Harmonic Serialism and Finite-State Optimality Theory

Yiding Hao
Department of Linguistics
Yale University
New Haven, CT, USA
yiding.hao@yale.edu

Abstract

This paper presents a new finite-state model of Optimality Theory (OT). In this model, two assumptions are imposed on the OT framework. Firstly, I adopt the Harmonic Serialism version of OT, in which output forms are derived from input forms via a series of incremental changes. Secondly, constraints are assumed to be *strictly local* in the sense that each markedness constraint specifies a set of banned sequences, each occurrence of which is penalized. I show that these two assumptions suffice to reduce the power of OT to rational relations.

1 Introduction

The seminal paper of Frank and Satta (1998) showed that grammars in the Optimality Theory (OT) framework can generate non-rational relations, but that a finite-state implementation is possible if each grammar specifies a bound on the number of violations that can be assigned by a constraint. Since then, various finite-state approximations of OT have been developed that achieve rationality by modifying the framework to reduce its computational power. Karttunen (1998), for example, implemented Frank and Satta's violation-bounded proposal by composing constraints using an operation called *lenient composition*. Improving upon this, Gerdemann and Van Noord (2000) and Gerdemann and Hulden (2012) developed a technique called *matching* that compares candidates based on the locations where violations are assigned. Eisner (2000) and Eisner (2002) propose a model called *directional OT* that prefers candidates whose violations are incurred as close as possible to the left or right boundary of the string. Finally, Riggle (2004) presents an algorithm, called the *Optimality Transducer Construction Algorithm* (OTCA), that takes an OT grammar as input and produces a finite-state transducer that correctly computes the grammar if and only if the grammar defines a rational relation.

In this paper, I present a new formalization of OT that limits the generative capacity of OT in two ways. Firstly, I adopt the *Harmonic Serialism* (HS) version of OT. Whereas the standard version of OT simply maps each input to the candidate that best satisfies a sequence of constraints, HS produces outputs by effecting a series of incremental changes to the input. Secondly, I assume that all constraints are *strictly local*, in the sense that each constraint designates a set of marked sequences and assigns a violation for each occurrence of a marked sequence. I show that these two assumptions suffice to reduce the power of OT to rational relations.

The structure of this paper is as follows. In Section 2, I introduce technical definitions and terminology used in this paper. Section 3 motivates the use of strictly local constraints and HS as restrictions on OT. Section 4 presents a formalization of HS, and Section 5 presents a finite-state model of HS. Section 6 concludes.

2 Preliminaries

As usual, $\mathbb{Z}$ is the set of integers, and $\mathbb{N} \subseteq \mathbb{Z}$ is the set of non-negative integers. Unless otherwise specified, Σ denotes a finite alphabet, Σ^* denotes the set of all strings over Σ, and Σ^+ denotes the set of all nonempty strings over Σ. The special symbols $\rtimes$ and $\ltimes$ are assumed not to be elements of Σ. When used, these symbols represent the left and right boundaries of a string, respectively. The *length* of a string x is denoted $|x|$, and λ denotes the *empty string*, the string of length 0. Symbols from Σ are identified with strings of length 1, and

Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing, pages 20–29,
Umeå, Sweden, 4–6 September 2017.
https://doi.org/10.18653/v1/W17-4003

for any k, Σ^k denotes the set of strings of length k over Σ. For any strings $a, b \in \Sigma^*$, ab is the concatenation of a and b. If $A, B \subseteq \Sigma^*$, then $AB = \{ab | a \in A, b \in B\}$. If $a \in \Sigma^*$ and $B \subseteq \Sigma^*$, then $aB = \{a\}B$ and $Ba = B\{a\}$. A string a is a *substring* or a *subsequence* of b if one can write $b = lar$ for some $l, r \in \Sigma^*$.

For any sets A and B, $A \times B$ denotes the set $\{\langle a, b\rangle | a \in A, b \in B\}$. A *relation over A and B* is a subset $R \subseteq A \times B$. The *transitive closure* of a relation $R \subseteq A \times A$ is the smallest relation $\hat{R} \subseteq A \times A$ such that $R \subseteq \hat{R}$ and if $\langle x, y\rangle, \langle y, z\rangle \in \hat{R}$, then $\langle x, z\rangle \in \hat{R}$. For any relations R and S, the *composition* of R and S is the relation

$$R \circ S = \{\langle x, z\rangle | \exists y[\langle x, y\rangle \in S, \langle y, z\rangle \in R]\}.$$

A *finite-state transducer* is a 6-tuple $T = \langle Q, \Sigma, \Gamma, I, F, \delta\rangle$, where

- Q is a finite set of *states*;
- Σ is an alphabet called the *input alphabet*;
- Γ is an alphabet called the *output alphabet*;
- $I \subseteq Q$ is the set of *initial states*;
- $F \subseteq Q$ is the set of *final states*; and
- $\delta \subseteq Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q$ is the *transition relation*.

The *extended transition relation* of T is the smallest set $\hat{\delta} \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$ such that $\delta \subseteq \delta^*$; for every $q \in Q$, $\langle q, \lambda, \lambda, q\rangle \in \hat{\delta}$; and if $\langle q, x, y, r\rangle \in \hat{\delta}$ and $\langle r, a, b, s\rangle \in \delta$, then $\langle q, xa, yb, s\rangle \in \hat{\delta}$. The *behavior* of T is the relation $[T]$ such that $\langle x, y\rangle \in T$ if and only if for some $q \in I$ and $r \in F$, $\langle q, x, y, r\rangle \in \hat{\delta}$. A relation is *rational* if it is the behavior of a finite-state transducer.

A language L is *k-strictly local* if, for some set $S \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$, $\rtimes L \ltimes$ is the set of strings such that every substring of length k is in S.

3 Restrictions on Constraints

Finite-state models of OT have typically achieved finite-stateness by imposing limitations on the power of constraints. The standard assumption, due to Ellison (1994), is that markedness constraints are finite-state mappings from strings to sequences of violation marks. The violation bound of Frank and Satta (1998) and Karttunen (1998) and the directional evaluation mechanism of Eisner (2000) and Eisner (2002) both refine the class of possible constraints to a strict subclass of rational functions from Σ^* to $\mathbb{N}$.

aaabb	DEP	ID	AGR	MAX
a. *aaabb*			$*!$	
b. *aaacbb*	$*!$			
c. *aaaaa*		$*!$		
☞ d. *aaa*				$**$
e. *bb*				$***!$

Figure 1: Tableau for a non-rational OT grammar

In this paper, I propose to restrict constraints to a proper subclass of rational functions motivated by recent work on the *subregular hierarchy*. The subregular hierarchy consists of subclasses of regular languages and rational relations that characterize empirically attested patterns in phonology. Among these are the *strictly local* languages of McNaughton and Papert (1971), the *tier-based strictly local* languages of Heinz et al. (2011), and the *input strictly local* and *output strictly local* functions of Chandlee (2014). All four subclasses formalize the observation that markedness constraints in phonology generally designate a set of undesirable sequences as *marked*, and penalize strings that contain such sequences. Based on this intuition, I define a class of constraints called *strictly local constraints*.

Definition 1. A *strictly local constraint* is a function $c : \Sigma^* \to \mathbb{N}$ such that for some finite set $S_c \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^*$, $c(x)$ is the number of unique decompositions $\rtimes x \ltimes = wyz$ such that $y \in S_c$. We say that c *bans* the sequence y if $y \in S_c$.

A strictly local constraint is a constraint of the form "assign one violation for every instance of s_1, s_2, ..., or s_n," where each s_i is a marked sequence. It can be easily shown that strictly local constraints are input strictly local functions from Σ^* to $\mathbb{N}$.

Unfortunately, strict locality of markedness constraints is not a sufficient condition for finite-stateness. Gerdemann and Hulden (2012) construct a non-finite-state OT grammar using only strictly local constraints as follows. The sole markedness constraint is AGR, which penalizes occurrences of the sequences ab and ba. AGR is outranked by the standard faithfulness constraints DEP and ID, which penalize insertion and substitution of symbols, respectively, while MAX, which penalizes deletion, ranks below all other constraints. This constraint ranking requires that

$aaabb$	DEP	ID	AGR	MAX
☞ a. $aaabb$			$*$	
b. $aaacbb$	$*!$			
c. $aaaab$		$*!$	$*$	
d. $aaab$			$*$	$*!$
e. $aabb$			$*$	$*!$

Figure 2: HS version of Figure 1

ab and ba sequences be destroyed by deleting segments. In order to remove all instances of ab or ba, either all as must be deleted, or all bs must be deleted. Between these two options, MAX favors the one that involves less deletion. Thus, if the input has more as than bs, then the bs will be deleted; otherwise, the as will be deleted. To illustrate, the tableau in Figure 1 shows the derivation of the output aaa, obtained by deleting all bs from the input $aaabb$.

Deleting the least frequent symbol from a string is non-finite-state because such a mapping requires counting the number of occurrences of each symbol. Since MAX adjudicates between the two candidates obtained by deletion, MAX is responsible for counting in this example. While strict locality limits the power of markedness constraints, this observation suggests that the power of faithfulness constraints should be limited as well. I propose to do this using *Harmonic Serialism* (HS), an alternate version of OT described in McCarthy (2000). In HS, GEN only produces candidates that differ from the input by one symbol. The winner chosen by EVAL is fed back into the grammar until a faithful mapping is obtained. To show how HS can restrict the power of MAX, Figure 2 shows an HS version of the tableau in Figure 1. Due to the restricted power of GEN, only one deletion can be performed at a time. The ab sequence in the input $aaabb$ cannot be destroyed by deleting only one symbol, so MAX simply chooses the faithful candidate. Since a faithful mapping is obtained, this candidate is not fed back into the grammar.

4 Formalization of Harmonic Serialism

Having motivated the use of HS and strictly local constraints, I now present a formal model of HS.

An HS grammar, like a standard OT grammar, computes a relation $R \subseteq \Sigma^* \times \Sigma^*$ via the three components GEN, CON, and EVAL. At the beginning of the computation, the grammar takes a string x as input. GEN reads this input and returns a set of *candidates*. CON assigns to each candidate a vector of natural numbers known as *violations*. Finally, EVAL, based on a linear ordering of $\mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}$, reads the set of candidates and their violations and returns the candidate y with the optimal violation vector. If $y = x$, then y is the output of the grammar. Otherwise, a recursive call to the grammar is made with y as the input, and the output from this call is the output of the grammar.

As discussed in the previous section, HS differs from standard OT in two ways. Firstly, recursive calls to the grammar are not featured in standard OT; instead, EVAL chooses the output in "one fell swoop." Secondly, in HS, GEN is restricted so that changes can only be made to the input "one at a time," so that each call to the grammar produces an optimal candidate that is only minimally different from the input.

These ideas are formalized in the remainder of this section. Let us begin with the notion of a *change*. A single change to a string is defined as insertion, substitution, or deletion of a single symbol in that string.

Definition 2. An *operation* is an ordered pair $\langle x, y\rangle \in ((\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\}))\backslash\{\langle \lambda, \lambda\rangle\}$. An operation $\langle x, y\rangle$ is an *insertion* if $x = \lambda$, a *deletion* if $y = \lambda$, a *substitution* if $\lambda \neq x \neq y \neq \lambda$, and an *identity* if $x = y$.

Definition 3. A pair $\langle a, b\rangle \in \Sigma^* \times \Sigma^*$ is an *application of operation* $\langle x, y\rangle$ if there exist $u, v \in \Sigma^*$ such that $a = uxv$ and $b = uyv$. An application of an operation $\langle x, y\rangle$ is called a *change* when the operation $\langle x, y\rangle$ is not specfied.

Strictly local constraints, defined in the previous section, formalize markedness constraints. To treat faithfulness constraints, I adopt the standard view that faithfulness constraints militate against certain kinds of changes to the input. I will assume that GEN is restricted so that on input a, GEN only produces candidates b such that $\langle a, b\rangle$ is a change. Since only one change can be made to a, faithfulness constraints can be seen as binary functions that penalize applications of banned operations.

Definition 4. A *faithfulness constraint* is a function $f : \Sigma^* \times \Sigma^* \to \mathbb{N}$ such that for some set O_f of operations not including identities, $f(a, b) = 1$ if $\langle a, b\rangle$ is an application of some $\langle x, y\rangle \in O_f$, and $f(a, b) = 0$ otherwise. If $\langle x, y\rangle \in O_f$, then we say that f *bans* $\langle x, y\rangle$.

CON contains a set of constraints, which are as-

sumed to be *ranked* with respect to one another. The ranking is represented here as a sequence of constraints.

Definition 5. For any strictly local constraint $c : \Sigma^* \to \mathbb{N}$, let c be extended to a function $c : \Sigma^* \times \Sigma^* \to \mathbb{N}$ defined by $c(x, y) = c(y)$. A *constraint ranking* is a sequence of functions $\langle c_1, c_2, \ldots, c_n \rangle$ where for each i, $c_i : \Sigma^* \times \Sigma^* \to \mathbb{N}$ is either a strictly local constraint or a faithfulness constraint. For any constraint ranking C, the number $k_C \geq 0$ is the length of the longest sequence banned by a strictly local constraint of C.

Among the candidates produced by GEN, EVAL chooses the one that violates the constraints the least. Given a constraint ranking $C = \langle c_1, c_2, \ldots, c_n \rangle$ and an input x, this is determined by considering for each candidate y the value $c_i(x, y)$. The winner chosen by EVAL is the one that minimizes this value for the most highly ranked constraints possible. To compare different candidates, I define here the notions of *cost*, *benefit*, and *harmonicity*.

Definition 6. The *cost* of a change $\langle x, y \rangle$ with respect to a constraint ranking $C = \langle c_1, c_2, \ldots, c_n \rangle$ is the vector

$$\mathfrak{c}_C(x, y) = \langle c_1(x, y), c_2(x, y), \ldots, c_n(x, y) \rangle.$$

The *benefit* of $\langle x, y \rangle$ is

$$\mathfrak{b}_C(x, y) = \mathfrak{c}_C(x, y) - \mathfrak{c}_C(x, x).$$

Definition 7. A vector $a = \langle a_1, a_2, \ldots, a_n \rangle \in \mathbb{Z}^n$ is *more harmonic* than a vector $b = \langle b_1, b_2, \ldots, b_n \rangle \in \mathbb{Z}^n$ if there exists j such that $a_j < b_j$ and for all $i < j$, $a_i = b_i$. We denote this by $a \succ_H b$. We write $a \succeq_H b$ if $a \succ_H b$ or $a = b$.

Putting these definitions together, an HS grammar is defined as a system, parameterized by a constraint ranking C, that takes a string as input and applies the change that results in the greatest benefit with respect to C. Recursion is performed until the most beneficial change is an identity.

Definition 8. An *HS grammar* is an ordered triple $\langle C, \mathscr{H}_C, \mathscr{H}_C^* \rangle$, where

- C is a constraint ranking;
- the relation $\mathscr{H}_C \subseteq \Sigma^* \times \Sigma^*$ is defined by $\langle u, v \rangle \in \mathscr{H}_C$ if and only if

$$\mathfrak{b}_C(u, v) = \max_y \mathfrak{b}_C(u, y),$$

  where max is taken with respect to $\succ_H$ over strings y such that $\langle u, y \rangle$ is a change; and
- letting $\hat{\mathscr{H}_C}$ be the transitive closure of $\mathscr{H}_C$, the relation $\mathscr{H}_C^*$ is defined by

$$\mathscr{H}_C^* = \{\langle x, y \rangle \in \hat{\mathscr{H}_C} | \langle y, y \rangle \in \mathscr{H}_C\}.$$

5 Finite-State Harmonic Serialism

The central result of this paper is that for any HS grammar $\langle C, \mathscr{H}_C, \mathscr{H}_C^* \rangle$, the relation $\mathscr{H}_C^*$ is rational. In section, I derive this result in two steps. Firstly, I construct a finite-state transducer whose behavior is $\mathscr{H}_C$. This shows that a single non-recursive call to a Harmonic Serialism grammar can be modelled as a rational relation. Secondly, I show that this transducer can be extended to a transducer whose behavior is $\mathscr{H}_C^*$.

5.1 $\mathscr{H}_C$ as a Rational Relation

This subsection describes a construction for a finite-state transducer that, for any constraint ranking C, computes $\mathscr{H}_C$. The construction relies on the property that the benefit of an application $\langle a, b \rangle$ of $\langle x, y \rangle$ can be computed using only information about a context of bounded size around the position of x in a and y in b. Since there are only finitely many such contexts, this locality property allows us to reduce the set of possible changes performed by $\mathscr{H}_C$ to a finite number of cases—one for each possible context. For each context, we can then construct a transducer that effects the most beneficial change for that context while ensuring that no context allowing for a more beneficial change is available. The union of all such transducers computes $\mathscr{H}_C$.

Let us now prove the locality property. To that end, I first introduce the definition of a context-sensitive rule, which captures the notion of an operation that only applies in a certain context.

Definition 9. A *rule* is an ordered quadruple $\langle x, y, c, d \rangle$, where $\langle x, y \rangle$ is an operation and $c, d \in (\Sigma \cup \{\rtimes, \ltimes\})^*$. We denote $\langle x, y, c, d \rangle$ by $x \to y \,/\, c \_ d$.

Definition 10. An *application of a rule* $x \to y \,/\, c \_ d$ is a pair $\langle a, b \rangle$ such for some $u, v \in (\Sigma \cup \{\rtimes, \ltimes\})^*$, $\rtimes a \ltimes = ucxdv$ and $\rtimes b \ltimes = ucydv$.

The locality property then states that every application of a rule with a sufficiently large context has the same benefit.

Proposition 11. *Let C be a constraint ranking, and suppose $\langle a_1, b_1 \rangle$ and $\langle a_2, b_2 \rangle$ are applications of a rule $x \to y \,/\, c \_ d$. If $|c|, |d| \geq k_C - 1$, then $\mathfrak{b}_C(a_1, b_1) = \mathfrak{b}_C(a_2, b_2)$.*

Proof. Write $C = \langle c_1, c_2, \ldots, c_n \rangle$. We need to show that for each i,

$$c_i(a_1, b_1) - c_i(a_1, a_1) = c_i(a_2, b_2) - c_i(a_2, a_2).$$

Fix any $i \in \{1, 2, \ldots, n\}$. Note that $\langle a_1, b_1 \rangle$ and $\langle a_2, b_2 \rangle$ are applications of the same operation. Therefore, if c_i is a faithfulness constraint, then $c_i(a_1, b_1) = c_i(a_2, b_2)$. Since a faithfulness constraint cannot ban the application of an identity, $c_i(a_1, a_1) = c_i(b_2, b_2) = 0$. From this the equation above follows.

Now suppose c_i is a strictly local constraint. The equation above can then be rewritten as follows.

$$c_i(b_1) - c_i(a_1) = c_i(b_2) - c_i(a_2)$$

Let S_i be the set of sequences banned by c_i. For any string w, $c_i(w)$ is the number of occurrences of elements of S_i in w. Since b_1 and a_1, as well as b_2 and a_2, only differ by x and y, any occurrence of an element of S_i in b_1 but not a_1 or in b_2 but a_2 must contain the x that is replaced by y. Similarly, any occurrence of an element of S_i in a_1 but not b_1 or a_2 but not b_2 must contain the y that replaces x. Since $|c|, |d| \geq k_C - 1$ and $|s| \leq k_C$ for all $s \in S_i$, any occurrence of some $s \in S_i$ that includes either the x or the y must be a substring of cxd or cyd. Thus, we have

$$\begin{aligned} c_i(b_1) - c_i(a_1) &= c_i(cyd) - c_i(cxd) \\ &= c_i(b_2) - c_i(a_2), \end{aligned}$$

giving us the equation above. □

Definition 12. Let C be a constraint ranking. The *benefit* of a rule $r = x \to y \,/\, c \_ d$ with respect to a constraint ranking C, denoted $\mathfrak{b}_C(r)$, is defined by $\mathfrak{b}_C(r) = \mathfrak{b}_C(c'xd', c'yd')$, where c' and d' are c and d, respectively, with occurrences of $\rtimes$ and $\ltimes$ replaced by λ.

Using Proposition 11, we can now construct a transducer computing $\mathscr{H}_C$ by considering all possible rules $x \to y \,/\, c \_ d$, where $|c| = |d| = k_C - 1$. For any input a, $\langle a, b \rangle \in \mathscr{H}_C$ if $\langle a, b \rangle$ is an application of the most beneficial rule that could be applied to a. Thus, for each rule r, we can construct a transducer that checks whether r is the most beneficial rule that is applicable to its input, and if so, apply r to its input. The following lemma gives us a way to check whether r is the most beneficial rule possible.

Lemma 13. *Let C be a constraint ranking, and suppose that $\langle a, b \rangle$ is an application of $r = x \to y \,/\, c \_ d$, where $|c| = |d| = k_C - 1$. Suppose further that for all $y' \in \Sigma \cup \{\lambda\}$,*

$$\mathfrak{b}_C(x \to y' \,/\, c \_ d) \preceq_H \mathfrak{b}_C(r).$$

Then, there is a set $F_r \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^{2k_C - 1}$ such that $\langle a, b \rangle \in \mathscr{H}_C$ if and only if a does not contain any element of F_r as a substring.

Proof. Since $\langle a, b \rangle$ is an application of r, the length of a must be at least $|cxd| = 2k_C - 1$. Thus, for any $b' \in \Sigma^*$, $\langle a, b' \rangle \in \mathscr{H}_C$ only if $\langle a, b' \rangle$ is an application of a rule $r' = x' \to y' \,/\, c' \_ d'$ with $|c'| = |d'| = k_C - 1$.

Such a rule r' is applicable to a if and only if a contains the substring $c'x'd'$. Thus, let us define F_r by

$$\begin{aligned} F_r = \{c'x'd' | &\mathfrak{b}_C(r') \succ_H \mathfrak{b}_C(r), \\ &|c'| = |d'| = k_C - 1\}. \end{aligned}$$

By hypothesis, $cxd \notin F_r$, so a rule more beneficial than r can be applied to a if and only if a contains a substring from F_r. But $\langle a, b \rangle \in \mathscr{H}_C$ if and only if no rule more beneficial than r can be applied to a, hence the lemma. □

To use Lemma 13, we only consider rules $r = x \to y \,/\, c \_ d$ such that no y' satisfies

$$\mathfrak{b}_C(x \to y' \,/\, c \_ d) \succ_H \mathfrak{b}_C(r).$$

To check whether a rule r is the most benefical rule applicable to a string a, we simply construct the set F_r and check that a does not contain any element of F_r as a substring.

We are now ready to present the construction of a finite-state transducer for $\mathscr{H}_C$.

Theorem 14. *Let $C = \langle c_1, c_2, \ldots, c_n \rangle$ be a constraint ranking. Then, $\mathscr{H}_C$ is a rational relation.*

Proof. We need to construct a finite-state transducer T such that on input a, T outputs b if and only if $\langle a, b \rangle \in \mathscr{H}_C$. To do this, we consider two possible cases: either $|a| < 2k_C - 1$, or $|a| \geq 2k_C - 1$. In the first case, $\langle a, b \rangle$ is generally not an application of a rule $x \to y \,/\, c \_ d$ with $|c|, |d| \geq k_C - 1$, so Proposition 11 does not apply. Instead, we simply observe that the relation

$$\{\langle a, b \rangle \in \mathscr{H}_C | |a| < 2_C - 1\}$$

is finite, so it is automatically rational. Let T_0 be a transducer whose behavior is this relation.

Now, let us assume that $|a| \geq 2k_C - 1$. Then, a is an application of a rule $x \rightarrow y \,/\, c\_d$ with $|c| = |d| = k_C - 1$, so we can use the technique discussed in this subsection. To that end, let R be the set of all rules $r = x \rightarrow y \,/\, c\_d$ such that

- $|c| = |d| = k_C - 1$;
- $\mathfrak{b}_C(r) \succ_H \langle 0, 0, \ldots, 0 \rangle$; and
- there is no $r' = x \rightarrow y' \,/\, c\_d$ such that $\mathfrak{b}_C(r') \succ_H \mathfrak{b}_C(r)$.

R is precisely the set of all rules r with $|c| = |d| = k_C - 1$, other than the identity, such that some application of r is in $\mathscr{H}_C$. For each $r \in R$, let F_r be defined as in Lemma 13.

For each rule $r = x \rightarrow y \,/\, c\_d \in R$, we need to construct a transducer T_r that applies r if it is the most beneficial rule applicable to its input. As discussed earlier, this amounts to checking that the input does not contain any element of F_r as a substring, and then applying the rule. Observe that the set of strings without substrings from F_r forms a $(2k_C - 1)$-strictly local language S_r, so to achieve this effect, we can simply take a transducer applying the rule and restricting its domain to S_r. Let us call this transducer I_r, whose behavior is defined below.

$$[I_r] = \{\langle ucxdv, ucydv\rangle | ucxdv \in S_r, \\ y \in \Sigma \cup \{\lambda\}\}$$

To construct T_r, we simply add the boundary symbols $\rtimes$ and $\ltimes$ to the input, apply I_r, and then remove the boundary symbols.

$$[T_r] = \{\langle \rtimes s \ltimes, s\rangle | s \in \Sigma^*\} \circ I_r \\ \circ \{\langle s, \rtimes s \ltimes\rangle | s \in \Sigma^*\}.$$

Finally, to construct a transducer T computing $\mathscr{H}_C$, we simply take the union of all the T_rs, along with T_0 and the identity relation on any string for which no rule in R is applicable.

$$[T] = \left(\bigcup_{r \in R} [T_r]\right) \cup [T_0] \cup \{\langle s, s\rangle | \forall r \in R[s \notin S_r]\}$$

It is clear that $[T] = \mathscr{H}_C$, so $\mathscr{H}_C$ is rational. □

5.2 Transducing Recursion

Having shown that $\mathscr{H}_C$ is a rational relation for any constraint ranking C, it remains to show that $\mathscr{H}_C^*$ is rational as well. Recall that the behavior of $\mathscr{H}_C^*$ is to repeatedly apply $\mathscr{H}_C$ until a fixed point is reached. Since rational relations are closed under composition, the naïve approach to transducing $\mathscr{H}_C^*$ is to take the transducer T constructed in the previous subsection and compose it with itself multiple times. This approach is not correct, however, because there is no bound on the recursion depth of an HS grammar. A string can, in principle, contain arbitrarily many instances of a sequence banned by a strictly local constraint, and such a string could require a recursive call for each instance of a banned sequence.

To address the problem of unbounded recursion depth, I rely on techniques from *regular model checking*, a discipline that analyzes automated systems with infinitely many state configurations and attempts to find the set of states reachable from the initial states. Under a paradigm introduced by Jonsson and Nilsson (2000) and Bouajjani et al. (2000), the set of possible states of a system are represented as a regular language, and the possible transitions between states are modelled as a rational relation. Finding the set of reachable states amounts to finding the transitive closure of the transition relation, since the transitive closure is exactly the relation obtained by applying the transition relation to itself arbitrarily many times. Accordingly, much has been written in the model checking literature about how the transitive closure of rational relations might be computed. Surveys of these results can be found in Nilsson (2000), Abdulla et al. (2004), and Abdulla (2012).

Using these techniques, we can take the transducer T computing $\mathscr{H}_C$ and compute its transitive closure $\hat{T}$. The effect of $\hat{T}$ is to apply $\mathscr{H}_C$ to a string arbitrarily many times, so $\hat{T}$ is able to handle the problem of arbitrary recursion depth.

The intuition behind the technique for computing the transitive closure is as follows. Consider the transducer T shown at the top of Figure 3. This transducer reads strings over $\Sigma = \{a, b\}$ and changes the first b to an a. This is done by having T begin in state 0, and enter state 1 when a b is read. Now, let us consider how a transducer computing $[T] \circ [T]$, which changes the first two bs in the input to as, might be constructed. Recall that on an input x, the *run* of T on x is the sequence $q_0 q_1 \ldots q_n$ of states that T enters into during its computation. On input x, T produces an output y by changing the first b of x to an a. To compute $[T] \circ [T]$, we must then feed y back into T,

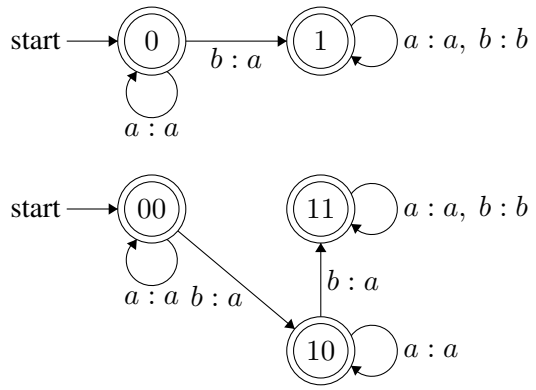

Figure 3: A transducer (top) and its composition with itself (bottom)

x:		a	a	b	b	...
Run 1:	0	0	0	1	1	...
y:		a	a	a	b	...
Run 2:	0	0	0	0	1	...
z:		a	a	a	a	...

Figure 4: The runs of two applications of the transducer from the top of Figure 3

producing an output z. This produces another run $p_0 p_1 \dots p_n$. The two runs are visualized in Figure 4, taking x to be a sample input beginning with $aabb$. A transducer T^2 computing $[T] \circ [T]$ can be constructed by stacking the two runs on top of one another. Each state of T^2 represents a *column* of the diagram in Figure 4—an ordered pair encoding the state of T during its first and its second iterations. Transitions can then be defined between the columns so as to match the behavior of T during its two passes. The resulting transducer T^2 is shown at the bottom of Figure 3. By inspection, it is clear that this transducer changes the first two bs of its input to as.

Let us now make these ideas explicit by defining the notion of a *column transducer*. This definition was introduced by Abdulla et al. (2002).

Definition 15. Let $T = \langle Q, \Sigma, \Sigma, \{q_0\}, F, \delta \rangle$ be a finite-state transducer such that $\langle x, y \rangle \in T$ implies $|x| = |y|$. The *column transducer for* T is the transducer $T^+ = \langle Q^+, \Sigma, \Sigma, q_0^+, F^+, \rho \rangle$, where $\langle q_1 q_2 \dots q_m, a, b, r_1 r_2 \dots r_m \rangle \in \rho$ if and only if there exist $a_0, a_1, \dots, a_m$ such that $a = a_0$, $b = a_m$, and for each i, $\langle q_1, a_{i-1}, a_i, r_i \rangle \in \delta$.

Abdulla et al. (2002) show that for any transducer T computing a length-preserving relation, $[T^+]$ is indeed the transitive closure of $[T]$. However, this is not enough to show that the transitive closure of $[T]$ is rational. In the example of Figures 3 and 4, the transducer T^2 only computes two iterations of T, so the states of T^2 are columns of length 2. However, the column transducer T^+ has states of arbitrary length, so T^+ has infinitely many states, and is therefore not a finite-state transducer.

To remedy this, Abdulla et al. (2002), noting that different states often exhibit the same behavior, define an equivalence relation $\simeq$ on Q^+ in hopes that only finitely many columns in $Q^+/\simeq$ might be reachable.

Definition 16. Let $T = \langle Q, \Sigma, \Sigma, \{q_0\}, \{q_f\}, \delta \rangle$ be a finite-state transducer. A state $q \in Q$ is *left-copying* if $\langle q_0, x, y, q \rangle \in \hat{\delta}$ implies $x = y$. A state $q \in Q$ is *right-copying* if $\langle q, x, y, q_f \rangle \in \hat{\delta}$ implies $x = y$. A state is *non-copying* if it is neither left-copying nor right-copying.

Definition 17. Let $p, q \in Q^+$. We write $p \simeq q$ if there exist $m_1, m_2, \dots, m_k, n_1, n_2, \dots, n_k > 0$ and $q_1, q_2, \dots, q_k \in Q$ such that

- $p = q_1^{m_1} q_2^{m_2} \dots q_k^{m_k}$,
- $q = q_1^{n_1} q_2^{n_2} \dots q_k^{n_k}$, and
- for each i, if q_i is non-copying, then $m_i = n_i = 1$.

Taking the quotient of Q^+ by $\simeq$ does not affect the behavior of T^+.

Theorem 18 (Abdulla et al. (2002))**.** *Define the quotient transducer of* T *by* $T_\simeq = \langle Q^+/\simeq, \Sigma, \Sigma, [q_0^+]_\simeq, F^+/\simeq, \psi \rangle$*, where*

$$\psi = \{\langle [p]_\simeq, a, b, [q]_\simeq \rangle | \langle p, a, b, q \rangle \in \rho\}.$$

Then, $[T_\simeq] = [T^+]$ *is the transitive closure of* $[T]$.

This result shows that the transitive closure of $[T]$ is rational if only finitely many states in $Q^+/\simeq$ are reached. By inspecting Definition 17, we see that this is possible if each reachable column contains finitely many non-copying states, and if each column contains finitely many alternations between different copying states. Abdulla et al. (2003) introduce a technique known as *bi-determinization* for ensuring that the latter condition is always met, so the former condition is sufficient to ensure that the transitive closure of $[T]$ is rational.

We are now ready to use Theorem 18 to show that $\mathscr{H}_C^*$ is rational. To do so, we first need to modify the construction from Theorem 14 for the

transducer T computing $\mathscr{H}_C$ so that $[T]$ is length-preserving. This can be done by padding strings with symbols that are treated like λs. Insertions are then performed by replacing these special symbols with symbols from Σ, while deletions are performed by replacing symbols from Σ with special symbols. This allows insertions and deletions to be simulated without changing the length of the input. Once T has been made to preserve length, we construct $T_{\simeq}$, and restrict its output to strings x such that $\langle x, y\rangle \in \mathscr{H}_C$ only if $x = y$.

Theorem 19. *Let C be a constraint ranking. Then, $\mathscr{H}_C^*$ is rational.*

Proof. Let $T = \langle Q, \Sigma, \Sigma, \{q_0\}, F, \delta\rangle$ be the finite-state transducer such that $[T] = \mathscr{H}_C$. We shall first show that the transitive closure of T is rational, and then use the transitive closure to construct a finite-state transducer whose behavior is $\mathscr{H}_C^*$.

Let $B = \{\mathtt{i}, \mathtt{d}\}$ be the special symbols used to pad strings so that $[T]$ can be made finite-state. Insertions are made by changing `i`s to other symbols, and deletions are made by changing symbols to `d`s. For any string $x = x_1 x_2 \dots x_n$, define

$$\iota(x) = B^* x_1 B^* x_2 \dots x_{n-1} B^* x_n B^*.$$

In other words, ι freely inserts special symbols to a string. Now, let us modify T by again considering the two cases where the length of T's input a is at most or greater than $2k_C - 1$. In the former case, for any $\langle a, b\rangle \in [T]$, write $a = uxv$ and $b = uyv$. We replace $\langle a, b\rangle$ with

- $\langle \iota(u)x\iota(v), \iota u \mathtt{d} \iota v\rangle$ if $y = \lambda$,
- $\langle \iota(u)\mathtt{i}\iota(v), \iota u y \iota v\rangle$ if $x = \lambda$, and
- $\langle \iota(u)x\mathtt{i}\iota(v), \iota u \mathtt{d} y \iota v\rangle$ if $x \neq \lambda$ and $y \neq \lambda$.

In the case where $|a| > 2k_C - 1$, let R be defined as in the proof of Theorem 14. Each rule $r = x \to y \,/\, c \_ d$ in R is replaced by

- $x \to \mathtt{d} \,/\, \iota(c) \_ \iota(d)$ if $y = \lambda$,
- $\mathtt{i} \to y \,/\, \iota(c) \_ \iota(d)$ if $x = \lambda$, and
- $x\mathtt{i} \to \mathtt{d}y \,/\, \iota(c) \_ \iota(d)$ if $x \neq \lambda$ and $y \neq \lambda$.

Let us call the set of these new rules R'.

In this modified version of T, the only modifications that could be made to the input are changing `i`s to alphabet symbols and changing alphabet symbols to `d`s. In particular, `d`s can never be changed by T. Therefore, if T is applied to an input arbitrarily many times, for any i, the ith position only changes at most twice. This means that in the column transducer T^+, the column reached at the ith position can only contain at most two non-copying states, so in the quotient transducer $T_{\simeq}$, only finitely many states are reachable. Removing unreachable states makes $T_{\simeq}$ a finite-state transducer, so by Theorem 18, the transitive closure of $[T]$ is rational.

To complete the proof, let us use $T_{\simeq}$ to construct a finite-state transducer M for $\mathscr{H}_C^*$. Define the transducers E and D, which freely insert and remove padding symbols, respectively, as follows.

$$[E] = \{\langle x, y\rangle | x \in \Sigma^*, y \in \iota(x)\}$$
$$[D] = \{\langle x, y\rangle | y \in \Sigma^*, x \in \iota(y)\}$$

M must first insert padding symbols to its input, then apply $T_{\simeq}$, and then remove padding symbols. Afterwards, the range of these operations must be intersected with the set of strings such that the most beneficial change is the identity. Letting S_r be defined for each r as in Theorem 14, recall that this set of strings is precisely

$$S = \bigcup_{r \in R} S_r.$$

Since each S_r is regular, so is S. Therefore, we write

$$[M] = \{\langle x, y\rangle \in [D] \circ [T_{\simeq}] \circ [E] | y \in S\},$$

completing the construction. □

6 Conclusion

In this paper, I have shown that the Harmonic Seralism version of Optimality Theory defines rational relations if markedness constraints are assumed to be strictly local. This was done by constructing a finite-state transducer relating each input with the winner chosen by EVAL after a single iteration of the grammar. This transducer was extended to a transducer that makes recursive calls to the grammar by relying on techniques from regular model checking for computing the transitive closure of rational relations satisfying certain conditions. The assumption that markedness constraints are strictly local allowed us to show that $\mathscr{H}_C$ is regular by partitioning the space of possible changes effected by the grammar into a finite number of cases. The limitation of GEN to

"one change at a time" allowed us to construct the transitive closure of $\mathscr{H}_C$ in such a way that only finitely many states in the quotient transducer are reachable.

For computational phonology, this paper contributes a new finite-state model of OT that incorporates ideas from recent work on the subregular hierarchy and provides an example of how the property of locality could be exploited to restrict the power of OT to rational relations. The model presented here is also the first to achieve finite-stateness using restrictions on OT originating in the phonological literature: most markedness constraints proposed in OT analyses are indeed strictly local, and Harmonic Serialism was first introduced in the original manuscript of Prince and Smolensky (1993).

This paper also has implications for theoretical phonology. While Harmonic Serialism is generally known as a way to model phonological opacity, McCarthy (2000) mentions that in many cases, HS analyses are not distinguishable from standard OT analyses. On the other hand, the ability of HS grammars to make recursive calls is traditionally seen as a significant increase in the complexity of OT, so a standard OT analysis is usually preferable to a similar HS analysis. The proposal of this paper, however, provides evidence against that intuition: since standard OT with strictly local constraints is more powerful than rational relations, the finite-state model presented here shows that HS is weaker than standard OT in language-theoretic terms. Thus, this paper supports the viewpoint, originating from Moreton (1999)'s proof that the recursion of EVAL always converges to a fixed point, that HS is in fact *less* complex than standard OT. While the ability to feed the output of EVAL back into GEN seems to increase the power of OT, this increase in power is offset by the restriction of GEN to one operation at a time. The rationality of HS provides an interesting distinction between standard OT and HS, and presents motivation for further work on HS phonology.

To conclude, several issues should be addressed in future work on this topic. Firstly, the results presented in this paper are purely theoretical. An implementation of the two constructions described in Section 5 needs to be developed if the ideas from this paper are to be used in NLP applications. Secondly, while the class of strictly local constraints is motived in part by empirical studies regarding phonological patterns in natural language, many constraints found in OT fall outside of this class. Future work should determine the extent to which the power of constraints can be extended while still ensuring that HS grammars define rational relations. One possibility would be to extend strictly local constraints to a class of constraints corresponding to the tier-based strictly local languages. Finally, for the sake of formal completeness, many commitments were made in Section 4 in the development of the formal model of HS used in this paper. In particular, I have assumed that "one change at a time" means insertion, deletion, or substitution of a single symbol. I have also assumed that if a single iteration of EVAL chooses multiple winners, each of these winners is passed back to GEN independently. In reality, multiple proposals exist in the HS literature regarding the implementational details of the framework. By modifying the formalism of Section 4, further studies could investigate which of these details affect the generative power of HS, and which do not.

Acknowledgments

I would like to thank Ryan Bennett, Robert Frank, and the reviewers for their valuable feedback and discussion. Any remaining errors are my own.

References

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. 2002. Regular model checking made simple and effcient. In *International Conference on Concurrency Theory*, pages 116–131. Springer.

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. 2003. Algorithmic improvements in regular model checking. In *International Conference on Computer Aided Verification*, pages 236–248. Springer.

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saksena. 2004. A survey of regular model checking. In *International Conference on Concurrency Theory*, pages 35–48. Springer.

Parosh Aziz Abdulla. 2012. Regular model checking. *International Journal on Software Tools for Technology Transfer (STTT)*, 14(2):109–118.

Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. 2000. Regular model checking. In *CAV*, volume 1855, pages 403–418. Springer.

Jane Chandlee. 2014. *Strictly local phonological processes*. Ph.D. thesis, University of Delaware.

Jason Eisner. 2000. Directional constraint evaluation in optimality theory. *Proceedings of COLING*.

Jason Eisner. 2002. Comprehension and compilation in optimality theory. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 56–63. Association for Computational Linguistics.

T Mark Ellison. 1994. Phonological derivation in optimality theory. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 1007–1013. Association for Computational Linguistics.

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

Dale Gerdemann and Mans Hulden. 2012. Practical finite state optimality theory. In *FSMNLP*, pages 10–19.

Dale Gerdemann and Gertjan Van Noord. 2000. Approximation and exactness in finite state optimality theory. *arXiv preprint cs/0006038*.

Jeffrey Heinz, Chetan Rawal, and Herbert G Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 58–64. Association for Computational Linguistics.

Bengt Jonsson and Marcus Nilsson. 2000. Transitive closures of regular relations for verifying infinite-state systems. In *TACAS*, volume 1785, pages 220–234. Springer.

Lauri Karttunen. 1998. The proper treatment of optimality in computational phonology: plenary talk. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 1–12. Association for Computational Linguistics.

John J McCarthy. 2000. Harmonic serialism and parallelism.

Robert McNaughton and Seymour A Papert. 1971. *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press.

Elliott Moreton. 1999. Non-computable functions in optimality theory.

Marcus Nilsson. 2000. *Regular model checking*. Ph.D. thesis, Uppsala universitet.

Alan Prince and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. ms.

Jason Alan Riggle. 2004. *Generation, recognition, and learning in finite state Optimality Theory*. Ph.D. thesis, Citeseer.