Optimality Theory is not computable*

Andrew Lamont
University College London
andrew.lamont@ucl.ac.uk

22 March 2023

- Non-cyclic rule-based models of phonology such as SPE (Chomsky and Halle, 1968) are *finite-state* (Johnson, 1972; Koskenniemi, 1983; Kaplan and Kay, 1994)
  - Insofar as these models are empirically adequate, natural language phonology also appears to be finite-state (see Heinz, 2018 for discussion)
  - Among other things, this hypothesis implies that phonology **cannot calculate unbounded numbers**[1] and is **easily computed**
- *Optimality Theory* (OT; Prince and Smolensky, 1993/2004) is known to be more computationally complex than rule-based phonology (Eisner, 1997b, 2000; Frank and Satta, 1998; Gerdemann and Hulden, 2012)
  - While some variants of OT are finite-state (Eisner, 2000, 2002; Frank and Satta, 1998; Riggle, 2004; see Hulden, 2017 for discussion), most are not
  - Complexity obtains regardless of the constraint types (Lamont, 2021, 2022b)
- Eisner (1997a); Idsardi (2006) have shown that OT is not easily computed; the amount of time required to generate an output is impractically large (see Heinz et al., 2009 for discussion of these results)

⇒ OT requires more complex computation than expected as a model of phonology

This talk contributes to the computational characterization of OT by demonstrating that it is **not computable** in general. In other words, it is impossible to write an algorithm that takes an arbitrary OT grammar and determines the output for an arbitrary input.

§1 Computability and the Halting Problem
§2 The Post Correspondence Problem, a non-computable problem
§3 How OT implements the Post Correspondence Problem
§4 Discussion

*For his comments on this work at various stages of development, I am grateful to Neil Immerman. All remaining errors are of course my own.

[1] This is distinct from the sense of *counting* discussed by Paster (2019).

1 Introduction

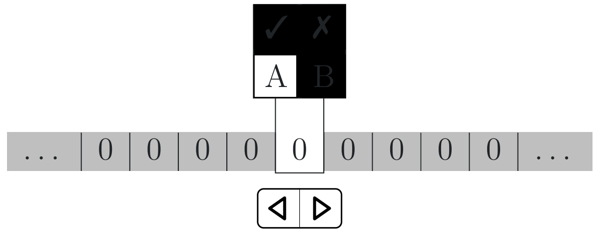
This section draws heavily from Sipser (2013); see also Partee et al. (1990) and Hopcroft et al. (2008) for discussion.

1.1 Turing Machines

- Turing Machines are abstract computing devices equipped with a *finite control* and an *infinite memory* (Turing, 1937)
- Formally, a Turing Machine is defined as a 6-tuple $(Q, \Sigma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where
  - Q is a finite set of states
  - Σ is a finite set of symbols, the *alphabet*
  - $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\text{L}, \text{R}\}$ is the *transition function*
  - $q_{\text{start}} \in Q$ is the *start state*
  - $q_{\text{accept}} \in Q$ is the *accept state*
  - $q_{\text{reject}} \in Q$ is the *reject state*
- Turing Machines begin the computation of an input in the start state. On each successive step, they *read* a symbol from the input, *write* a symbol in its place, *update* their state, and *move* one symbol to the left or right. This continues recursively until the machine reaches the accept state or reject state, at which point the computation *halts*. If the computation does not halt, the computation enters an infinite *loop*.
  - For example, consider `Te reo Māori`, which determines whether a string is a possible word of Māori. Among other things, Māori does not allow words to contain consonant clusters (Harlow, 2007). If `Te reo Māori` identifies a consonant cluster, it should transition into its reject state and halt
  - If phonology is in fact finite-state, `Te reo Māori` can be implemented as a *finite-state automaton*, a read-only machine that only moves to the right
- An illustration: the *Busy Beaver* problem asks for the maximum number of 1's a machine with n states can write to a tape of all 0's before halting (Radó, 1962)
- With two states, the maximum number of 1's is four

(1) Definition of `Busy Beaver 2`
   - $Q = \{\text{A, B, ✓, ✗}\}$
   - $\Sigma = \{0, 1\}$
   - $\delta = \{(\text{A,0,B,1,R}), (\text{A,1,B,1,L}), (\text{B,0,A,1,L}), (\text{B,1,✓,1,R})\}$
   - $q_{\text{start}} = \text{A}$
   - $q_{\text{accept}} = $ ✓
   - $q_{\text{reject}} = $ ✗

(2) Computation by `Busy Beaver 2`

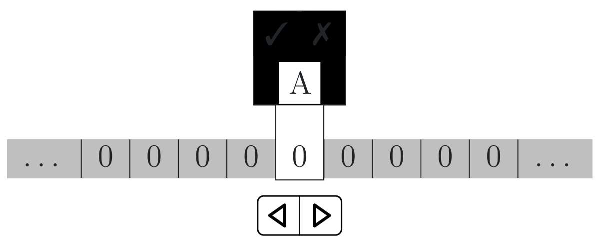


- Some Turing Machines, like `Tsà`[2], do not halt. `Tsà` begins computations in state A, and never transitions out of it

(3) Definition of `Tsà`

  - $Q = \{\text{A}, ✓, ✗\}$
  - $\Sigma = \{0, 1\}$
  - $\delta = \{(\text{A},0,\text{A},1,\text{R}), (\text{A},1,\text{A},1,\text{R})\}$
  - $q_{\text{start}} = \text{A}$
  - $q_{\text{accept}} = ✓$
  - $q_{\text{reject}} = ✗$

(4) Partial computation by `Tsà`



1.2 Computability

- A problem is said to be **computable** if can be solved by an *algorithm*, a Turing Machine that is guaranteed to halt on all inputs

- The function BB(n), the maximum number of 1's a Turing Machine with n states can write to a tape of all 0's, is not computable. Only five values are known:

  - BB(0) = 0 (Radó, 1962)
  - BB(1) = 1 (Radó, 1962)
  - BB(2) = 4 (Radó, 1962)
  - BB(3) = 6 (Lin and Radó, 1965)
  - BB(4) = 13 (Brady, 1983)

[2]Named for the Tłı̨chǫ beaver spirit who teaches productivity – thanks to Shay Hucklebridge

⇒ BB(n) grows faster than any computable function; it is impossible to write an algorithm that solves BB(n) (Radó, 1962)

Proof (following Hopcroft, 1984):

BB(n) strictly grows in n; i.e., $\mathrm{BB}(n+1) > \mathrm{BB}(n)$. You can always add a state that finds the first 0 to the right, writes a 1, and transitions into q_{accept}.

For any $n \in \mathbb{N}$, a Turing Machine with $n + 19$ states can be built that writes n^2 1's onto a tape of all 0's.

Suppose *by way of contradiction* that the Turing Machine `BusyBeaverSolver` exists with k states. `BusyBeaverSolver` reads a string of n 1's and writes BB(n) 1's.

$$\ldots \mid 0 \mid \mathbf{1} \mid \mathbf{1} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid \ldots \quad \rightarrow \quad \ldots \mid 0 \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid 0 \mid 0 \mid 0 \mid \ldots$$

For any $n \in \mathbb{N}$, build a Turing Machine `SquareAndSolve` which calls `BusyBeaverSolver` as a subroutine. `SquareAndSolve` writes n^2 1's and then calls `BusyBeaverSolver`. `SquareAndSolve` writes BB(n^2) 1's onto a tape of all 0's; it has $n + 19 + k$ states.

$$\ldots \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid \ldots \quad \rightarrow$$

$$\ldots \mid 0 \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid \ldots \quad \rightarrow$$

$$\ldots \mid 0 \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid \mathbf{1} \mid 0 \mid \ldots$$

It follows that a Turing Machine with $n + 19 + k$ states can write at least as many 1's onto a tape of all 0's as a machine with n^2 states.

$$\mathrm{BB}(n + 19 + k) \geq \mathrm{BB}(n^2)$$

This contradicts the fact that BB(n) strictly grows in n, when $n > 2$:

$$\begin{aligned}
&\text{Let } n = 20 + k \\
&n + 19 + k = n + (n-1) = 2n - 1 = n^2 - (n-1)^2 \\
&n^2 - 1 > n^2 - (n-1)^2 \\
&\mathrm{BB}(n^2 - 1) > \mathrm{BB}(n + 19 + k) \geq \mathrm{BB}(n^2) \\
&\mathrm{BB}(n^2 - 1) > \mathrm{BB}(n^2)
\end{aligned}$$

Therefore, `BusyBeaverSolver` does not exist, and BB(n) is not computable. □

⇒ This result implies that the *Halting Problem* is also not computable; see Church (1936); Turing (1937) for alternate proofs. The Halting Problem asks whether an arbitrary Turing Machine will halt when run on an arbitrary input

BB(n) grows faster than any computable function. A related function S(n) gives the maximum number of computational steps a Turing Machine with n states can take before halting. S(n) is at least as large as BB(n) because not every step of a Busy Beaver computation writes a new 1. S(n) is therefore itself not computable.

If S(n) were computable, it would imply a solution to the Halting Problem: if a machine does not halt within S(n) steps, it never will. However, because S(n) is not computable, the Halting Problem is not computable.

⇒ If the solution to a formal problem would imply a solution to the Halting Problem, it is itself not computable

2 The Post Correspondence Problem

The formal problem of interest is the *Post Correspondence Problem* (PCP; Post, 1946),[3] which is itself not computable. This section also draws heavily from Sipser (2013).

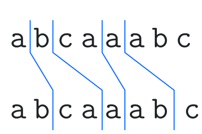
- An instance of the PCP defines a set of domino *types*, such as

$$\left\{ \left[\frac{\mathtt{b}}{\mathtt{c\ a}}\right], \left[\frac{\mathtt{a}}{\mathtt{a\ b}}\right], \left[\frac{\mathtt{c\ a}}{\mathtt{a}}\right], \left[\frac{\mathtt{a\ b\ c}}{\mathtt{c}}\right] \right\}$$

- A solution is a sequence of domino *tokens*, where the string along the *top* matches the string along the *bottom*

$$\left[\frac{\mathtt{a}}{\mathtt{a\ b}}\right] \left[\frac{\mathtt{b}}{\mathtt{c\ a}}\right] \left[\frac{\mathtt{c\ a}}{\mathtt{a}}\right] \left[\frac{\mathtt{a}}{\mathtt{a\ b}}\right] \left[\frac{\mathtt{a\ b\ c}}{\mathtt{c}}\right]$$

- Following Sipser, I vertically align the strings with domino boundaries deformed



- Analogous to Busy Beaver machines, Lorentz (2001) demonstrates that small sets of domino types can have surprisingly long solutions. The solution to the instance below has a minimum length of 76 characters

$$\left\{ \left[\frac{\mathtt{0\ 1}}{\mathtt{0}}\right], \left[\frac{\mathtt{0\ 1}}{\mathtt{1\ 1\ 0}}\right], \left[\frac{\mathtt{0}}{\mathtt{1\ 0\ 0}}\right], \left[\frac{\mathtt{1\ 0\ 0}}{\mathtt{1\ 0}}\right] \right\}$$

[3] I am grateful to Cerek Hillen for first drawing my attention to the PCP

- Some instances of the PCP do not have solutions

$$\left\{ \left[\frac{\texttt{a b c}}{\texttt{a b}}\right], \left[\frac{\texttt{c a}}{\texttt{a}}\right], \left[\frac{\texttt{a c c}}{\texttt{b a}}\right] \right\}$$

⇒ **The PCP is not computable** (Post, 1946). Sipser (2013:§5.2) provides an algorithm to translate an arbitrary Turing Machine and input into an instance of the PCP, which has a solution if and only if the corresponding computation halts

(5) The PCP instance for `Busy Beaver 2` (slightly simplified for presentation)

$$\left\{ \begin{array}{l} \left[\frac{\texttt{\#}}{\texttt{\# 0 0 A 0 0 \#}}\right], \\ \left[\frac{\texttt{A 0}}{\texttt{1 B}}\right], \left[\frac{\texttt{B 1}}{\texttt{1 } \checkmark}\right], \left[\frac{\texttt{0 A 1}}{\texttt{B 0 1}}\right], \left[\frac{\texttt{1 A 1}}{\texttt{B 1 1}}\right], \left[\frac{\texttt{0 B 0}}{\texttt{A 0 1}}\right], \left[\frac{\texttt{1 B 0}}{\texttt{A 1 1}}\right], \\ \left[\frac{\texttt{0}}{\texttt{0}}\right], \left[\frac{\texttt{1}}{\texttt{1}}\right], \left[\frac{\texttt{\#}}{\texttt{\#}}\right], \left[\frac{\texttt{\#}}{\texttt{0 \#}}\right], \\ \left[\frac{\texttt{0 } \checkmark}{\checkmark}\right], \left[\frac{\checkmark \texttt{ 0}}{\checkmark}\right], \left[\frac{\texttt{1 } \checkmark}{\checkmark}\right], \left[\frac{\checkmark \texttt{ 1}}{\checkmark}\right], \left[\frac{\checkmark \texttt{ \# \#}}{\texttt{\#}}\right] \end{array} \right\} \begin{array}{l} \left.\right\} \text{Initial domino } (q_{\text{start}}, \textit{input}) \\ \left.\right\} \text{Transition function } (\delta) \\ \left.\right\} \text{Alphabet } (\Sigma) \\ \left.\right\} \text{Final dominoes } (q_{\text{accept}}) \end{array}$$

- There's a solution to this instance of the PCP that begins with the initial domino if and only if `Busy Beaver 2` halts, which we know to be the case. The minimal solution below encodes the computation, step-by-step:

(6) Simulation of `Busy Beaver 2`

$$\texttt{\#|0|0|A 0|0|\#|0|0|1 B 0|\#|0|0|A 1|1|\#|0 B 0|1|1|\#|A 0 1|1|1|\#|1|B 1|1|1|\#|1|1|}\checkmark\texttt{ 1|1|\#|1|1|}\checkmark\texttt{ 1|\#|1|1|}\checkmark\texttt{|\#|1|}\checkmark\texttt{|\#|}\checkmark\texttt{ \# \#}$$

$$\underbrace{\underbrace{\texttt{\# 0 0 A 0 0}}_{\text{Start}}\underbrace{\texttt{\# 0 0 1 B 0}}_{\text{Step 1}}\underbrace{\texttt{\# 0 0 A 1 1}}_{\text{Step 2}}\underbrace{\texttt{\# 0 B 0 1 1}}_{\text{Step 3}}\underbrace{\texttt{\# A 0 1 1 1}}_{\text{Step 4}}\underbrace{\texttt{\# 1 B 1 1 1}}_{\text{Step 5}}\underbrace{\texttt{\# 1 1 } \checkmark \texttt{ 1 1}}_{\text{Step 6}}}_{\text{Computation}}\underbrace{\texttt{\# 1 1 } \checkmark \texttt{ 1 \# 1 1 } \checkmark \texttt{ \# 1 } \checkmark \texttt{ \# } \checkmark \texttt{ \# \#}}_{\text{Cleaning up}}$$

- If an instance of the PCP has one solution, it has infinitely many, because the sequence of domino tokens can be repeated arbitrarily many times

(7) The PCP instance for `Tsà` (slightly simplified for presentation)

$$\left\{ \begin{array}{l} \left[\frac{\texttt{\#}}{\texttt{\# 0 0 A 0 0 \#}}\right], \\ \left[\frac{\texttt{A 0}}{\texttt{1 A}}\right], \left[\frac{\texttt{A 1}}{\texttt{1 A}}\right], \\ \left[\frac{\texttt{0}}{\texttt{0}}\right], \left[\frac{\texttt{1}}{\texttt{1}}\right], \left[\frac{\texttt{\#}}{\texttt{\#}}\right], \left[\frac{\texttt{\#}}{\texttt{0 \#}}\right], \\ \left[\frac{\texttt{0 } \checkmark}{\checkmark}\right], \left[\frac{\checkmark \texttt{ 0}}{\checkmark}\right], \left[\frac{\texttt{1 } \checkmark}{\checkmark}\right], \left[\frac{\checkmark \texttt{ 1}}{\checkmark}\right], \left[\frac{\checkmark \texttt{ \# \#}}{\texttt{\#}}\right] \end{array} \right\} \begin{array}{l} \left.\right\} \text{Initial domino } (q_{\text{start}}, \textit{input}) \\ \left.\right\} \text{Transition function } (\delta) \\ \left.\right\} \text{Alphabet } (\Sigma) \\ \left.\right\} \text{Final dominoes } (q_{\text{accept}}) \end{array}$$

- There is no solution for this instance that starts with the initial domino
  - The initial domino creates an imbalance of `A`'s and `#`'s
  - The `A` imbalance is unresolvable: no domino has more `A`'s on top
  - The `#` imbalance is unresolvable: only the final domino has more `#`'s on top, but it cannot be reached unless `Tsà` transitions into the state ✓, which it does not
- The partial match constantly expands the visible portion of the tape

(8) Simulation of `Tsà`

$$\texttt{\# 0 0 A 0 0 \# 0 0 1 A 0 \# 0 0 1 1 A 0 \#}\ \ldots$$

$$\underbrace{\texttt{\# 0 0 A 0 0}}_{\text{Start}}\ \underbrace{\texttt{\# 0 0 1 A 0}}_{\text{Step 1}}\ \underbrace{\texttt{\# 0 0 1 1 A 0}}_{\text{Step 2}}\ \underbrace{\texttt{\# 0 0 1 1 1 A 0}}_{\text{Step 3}}\ \texttt{\#}\ldots$$

- If there were an algorithm to solve arbitrary instances of the PCP, it would imply a solution to the Halting Problem: given an arbitrary Turing Machine and input, just translate it into an instance of the PCP and run the solver

3 The Post Correspondence Problem as phonological optimization

This section demonstrates how to translate the Post Correspondence Problem into an Optimality Theoretic grammar, which returns the input faithfully if the PCP has no solution and otherwise returns the shortest unfaithful candidate that encodes a solution to the PCP. Because the PCP is not computable, it is impossible to determine the output algorithmically.

3.1 Optimality Theory

- Optimality Theory (OT; Prince and Smolensky, 1993/2004) is a constraint-based framework for modeling input-output mappings, most commonly applied to phonology[4]
- An OT grammar comprises three parts, Gen, Con, and Eval, where
  - Gen is a function that *generates* an infinite candidate set from an input[5]
  - Con is an ordered list of *constraints* that either penalize *phonotactic* structures or *unfaithful* mappings
  - Eval is a function that *evaluates* the candidate set, choosing the candidate that best satisfies Con as output

[4]For general introductions to OT, see Kager (1999); McCarthy (2002, 2008a).

[5]This is not a problem in and of itself; finite-state variants of OT evaluate infinite candidate sets.

- Following the principle of *freedom of analysis*, GEN is unrestricted: if it can produce candidates that differ from the input via one application of an operation, if can produce candidates that differ from the input via arbitrarily many applications of an operation (cf. proposals by Łubowicz, 2003; de Lacy, 2007 to bound epenthesis)

$$\textsc{Gen}(/\mathrm{ba}/) = \{[\mathrm{ba}], [\mathrm{baʔ}], [\mathrm{baʔʔ}], [\mathrm{baʔʔʔ}], \ldots, [\mathrm{baʔ}^{\aleph_0}], \ldots\}$$

- Assuming *Correspondence Theory* (McCarthy and Prince, 1994, 1995, 1999), faithfulness constraints are defined over explicit relations between related pairs of strings
  - DEP: Assign one violation for every segment in a candidate that does not have a correspondent in the input.

$$\begin{aligned}\textsc{Dep}(/\mathrm{b}_1\mathrm{a}_2/, [\mathrm{b}_1\mathrm{a}_2]) &= 0\\ \textsc{Dep}(/\mathrm{b}_1\mathrm{a}_2/, [\mathrm{b}_1\mathrm{a}_2\mathrm{ʔ}_3]) &= 1\\ \textsc{Dep}(/\mathrm{b}_1\mathrm{a}_2/, [\mathrm{b}_1\mathrm{a}_2\mathrm{ʔ}_3\mathrm{ʔ}_4]) &= 2\\ &\vdots\end{aligned}$$

- EVAL interprets CON *lexicographically*: for a pair of constraints A, B, where A $\gg$ B, one violation of A is **strictly worse** than **arbitrarily many** violations of B

(9) Unbounded nasalization in Optimality Theory: one violation of AGREE(nasal) or MAX(nasal) is strictly worse than n violations of DEP(nasal)

/w̃a^n/; $n > 0$	AGREE(nasal)	MAX(nasal)	DEP(nasal)
a. [w̃a^n]	W 1		L
b. [wa^n]		W 1	L
→ c. [w̃ã^n]			n

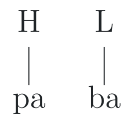
  - AGREE(nasal): Assign one violation for every pair of adjacent segments in a candidate with different specifications of the feature [nasal].
  - MAX(nasal): Assign one violation for every [nasal] feature in the input that does not have a correspondent in the candidate.
  - DEP(nasal): Assign one violation for every [nasal] feature in a candidate that does not have a correspondent in the input.

3.2 Dominoes as autosegmental structures

- (Sequences of) dominoes in the PCP define two sequences of symbols, the top string $\mathrm{t}_1\mathrm{t}_2\ldots\mathrm{t}_m$ and the bottom string $\mathrm{b}_1\mathrm{b}_2\ldots\mathrm{b}_n$. In a solution, $|\mathrm{t}_1\mathrm{t}_2\ldots\mathrm{t}_m| = |\mathrm{b}_1\mathrm{b}_2\ldots\mathrm{b}_n|$ and, for every i, $\mathrm{t}_i = \mathrm{b}_i$. There is no correspondence between the top and bottom strings except for vertical matching

⇒ This two-tiered structure is straightforwardly translated phonologically into an *autosegmental representation* (Goldsmith, 1976)

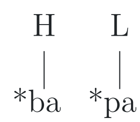
- The top string is translated into *tones* and the bottom into *syllables*
- Because any string can be translated into a binary representation, only two symbols are needed on each tier: {H, L} and {pa, ba}
- Cross-linguistically, high tones avoid syllables with initial voiced stops and low tones avoid syllables with initial voiceless obstruents (Lee, 2008). Well-formed associations link H to [pa] and L to [ba]



- Other associations violate the phonotactic constraints *H/b, *L/p

(10) *H/b: Assign one violation for every association between a high tone and a syllable with a voiced obstruent onset.

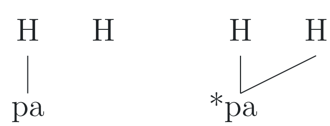
(11) *L/p: Assign one violation for every association between a low tone and a syllable with a voiceless obstruent onset.



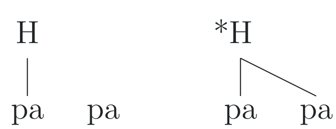
- Mismatches like these correspond to domino sequences where $t_i \neq b_i$

- Autosegmental structures can represent more than simple vertical matches. For example, one tone may be linked to multiple syllables and a syllable may be unlinked to any tone. One-to-one associations between tones and syllables are enforced by standard phonotactic constraints (Yip, 2002)

(12) *Contour: Assign one violation for every syllable associated to more than one tone.



(13) NoLongT: Assign one violation for every tone associated to more than one syllable.



(14) SpecifyT: Assign **one violation** if one or more syllables are not associated to any tone.

$$\begin{array}{c} \text{H} \\ | \\ \text{pa} \end{array} \qquad \begin{array}{c} \text{H} \\ \\ {}^{*}\text{pa} \end{array}$$

(15) *Float: Assign **one violation** if one or more tones are not associated to any syllable.

$$\begin{array}{c} \text{H} \\ | \\ \text{pa} \end{array} \qquad \begin{array}{c} {}^{*}\text{H} \\ \\ \text{pa} \end{array}$$

For reasons discussed below, SpecifyT and *Float are defined as *binary* constraints (Frank and Satta, 1998); they assign either 0 or 1 violations

⇒ In structures that satisfy these six phonotactic constraints, every high tone is associated to exactly one syllable [pa], every low tone is associated to exactly one syllable [ba], every syllable [pa] is associated to exactly one high tone, and every syllable [ba] is associated to exactly one low tone. This is *isomorphic* to a solution to the PCP

- There are equal numbers of tones and syllables
- The ith tone is high if and only if the ith syllable is [pa]

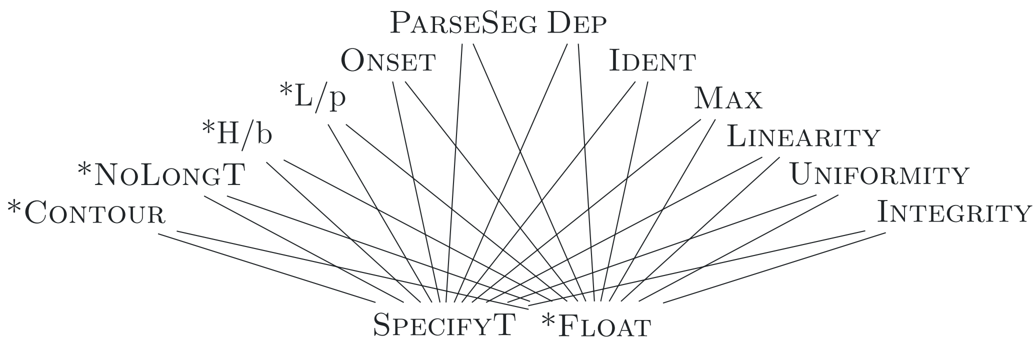
- I assume Gen does not violate the No Crossing Constraint and cannot produce structures like $\begin{array}{cc} \text{H} & \text{L} \\ \times & \\ \text{ba} & \text{pa} \end{array}$ (Goldsmith, 1976; cf. Bagemihl, 1989; Coleman and Local, 1991; Frampton, 2009)

- Henceforth, I refer to dominoes as *morphemes* and as sets of domino types as *lexicons*
  - Morphemes do not contain any underlying associations; all tones are floating
  - The analogy to a lexicon is appropriate: there are languages with morphemes that combine floating features and segments (Wolf, 2005, 2007)

$$\left\{ \left[\frac{0\,1}{0}\right], \left[\frac{0\,1}{1\,1\,0}\right], \left[\frac{0}{1\,0\,0}\right], \left[\frac{1\,0\,0}{1\,0}\right] \right\} \Leftrightarrow \left\{ \begin{array}{c} \text{H L} \\ /\ \text{pa}\ / \end{array}, \begin{array}{c} \text{H L} \\ /\ \text{babapa}\ / \end{array}, \begin{array}{c} \text{H} \\ /\ \text{bapapa}\ / \end{array}, \begin{array}{c} \text{L H H} \\ /\ \text{bapa}\ / \end{array} \right\}$$

3.3 Motivating and restricting repairs

- Morphemes are not phonotactically well-formed; their faithful realizations violate SpecifyT and/or *Float
- Ranking SpecifyT and *Float below the other phonotactic constraints and the eight constraints below (16-23) prevents the grammar from making various changes that would satisfy SpecifyT and *Float



(16) Ident: Assign one violation for every tone or segment in the input whose correspondent in the output is different.

(17) Dep: Assign one violation for every tone or segment in the output without a correspondent in the input.

While rare, tone-driven epenthesis is attested (Gleim, 2019; Rolle and Merrill, 2022)

(18) Max: Assign one violation for every tone or segment in the input without a correspondent in the output.

(19) Linearity: For every pair of tones T_1, T_2 in the input, and every pair of tones T'_1, T'_2 in the output, where T_1 corresponds to T'_1 and T_2 corresponds to T'_2, if T_1 precedes T_2 and T'_2 precedes T'_1, assign one violation. Likewise for segments, mutatis mutandis.

(20) Uniformity: For every pair of tones T_1, T_2 in the input, assign one violation if there is a tone in the output that corresponds to both T_1 and T_2. Likewise for segments, mutatis mutandis.

(21) Integrity: For every pair of tones T_1, T_2 in the output, assign one violation if there is a tone in the input that corresponds to both T_1 and T_2. Likewise for segments, mutatis mutandis.

(22) Onset: Assign one violation for every syllable without an onset.

(23) ParseSeg: Assign one violation for every segment not parsed into a syllable.

(24) Tones and segments cannot be changed, inserted, or deleted

H_1 b_2a_3	Ident	Dep	Max	*H/b	SpecifyT	*Float
→ a. H_1 b_2a_3					1	1
b. H_1 $\vert$ b_2a_3				W 1	L	L
c. L_1 $\vert$ b_2a_3	W 1				L	L
d. H_1 $\vert$ p_2a_3	W 1				L	L
e. L_4 H_1 $\backslash$ $\backslash$ b_2a_3 p_5a_6		W 3			L	L
f. ϵ ϵ			W 3		L	L

(25) Tones and segments cannot be reordered

H_1 L_2 b_3a_4 p_5a_6	Linearity	*H/b	*L/p	SpecifyT	*Float
→ a. H_1 L_2 b_3a_4 p_5a_6				1	1
b. H_1 L_2 $\vert$ $\vert$ b_3a_4 p_5a_6		W 1	W 1	L	L
c. L_2 H_1 $\vert$ $\vert$ b_3a_4 p_5a_6	W 1			L	L
d. H_1 L_2 $\vert$ $\vert$ p_5a_4 b_3a_6	W 3			L	L

(26) Tones and segments cannot be fused or fissioned

H_1 H_2 L_3 / p_4a_5 b_6a_7 b_8a_9	UNIFORMITY	INTEGRITY	LINEARITY	SPECIFYT	*FLOAT
→ a. H_1 H_2 L_3 / p_4a_5 b_6a_7 b_8a_9				1	1
→ b. H_1 H_2 L_3 / p_4a_5 b_6a_7 b_8a_9				1	1
c. $H_{1,2}$ L_3 / p_4a_5 $b_{6,7}a_{8,9}$	W 3			L	L
d. H_1 H_2 L_3 L_3 / p_4a_5 p_4a_5 b_6a_7 b_8a_9		W 3	W 1	L	L

(27) Segments must be parsed into CV syllables

H_1 L_2 / p_3a_4 p_5a_6	ONSET	PARSESEG	SPECIFYT	*FLOAT
→ a. H_1 L_2 / $.p_3a_4.p_5a_6.$			1	1
→ b. H_1 L_2 / $.p_3a_4.p_5a_6.$			1	1
c. H_1 L_2 / $.p_3a_4p_5.a_6.$	W 1		L	L
d. H_1 L_2 / $.p_3a_4.<p_5>.a_6.$	W 1	W 1	L	L

- If GEN can insert or copy entire morphemes, then SPECIFYT and *FLOAT can motivate changes, but, **only if** there would be one-to-one correspondences between high tones and [pa] syllables and low tones and [ba] syllables
  - SPECIFYT and *FLOAT are binary constraints which are either satisfied or vio-

lated. Partial improvement is impossible

3.4 Two constructions that model the PCP

3.4.1 Lexical insertion

- Xu (2007, 2011); Wolf (2008, 2015); Rolle (2020) have argued that Gen can **insert freely from the lexicon**

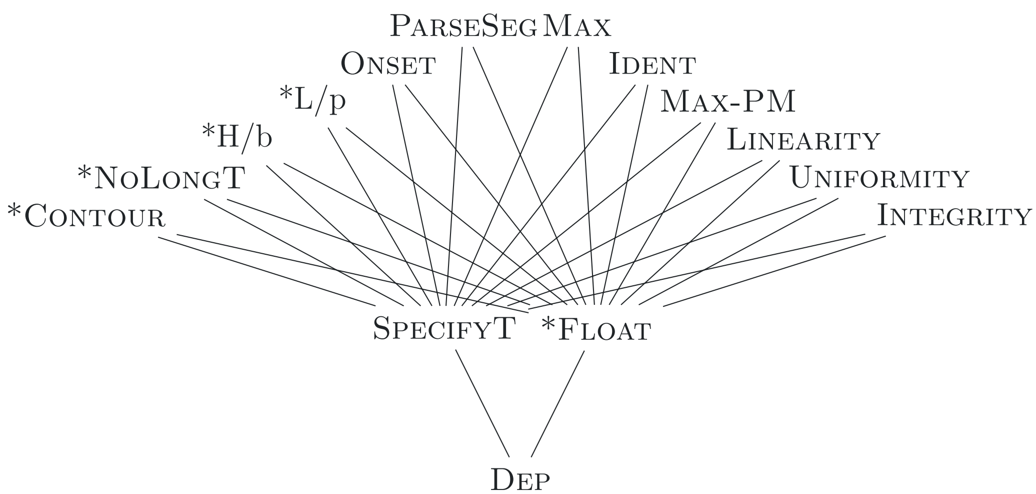
$$\textsc{Gen}(/\mathrm{b}_1\mathrm{a}_2/) = \{[\mathrm{b}_1\mathrm{a}_2], [\mathrm{b}_1\mathrm{a}_2\mathrm{d}_3\text{ɔ}_4\mathrm{g}_5], [\mathrm{b}_1\mathrm{a}_2\underline{\mathrm{d}_3\text{ɔ}_4\mathrm{g}_5}], \ldots\}$$

- I assume that lexical insertion and ordinary phonological insertion both violate Dep. The constraint Max-PM distinguishes them (Walker and Feng, 2004)

(28) Max-PM: Assign one violation for every tone/segment in the output that is not associated to a morpheme in the output.

$$\textsc{Max-PM}(/\mathrm{b}_1\mathrm{a}_2/, [\mathrm{b}_1\mathrm{a}_2\mathrm{d}_3\text{ɔ}_4\mathrm{g}_5]) = 3$$
$$\textsc{Max-PM}(/\mathrm{b}_1\mathrm{a}_2/, [\mathrm{b}_1\mathrm{a}_2\underline{\mathrm{d}_3\text{ɔ}_4\mathrm{g}_5}]) = 0$$

- Assuming *consistency of exponence*, Gen cannot change morphological affiliation
- Under a lexical insertion approach, OT grammars are equipped with lexicons
- Consider a grammar with the lexicon {/papapa/, /H H/, /H/} and a Con, where



- The output for any mono-morphemic input is

```
H  H  H     H  H  H     H  H  H
|  |  |     |  |  |     |  |  |
pa pa pa ,  pa pa pa ,  pa pa pa
```

(29) SPECIFYT and *FLOAT motivate lexical insertion

H	MAX-PM	SPECIFYT	*FLOAT	DEP
a. H			W 1	L
b. H | pa	W 2			L 2
c. H | pa pa pa		W 1		L 6
→ d. H H H | | | pa pa pa				8
→ e. H H H | | | pa pa pa				8
→ f. H H H | | | pa pa pa				8
g. H H H H H H | | | | | | pa pa pa pa pa pa				W 15

- GEN inserts morphemes *faithfully* from the lexicon; it cannot insert the string [pabapa]
  - Otherwise, additional faithfulness constraints to regulate faithfulness to the lexicon would be required, generalizing a proposal by Landman (2002)
  - Or output-output correspondence (Benua, 1997) could be used provided there is a context in which the morphemes surface faithfully. One can be easily generated artificially with a high ranked, lexically-indexed DEP (Pater, 2007, 2010)
  - For ease of presentation, I assume faithful lexical insertion
- As discussed above, no other change, such as deleting the underlying tone, is optimal
- In general, given an *arbitrary lexicon*, an OT grammar with this constraint ranking will do one of two things
  - Return the *faithful* candidate, with violation(s) of SPECIFYT and *FLOAT (30)
  - Return the *shortest* candidates that satisfy both SPECIFYT and *FLOAT (31)

(30) The grammar returns the faithful candidate

/x/	SpecifyT	*Float	Dep
→ a. [x]	(1)	(1)	
b. [yxz]	(1)	(1)	W |yz|

(31) The grammar returns one or more unfaithful candidates

/x/	SpecifyT	*Float	Dep
a. [x]	(W 1)	(W 1)	L
→ b. [yxz]			|yz|

- The latter occurs when it is possible to create one-to-one associations between high tones and [pa]'s and low tones and [ba]'s, and the former occurs otherwise

⇒ Because this is equivalent to solving an arbitrary instance of the PCP (it is straightforward to translate an instance of the PCP into one of these lexicons), it is impossible to determine algorithmically what the grammar will do.

⇒ Therefore, **it is impossible to determine algorithmically the output for an arbitrary OT grammar**, and I have shown that OT is not computable

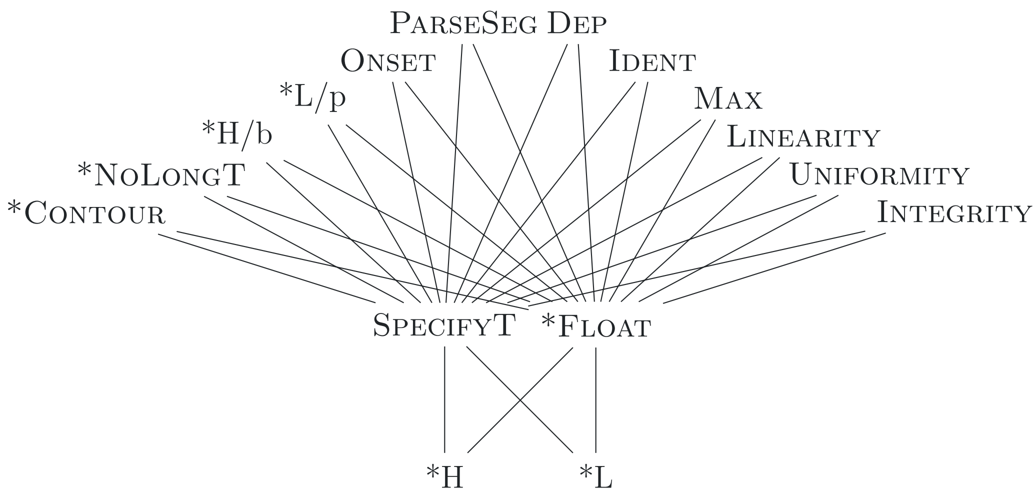
3.4.2 Reduplication

- Instead of copying from the lexicon, Gen is also free to copy from candidates, modeling the PCP as **non-reduplicative copying** (Gafos, 1998; Kawu, 2000; Yu, 2003, 2005, 2007; Kawahara, 2004; Elfner and Kimper, 2008); i.e., copying that fulfills a phonological function, rather than exponing some morpheme
- Under *Base-Reduplicant Correspondence Theory*, copying per se does not violate any constraints. To limit the size of outputs, the constraints *H and *L, which penalize high and low tones, respectively, take the place of Dep
  - These act like general constraints on phonological structure (Gouskova, 2003)
  - Without correspondence, as in Minimal Reduplication (Saba Kirchner, 2010), it is not obvious how to force the grammar to copy entire morphemes

(32) *H: Assign one violation for every high tone.

(33) *L: Assign one violation for every low tone.

- The ranking below disallows operations other than copying



- To allow Gen access to the full set of morphemes, they are all included in the input. Consider the simple lexicon {/papa/, /H/}
  - This instantiates an instance of the *Modified Post Correspondence Problem*, which asks for a solution given a particular starting point
  - Unsurprisingly, the MPCP is not computable; see Harrison (1978) for discussion

(34) SpecifyT and *Float motivate reduplication

H , pa pa	SpecifyT	*Float	*H	*L
a. H_1 | pa pa	W 1		L 1	
→ b. H_1 H_1 | | pa pa			2	

- Note that copying does not violate Dep: every high tone in the output corresponds to the high tone in the input

- To avoid partial copying, there are two options
  - Input-reduplicant faithfulness (McCarthy and Prince, 1999; Fitzgerald, 2000; Sloos and van Engelenhoven, 2011)
  - Base-reduplicant faithfulness augmented with anchor constraints that require the left/right edges of the base of reduplicants to correspond to left/right edges of morphological constituents (Shaw, 2005; Haugen, 2009)

⇒ With either set of constraints ranked above SPECIFYT and *FLOAT, the behavior of the grammar is identical to the lexical insertion construction: it returns either the faithful candidate (35) or the shortest unfaithful candidates with one-to-one associations (36)

(35) The grammar returns the faithful candidate

/x/	SPECIFYT	*FLOAT	*H	*L
→ a. [x]	(1)	(1)	i	j
b. [yxz]	(1)	(1)	(W $i+i'$)	(W $j+j'$)

(36) The grammar returns one or more unfaithful candidates

/x/	SPECIFYT	*FLOAT	*H	*L
a. [x]	(W 1)	(W 1)	(L) i	(L) j
→ b. [yxz]			($i+i'$)	($j+j'$)

4 Discussion

- This talk has demonstrated that **Optimality Theory is not computable**
  - This means that it is impossible to write an algorithm that takes an arbitrary OT grammar and input and determine the output
- As such, OT joins the ranks of other constraint-based formalisms that have been shown not to be computable including unrestricted variants of LFG (Kaplan and Bresnan, 1982), Attribute-Value Grammars (Johnson, 1988), HPSG (Kepser, 2004), and Unification Grammars (Francez and Wintner, 2012) (see Kaplan and Wedekind, forthcoming; Przepiórkowski, forthcoming for discussion). It is also possible to demonstrate that the recursive definition of targeted constraints (Wilson, 2000, 2001, 2003) is not computable
- The result depends on **strictly ordered constraints** and **freedom of analysis**
- With weighted constraints, as in Harmonic Grammar (Legendre et al., 1990), the cumulative penalty from an large number of operations eventually outweighs the benefit (Pater et al., 2007; Farris-Trimble, 2008a,b, 2010; Pater, 2009a,b, 2012, 2016; O'Hara, 2016)

(37) Bounded nasalization in Harmonic Grammar

/w̃a^n/; $n > 0$	AGREE(nasal) x	MAX(nasal) y	DEP(nasal) z	$\mathscr{H}$
(→) a. w̃a^n	-1			$-x$
b. w̃ã^n		-1		$-y$
(→) c. wa^n			$-n$	$-nz$

- Candidate (b) will always lose to candidate (a) if $y > x$. However, there is no weighting of x, y, z that will *always* choose candidate (c) over candidate (a). Eventually the accumulated violations of DEP(nasal) will exceed the violation of AGREE(nasal)[6]

$$\tilde{\text{w}}\text{a}^n \mapsto \begin{cases} \tilde{\text{w}}\text{a}^n & \text{if } n > \frac{x}{z} \\ \tilde{\text{w}}\tilde{\text{a}}^n & \text{if } n < \frac{x}{z} \end{cases}$$

- Crucially, this assumes that all weights are negative (cf. Smolensky, 1992)

• With a restricted GEN, as in Harmonic Serialism (Prince and Smolensky, 1993/2004; McCarthy, 2000, 2006, 2008b, 2016) the violations of SPECIFY must be strictly decreasing with each domino insertion, bounding the length of possible solutions

  - However, there are variants of HS that can fall into infinite loops (Kimper, 2016; Lamont, 2022a; Müller, 2020)

• This result should give us pause as phonologists. While OT has in many ways been a dominant theory for the last thirty years, it has undesirable properties. **How can we salvage its benefits (conspiracies, typologies, etc.) without sacrificing restrictiveness?**

[6]This property makes it possible to show that Harmonic Grammar is finite-state with string constraints like AGREE(nasal) (work in prep).

References

Bagemihl, Bruce (1989). The Crossing Constraint and 'backwards languages'. *Natural Language & Linguistic Theory* **7**(4). 481–549.

Benua, Laura (1997). *Transderivational identity: Phonological relations between words*. Ph.D. dissertation, University of Massachusetts Amherst.

Brady, Allen H. (1983). The determination of the value of Rado's noncomputable function $\sigma(k)$ for four-state Turing machines. *Mathematics of Computation* **40**(162). 647–665.

Chomsky, Noam and Morris Halle (1968). *The sound pattern of English*. New York: Harper & Row.

Church, Alonzo (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics* **58**(2). 345–363.

Coleman, John and John Local (1991). The "No Crossing Constraint" in Autosegmental Phonology. *Linguistics and Philosophy* **14**(3). 295–338.

de Lacy, Paul (2007). Fredom, interpretability, and the loop. In Sylvia Blaho, Patrik Bye and Martin Krämer (eds.), *Freedom of analysis?*, Berlin and New York: Mouton de Gruyter. 175–202.

Eisner, Jason (1997a). Efficient generation in Primitive Optimality Theory. In Philip R. Cohen and Wolfgang Wahlster (eds.), *Proceedings of the 35th Annual ACL and 8th European ACL*. Association for Computational Linguistics. 313–320.

Eisner, Jason (1997b). What constraints should OT allow? Paper presented at LSA 71. Available at `http://roa.rutgers.edu/article/view/215`.

Eisner, Jason (2000). Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18th International Conference on Computational Linguistics*. The Association for Computational Linguistics. 257–263.

Eisner, Jason (2002). Comprehension and compilation in Optimality Theory. In Pierre Isabelle, Eugene Charniak and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics. 56–63.

Elfner, Emily and Wendell Kimper (2008). Reduplication without RED: Evidence from *diddly*-infixation. In Natasha Abner and Jason Bishop (eds.), *Proceedings of the 27th West Coast Conference on Formal Linguistics*. Somerville, MA: Cascadilla Proceedings Project. 150–158.

Farris-Trimble, Ashley W. (2008a). *Cumulative faithfulness effects in phonology*. Ph.D. dissertation, Indiana University.

Farris-Trimble, Ashley W. (2008b). Cumulative faithfulness effects: Opaque or transparent? In Ashley W. Farris-Trimble and Daniel A. Dinnsen (eds.), *IUWPL6: Phonological opacity effects in Optimality Theory*, Bloomington, IN: IULC Publications. 119–145.

Farris-Trimble, Ashley W. (2010). Nothing is better than being unfaithful in multiple ways. In Max Bane, Juan Bueno, Tommy Grano, April Grotberg and Yaron McNabb (eds.), *Proceedings from the Annual Meeting of the Chicago Linguistic Society 44*. Chicago: Chicago Linguistic Society. 79–93.

Fitzgerald, Colleen M. (2000). Vowel hiatus and faithfulness in Tohono O'odham reduplication. *Linguistic Inquiry* **31**(4). 713–722.

Frampton, John (2009). *Distributed reduplication*, vol. 52 of *Linguistic Inquiry Monographs*. Cambridge, MA and London: The MIT Press.

Francez, Nissim and Shuly Wintner (2012). *Unification grammars*. Cambridge: Cambridge University Press.

Frank, Robert and Giorgia Satta (1998). Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics* **24**(2). 307–315.

Gafos, Diamandis (1998). Eliminating long-distance consonantal spreading. *Natural Language & Linguistic Theory* **16**(2). 223–278.

Gerdemann, Dale and Mans Hulden (2012). Practical finite state Optimality Theory. In Iñaki Alegria and Mans Hulden (eds.), *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*. The Association for Computational Linguistics. 10–19.

Gleim, Daniel (2019). A feeding Duke-of-York interaction of tone and epenthesis in Arapaho. *Glossa: a journal of general linguistics* **4**(1). 97.

Goldsmith, John (1976). *Autosegmental phonology*. Ph.D. dissertation, Massachusetts Institute of Technology.

Gouskova, Maria (2003). *Deriving economy: Syncope in Optimality Theory*. Ph.D. dissertation, University of Massachusetts Amherst.

Harlow, Ray (2007). *Māori: A linguistic introduction*. Cambridge: Cambridge University Press.

Harrison, Michael A. (1978). *Introduction to formal language theory*. Addison-Wesley Series in Computer Science.

Haugen, Jason D. (2009). What is the base for reduplication? *Linguistic Inquiry* **40**(3). 505–514.

Heinz, Jeffrey (2018). The computational nature of phonological generalizations. In Larry M. Hyman and Frans Plank (eds.), *Phonological typology*, Berlin: De Gruyter Mouton. 126–195.

Heinz, Jeffrey, Gregory M. Kobele and Jason Riggle (2009). Evaluating the complexity of Optimality Theory. *Linguistic Inquiry* **40**(2). 277–288.

Hopcroft, John E. (1984). Turing machines. *Scientific American* **250**. 86–98.

Hopcroft, John E., Rajeev Motwani and Jeffrey D. Ullman (2008). *Introduction to automata theory, languages, and computation*. Pearson, 3 edition.

Hulden, Mans (2017). Formal and computational verification of phonological analyses. *Phonology* **34**(2). 407–435.

Idsardi, William J. (2006). A simple proof that Optimality Theory is computationally intractable. *Linguistic Inquiry* **37**(2). 271–275.

Johnson, C. Douglas (1972). *Formal aspects of phonological description*. The Hague: Mouton.

Johnson, Mark (1988). *Attribute-value logic and the theory of grammar*, vol. 16 of *CSLI Lecture Notes*. Stanford, CA: Center for the Study of Language and Information.

Kager, René (1999). *Optimality Theory*. Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.

Kaplan, Ronald M. and Joan Bresnan (1982). Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan (ed.), *The mental representation of grammatical representation*, Cambridge, MA: The MIT Press. 173–281.

Kaplan, Ronald M. and Martin Kay (1994). Regular models of phonological rule systems. *Computational Linguistics* **20**(3). 331–378.

Kaplan, Ronald M. and Jürgen Wedekind (forthcoming). Formal and computational properties of LFG. In Mary Dalrymple (ed.), *The handbook of Lexical Functional Grammar*, Berlin: Language Science Press. Empirically Oriented Theoretical Morphology and Syntax. `https://langsci-press.org/catalog/view/312/3316/2510-2`.

Kawahara, Shigeto (2004). Locality in echo epenthesis: Comparison with reduplication. In Keir Moulton and Matthew Wolf (eds.), *Proceedings of NELS 34*. Amherst, MA: Graduate Linguistics Students Association. vol. 2, 295–310.

Kawu, Ahmadu Ndanusa (2000). Structural markedness and nonreduplicative copying. In Masako Hirotani, Andries Coetzee, Nancy Hall and Ji-yung Kim (eds.), *Proceedings of NELS 30*, Amherst, MA: Graduate Linguistics Students Association. vol. 1, 377–388.

Kepser, Stephan (2004). On the complexity of RSRL. *Electronic Notes in Theoretical Computer Science* **53**. 146–162.

Kimper, Wendell (2016). Positive constraints and finite goodness in Harmonic Serialism. In John J. McCarthy and Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*, Sheffield: Equinox Publishing. 221–235.

Koskenniemi, Kimmo (1983). *Two-level morphology: A general computational model for word-form recognition and production*, vol. 11 of *Publications*. Helsinki: University of Helsinki.

Lamont, Andrew (2021). Optimizing over subsequences generates context-sensitive languages. *Transactions of the Association for Computational Linguistics* **9**. 528–537.

Lamont, Andrew (2022a). *Directional Harmonic Serialism*. Ph.D. dissertation, University of Massachusetts Amherst.

Lamont, Andrew (2022b). Optimality theory implements complex functions with simple constraints. *Phonology* **38**(4). 729–740.

Landman, Meredith (2002). Morphological contiguity. In Angela C. Carpenter, Andries W. Coetzee, and Paul de Lacy (eds.), *UMOP 26: Papers in Optimality Theory II*, Amherst, MA: Graduate Linguistics Students Association. 141–169.

Lee, Seunghun Julio (2008). *Consonant-tone interaction in Optimality Theory*. Ph.D. dissertation, Rutgers, The State University of New Jersey.

Legendre, Géraldine, Yoshiro Miyata and Paul Smolensky (1990). Harmonic Grammar: a formal multi-level connectionist theory of linguistic well-formedness: an application. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum. 884–891.

Lin, Shen and Tibor Radó (1965). Computer studies of Turing machine problems. *Journal of the Association for Computing Machinery* **12**(2). 196–212.

Lorentz, Richard J. (2001). Creating difficult instances of the Post Correspondence Problem. In Tony Marsland and Ian Frank (eds.), *Computers and games*, Berlin: Springer. vol. 2063 of *Lecture Notes in Computer Science*, 214–228.

Łubowicz, Anna (2003). *Contrast preservation in phonological mappings*. Ph.D. dissertation, University of Massachusetts Amherst.

McCarthy, John J. (2000). Harmonic Serialism and Parallelism. In Masako Hirotani, Andries Coetzee, Nancy Hall and Ji-yung Kim (eds.), *Proceedings of NELS 30*. Amherst, MA: Graduate Linguistics Students Association. vol. 2, 501–524.

McCarthy, John J. (2002). *A thematic guide to Optimality Theory*. Research Surveys in Linguistics. Cambridge: Cambridge University Press.

McCarthy, John J. (2006). Restraint of analysis. In Eric Baković, Junko Itô and John J. McCarthy (eds.), *Wondering at the natural fecundity of things: Essays in honor of Alan Prince*, Santa Cruz, CA: Linguistics Research Center. 195–219.

McCarthy, John J. (2008a). *Doing Optimality Theory*. Malden, MA: Blackwell Publishing.

McCarthy, John J. (2008b). Restraint of analysis. In Sylvia Blaho and Patrik Bye (eds.), *Freedom of analysis?*, Berlin: De Gruyter. 203–231.

McCarthy, John J. (2016). The theory and practice of Harmonic Serialism. In John J. McCarthy and Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*, Sheffield: Equinox Publishing. 47–87.

McCarthy, John J. and Alan Prince (1994). The emergence of the unmarked: Optimality in prosodic morphology. In Mercè Gonzàlez (ed.), *Proceedings of NELS 24*, Amherst, MA: Graduate Linguistics Students Association. vol. 2, 333–379.

McCarthy, John J. and Alan Prince (1995). Faithfulness and reduplicative identity. In Jill Beckman, Suzanne Urbanczyk and Laura Walsh Dickey (eds.), *Papers in Optimality Theory*, Amherst, MA: Graduate Linguistics Students Association. 249–384.

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

Müller, Gereon (2020). *Inflectional morphology in harmonic serialism*. Advances in Optimality Theory. Sheffield, UK and Bristol CT: Equinox Publishing Ltd.

O'Hara, Charlie (2016). Harmony in Harmonic Grammar by reevaluating faithfulness. In Christopher Hammerly and Brandon Prickett (eds.), *Proceedings of NELS 46*. Amherst, MA: Graduate Linguistics Students Association. vol. 3, 71–84.

Partee, Barbara H., Alice ter Meulen and Robert E. Wall (1990). *Mathematical methods in linguistics*, vol. 30 of *Studies in Linguistics and Philosophy*. Dordrecht / Boston / London: Kluwer Academic Publishers.

Paster, Mary (2019). Phonology counts. *Radical: A Journal of Phonology* **1**. 1–61.

Pater, Joe (2007). The locus of exceptionality: Morpheme-specific phonology as constraint indexation. In Leah Bateman, Michael O'Keefe, Ehren Reilly and Adam Werle (eds.), *Papers in Optimality Theory III*, Amherst, MA: Graduate Linguistics Students Association. University of Massachusetts Occasional Papers, 259–296.

Pater, Joe (2009a). Review of the harmonic mind: From neural computation to optimality-theoretic grammar. *Phonology* **26**(1). 217–226.

Pater, Joe (2009b). Weighted constraints in generative linguistics. *Cognitive Science* **33**(6). 999–1035.

Pater, Joe (2010). Morpheme-specific phonology: Constraint indexation and inconsistency resolution. In Steve Parker (ed.), *Phonological argumentation: Essays on evidence and motivation*, London: Equinox Publishing. 123–154.

Pater, Joe (2012). Serial Harmonic Grammar and Berber syllabification. In Toni Borowsky, Shigeto Kawahara, Mariko Sugahara and Takahito Shinya (eds.), *Prosody matters: Essays in honor of Elisabeth Selkirk*, Sheffield: Equinox Publishing. 43–72.

Pater, Joe (2016). Universal grammar with weighted constraints. In John J. McCarthy and Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*, Sheffield: Equinox Publishing. 1–46.

Pater, Joe, Rajesh Bhatt and Christopher Potts (2007). Linguistic optimization. Unpublished manuscript. University of Massachusetts. Available at `http://roa.rutgers.edu/article/view/954`.

Post, Emil L. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society* **52**(4). 264–268.

Prince, Alan and Paul Smolensky (1993/2004). *Optimality Theory: Constraint interaction in generative grammar*. Malden, MA: Blackwell Publishing.

Przepiórkowski, Adam (forthcoming). LFG and Head-Driven Phrase Structure Grammar. In Mary Dalrymple (ed.), *The handbook of Lexical Functional Grammar*, Berlin: Language Science Press. Empirically Oriented Theoretical Morphology and Syntax. `https://langsci-press.org/catalog/view/312/3337/2439-1`.

Radó, Tibor (1962). On non-computable functions. *Bell System Technical Journal* **41**(3). 877–884.

Riggle, Jason (2004). *Generation, recognition, and learning in finite state Optimality Theory*. Ph.D. dissertation, University of California Los Angeles.

Rolle, Nicholas (2020). In support of an OT-DM model: Evidence from clitic distribution in Degema serial verb constructions. *Natural Language & Linguistic Theory* **38**(1). 201–259.

Rolle, Nicholas and John T. M. Merrill (2022). Tone-driven epenthesis in Wamey. *Phonology* **39**(1). 113–158.

Saba Kirchner, Jesse (2010). *Minimal reduplication*. Ph.D. dissertation, University of California, Santa Cruz.

Shaw, Patricia A. (2005). Non-adjacency in reduplication. In Bernhard Hurch (ed.), *Studies on reduplication*, Berlin and New York: Mouton de Gruyter. Empirical Approaches to Language Typology, 161–210.

Sipser, Michael (2013). *Introduction to the theory of computation*. Boston, MA: Cengage Learning, 3 edition.

Sloos, Marjoleine and Aone van Engelenhoven (2011). Input-Reduplicant correspondence in Leti. In Rick Nouwen and Marion Elenbaas (eds.), *Linguistics in the Netherlands 2011*, Amsterdam: John Benjamins Publishing Company. vol. 28 of *AVT Publications*, 112–124.

Smolensky, Paul (1992). Harmonic Grammars for formal languages. In Stephen José Hanson, Jack D. Cowan and C. Lee Giles (eds.), *Advances in neural information processing systems 5*. San Francisco: Morgan Kaufmann Publishers Inc. 847–854.

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* **s2-42**(1). 230–265.

Walker, Rachel and Bella Feng (2004). A ternary model of morphology-phonology correspondence. In Vineeta Chand, Ann Kelleher, Angelo J. Rodríguez and Benjamin Schmeiser (eds.), *Proceedings of WCCFL 23*. Somerville, MA: Cascadilla Press. 787–800.

Wilson, Colin (2000). *Targeted constraints: An approach to contextual neutralization in Optimality Theory*. Ph.D. dissertation, The Johns Hopkins University.

Wilson, Colin (2001). Consonant cluster neutralisation and targeted constraints. *Phonology* **18**(1). 147–197.

Wilson, Colin (2003). Analyzing unbounded spreading with constraints: marks, targets and derivations. Unpublished manuscript. UCLA.

Wolf, Matthew (2005). An autosegmental theory of quirky mutations. In John Alderete, Chung-hye Han and Alexei Kochetov (eds.), *Proceedings of the 24th West Coast Conference on Formal Linguistics*. Somerville, MA: Cascadilla Proceedings Project. 370–378.

Wolf, Matthew (2007). For an autosegmental theory of mutation. In Leah Bateman, Michael O'Keefe, Ehren Reilly and Adam Werle (eds.), *Papers in Optimality Theory III*, Amherst, MA: Graduate Linguistics Students Association. University of Massachusetts Occasional Papers, 315–404.

Wolf, Matthew (2008). *Optimal interleaving: Serial phonology-morphology interaction in a constraint-based model*. Ph.D. dissertation, University of Massachusetts Amherst.

Wolf, Matthew (2015). Lexical insertion occurs in the phonological component. In Eulàlia Bonet, Maria-Rosa Lloret and Joan Mascaró Altimiras (eds.), *Understanding allomorphy: Perspectives from optimality theory*, Sheffield: Equinox. 361–407.

Xu, Zheng (2007). *Inflectional morphology in Optimality Theory*. Ph.D. dissertation, Stony Brook University.

Xu, Zheng (2011). Optimality Theory and morphology. *Language and Linguistics Compass* **5**(7). 424–508.

Yip, Moira (2002). *Tone*. Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.

Yu, Alan C. L. (2003). Reduplication in English Homeric infixation. In Keir Moulton and Matthew Wolf (eds.), *Proceedings of NELS 34*, Amherst, MA: Graduate Linguistics Students Association. vol. 2, 619–633.

Yu, Alan C. L. (2005). Toward a typology of compensatory reduplication. In John Alderete, Chung-hye Han and Alexei Kochetov (eds.), *Proceedings of the 24th West Coast Conference on Formal Linguistics*. Somerville, MA: Cascadilla Proceedings Project. 397–405.

Yu, Alan C. L. (2007). *A natural history of infixation*. Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.