often be used to eliminate large subsets of the full space of systems without even running the simplex on them.

As theorists, we may be working with infinite candidate sets (cf. McCarthy 2006). For OT, we have results showing that, with certain assumptions in place, the typological predictions are finite (Samek-Lodovici and Prince, 1999; Coetzee, 2003). These results build on the harmonic bounding properties of OT, which as Prince (2002a) shows, hold under HG (except for collective bounding; see footnote 5). Properly addressing this issue goes beyond the scope of this paper, but we suspect that finiteness in HG will depend on the constraints. The interaction of ALIGN-HEAD and WEIGHT-TO-STRESS in HG does produce infinitely many languages (Legendre et al., 2006b), as is evident from the following generalization of the pattern suggested by (11) above:

(39)

| Weights | | Language generated |
|---|---|---|
| W-to-S | Align-H-R | |
| 2 | 1.1 | ⇒ { bán.ta, **ban.ta.ná**, **ban.ta.na.má**, **ban.ta.na.ma.lá**, . . . } |
| 3 | 1.1 | ⇒ { bán.ta, bán.ta.na, **ban.ta.na.má**, **ban.ta.na.ma.lá**, . . . } |
| 4 | 1.1 | ⇒ { bán.ta, bán.ta.na, bán.ta.na.ma, **ban.ta.na.ma.lá**, . . . } |
| ⋮ | ⋮ | ⋮ |

The sequence of weightings in (39) produces an infinite set of languages because there is no limit (besides word-size) to the number of Align-Head-R violations that can trade off against a single Weight-to-Stress violation. It remains an open question which sets of constraints will produce trade-offs that lead to this result.

# 5 Harmonic grammar and local constraint conjunction

Local constraint conjunction (Smolensky, 2006) is an enrichment of standard OT in which two (or more) constraints can be conjoined to create a new separately rankable constraint that is violated if and only if the conjuncts are both (all) violated. In this section, we show that some of the effects of local conjunction can be produced by additive interaction in HG. We also show that there are strict limits on the types of gang effects that can be modeled in HG, which may lead to a more restrictive theory of constraint interaction than OT with freely conjoined constraints. This discussion also serves a number of other purposes: it further exemplifies the use of weighting conditions in HG analysis, it provides an additional example of how our reduction to linear systems works, and it illustrates the usefulness of infeasibility detection for revealing typological predictions.

Our example is a case of what McCarthy (2002, 2003a) refers to as a *grandfather effect*: a structure is banned as the output of a process like assimilation but can surface if present underlyingly. Baković (2000) and Smolensky (2006) analyze this sort of pattern as the joint product of a markedness constraint against the structure and of a faithfulness constraint against changes in feature value. In their OT analyses, this additive effect requires the postulation of a separate conjoined constraint. As we show in what follows, in HG, only the basic constraints are needed.

In McCarthy's (2003b) Meccan Arabic paradigm (Bakalla, 1973; Abu-Mansour, 1996; Mascaró and Wetzels, 2001), voicing assimilation (e.g., (40a)) fails to create voiced obstruents (40b), although underlying voiced obstruents surface faithfully, as in (40c).

(40) a. /ʔagsam/ → [ʔaksam]

b. /ʔakbar/ → [ʔakbar]

c. /ʔibnu/ → [ʔibnu]

We assume three standard OT constraints: two markedness constraints assessing output voicing on obstruents (Lombardi, 1999) and a faithfulness constraint demanding that the voicing specification remain constant between input and output (McCarthy and Prince, 1999):

(41) Constraint definitions

a. Agree-Voice: Adjacent obstruents have the same value for [voice].

b. *Voice: Obstruents are [−voice].

c. Ident-Voice: Segments in correspondence have the same value for [voice].

We first present the tableaux without weightings, so as to demonstrate the steps involved in constructing an HG analysis. The first tableau shows the violation marks incurred by a voiced obstruent that occurs in isolation, and for a candidate in which underlying [+voice] is lost. For the surface voiced obstruent to be optimal, the weight of Ident-Voice must be greater than that of *Voice.

(42) Voiced obstruents are allowed: $w$(Ident-Voice) > $w$($^*$Voice)

| /ʔibnu/ | Agree-Voice | Ident-Voice | *Voice |
|---|---|---|---|
| ☞ [ʔibnu] | 0 | 0 | 1 |
| [ʔipnu] | 0 | 1 | 0 |

In the context of a following voiceless obstruent, an underlying voiced segment surfaces as voiceless, which yields the weighting condition shown above the tableau in (43).

(43) Voiced obstruents assimilate: $w$(Agree-Voice) + $w$($^*$Voice) > $w$(Ident-Voice)

| /ʔagsam/ | Agree-Voice | Ident-Voice | *Voice |
|---|---|---|---|
| [ʔagsam] | 1 | 0 | 1 |
| ☞ [ʔaksam] | 0 | 1 | 0 |

Finally, the underlying voiceless obstruent fails to assimilate to a following voiced one, which yields a third weighting condition:

(44) Voiceless obstruents do not assimilate: $w$(Ident-Voice)+$w$($^*$Voice) > $w$(Agree-Voice)

| /ʔakbar/ | Agree-Voice | Ident-Voice | *Voice |
|---|---|---|---|
| [ʔagbar] | 0 | 1 | 2 |
| ☞ [ʔakbar] | 1 | 0 | 1 |

The full set of strict inequalities that must hold so that the optima beat their rivals are as in (45):

(45)
a. $w(\textsc{Ident-Voice}) > w({}^{*}\textsc{Voice})$

b. $w(\textsc{Agree-Voice}) + w({}^{*}\textsc{Voice}) > w(\textsc{Ident-Voice})$

c. $w(\textsc{Ident-Voice}) + w({}^{*}\textsc{Voice}) > w(\textsc{Agree-Voice})$

The difficulty of finding a set of values that satisfy these conditions will depend on one's familiarity with solving problems involving strict inequalities. This process can be circumvented by using a computational solver. The procedure described in the previous section first translates the above tableau into the linear system in (46).

(46)
$$\begin{array}{rrcrcrcl}
\text{minimize} & 1w_{\textsc{Agree-Voice}} & + & 1w_{\textsc{Ident-Voice}} & + & 1w_{{}^{*}\textsc{Voice}} & & \\
\text{subject to} & 0w_{\textsc{Agree-Voice}} & + & 1w_{\textsc{Ident-Voice}} & - & 1w_{{}^{*}\textsc{Voice}} & \geqslant & a \\
 & 1w_{\textsc{Agree-Voice}} & - & 1w_{\textsc{Ident-Voice}} & + & 1w_{{}^{*}\textsc{Voice}} & \geqslant & a \\
 & -1w_{\textsc{Agree-Voice}} & + & 1w_{\textsc{Ident-Voice}} & + & 1w_{{}^{*}\textsc{Voice}} & \geqslant & a \\
 & 1w_{\textsc{Agree-Voice}} & & & & & \geqslant & b \\
 & & & 1w_{\textsc{Ident-Voice}} & & & \geqslant & b \\
 & & & & & 1w_{{}^{*}\textsc{Voice}} & \geqslant & b \\
 & & & & & a, b & > & 0 \\
 & & & & & \text{all } w_i & \geqslant & 0
\end{array}$$

This system can be solved using the simplex algorithm. If we set $a = b = 1$, then we get (47) as the optimal (minimal) weighting that delivers the attested pattern.

(47)
a. $w(\textsc{Agree-Voice}) = 2$

b. $w({}^{*}\textsc{Voice}) = 2$

c. $w(\textsc{Ident-Voice}) = 1$

Inserting these values into the tableaux in (42)–(44) shows that this analysis makes use of two additive interactions: the one in (44) between Ident-Voice and *Voice that mimics the local conjunction analysis of Baković (2000) and Smolensky (2006), as well as one in (43) between Agree-Voice and *Voice. One of these is demanded by the pattern under the current constraint set; no OT ranking will work (though see McCarthy 2002, 2003a and de Lacy 2006 for other expansions of the constraint set). The ERCs for the above tableaux are in (48).

(48)
a. IdentVoice ≫ *Voice

b. Agree-Voice ≫ Ident-Voice *or* *Voice ≫ Ident-Voice

c. *Voice ≫ Agree-Voice *or* Ident-Voice ≫ Agree-Voice

Because of (48a), the ERC in (48b) must be met by Agree-Voice ≫ Ident-Voice. This in turn leads (48c) to require *Voice ≫ Agree-Voice. These three requirements cannot be expressed in the linear order required for OT grammars.

As Tesar (1998) and Prince (2002b) emphasize, an important property of the Constraint Demotion Algorithm (Tesar and Smolensky, 1998) is that it detects such inconsistency. In section 4.4, we discussed the importance of inconsistency detection for the calculation of the typological predictions of a constraint set: if a set of optima is inconsistent, the language they represent is predicted to be impossible (see Tesar 1998 et seq. for learnability applications).

As we mentioned in the introduction, the primary advantage of finding HG weightings using the simplex algorithm as opposed to connectionist or other probabilistic learners is that, like the Constraint Demotion Algorithm, it detects inconsistency. When a linear programming solver is given a set of conditions that are inconsistent, it returns a verdict of ‘infeasible’. As an example, we might wonder whether the constraint Agree-Place can also participate in a gang effect with Agree-Voice, producing a language in which clusters assimilate in voicing only if they disagree in place.

(49) Meccan′

a. *w*(Agree-Voice) > *w*(Ident-Voice)

| /ʔagsam/ | Agree-Voice | Ident-Voice | Agree-Place |
|---|---|---|---|
| [ʔagsam] | 1 | 0 | 1 |
| ☞ [ʔaksam] | 0 | 1 | 1 |

b. *w*(Ident-Voice) > *w*(Agree-Voice)

| /ʔadsam/ | Agree-Voice | Ident-Voice | Agree-Place |
|---|---|---|---|
| ☞ [adsam] | 1 | 0 | 0 |
| [atsam] | 0 | 1 | 0 |

In this simple example, it is clear that the weighting conditions are inconsistent. As the size of the systems grow, however, it quickly becomes difficult to spot such inconsistencies by eye (see (14) and (15)). This is where our simplex-based methods can be so valuable.

This example highlights an important difference between local conjunction and HG. As discussed in section 2, gang effects in HG are possible only when there is an asymmetric trade-off in constraint violations between candidates. In the case in (49a), because both candidates have a cluster that disagrees in place, its presence is irrelevant to whether there is voicing assimilation. For two constraints to produce a gang effect in HG, it must be the case that both violations can be avoided by incurring a single violation of another constraint, as in the additive interactions in our analysis of real Meccan.

Because local conjunction is not subject to the restrictions of additive HG, Legendre et al. (2006b) refer to it as *super-additive*. While they argue in favor of super-additivity, the additional power can produce unwanted effects, labeled as problems of *locality* and *relevance* by McCarthy (2002, 2003a). The relevance problem can be illustrated by the fact that if Agree-Place and Agree-Voice were conjoined, the resulting constraint would favor the optimal outcome in (49a), but not (49b), and thus it could be used to produce this unattested pattern. The locality problem could be illustrated by using a self-conjoined version of NoCoda to produce the pattern in (8). While some attempts have been made to restrict the power of local conjunction by imposing limits on domains and conjuncts, at least some restrictions on locality and relevance are entailed by the nature of additive interaction in HG.[9]

The comparisons in this paper have shown that the power of HG appears to lie somewhere between standard OT and OT with locally conjoined constraints. Determining whether its expressive power is desirable will require careful study of the predictions of various constraint sets, and this should focus attention on what counts as a suitable constraint. Constraints that are suitable for OT may not be suitable for HG, and vice versa. To ward off unwanted gang effects, for instance, HG constraints must be local in their scope. (See section 2 above on gradient Alignment constraints, as well as the discussion of the Null Parse in Pater 2006). On the other hand, local scalar constraints are compatible with HG, but they are of limited utility in OT due to strict domination (Prince and Smolensky 1993/2004:8, McCarthy 2003b:84). The ability to express a scalar generalization with a single constraint rather than with a fixed ranking of a set of constraints may count as an important argument for HG (see further Flemming 2001; cf. Boersma 1998:211). To fully appreciate the ramifications of our decisions about which constraints to admit and which to exclude, we must look closely at the predictions of a wide and varied array of constraint sets, subjecting each to considerable amounts of data. And insofar as evidence for strict domination will be rather subtle when constraints are local and/or categorical, deciding between ranking and weighting will also require this kind of careful work This study can be greatly facilitated by the availability of HaLP.

[9] If Ident-Voice is replaced by Max in the tableaux in (49), the additive interaction is possible. There are at least two paths to a solution, and choosing between them would take us too far afield. One possibility is to adopt a theory of faithfulness in which deletion is not a possible repair for Agree-Voice (e.g., Lombardi 2001; Steriade 2001). Another is to adopt a theory of Gen that limits each operation to a single faithfulness violation (McCarthy, 2006).

# 6 Conclusion

We have shown that constraint weighting problems translate into linear systems and are thus solvable using the simplex algorithm (section 4). This is an important mathematical connection, and it should provide significant insights into the nature of linguistic optimization. It has a practical component as well: our linear solver HaLP (Potts et al., 2006) should facilitate comparison between weighting and other constraint-based approaches. The present version runs in Web browsers. The user uploads two Praat files (Boersma and Weenink, 2006), one specifying the intended winners and another specifying violation profiles for the full constraint-set. The program processes these using the simplex and returns a tableau-formatted optimal weighting or else 'infeasible'. This implementation, freely available and requiring no software downloads or specialized user expertise, gets us over the intrinsic practical obstacles to exploring weighting systems. We can then focus attention on the linguistic usefulness of HG and other weighting-based approaches. Section 2 calls into question the notion that HG is too unconstrained to provide suitable typological theories (see also section 4.4), and the results of section 5 suggest that HG might fare well empirically when compared with standard OT and the enriched version of it that permits local constraint conjunction.

The formal results of this paper are best summarized by drawing an explicit connection with the fundamental theorem of linear programming (Cormen et al. 2001:816):

(50) The fundamental theorem of linear programming

If *L* is a linear system in standard form, then there are just three possibilities:

1. *L* has an optimal solution with a finite objective function.
2. *L* is unbounded (in which case we can return a solution, though the notion of optimal is undefined).
3. *L* is infeasible (no solution satisfies all its conditions).

Our method reveals a deep connection between this theorem and HG. The 'unbounded' output is not of much concern to us. The sense it has here is not 'the feasible region is unbounded', but rather 'there is no limit to how good the solution can be'. Thus, even though the feasible regions for phonological systems are unbounded ("open at the top"), we solve minimization problems, thereby avoiding this pitfall. The 'infeasible' verdict is essential. It tells us that the current grammar cannot deliver the set of optimal constraints we have specified. This might be a signal that the analysis must change, or it might prove that a predicted typological gap in fact exists for the current constraint set. And if we are presented with an optimal solution, then we know our grammar delivers the specified set of forms as optimal. Moreover, we can then analyze the solution to learn about the relations among our constraints.

We obtain these results efficiently, and they hold for the full range of linear systems, including very large ones. We therefore see the translation of HG systems into linear systems as providing a valuable tool for the serious exploration of constraint weighting in linguistics. We also see great promise in the approach for developing theories of learning, for determining the nature of the constraint set, and for gaining a deeper understanding of the theory's main building blocks (e.g., Gen and Eval). What's more, we see great potential for applying to these endeavors the wealth of existing mathematical results about linear systems.

# References

Abu-Mansour, Mahasen Hasan. 1996. Voice as a privative feature. In *Perspectives on arabic linguistics viii: Papers from the eighth annual symposium on arabic linguistics*, ed. Mushira Eid, 201–231. Amsterdam and Philadelphia: John Benjamins.

Alderete, John. 2001. Dominance effects as trans-derivational anti-faithfulness. *Phonology* 18:201–253.

Bakalla, Mohammed. 1973. The morphology and phonology of meccan arabic. Doctoral Dissertation, SOAS, University of London.

Baković, Eric. 2000. Harmony, dominance, and control. Doctoral Dissertation, Rutgers University, New Brunswick, NJ.

Bakovic, Eric. 2004. Unbounded stress and factorial typology. In *Optimality Theory in phonology: A reader*, ed. John J. McCarthy, 202–214. Oxford: Blackwell. Originally published in RuLing Papers 1 (Working Papers from Rutgers University), ed. Ron Artstein and Madeline Holler, 1998.

Boersma, Paul. 1998. Functional phonology: Formalizing the interactions between articulatory and perceptual drives. Doctoral Dissertation, University of Amsterdam.

Boersma, Paul, and David Weenink. 2006. Praat: Doing phonetics by computer, version 4.4. URL `http://www.praat.org/`, Developed at the Institute of Phonetic Sciences, University of Amsterdam.

Chvátal, Vašek. 1983. *Linear programming*. New York: W. H. Freeman and Company.

Coetzee, Andries. 2003. Just how many languages are there? In *Proceedings of the 33rd meeting of the North East Linguistic Society*, ed. Makoto Kadowaki and Shigeto Kawahara, 103–114. Amherst, MA: GLSA.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. 2001. *Introduction to algorithms*. Cambridge, MA and New York: MIT Press and McGraw-Hill, 2 edition.

Dantzig, George B. 1981/1982. Reminiscences about the origins of linear programming. *Operations Research Letters* 1:43–48.

Eisner, Jason. 1998. FootForm decomposed: Using primitive constraints in OT. In *Proceedings of SCIL VIII*, ed. Benjamin Bruening, number 31 in MIT Working Papers in Linguistics, 115–143. Cambridge, MA. URL `http://cs.jhu.edu/~jason/papers/#scil96`.

Flemming, Edward. 2001. Scalar and categorical phenomena in a unified model of phonetics and phonology. *Phonology* 18:7–44.

Goldwater, Sharon, and Mark Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. In *Proceedings of the stockholm workshop on variation within optimality theory*, ed. Jennifer Spenader, Anders Eriksson, and Östen Dahl, 111–120. Stockholm: Stockholm University.

Hayes, Bruce, and Paul Boersma. 2001. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32:45–86.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw. 2003. OTSoft 2.1. URL `http://www.linguistics.ucla.edu/people/hayes/otsoft/`, Software package developed at UCLA.

Hayes, Bruce, and Colin Wilson. 2006. A maximum entropy model of phonotactics and phonotactic learning. Ms., UCLA.

Jäger, Gerhard. to appear. Maximum entropy models and Stochastic Optimality Theory. In *Architectures, rules, and preferences: A festschrift for Joan Bresnan*, ed. Jane Grimshaw, Joan Maling, Chris Manning, Jane Simpson, and Annie Zaenen. Stanford, CA: CSLI.

Jesney, Karen, Joe Pater, and Anne-Michelle Tessier. in prep. Restrictiveness and gang effects in learning. Ms., University of Massachusetts, Amherst, and University of Alberta.

Kager, René. 2001. Rhythmic directionality by positional licensing. Paper presented at the 5th HIL Phonology Conference, University of Potsdam. Handout available as ROA-514 from the Rutgers Optimality Archive.

Keller, Frank. 2006. Linear optimality theory as a model of gradience in grammar. In *Gradience in grammar: Generative perspectives*, ed. Gisbert Fanselow, Caroline Féry, Ralph Vogel, and Matthias Schlesewsky. Oxford: Oxford University Press.

de Lacy, Paul. 2006. *Markedness: Reduction and preservation in phonology*. Cambridge Studies in Linguistics 112. Cambridge: Cambridge University Press.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990a. Harmonic Grammar – a formal multi-level connectionist theory of linguistic wellformedness: An application. In *Proceedings of the twelfth annual conference of the Cognitive Science Society*, 884–891. Cambridge, MA: Lawrence Erlbaum.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990b. Harmonic Grammar – a formal multi-level connectionist theory of linguistic wellformedness: Theoretical foundations. In *Proceedings of the twelfth annual conference of the Cognitive Science Society*, 388–395. Cambridge, MA: Lawrence Erlbaum.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 2006a. The interaction of syntax and semantics: A Harmonic Grammar account of split intransitivity. In Smolensky and Legendre (2006), 417–452.

Legendre, Géraldine, Antonella Sorace, and Paul Smolensky. 2006b. The Optimality Theory–Harmonic Grammar connection. In Smolensky and Legendre (2006), 903–966.

Lombardi, Linda. 1999. Positional faithfulness and voicing assimilation in Optimality Theory. *Natural Language and Linguistic Theory* 17:267–302.

Lombardi, Linda. 2001. Why place and voice are different: Constraint-specific alternations and Optimality Theory. In *Segmental phonology in Optimality Theory: Constraints and representations*, ed. Linda Lombardi, 13–45. Cambridge: Cambridge University Press.

Mascaró, Joan, and Leo Wetzels. 2001. The typology of voicing and devoicing. *Language* 77:207–244.

McCarthy, John, and Alan Prince. 1993. Prosodic morphology I: Constraint interaction and satisfaction. RuCCS 3, University of Massachusetts, Amherst, and Rutgers University. ROA-482.

McCarthy, John J. 2002. Comparative markedness [long version]. In *Papers in Optimality Theory II (University of Massachusetts occasional papers in linguistics)*, ed. Angela C. Carpenter, Andries W. Coetzee, and Paul de Lacy, volume 26, 171–246. Amherst, MA: GLSA.

McCarthy, John J. 2003a. Comparative markedness [short version]. *Theoretical Linguistics* 29:1–51.

McCarthy, John J. 2003b. OT constraints are categorical. *Phonology* 20:75–138.

McCarthy, John J. 2006. Restraint of analysis. In *Wondering at the natural fecundity of things: Essays in honor of Alan Prince*, chapter 10. Linguistics Research Center. URL `http://repositories.cdlib.org/lrc/prince/10`.

McCarthy, John J., and Alan Prince. 1999. Faithfulness and identity in prosodic morphology. In *The prosody–morphology interface*, ed. René Kager, Harry van der Hulst, and Wim Zonneveld, 218–309. Cambridge: Cambridge University Press. ROA-216.

Murty, Katta G. 1995. *Operations research: Deterministic optimization models*. Upper Saddle River, NJ: Prentice Hall.

Pater, Joe. 2006. Additive optimization and phonological typology. URL `http://people.umass.edu/pater/AO+typology.pdf`, Handout from a talk presented at the MIT/UMass Phonology Workshop, February 11.

Pater, Joe. to appear. Non-convergence in the Gradual Learning Algorithm. *Linguistic Inquiry* .

Pater, Joe, and Andries Coetzee. 2005. Lexically specific constraints: Gradience, learnability, and perception. In *Proceedings of the 3rd seoul international conference on phonology*, 85–119. Seoul: The Phonology-Morphology Circle of Korea.

Potts, Christopher, Joe Pater, Rajesh Bhatt, and Michael Becker. 2006. HaLP: Harmonic grammar with linear programming, version 1. Software available online at http://web.linguist.umass.edu/˜halp/.

Prince, Alan. 2002a. Anything goes. In *New century of phonology and phonological theory*, ed. Takeru Honma, Masao Okazaki, Toshiyuki Tabata, and Shin ichi Tanaka, 66–90. Tokyo: Kaitakusha. ROA-536.

Prince, Alan. 2002b. Arguing optimality. In *Papers in Optimality Theory II*, ed. Andries Coetzee, Angela Carpenter, and Paul de Lacy, 269–304. Amherst, MA: GLSA. ROA-562.

Prince, Alan. 2002c. Entailed ranking arguments. Ms., Rutgers University, New Brunswick, NJ. ROA 500-0202.

Prince, Alan, and Paul Smolensky. 1993/2004. Optimality Theory: Constraint interaction in generative grammar. RuCCS Technical Report 2, Rutgers University, Piscateway, NJ: Rutgers University Center for Cognitive Science. Revised version published 2004 by Blackwell. Page references to the 2004 version.

Prince, Alan, and Paul Smolensky. 1997. Optimality: From neural networks to universal grammar. *Science* 275:1604–1610.

Samek-Lodovici, Vieri, and Alan Prince. 1999. Optima. RuCCS Technical Report no. 57, Rutgers University, Piscateway, NJ: Rutgers University Center for Cognitive Science. ROA 363-1199.

Sedgewick, Robert. 1992. *Algorithms in C++*. Reading, MA: Addison-Wesley, 2 edition.

Smolensky, Paul. 2006. Optimality in phonology II: Harmonic completeness, local constraint conjunction, and feature domain markedness. In Smolensky and Legendre (2006), 27–160.

Smolensky, Paul, and Géraldine Legendre. 2006. *The harmonic mind: From neural computation to Optimality-Theoretic grammar*, volume I: cognitive architecture. Cambridge, MA: MIT Press.

Smolensky, Paul, Géraldine Legendre, and Bruce Tesar. 2006. Optimality Theory: The structure, acquisition and use of grammar. In Smolensky and Legendre (2006), 453–535.

Steriade, Donca. 2001. The phonology of perceptibility effects: The P-map and its consequences for constraint organization. Ms., MIT.

Tesar, Brice. 1998. Using the mutual inconsistency of structural descriptions to overcome ambiguity in language learning. In *Proceedings of NELS 28*, ed. Pius N. Tamanji and Kiyomi Kusumoto, 469–483. Amherst, MA: GLSA.

Tesar, Bruce, and Paul Smolensky. 1998. Learning Optimality-Theoretic grammars. *Lingua* 106:161–196.

Wilson, Colin. to appear. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science* .