Finite-state Optimality Theory: non-rationality of Harmonic Serialism

Yiding Hao
Department of Linguistics
Department of Computer Science
Yale University
New Haven, Connecticut, USA

ABSTRACT

This paper analyzes the language-theoretic complexity of *Harmonic Serialism* (HS), a derivational variant of Optimality Theory. I show that HS can generate non-rational relations using *strictly local* markedness constraints, proving the "result" of Hao (2017), that HS is rational under those assumptions, to be incorrect. This is possible because deletions performed in a particular order have the ability to enforce nesting dependencies over long distances. I argue that coordinated deletions form a canonical characterization of non-rational relations definable in HS.

Keywords: optimality theory, harmonic serialism, phonology, finite-state, strictly local, subregular

1 INTRODUCTION

A classical question in mathematical linguistics concerns whether or not patterns describable by grammar formalisms resemble natural language dependencies (Chomsky 1959; Berwick 1984). An ideal grammar formalism should be expressive enough to generate all attested languages, but also restrictive enough to exclude patterns thought to be impossible in natural language.

In phonology, the seminal work of Johnson (1970, 1972) and Kaplan and Kay (1994) showed that the Sound Pattern of English (SPE) formalism of Chomsky and Halle (1968) is equivalent in generative

power to *finite-state transducers* (FSTs). Henceforth, it has been generally accepted that the mapping from underlying phonological representations (URs) to surface phonetic representations (SRs) in a given language can be computed using an FST. Empirical evidence in favor of this hypothesis can be found in the SPE phonology literature, as well as in the success of hidden-Markov-model-based approaches to automatic speech recognition (Baker 1975; Lowerre 1976; Jelinek 1976).

If it is empirically true that phonological mappings are finite-state, then theoretical frameworks for phonology should ideally describe only mappings that are finite-state. Frank and Satta (1998) carry out an analysis of Optimality Theory (OT, Prince and Smolensky 1993, 2004), showing that the full OT framework can describe non-rational relations, and is therefore too powerful according to the criterion of finite-stateness. They do this by following Ellison (1994) in thinking of OT constraints as FSTs that read input–output pairs and emit violation marks. However, Frank and Satta also find that OT can be made equivalent to FSTs by assuming that for each constraint there is an upper bound on the number of violation marks that the constraint may assign to any given input–output pair. Accordingly, Karttunen (1998) has developed a finite-state calculus for implementing this *violation-bounded* version of OT.

This paper presents an analysis of *Harmonic Serialism* (HS), a variant of OT in which surface forms are computed by recursively applying incremental changes to underlying forms. These incremental changes are chosen from a collection of basic operations based on an OT-style constraint ranking system. Previous work on the expressive power of HS includes Lamont (2018a,b), who shows that HS can implement bounded alphabetical sorting if constraints are *strictly piecewise* and if the basic operations include metathesis. Hao (2017), on the other hand, arrives at the main conclusion that if markedness constraints are strictly local and if the basic operations only include insertion, deletion, and substitution of a single symbol, then HS only produces rational relation. The present paper disproves the latter "result" by constructing an HS grammar that produces a non-rational relation while fulfilling Hao's assumptions.

The structure of this paper is as follows. Section 2 defines terminology and notation used in the rest of this paper. Section 3 introduces

the formalism of OT and reviews existing literature in finite-state OT. Section 4 defines HS along with various kinds of markedness constraints. In Section 5, I show that the model from Section 4 can produce a non-rational relation by relying on carefully coordinated deletions. Section 6 argues that all non-rational mappings in HS are implemented in a manner similar to the non-rational mapping from Section 5. Section 7 concludes.

2 PRELIMINARIES

Let us adopt the following standard notations and definitions. $\mathbb{Z}$ is the set of integers, and $\mathbb{N} \subsetneq \mathbb{Z}$ is the set of non-negative integers. Unless otherwise specified, the letters Σ, Δ, and Γ denote finite alphabets not including the special symbols $\rtimes$ and $\ltimes$. When used, these special symbols represent the left and right boundaries of a string, respectively.

The length of a string x is denoted by $|x|$, and λ denotes the empty string, the unique string of length 0. Alphabet symbols are identified with strings of length 1, Σ^k denotes the set of strings of length k, $\Sigma^{\leq k}$ denotes the set of strings of length at most k, Σ^* denotes the set of all strings over Σ, and Σ^+ denotes the set $\Sigma^* \backslash \{\lambda\}$. For strings a and b, ab denotes the concatenation of a and b. This notation is extended to sets of strings A and B in the usual way. We say that a is a *substring* of b if there exist strings l and r such that $b = lar$.

For sets A and B, $A \times B$ is the Cartesian product of A and B. An *n-ary relation over* $A_1, A_2, \ldots, A_n$ is a subset $R \subseteq A_1 \times A_2 \times \cdots \times A_n$. If $\langle a_1, a_2, \ldots, a_n \rangle \in R$, then we may write $R(a_1, a_2, \ldots, a_n)$. If $n = 2$, then we may also write $a_1 \; R \; a_2$. For any sets A, B, and C, the *composition* of a relation $R \subseteq B \times C$ with a relation $S \subseteq A \times B$ is the unique binary relation $R \circ S \subseteq A \times C$ such that $a \; R \circ S \; c$ if and only if there exists b such that $a \; S \; b$ and $b \; R \; c$. The *transitive closure* of a relation R is defined as

$$R^+ := \bigcup_{i=1}^{\infty} \underbrace{R \circ R \circ \cdots \circ R}_{i\text{-many times}}.$$

An *equivalence relation over* A is a relation $R \subseteq A \times A$ satisfying the following properties.

- For all $a \in A$, $a \; R \; a$ (i.e., R is *reflexive*).
- For all $a, b \in A$, if $a \; R \; b$, then $b \; R \; a$ (i.e., R is *symmetric*).

- For all $a, b, c \in A$, if $a \mathrel{R} b$ and $b \mathrel{R} c$, then $a \mathrel{R} c$ (i.e., R is *transitive*).

For each $a \in A$, the *equivalence class of a with respect to R* is the set $[a]_R := \{b | b \in A, b \mathrel{R} a\}$. The *quotient of A under R* is the set $A/R := \{[a]_R | a \in A\}$. Each element of A/R is called an *equivalence class*.

A *finite-state transducer* (FST) is a 6-tuple $T = \langle Q, \Sigma, \Gamma, I, F, \rightarrow \rangle$, 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
- $\rightarrow \subseteq Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q$ is the *transition relation*.

We assume without loss of generality that if $\rightarrow(q, x, y, r)$, then $xy \neq \lambda$. The *extended transition relation* is denoted by $\rightarrow_*$. The notations $q \xrightarrow{x:y} r$ and $q \xrightarrow{x:y}_* r$ denote $\rightarrow (q, x, y, r)$ and $\rightarrow_* (q, x, y, r)$, respectively. The *behavior* of an FST T is the relation $[\![T]\!]$ defined by $x \mathrel{[\![T]\!]} y$ if and only if T has an initial state q and a final state r such that $q \xrightarrow{x:y}_* r$. A relation is *rational* if it is the behavior of an FST.

A *finite-state automaton* (FSA) is a 5-tuple $A = \langle Q, \Sigma, I, F, \rightarrow \rangle$, where

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

The *extended transition relation* is denoted by $\rightarrow_*$. The notations $q \xrightarrow{x} r$ and $q \xrightarrow{x}_* r$ denote $\rightarrow (q, x, r)$ and $\rightarrow_* (q, x, r)$, respectively. We say that A *accepts* a string w if and only if A has an initial state q and a final state r such that $q \xrightarrow{x}_* r$. The *language recognized by A* is the set of all strings accepted by A. A language is *regular* if it is recognized by an FSA.

3 BACKGROUND

Optimality Theory (OT, Prince and Smolensky 1993, 2004) is a formalism that defines mappings between URs and SRs using ranked, violable

constraints. An OT grammar is standardly considered to be given by three components. Firstly, the function GEN takes an input and returns a set of *candidates*. Then, the function EVAL chooses one or more of these candidates to be the output of the grammar. The SR of a word is assumed to be the output of the grammar when given the UR as input. The computation performed by EVAL is parameterized by the input and by CON, a set of *constraints* that must be satisfied by input–output pairs. These constraints typically contradict one another, so EVAL specifies a *ranking* over CON that determines which constraints are prioritized over others.[1] The relationships among these three components are shown visually in Figure 1.

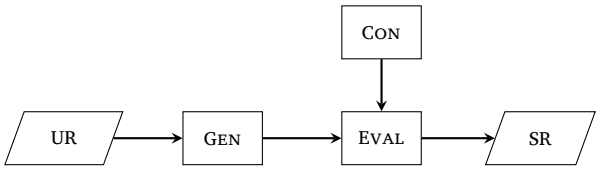


Figure 1:
The three components of standard OT

To illustrate, let us consider a concrete example. (1) shows three words from the Māori language. If a UR ends with a consonant, the SR is produced by deleting this consonant.

(1) Coda Deletion in Māori (Hohepa 1967; Hale 1973)
   a. /hopuk/ ⇝ [hopu] “catch”
   b. /arum/ ⇝ [aru] “follow”
   c. /maur/ ⇝ [mau] “carry”

A typical OT implementation of this mapping is as follows. GEN takes a UR as input, and produces as candidates all possible strings that may be obtained by inserting symbols to the input, deleting symbols from the input, or changing symbols from the input to other symbols. From among these possibilities, EVAL chooses an output based on the following constraints from CON. This output is taken to be the SR.

[1] Other formalisms with violable constraints may feature other methods for adjudicating between constraints. For example, *Harmonic Grammar* (Pater 2009; Potts *et al.* 2010) features *weighted constraints* that contribute additively to the computation of SRs from URs. Ranked constraints give rise to explanations of linguistic universals based on *factorial typology* (see Prince and Smolensky 1993, 2004), while weighted constraints account for *gang effects* (see Pater 2009).

(2) a. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.

b. DEP: Assign one violation for each symbol inserted into the input.

c. NOCODA: Assign one violation if the input ends with a consonant.[2]

d. MAX: Assign one violation for each symbol deleted from the input.

Each constraint assigns to each candidate a score, expressed in units called *violations*, that measures the degree to which the candidate violates the constraint. While several possible rankings of the four constraints above may yield the mapping shown in (1), for simplicity let us assume that the four constraints are ranked in the order shown, with higher-ranking constraints taking priority over lower-ranking ones. We denote the constraint ranking by

$$\textsc{Id} \gg \textsc{Dep} \gg \textsc{NoCoda} \gg \textsc{Max}.$$

Now, let us consider an input ending with a consonant. Based on this constraint ranking, the output cannot be produced by performing substitutions or insertions, as candidates produced in this manner violate ID and DEP. The output also cannot end with a consonant, lest it violate NOCODA. Deleting the final consonant violates MAX, but this is tolerated because MAX is the lowest-ranked constraint. All candidates produced by GEN without deletion either end with a consonant, thus violating NOCODA, or avoid a NOCODA violation by inserting or changing symbols, thus violating ID or DEP. Constraints may be violated to varying degrees; thus, deleting additional symbols beyond the final consonant is not possible, because such candidates violate MAX more severely than the candidate that only deletes the final consonant.

The computation of an output is shown in a table called a *tableau*.[3] An example of a tableau is shown in (3). The columns represent the

[2] Typically, NOCODA assigns a violation for each *syllable* ending with a consonant (Prince and Smolensky 1993, 2004). For simplicity, syllabification is not discussed here, hence this alternate statement of NOCODA.

[3] For simplicity, only *violation tableaux* are used in this paper. *Comparative tableaux* (Prince 2002, 2003) are also commonly used in OT phonology.

constraints, from highest-ranked to lowest-ranked. The rows represent a selection of candidates produced by GEN. Typically, the candidates shown in a tableau are those used to illustrate or justify claims about the behavior of the grammar, such as the effect of changing the constraint ranking. Each cell shows the number of violations the constraint assigns to the candidate.[4] The candidate that is identical to the input is known as the *faithful* candidate. The candidate that is chosen as the output, marked with the symbol ☞, is known as the *winning* candidate, or *winner*. For a non-winning candidate, the cell associated with the highest-ranking constraint that distinguishes the candidate from the winner is annotated with the symbol !.

(3) /hopuk/ ⇝ [hopu]

hopuk	ID	DEP	NOCODA	MAX
a. hopuk	0	0	1!	0
☞ b. hopu	0	0	0	1
c. hopuku	0	1!	0	0
d. hopuu	1!	0	0	0
e. ho	0	0	0	3!

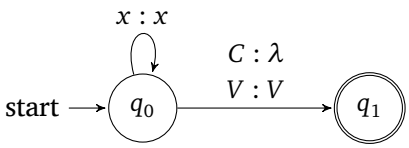
In (3), we consider the tableau of the grammar described by (2) for the UR /hopuk/. The faithful candidate violates NOCODA. Candidates c and d satisfy NOCODA, but violate DEP and ID, respectively. Candidates b and e, obtained by deleting the final consonant, violate only the lowest-ranked constraint MAX. However, because candidate e violates MAX thrice while candidate b violates MAX only once, candidate b is the winner.

3.1 *Finite-State Optimality Theory*

Māori coda deletion, as described here, is implemented by the FST in Figure 2. Recall that an FST is a device that reads an input string from left to right and produces an output while doing so. Throughout the course of its computation, the FST can be in one of a finite collection of *states*, serving as a limited form of memory. The diagram in Figure 2

[4] Numbers of violations are typically represented in unary notation over the symbol ∗. Arabic numerals are used here because some parts of this paper represent numbers of violations using algebraic expressions.

Figure 2:
An FST that deletes codas



is interpreted as follows. The FST begins in state q_0. After reading each symbol, the FST may choose to add that symbol to the output while remaining in state q_0 ($x : x$), or *transition* to state q_1. Should the machine choose the latter, it must either read a vowel and add it to the output ($V : V$), or read a consonant and omit it from the output ($C : \lambda$). The two circles around q_1 indicate that the computation is allowed to end there; the FST crashes if its computation ends at q_0. Since the FST in Figure 2 has no permissible actions once it has entered state q_1, on any given input it must remain in q_0, outputting a copy of its input, and then transition to q_1 while reading the last input symbol. If the last input symbol is a consonant, that consonant is deleted. We say that Māori coda deletion is *rational*, since it can be implemented by an FST.

In general, OT can define mappings that cannot be computed using an FST. To see how this is possible, let us consider an example of a grammar defining a non-rational relation. In this grammar, given by Gerdemann and Hulden (2012), GEN once again produces candidates by inserting symbols into the UR, deleting symbols from the UR, and changing symbols of the UR into other symbols. CON contains the following four constraints, shown in order from highest-ranking to lowest-ranking.[5]

(4) a. DEP: Assign one violation for each symbol inserted into the input.

b. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.

c. AGR: Assign one violation for each occurrence of the substring ab or ba in the output.

d. MAX: Assign one violation for each symbol deleted from the input.

[5] Again, this is not the only ranking with the intended behavior; for example, DEP and ID may be switched without consequence.

Suppose that URs are strings over the alphabet $\Sigma = \{a, b\}$ and SRs are strings over the alphabet $\Delta = \{a, b, c\}$. Consider an input of the form a^*b^*, such as $aaabb$. A tableau for $aaabb$ is shown in (5).

(5) $aaabb \rightsquigarrow aaa$

$aaabb$	DEP	ID	AGR	MAX
a. $aaabb$	0	0	1!	0
b. $aaacbb$	1!	0	0	0
c. $aaaaa$	0	2!	0	0
d. $bbbbb$	0	3!	0	0
☞ e. aaa	0	0	0	2
f. bb	0	0	0	3!

The faithful candidate a violates AGR, since it contains the substring ab. Candidates satisfying AGR cannot be chosen as the winner if they are formed by inserting or changing symbols, since such candidates violate the higher-ranking constraints DEP and ID, respectively. Thus, candidates b, c, and d cannot be the winner. An AGR-obeying candidate may be obtained by deleting all the as or the bs, resulting in candidates f and e, respectively. Because AGR $\gg$ MAX, violation of MAX in order to satisfy AGR is warranted. Between candidates f and e, the candidate that involves less deletion, namely candidate e, violates MAX to a lesser degree, and is therefore chosen as the winner.

In general, when presented with a UR of the form a^*b^*, this grammar deletes all instances of either a or b, whichever symbol occurs less frequently. This kind of mapping, in which the SR depends on the frequency of each symbol in the UR, is known as a *majority-rules* mapping (Baković 1999, 2000). Since counting the number of as and bs in a string requires infinite-state memory, finite-state transducers cannot compute majority-rules mappings.

Examining tableau (5), we see that the adjudication between candidates e and f is done by counting the number of symbols deleted from the UR. Thus, OT is endowed with the ability to count by the fact that constraints may be violated to varying degrees. Because of this insight, existing approaches to finite-state OT have sought to strip constraints of counting power by imposing restrictions on how constraints may assign violation marks, or how two candidates may be compared with

one another. Using the nomenclature of Eisner (2002), the current proposals for restricted constraints are listed below.

(6) a. n-Bounded Approximation (Frank and Satta 1998; Karttunen 1998): Each constraint may assign at most n-many violations.

b. Matching (Gerdemann and van Noord 2000; Gerdemann and Hulden 2012): Each constraint is computed by an FST that reads candidates and emits violation marks, and candidate y is considered worse than candidate z if the set of positions where violations are assigned to y is a strict superset of those for z.

c. Directional Evaluation (Eisner 2000): Each constraint is computed by an FST. Candidates are compared to each other by scanning them left-to-right or right-to-left in parallel, and a candidate is eliminated as soon as it receives a violation that at least one other candidate does not receive.

While these approaches do not reflect the version of OT used in phonology, each of them has a finite-state implementation, and therefore none of them can generate majority-rules mappings. Beyond these approaches, Riggle (2004) proposes an algorithm called the *Optimality Transducer Construction Algorithm* (OTCA) that takes an OT grammar and attempts to produce an FST computing the mapping defined by the grammar. However, this algorithm is not guaranteed to terminate.

4 HARMONIC SERIALISM

Harmonic Serialism (HS) is a variation of OT in which SRs are produced by making incremental changes to URs. In HS, for any given input, GEN is assumed only to produce candidates that may be obtained by applying to the input one of a small collection of basic operations. Most existing HS analyses assume that these operations may insert, delete, or change at most one symbol of the input, so that candidates differ from the input by an edit distance of at most 1 (McCarthy 2007). Other proposals for basic operations include applying multiple instances of the same one-symbol change (McCarthy 2008; Walker 2008, 2010), creating and adjoining syllables (Elfner 2009,

2016), creating feet and assigning stress (Pruitt 2008, 2012), and inserting or deleting autosegmental association lines (McCarthy 2009). In order to effect more dramatic changes to URs, HS stipulates that recursive calls to the grammar are made until a fixed point is reached. In other words, suppose y is the winning candidate chosen by EVAL for the UR x. If $y = x$, then y is the SR for x. If not, then the SR for x is the SR for y. This process is illustrated in Figure 3.

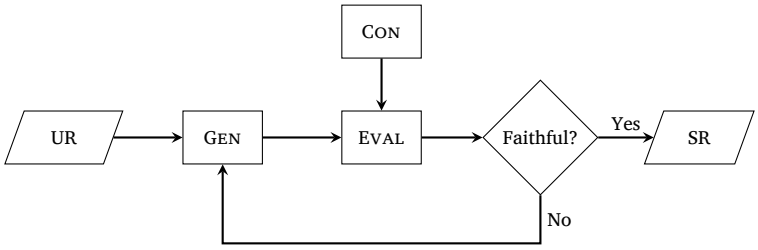


Figure 3: Harmonic Serialism

To see how HS works, let us consider an example due to McCarthy (2010). In Classical Arabic, the two symbols [ʔi] are appended to the beginning of the SR if the UR begins with more than one consonant. For example, the SR for the UR /fʕal/ "do!" is [ʔifʕal]. One possible constraint ranking deriving the correct SR is shown below.

(7) a. *CO: Assign one violation if the word begins with more than one consonant.

b. MAX: Assign one violation for each symbol deleted from the input.

c. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.

d. ONSET: Assign one violation if the word does not begin with a consonant.

e. DEP: Assign one violation for each symbol inserted into the input.

Since the constraint *CO outranks all faithfulness constraints, any violations of *CO occurring in a UR must be repaired in the SR. The fact that DEP is the lowest-ranking faithfulness constraint means that *CO will be repaired via insertion. Let us assume that low-ranking markedness constraints not shown above ensure that any inserted vowel is i,

and any inserted consonant is ʔ.[6] Accordingly, on input fʕal, we see that the winner repairs the *CO violation by inserting a vowel.

(8) Step 1: fʕal ⇝ ifʕal

fʕal	*CO	MAX	ID	ONSET	DEP
a. fʕal	1!	0	0	0	0
b. ʕal	0	1!	0	0	0
c. iʕal	0	0	1!	1	0
☞ d. ifʕal	0	0	0	1	1

The tableau above follows McCarthy (2007) in assuming that GEN can only change a single symbol. Thus, the final SR [ʔifʕal] is not available among the candidates shown. Since the winner in (8) is not the faithful candidate, the grammar is called a second time.

(9) Step 2: ifʕal ⇝ ʔifʕal

ifʕal	*CO	MAX	ID	ONSET	DEP
a. ifʕal	0	0	0	1!	0
b. fʕal	1!	1	0	0	0
c. ʔfʕal	1!	0	1	0	0
☞ d. ʔifʕal	0	0	0	0	1

This time, the input violates ONSET, since it begins with a vowel. Since DEP is still the lowest-ranking faithfulness constraint, the winning candidate is the one that repairs the ONSET violation by inserting a consonant. Since the winner is not the faithful candidate, the grammar is called for a third time.

(10) Step 3: Convergence

ʔifʕal	*CO	MAX	ID	ONSET	DEP
☞ a. ʔifʕal	0	0	0	0	0
b. ifʕal	0	1!	0	1	0
c. iifʕal	0	0	1!	1	0
d. iʔifʕal	0	0	0	1!	1

This time, the input does not violate any of the markedness constraints. The unfaithful candidates introduce gratuitous violations of faithful-

[6] This is known as *the emergence of the unmarked* (McCarthy and Prince 1994).

ness constraints, so they are all eliminated. Since the faithful candidate is chosen as the winner, the grammar terminates here. We say that the grammar has *converged* to the output [ʔifʕal], so it is chosen as the final SR for the UR /fʕal/.

Applications of HS in OT phonology are typically motivated by phonological phenoma that are most elegantly explained by decomposing the UR–SR mapping into several derivational steps. Such phenomena famously include examples of *opacity*. Elfner (2009, 2016), for example, studies opaque interactions between vowel insertion and stress assignment, and adopts an HS analysis that implements the two processes separately, allowing them to interfere with one another. Other arguments in favor of HS note that phonological processes seem to be composed of small, incremental operations. Pruitt (2008), for example, argues that locality effects in foot parsing are best explained by the gradual nature of GEN in HS. A brief survey of phonological research in HS can be found in McCarthy (2010).

The remainder of this section declares the assumptions about HS that I make in order to construct the non-rational HS grammar in Section 5. Subsection 4.1 formally defines the version of HS studied in this paper. Subsection 4.2 defines the basic operations of GEN this paper utilizes, as well as a restrictive class of markedness constraints that includes the constraints appearing in Section 5.

4.1 *Formalization of Harmonic Serialism*

For completeness, this section presents a formal definition of HS. The formalization here roughly follows Ellison's (1994) formalization of standard OT, which forms the basis of other formalizations appearing in finite-state OT. There, GEN is taken to be an FST producing candidates. Each constraint is modelled by an FST that reads candidates and emits numbers in unary notation. The behavior of EVAL is described by an ordering relation on candidates induced by the constraint ranking mechanism.

Definition 11. A *constraint over* Σ is a rational function $c : \left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^* \to \mathbb{N}$. A *constraint ranking over* Σ is a sequence $\langle c_1, c_2, \ldots, c_n \rangle$, where each c_i is a constraint over Σ. For each i, j, we say that c_i *outranks* c_j and write $c_i \gg c_j$ if $i < j$.

Definition 12. An *HS Grammar* is an ordered triple $\langle \Sigma, \textsc{Gen}, \textsc{Con} \rangle$, where

- Σ is an alphabet;
- $\textsc{Gen} : \Sigma^* \to \left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^*$ is a rational relation; and
- $\textsc{Con}$ is a constraint ranking over Σ.

The above definition departs from Ellison (1994) in that candidates are represented as strings of pairs rather than strings of alphabet symbols. This kind of representation allows Definition 12 to model faithfulness constraints (Chen-Main and Frank 2003), which depend on both the input and the potential output represented by a candidate. The pair-string representation is also standardly used in OT analyses following *Correspondence Theory* (McCarthy and Prince 1995). I follow Riggle (2004) in modelling constraints as FSTs that read pair strings and emit violations.

Definition 13. A *candidate over* Σ is a string in $\left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^*$. We may sometimes denote the candidate $\langle x_1, y_1 \rangle \langle x_2, y_2 \rangle \dots \langle x_n, y_n \rangle$ using the notation $x_1 x_2 \dots x_n \mapsto y_1 y_2 \dots y_n$.

EVAL was formalized by Samek-Lodovici and Prince (1999, 2002), who noted that the behavior of an HS grammar towards a candidate $x \mapsto y$ is completely dependent on the number of violations assigned to $x \mapsto y$ by each constraint. To that end, Samek-Lodovici and Prince identify candidates with their *violation profiles*, or *costs*.

Definition 14. Let $C = \langle c_1, c_2, \dots, c_n \rangle$ be a constraint ranking over Σ. For $x \mapsto y \in \left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^*$, the *cost of* $x \mapsto y$ *with respect to* C is the vector

$$\mathfrak{c}_C(x \mapsto y) := \langle c_1(x \mapsto y), c_2(x \mapsto y), \dots, c_n(x \mapsto y) \rangle \in \mathbb{N}^n.$$

Candidates violating lower-ranked constraints are preferred over candidates violating higher-ranked constraints, and candidates incurring few violations of a particular constraint are preferred over candidates incurring many violations of that constraint. This preference relation is represented by an ordering over cost vectors.

Definition 15. Let $a = \langle a_1, a_2, \dots, a_n \rangle$ and $b = \langle b_1, b_2, \dots, b_n \rangle$ be vectors in $\mathbb{Z}^n$. We say that a is *more harmonic than* b, and write $a \succ_H b$, if there exists $j \in \{1, 2, \dots, n\}$ such that $a_j < b_j$ and for all $i < j$, $a_i \leq b_i$. We write $a \succeq_H b$ if $a \succ_H b$ or $a = b$.[7]

[7] Note that $\succ_H$ is the lexicographic ordering on $\mathbb{Z}^n$.

Under Definition 15, candidates with more harmonic cost vectors are preferred over candidates with less harmonic cost vectors. Therefore, among a set of candidates, EVAL chooses the candidate with the most harmonic cost vector as the winner.

Definition 16. Let C be a constraint ranking over Σ. The function $\textsc{Eval}_C : \mathcal{P}\left(\left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^*\right) \to \mathcal{P}(\Sigma^*)$ is defined by

$$\textsc{Eval}_C(K) := \left\{ y \middle| x \mapsto y \in K, \mathfrak{c}_C(x \mapsto y) = \max_{\alpha \mapsto \beta \in K} \mathfrak{c}_C(\alpha \mapsto \beta) \right\},$$

where max is taken with respect to $\succeq_H$.[8]

Finally, let us conclude this subsection by defining the UR–SR mapping $\mathcal{H}_G^+$ generated by an HS grammar G.

Definition 17. Let $G = \langle \Sigma, \textsc{Gen}, \textsc{Con} \rangle$ be an HS grammar. The relation $\mathcal{H}_G$ is defined as follows: $x \; \mathcal{H}_G \; y$ if and only if $y \in \textsc{Eval}_{\textsc{Con}}(\textsc{Gen}(x))$. The relation $\mathcal{H}_G^+$ is defined as follows: $x \; \mathcal{H}_G^+ \; y$ if and only if $y \; \mathcal{H}_G \; y$ and there exist $x_1, x_2, \ldots, x_n$ such that $x_1 = x$, $x_n = y$, and for each i, $x_i \; \mathcal{H}_G \; x_{i+1}$. If $x \; \mathcal{H}_G \; y$, then we say that y is an *output of G on input x*. If $x \; \mathcal{H}_G^+ \; y$, then we say that *$G$ converges to y on input x*.

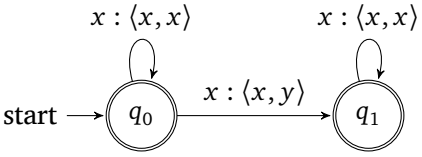
Example 18. Fix $\Sigma = \{a, b\}$. Let us construct a simple HS grammar in order to illustrate how GEN and CON may be implemented using FSTs. Suppose GEN inserts, deletes, or substitutes a single symbol from its input, and suppose CON consists of a single constraint, shown below.

(19) *CC: Assign one violation for each instance of bb in the output.

Intuitively, *CC declares a dispreference for outputs containing consonant clusters.

GEN is implemented by the FST in (20).

(20) FST for GEN ($x, y \in \Sigma^{\leq 1}$; $x \neq y$)

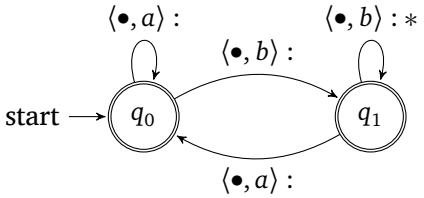


*CC is implemented by the FST in (21). Here, • represents any symbol from $\Sigma^{\leq 1}$, so that an arc from state q to state r labelled with

[8] Note that despite the notation $\max_{x' \mapsto y' \in K} \mathfrak{c}_C(x' \mapsto y')$, $\textsc{Eval}_C$ is not choosing the candidate with the "maximum cost," since what is maximal is the *harmonicity* of the cost vector.

$\langle \bullet, y \rangle : z$ means that $q \xrightarrow{\langle a,y \rangle : z} r$, $q \xrightarrow{\langle b,y \rangle : z} r$, and $q \xrightarrow{\langle \lambda,y \rangle : z} r$ are all possible transitions of the FST.

(21) FST for *CC ($x \in \Sigma \cup \{\lambda\}$)



On input $abbba$, GEN produces candidates such as $abbba \mapsto abbba$, $abbba \mapsto ababba$, and $abbba \mapsto abba$.[9] For the candidate $abbba \mapsto abbba$, the FST for *CC outputs $**$; for $abbba \mapsto ababba$ and $abbba \mapsto abba$, it outputs $*$. Thus, *CC($abbba \mapsto abbba$) = 2, while *CC($abbba \mapsto ababba$) = *CC($abbba \mapsto abba$) = 1. Since *CC is the only constraint, we have $\mathfrak{c}_{\text{CON}}(abbba \mapsto abbba) = \langle 2 \rangle$ and $\mathfrak{c}_{\text{CON}}(abbba \mapsto ababba) = \mathfrak{c}_{\text{CON}}(abbba \mapsto abba) = \langle 1 \rangle$. Observe that $\langle 1 \rangle \succ_H \langle 2 \rangle$.

4.2 *Assumptions about HS*

The previous subsection defined an HS grammar as a tuple $\langle \Sigma, \text{GEN}, \text{CON} \rangle$, but did not address the question of what kinds of FSTs may implement GEN or constraints of CON. This subsection presents the following weak assumptions about HS grammars, which suffice to construct the non-rational HS grammar in Section 5.

- GEN can insert a single symbol, delete a single symbol, or change a single symbol to another symbol.
- Markedness constraints are *strictly local*.
- Each faithfulness constraint is defined by a set of banned operations, assigning one violation to any candidate produced by applying a banned operation.

These assumptions are made explicit in Subsections 4.2.1, 4.2.2, and 4.2.3, respectively.

[9] Technically, the notation $abbba \mapsto abba$ does not specify which of the bs is deleted. This is not consequential for the rest of the paper.

4.2.1 Basic operations of GEN

I assume that GEN performs at least the following basic operations.

Definition 22. Let Σ be an alphabet. An *operation over* Σ is an ordered pair $\langle a, b\rangle$, denoted $a \mapsto b$, where $a, b \in \Sigma^{\leq 1}$ and $ab \neq \lambda$. We refer to $a \mapsto b$ simply as an *operation* when the alphabet Σ is clear from context. Additionally, we say that $a \mapsto b$ is

- an *insertion* if $a = \lambda$,
- a *deletion* if $b = \lambda$,
- a *substitution* if $\lambda \neq a \neq b \neq \lambda$, and
- an *identity* if $a = b$.

Definition 23. Let Σ be an alphabet, and define the string of pairs

$$\omega := \langle x_1, y_1\rangle\langle x_2, y_2\rangle \dots \langle x_n, y_n\rangle \in \left(\Sigma^{\leq 1} \times \Sigma^{\leq 1}\right)^*.$$

For any operation $a \mapsto b$ over Σ, we say that ω is an *application of* $a \mapsto b$ if there exists $i \in \{1, 2, \dots, n\}$ such that $a \mapsto b = x_i \mapsto y_i$ and $x_j \mapsto y_j$ is an identity as long as $j \neq i$. When ω is an application of an operation $a \mapsto b$, we may denote ω by $\omega = x_1 x_2 \dots x_n \mapsto y_1 y_2 \dots y_n$. Without explicitly specifying the operation $a \mapsto b$, we may refer to ω as a *change*. If $a \mapsto b$ is an identity, then we say that ω is *faithful*; otherwise, ω is *unfaithful*.

A change, as defined above, is an insertion, a deletion, or a substitution of a single input symbol. In practice, HS grammars considered in OT phonology rely on much richer operations. For example, the syllable-building operations of Elfner (2009, 2016) and the foot-building operations of Pruitt (2008, 2012) may change multiple symbols of the input, depending on how syllables and feet are represented. However, the availability of richer operations does not change the result of Section 5 as long as the basic operations above are available.

4.2.2 Markedness constraints

I assume that markedness constraints are *strictly local* in the following sense.

Definition 24. A constraint c is a *markedness constraint* if for every $x \mapsto z$ and $y \mapsto z$, $c(x \mapsto z) = c(y \mapsto z)$. Otherwise, c is a *faithfulness constraint*.

Definition 25. A markedness constraint c over Σ is *strictly k-local* (k-SL) if there exists a set $S \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$ such that for any candidate $x \mapsto y$, $c(x \to y)$ is the number of unique decompositions of the string $\rtimes^{k-1} y \ltimes^{k-1}$ into substrings l, s, r such that $\rtimes^{k-1} y \ltimes^{k-1} = lsr$ and $s \in S$. We say that c *bans* s if $s \in S$. A markedness constraint is *strictly local* (SL) if it is k-SL for some k.

SL constraints are constraints of the form "Assign one violation for each instance of ...." Thus, the constraint *CC from Example 18 is a 2-SL constraint that bans the substring bb. Other markedness constraints may be SL even if they are stated differently. For example, NoCoda from (2) is a 2-SL constraint banning substrings of the form $x\ltimes$, where x is a consonant. Similarly, *CO from (7) is a 3-SL constraint banning substrings of the form $\rtimes xy$, where x and y are both consonants. Observe also that the constraint Agr from the non-rational standard OT grammar of (4) is a 2-SL constraint banning the substrings ab and ba. Therefore, non-rational mappings in OT may be implemented using only markedness constraints that are SL.

The definition of SL constraints above is based on the *strictly local languages* of McNaughton and Papert (1971), a small subclass of the regular languages belonging to the *subregular hierarchy*. Intuitively, a language L is *k-strictly local* (k-SL) if there exists a k-SL markedness constraint c such that $c(x \mapsto x) = 0$ for every $x \in L$. A related subregular class of languages is the *tier-based strictly local* (TSL) *languages*, a generalization of the SL languages in which banned substrings may be interrupted by symbols from a designated subset of the alphabet. The class of TSL languages was defined by Heinz *et al.* (2011), who propose it as a formal characterization of the kinds of phonotactic dependencies that may occur in natural language phonology. Various forms of justification have been given for this hypothesis. Local phonotactic restrictions on what kinds of phonemes can occur adjacent to one another are clearly SL, and therefore TSL. Additionally, Heinz (2007), Edlefsen *et al.* (2008), Heinz (2009), and Heinz (2014) study the typology of stress patterns across languages, showing that they are usually TSL, while McMullin (2016) and McMullin and Hansson (2016) show that long-distance consonant interactions are tier-based strictly 2-local. Experimentally, Rogers and Pullum (2011), Lai (2012), Lai (2015), and McMullin (2016) investigate the ability of humans and non-human animals to learn linguistic and non-linguistic

patterns, and find that patterns that are learned successfully can be modelled by TSL languages.

4.2.3 Faithfulness constraints

Finally, I assume that each faithfulness constraint c is associated with a set O of operations such that $c(x \mapsto y) = 1$ if $x \mapsto y$ is an application of an operation in O, and $c(x \mapsto y) = 0$ otherwise. For each $o \in O$, we say that c *bans* o.

The faithfulness constraints DEP, MAX, and ID are all of this form, since they ban all insertions, all deletions, and all substitutions, respectively. Section 5 will make use of faithfulness constraints that ban deletions of a particular alphabet symbol.

5 NON-RATIONAL MAPPINGS IN HS

Define the function $f : \left(a^2b^2\right)^+ a^2ca^2\left(b^2a^2\right)^+ \to \Sigma^*$ as follows.

$$f\left(\left(a^2b^2\right)^{m+1} a^2ca^2\left(b^2a^2\right)^{n+1}\right) := \begin{cases} \left(a^2b^2\right)^{m-n} a^2ca^2, & m \geq n \\ a^2ca^2\left(b^2a^2\right)^{n-m}, & m \leq n \end{cases}$$

This function is defined on inputs consisting of a c preceded by a string in $\left(a^2b^2\right)^+ a^2$ and followed by a string in $a^2\left(b^2a^2\right)^+$. The behavior of f is to delete an integer number of b^2a^2 units to the left of the c, along with the *same* number of a^2b^2 units to the right of the c. The function deletes as many b^2a^2 and a^2b^2 units as possible, so that the total number of units deleted depends on the length of the material to the left of the c or the material on the right of the c, whichever is shorter. Since the number of b^2a^2s deleted must match the number of a^2b^2s deleted, I refer to f as the *matching deletion function.*

The goal of this section is to show that matching deletion is not finite-state, and that HS can implement it under the weak assumptions from Subsection 4.2. Intuitively, computing f requires comparing the lengths of two substrings of its input, separated by the special character c. Since FSTs are not capable of this kind of computation, f is non-rational. I begin this section by making this argument rigorous in Subsection 5.1. In Subsection 5.2, I construct an HS grammar with 6-SL markedness constraints that maps an underlying form x to the

surface form $f(x)$ as long as x is within the domain of f. Since rational relations are closed under domain restriction to regular subsets, the relation described by this HS grammar is not rational.

5.1 *Non-rationality of matching deletion*

The non-rationality of matching deletion can be proven using a straightforward application of the *Pumping Lemma*, a standard tool in formal language theory.[10]

Lemma 26 (Pumping Lemma)**.** *Let L be a regular language. There exists an integer $p \geq 1$ such that every string $\xi \in L$ of length at least p may be decomposed into three substrings $\xi = \alpha\beta\gamma$ such that $|\beta| > 1$, $|\alpha\beta| < p$, and $\alpha\beta^{+}\gamma \subseteq L$. The number p is called the pumping length of L.*

Intuitively, the Pumping Lemma describes the behavior of FSAs when reading long strings. Since an FSA A only has finitely many states, when ξ is sufficiently long, there must be some state q that A enters at least twice when reading ξ. The substring β is the substring that is read during the two occurrences of q; i.e., $q \xrightarrow{\beta}_{*} q$. Since $q \xrightarrow{\beta}_{*} q \xrightarrow{\beta}_{*} q$, the substring β may be repeated arbitrarily many times without producing a string not accepted by A.

To adapt the Pumping Lemma to rational relations, observe that an FST over input alphabet Σ and output alphabet Γ may be thought of as an FSA over the alphabet $\Sigma^{\leq 1} \times \Gamma^{\leq 1}$ by replacing each FST transition $q \xrightarrow{x:y} r$ with an FSA transition $q \xrightarrow{\langle x,y \rangle} r$ (Kaplan and Kay 1994). Thus, the matching deletion function f may be thought of as a language by encoding each pair $\langle x, f(x) \rangle$ as a string of pairs. Unlike in Subsection 4.2.1, here we cannot make any assumptions regarding which symbols of x are aligned with which symbols of $f(x)$ in the pair-string representation. Instead, since the representation of f as a language is not unique, we must show that *no* language representing f is regular.

The proof will proceed as follows. We will consider an input x with a long left-hand side and an even longer right-hand side, so that the left-hand side is completely deleted by f. We will show that the substring β represents some number of a^2b^2 units from the input and some number of b^2a^2 units from the output. Thus, repeating β results in increasing the number of a^2b^2s in x and b^2a^2s in $f(x)$. However,

[10] A complete treatment of the Pumping Lemma may be found in formal language theory textbooks such as Sipser (2013).

since the left-hand side is much shorter than the right-hand side, the number of b^2a^2s on the right-hand side of $f(x)$ must decrease when the number of a^2b^2s on the left-hand side of x increases, whence a contradiction.

Theorem 27. *The matching deletion function f is not rational.*

Proof. Suppose that f were computed by an FST T. Thinking of T as an FSA, let p be the pumping length of T. Let

$$x := \left(a^2b^2\right)^{2p} a^2ca^2 \left(b^2a^2\right)^{4p}, \text{ so that}$$
$$f(x) = a^2ca^2 \left(b^2a^2\right)^{2p},$$

and let ξ be the string over $\Sigma^{\leq 1} \times \Sigma^{\leq 1}$ corresponding to the pair $\langle x, f(x)\rangle \in [T]$.

By the Pumping Lemma, since $|\xi| > p$, there exist α, β, γ such that $\xi = \alpha\beta\gamma$, $|\beta| > 1$, $|\alpha\beta| < p$, and $\alpha\beta^+\gamma \subseteq [T]$. Writing $\beta = \langle y_1, z_1\rangle\langle y_2, z_2\rangle \ldots \langle y_n, z_n\rangle$, let $y := y_1y_2 \ldots y_n$ and $z := z_1z_2 \ldots z_n$, so that β represents T reading the substring y and emitting the substring z as output.

Now, observe that the domain of f is $\left(a^2b^2\right)^+ a^2ca^2 \left(b^2a^2\right)^+$, and the range of f is $\left(a^2b^2\right)^+ a^2ca^2 \cup a^2ca^2 \left(b^2a^2\right)^+$. Any string in either the domain or the range of f contains an integer number of a^2b^2s or b^2a^2s surrounding an a^2ca^2 in the middle. Therefore, since $\alpha\beta^+\gamma \subseteq [T]$, and since y must be a substring of the first p-many symbols of x, there must exist i and j such that the pair-string $\alpha\beta\beta\gamma$ represents the pair $\langle x', f(x')\rangle$, where

$$x' = \left(a^2b^2\right)^{2p+i} a^2ca^2 \left(b^2a^2\right)^{4p} \text{ and}$$
$$f(x') = a^2ca^2 \left(b^2a^2\right)^{2p+j}.$$

Since $|\alpha\beta| < p$, we must have $i < p$, so $2p + i < 4p$. According to the definition of f,

$$f(x') = a^2ca^2 \left(b^2a^2\right)^{4p-(2p+i)} = a^2ca^2 \left(b^2a^2\right)^{2p-i},$$

so $j = -i$. Since a string cannot have a negative length, we must have $j = i = 0$. However, this implies that $|\beta| = 0$, contradicting the Pumping Lemma, so we conclude that f cannot be computed by an FST. □

5.2 Matching deletion in HS

Having shown that f is not rational, let us now implement f in HS. We shall do this by constructing a grammar that, given a UR of the form $\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}$, behaves as follows.

- First, delete the a immediately to the right of the c:

$$\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1} \rightsquigarrow \left(a^2b^2\right)^{m+1}a^2ca\left(b^2a^2\right)^{n+1}.$$

- Next, delete the a immediately to the right of the c:

$$\left(a^2b^2\right)^{m+1}a^2ca\left(b^2a^2\right)^{n+1} \rightsquigarrow \left(a^2b^2\right)^{m+1}a^2c\left(b^2a^2\right)^{n+1}.$$

- Next, delete the a immediately to the left of the c:

$$\left(a^2b^2\right)^{m+1}a^2c\left(b^2a^2\right)^{n+1} \rightsquigarrow \left(a^2b^2\right)^{m+1}ac\left(b^2a^2\right)^{n+1}.$$

- Next, delete the a immediately to the left of the c:

$$\left(a^2b^2\right)^{m+1}ac\left(b^2a^2\right)^{n+1} \rightsquigarrow \left(a^2b^2\right)^{m+1}c\left(b^2a^2\right)^{n+1}.$$

- Next, do the same with the bs adjacent to the c; delete the two bs immediately to the right, and then the two bs immediately to the left:

$$\left(a^2b^2\right)^{m+1}c\left(b^2a^2\right)^{n+1} \rightsquigarrow \cdots \rightsquigarrow \left(a^2b^2\right)^{m}a^2ca^2\left(b^2a^2\right)^{n}.$$

The effect of these eight derivational steps is to delete a full b^2a^2 from the right and a full a^2b^2 from the left. This process continues until either the left side has no more a^2b^2s or the right side has no more b^2a^2s. If $m \leq n$, then the total number of b^2a^2s and a^2b^2s deleted is $m+1$, so the SR is $a^2ca^2\left(b^2a^2\right)^{n-m}$. If $m \geq n$, then the total number of b^2a^2s and a^2b^2s deleted is $n+1$, so the SR is $\left(a^2b^2\right)^{m-n}a^2ca^2$.

To obtain this behavior, the grammar we construct must fulfill three criteria. Firstly, the grammar needs to contain markedness constraints against substrings occurring near the c. This is because HS grammars can only perform unfaithful operations in order to repair violations of markedness constraints, so the deletions performed by the grammar must destroy banned substrings. Secondly, the grammar must contain a mechanism for ensuring that the symbols adjacent to the c are deleted in the correct order – first to the right of

the c, and then to the left of the c. If the grammar were allowed to delete symbols adjacent to the c in an arbitrary order, then there would be no non-rational dependency between the number of a^2b^2s deleted to the left of the c and the number of b^2a^2s deleted to the right of the c. Finally, the grammar must ensure that no more deletions occur when all the material either to the left or to the right of the a^2ca^2 center marker has been deleted. Otherwise, the grammar would associate each UR with the SR a^2ca^2, resulting in a rational map.

These three components of the construction are presented in Subsections 5.2.1, 5.2.2, and 5.2.3, respectively. Subsection 5.2.4 shows how the three components are combined together to form an HS grammar implementing f.

5.2.1 Motivating deletion

Let us assume that faithfulness constraints are ranked so that the only possible actions of the grammar are to delete an a, to delete a b, or to do nothing. The behavior we wish to implement is for the grammar to delete symbols adjacent to the c. These deletions are driven by two markedness constraints, ranked in the order shown below.

(28) a. $*ab$: Assign one violation for each occurrence of ab or ba.
b. $*caa$: Assign one violation for each occurrence of caa, aac, cbb, or bbc.

Consider a string $\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}$ in f's domain. Every symbol of this string other than those comprising the aca in the middle forms part of an ab or ba sequence banned by $*ab$. Since each a^2b^2 segment and each b^2a^2 segment introduces one ab substring and one ba substring, the total number of violations assigned by $*ab$ is $2(m+1)+2(n+1)=2(m+n)+4$. The a^2ca^2 segment in the middle consists of two overlapping instances of substrings banned by $*caa$, namely a^2c and ca^2. Thus, $*caa$ assigns 2 violations in total.

Since every a is adjacent to another a and every b is adjacent to another b, deleting a single a or a single b cannot repair a violation of $*ab$. This is seen in candidate d of tableau (29a): there, an a is deleted from the last a^2b^2 segment to the left of the c, but the number of violations of $*ab$ does not change. On the other hand, deleting one of the as adjacent to the c results in the destruction of either the ca^2

segment or the a^2c segment, so the number of violations assigned by $*caa$ is reduced by 1. Thus, the first step of the derivation is to delete an a adjacent to the c.

The result of the first step is a string in which the c is flanked by two as on one side and a single a on the other. These are shown in candidates b and c of (29a).[11] Since the single a is no longer adjacent to another a, it now forms an ab segment with the neighboring b that is vulnerable to deletion. This is shown in candidate b of (29b). While it is still possible to repair a violation of $*caa$ by deleting an a on the other side of the c, as in candidate c of (29b), repairing $*caa$ is dispreferred to repairing $*ab$ because the latter constraint outranks the former. Thus, the second step of the grammar's derivation is to delete the single a that occurs between the c and a b.

(29) a. First repair $*caa$…

$(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	$*ab$	$*caa$
a. $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	2!
☞ b. $(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$2(m+n)+4$	1
☞ c. $(a^2b^2)^{m+1}aca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	1
d. $(a^2b^2)^{m}ab^2a^2ca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	2!

b. …then repair $*ab$

$(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$*ab$	$*caa$
a. $(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$2(m+n)+4$!	1
☞ b. $(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$2(m+n)+3$	2
c. $(a^2b^2)^{m+1}aca(b^2a^2)^{n+1}$	$2(m+n)+4$!	0

Observe that while destroying the ab segment in (29b), the grammar has created a cb^2 segment, which is banned by $*caa$. This is permissible because $*ab$ outranks $*caa$, so introducing a new violation of the latter is justified by removing a violation of the former. The result of the two derivational steps is again a string in which $*caa$ is violated twice and $*ab$ cannot be repaired. In general, the effect of the two constraints of (28) is to ensure that deletions occur two at a time:

[11] Both candidates have been marked as winning candidates, but in the next subsection candidate c will be eliminated as a possible winner.

first an a or a b to the left or to the right of the c, and then the other a or b on the same side of the c.

5.2.2 Enforcing directionality

The constraints of (28) create a mechanism by which the grammar is now required to delete as and bs adjacent to the c two at a time. The next step is to ensure that these pairs of deletions occur in the correct order: first the as on the right, then the as on the left, then the bs on the right, then the bs on the left. This particular ordering of the deletions ensures that every eight derivational steps, exactly one complete a^2b^2 segment to the left of the c and one complete b^2a^2 segment to the right of the c are deleted. Therefore, no partial a^2b^2 or b^2a^2 segment may remain at the end of the derivation, and the number of a^2b^2s deleted must always match the number of b^2a^2s deleted.

In (29b), we see that a deletion repairing $*ab$ must always occur on the same side of the c as the previous deletion repairing $*caa$. However, in (29a) we see that the constraints of (28) allow for a choice between two possible actions: destroying a ca^2 or cb^2 segment by deleting to the right of the c, or destroying an a^2c or b^2c segment by deleting to the left. The goal of this subsection is to remove this choice by imposing a preference for correct deletions over incorrect ones.

To that end, consider the possible strings that may be obtained from $\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}$ by deleting symbols adjacent to the c in the manner described in Subsection 5.2.1. Any such string containing two violations of $*caa$ and no repairable violations of $*ab$ must adhere to one of the following patterns.

(30) a. $\Sigma^*a^2ca^2\Sigma^*$
b. $\Sigma^*a^2cb^2\Sigma^*$
c. $\Sigma^*b^2cb^2\Sigma^*$
d. $\Sigma^*b^2ca^2\Sigma^*$

After a $*caa$ violation is repaired, the resulting string contains either an a sandwiched between the c and a b or a b sandwiched between the c and an a. The location of the single a or b reflects the location of the $*caa$-repairing deletion: either both occur to the left of the c or both occur to the right of the c. (31) shows that eight unique configurations can be reached from one of the patterns in (30) by deleting either on the left or on the right.

(31) a. $\Sigma^* a^2 c a^2 \Sigma^* \rightsquigarrow \Sigma^* b a c a^2 \Sigma^*$ (left)/$\Sigma^* a^2 c a b \Sigma^*$ (right)
b. $\Sigma^* a^2 c b^2 \Sigma^* \rightsquigarrow \Sigma^* b a c b^2 \Sigma^*$ (left)/$\Sigma^* a^2 c b a \Sigma^*$ (right)
c. $\Sigma^* b^2 c b^2 \Sigma^* \rightsquigarrow \Sigma^* a b c b^2 \Sigma^*$ (left)/$\Sigma^* b^2 c b a \Sigma^*$ (right)
d. $\Sigma^* b^2 c a^2 \Sigma^* \rightsquigarrow \Sigma^* a b c a^2 \Sigma^*$ (left)/$\Sigma^* b^2 c a b \Sigma^*$ (right)

Since the eight configurations are unique, from each configuration on the right-hand side of an $\rightsquigarrow$ above it is possible to recover both the pattern from (30) on the left-hand side and whether the configuration was obtained by deleting on the left or the right.

When the deletions are performed in the correct order, the intermediate strings encountered in the derivation of the grammar cycle through the four patterns of (30), in the order shown. Because of this, whether the correct deletion occurs to the left or the right of the c is completely determined by whether the input matches pattern (30a), (30b), (30c), or (30d). Four of the eight configurations in (31) reflect the result of performing the correct action based on the pattern from (30). The remaining four configurations are obtained when incorrect actions are performed. Therefore, the correct order of the deletions can be enforced using a markedness constraint against the four configurations reflecting incorrect deletions.

(32) $*baca$: Assign one violation for each occurrence of $baca$, $acba$, $abcb$, or $bcab$.

To illustrate, consider again the tableau of (29a), but this time including the constraint $*baca$. For considerations of space, let $M := 2(m+n)+4$. While both candidates b and c were marked as winners in (29a), in (33) candidates b and c are distinguished by the latter's violation of $*baca$. Thus, only candidate b, in which deletion occurs to the right of the c, is a winner in (33).

(33) $*baca$ distinguishes between candidates b and c

$(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	M	2!	0
☞ b. $(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	M	1	0
c. $(a^2b^2)^{m+1}aca^2(b^2a^2)^{n+1}$	M	1	1!
d. $(a^2b^2)^{m}ab^2a^2ca^2(b^2a^2)^{n+1}$	M	2!	0

Since the purpose of $*baca$ is to distinguish between candidates that are treated identically by the other markedness constraints, the ranking of $*baca$ relative to $*ab$ and $*caa$ is inconsequential. This is indicated by the dashed line in tableau (33) separating the $*baca$ column from the other columns. For convenience, tableaux in the remainder of this section will show the three markedness constraints in the order presented in (33).

5.2.3 Stopping condition

We have now seen that (28) and (32) together create the eight-step effect of deleting a full b^2a^2 to the right of the c and a full a^2b^2 to the left of the c. The final step of the construction is to force the grammar to converge to a final SR when one of the two sides runs out of a^2b^2s or b^2c^2s. This has occurred when the string matches one of the following patterns.

(34) a. $\Sigma^* ba^2ca^2$
b. $a^2ca^2b\Sigma^*$
c. a^2ca^2

Pattern (34c) occurs when the original UR $\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}$ contains the same number of a^2b^2s to the left of the c as b^2a^2s to the right of the c; i.e., when $m = n$. (34a) occurs when $m > n$, and therefore the right side runs out of b^2a^2s before the left side runs out of a^2b^2s. (34b) occurs when $m < n$.

As in Subsection 5.2.2, we can consider the configurations that result when a symbol adjacent to the c is deleted from a string matching one of the patterns in (34).

(35) a. $\Sigma^* ba^2ca^2 \rightsquigarrow \Sigma^* baca^2$ (left)/$\Sigma^* ca$ (right)
b. $a^2ca^2b\Sigma^* \rightsquigarrow ac\Sigma^*$ (left)/$a^2cab\Sigma^*$ (right)
c. $a^2ca^2 \rightsquigarrow aca^2 \in ac\Sigma^*$ (left)/$a^2ca \in \Sigma^* ca$ (right)

Since no further deletions should occur when one of the patterns in (34) is reached, all the configurations on the right-hand side of an $\rightsquigarrow$ above reflect incorrect deletions, so all the configurations must be banned by a markedness constraint.

(36) STOP: Assign one violation for each occurrence of $baca^2\ltimes$, $ca\ltimes^4$, $\rtimes^4 ac$, or $\rtimes a^2cab$.

Each of the deletions shown in (35) repairs a violation of $*caa$, so STOP must rank above $*caa$.

To confirm that STOP behaves as intended, let us consider a string in which the left side contains no a^2b^2 segments, but the right side contains at least one b^2a^2 segment.

(37) Converging to a final UR

$a^2ca^2(b^2a^2)^{n+1}$	STOP	$*ab$	$*caa$	F
☞ a. $a^2ca^2(b^2a^2)^{n+1}$	0	$2n+2$	2	0
b. $aca^2(b^2a^2)^{n+1}$	1!	$2n+2$	1	1
c. $a^2ca(b^2a^2)^{n+1}$	1!	$2n+2$	1	1
d. $a^2ca^2ba^2(b^2a^2)^n$	0	$2n+2$	2	1!

Since no violation of $*baca$ can be introduced by deleting a single a or b from $a^2ca^2(b^2a^2)^{n+1}$, $*baca$ is not shown in the tableau above. Candidate b, obtained by deleting an a to the left of the c, contains the banned substring $\rtimes^4ac$, so it violates STOP. Candidate c, obtained by deleting an a to the right of the c, contains $\rtimes a^2cab$, so it also violates STOP. These violations of STOP eliminate b and c as potential winners. Observe that none of the markedness constraints distinguishes between candidate a, the faithful candidate, and candidate d, obtained by deleting a b not adjacent to the c. Instead, candidates like d are eliminated by low-ranking faithfulness constraints against deleting as and bs, represented collectively in the column labelled "F."

5.2.4 Constraint ranking

To complete the construction, all that remains is to arrange the markedness constraints of (28), (32), and (36) and relevant faithfulness constraints to form a ranking that produces the desired behavior. Let us briefly summarize the ranking requirements identified in the previous subsections.

- $*ab \gg *caa$.
- The ranking of $*baca$ is inconsequential.
- STOP $\gg *caa$.

The faithfulness constraints need to ensure that the only permissible actions, other than doing nothing, are deleting an a or a b. The constraints ID and DEP can be used to ban substitutions and insertions, re-

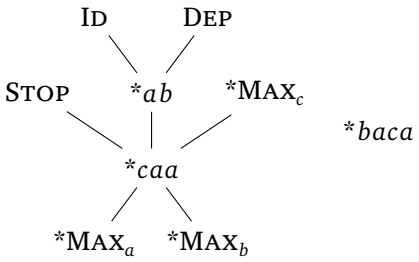


Figure 4:
Required ranking relations among constraints

spectively. The constraint MAX will be replaced with three constraints, each of which bans deletion of a particular alphabet symbol.

(38) a. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.

b. DEP: Assign one violation for each symbol inserted into the input.

c. MAX$_a$: Assign one violation for each a deleted from the input.

d. MAX$_b$: Assign one violation for each b deleted from the input.

e. MAX$_c$: Assign one violation for each c deleted from the input.

To allow deletions to occur, MAX$_a$ and MAX$_b$ must rank below $*caa$, and to ban substitutions and insertions, ID and DEP must rank above $*ab$. Since deleting a c cannot repair a violation of $*ab$, MAX$_c$ needs to rank above $*caa$, but not necessarily above $*ab$. These ranking requirements, along with the requirements for markedness constraints, are shown visually in Figure 4. Any ranking compatible with Figure 4 produces the desired behavior, so let us simply assume the ranking shown below.

ID $\gg$ DEP $\gg$ MAX$_c$ $\gg$ STOP $\gg$ $*ab$ $\gg$ $*caa$ $\gg$ $*baca$ $\gg$ MAX$_a$ $\gg$ MAX$_b$

Theorem 39. *There exists an HS grammar with strictly local markedness constraints that generates a non-rational mapping between underlying forms and surface forms.*

Proof. Let us confirm that the HS grammar we have constructed behaves as expected. Given a UR $x = (a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$, the grammar should produce $f(x)$ as the SR. As mentioned before, since the

restriction of a rational function to a regular subset of its domain is still rational, any grammar that computes f on its domain implements a non-rational function in general.

I will proceed by first showing that the grammar implements the eight-step derivational process that deletes exactly one a^2b^2 to the left of the c and one b^2a^2 to the right of the c. Then, I show that STOP correctly implements the stopping condition that terminates the derivation.

Subsections 5.2.1 and 5.2.2 showed that the first two steps of the eight-step process behave as expected. The following tableaux show the remaining six steps. As before, let us define $M := 2(m+n)+4$.

(40) a. Deleting an a on the left

$(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$M-1$	2!	0
☞ b. $(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$M-1$	1	0
c. $(a^2b^2)^{m+1}a^2cba^2(b^2a^2)^{n}$	$M-1$	1	1!

b. Deleting another a on the left

$(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$M-1$!	1	0
☞ b. $(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$M-2$	2	0

c. Deleting a b on the right

$(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$M-2$	2!	0
☞ b. $(a^2b^2)^{m+1}cba^2(b^2a^2)^{n}$	$M-2$	1	0
c. $(a^2b^2)^{m}a^2bc(b^2a^2)^{n+1}$	$M-2$	1	1!

d. Deleting another b on the right

$(a^2b^2)^{m+1}cba^2(b^2a^2)^{n}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}cba^2(b^2a^2)^{n}$	$M-2$!	1	0
☞ b. $(a^2b^2)^{m+1}ca^2(b^2a^2)^{n}$	$M-3$	2	0

e. Deleting a b on the left

$(a^2b^2)^{m+1}ca^2(b^2a^2)^n$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}ca^2(b^2a^2)^n$	$M-3$	2!	0
☞ b. $(a^2b^2)^{m}a^2bca^2(b^2a^2)^n$	$M-3$	1	0
c. $(a^2b^2)^{m+1}ca(b^2a^2)^n$	$M-3$	1	1!

f. Deleting another b on the left

$(a^2b^2)^{m}a^2bca^2(b^2a^2)^n$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m}a^2bca^2(b^2a^2)^n$	$M-3$!	1	0
☞ b. $(a^2b^2)^{m}a^2ca^2(b^2a^2)^n$	$M-4$	2	0

Tableaux (40a), (40c), and (40e) are analogous to tableaux (29a) and (33): $*caa$ is violated twice, $*ab$ violations cannot be repaired, and $*baca$ causes the grammar to choose the correct deletion to repair $*caa$ over the incorrect one. Tableaux (40b), (40d), and (40f), like (29b), repair a violation of $*ab$ while introducing a violation of $*caa$. The winner in (40f), after the eighth step of the derivation, is $(a^2b^2)^{m}a^2ca^2(b^2a^2)^n$ – the result of deleting exactly one a^2b^2 and one b^2a^2 from x.

The final part of the proof is to show that the stopping condition behaves as expected. (37) showed that STOP correctly terminates the derivation when $m < n$. The following tableaux show that the derivation terminates correctly when $m > n$ and when $m = n$.

(41) a. Converging when $m > n$

$(a^2b^2)^{m+1}a^2ca^2$	STOP	$*ab$	$*caa$	F
☞ a. $(a^2b^2)^{m+1}a^2ca^2$	0	$2m+2$	2	0
b. $(a^2b^2)^{m+1}a^2ca$	1!	$2m+2$	1	1
c. $(a^2b^2)^{m+1}aca^2$	1!	$2m+2$	1	1
d. $(a^2b^2)^{m}a^2ba^2ca^2$	0	$2m+2$	2	1!

b. Converging when $m = n$

a^2ca^2	STOP	$*ab$	$*caa$	F
☞ a. a^2ca^2	0	0	2	0
b. aca^2	1!	0	1	1
c. a^2ca	1!	0	1	1

Candidate b of (41a) is eliminated by STOP because it contains the offending substring $ca\ltimes^4$. Similarly, candidate c contains $baca^2\ltimes$. As in (37), candidate d is eliminated by low-ranking faithfulness constraints. In (41b), b and c are the only candidates that may be obtained by deleting an a. These candidates contain $\rtimes^4ac$ and $ca\ltimes^4$, respectively, so they are eliminated by STOP. □

6 CANONICAL NON-RATIONAL MAPPINGS

We have now seen that in general, HS grammars with SL markedness constraints can produce non-rational relations. Computing the matching deletion function requires the ability to enforce dependencies between arbitrarily many nesting pairs of a^2b^2 and b^2a^2 units separated by arbitrarily long distances. While the markedness constraints $*caa$ and $*ab$ are only sensitive to the material immediately adjacent to the c, deletion was able to extend the reach of these constraints by moving a^2b^2 and b^2a^2 units close to the c. The carefully choreographed manner in which the deletions were carried out allowed the grammar to maintain nesting dependencies reminiscent of context-free grammars.

The result of the previous section raises the question of what kinds of non-rational mappings HS can generate. The goal of this section is to argue that all non-rational relations generated by HS involve coordinated deletions that occur on opposite sides of some center marker. Thus, the matching deletion function is a canonical example of a non-rational mapping in HS, just as majority-rules mappings may be seen as canonical examples of non-rational mappings in standard OT.

Hao (2017) attempted to construct a finite-state model of HS by first showing that $\mathcal{H}_G$ is rational for any HS grammar G, and then applying an algorithm due to Abdulla *et al.* (2002, 2003) that takes an

FST as input and attempts to construct an FST computing its transitive closure. Subsection 6.1 reviews this algorithm in detail, including sufficient conditions for its termination. Subsection 6.2 argues that HS grammars violate these conditions exactly when performing coordinated deletions. In particular, I will show that HS grammars with SL markedness constraints are rational if they cannot perform deletion.

6.1 *Transducer iteration*

Computing the transitive closure of a relation is a difficult problem. Since transitions between Turing machine configurations can be modelled by rational relations, the halting problem can be reduced to finding the transitive closure of a rational relation. Nonetheless, the problem of computing transitive closures, or approximations thereof, is of substantial practical interest. The field of *model checking*, for example, is concerned with determining what states of a program or other computational system can be reached from an initial configuration, and in particular whether any of these reachable states indicate undesirable behavior. Several studies in this area have explored the possibility of performing reachability analysis using FSTs by modelling state transitions as rational relations. To that end, partial algorithms have been developed that attempt to produce FSTs computing the transitive closures of rational relations. One approach, developed by Bouajjani *et al.* (2000), Jonsson and Nilsson (2000), Abdulla *et al.* (2002), and Abdulla *et al.* (2003), considers infinite-state transducers computing transitive closures and defines a behavior-preserving equivalence relation under which the state set has a finite quotient. Another approach, due to Dams *et al.* (2001a,b, 2002), also attempts to produce a finite quotient of an infinite state set, but the equivalence relation is constructed algorithmically while computing increasingly large compositions of an FST with itself.

The transducer iteration algorithm used in Hao (2017) is the quotient-based algorithm of Abdulla *et al.* (2002) and Abdulla *et al.* (2003). Since this algorithm is only compatible with FSTs that perform substitutions, Hao's construction simulates insertions and deletions by thinking of them as substitutions involving a designated alphabet symbol representing λ. Abdulla *et al.*'s technique relies on the observation

that for any FST T, the composition of $[T]$ with itself n times for some fixed n is a rational relation. An infinite-state transducer for $[T]^+$ is produced using a construction for the n-fold composition of T by taking n to infinity. The size of the state set is reduced by identifying states with the same behavior and merging them. The algorithm terminates if finitely many states remain after merging is complete. This termination condition gives us a sufficient condition for the rationality of $[T]^+$.

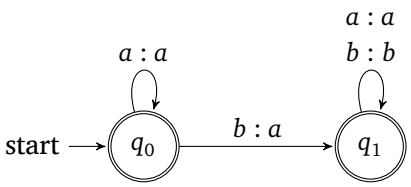
Subsection 6.1.1 describes the construction for the infinite-state transducer computing $[T]^+$. Subsection 6.1.2 presents the equivalence relation used to collapse the infinite state set into a possibly finite one.

6.1.1 Column transducers

To understand the construction for the n-fold composition of an FST, let us consider a concrete example.

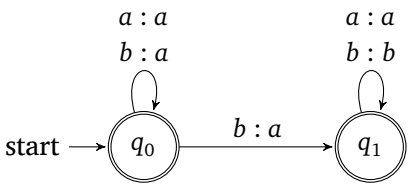
Example 42. Consider the following FST over the alphabet $\Sigma = \{a, b\}$.

(43) An FST that changes the first b to an a



The behavior of FST (43) is to change the first b it encounters to an a. The transitive closure of this relation would change the first n instances of b to as, where the value of n is chosen nondeterministically. This is clearly finite-state, since it can be computed by the FST shown below in (44).

(44) An FST that changes the first n-many bs to as



The intuition behind Abdulla *et al.*'s technique is as follows. We can understand the behavior of an FST on a particular input by inspecting its *run* – the sequence of states entered into during the course

of the computation. For example, (45) shows the runs of (43) when it is applied twice to the input $aabb$, producing first $aaab$ and then $aaaa$.

(45) Runs of FST (43) on input $aabb$

Time:	0		1		2		3		4
Input 1:		a		a		b		b	
Run 1:	q_0		q_0		q_0		q_1		q_1
Input 2:		a		a		a		b	
Run 2:	q_0		q_0		q_0		q_0		q_1
Input 3:		a		a		a		a	

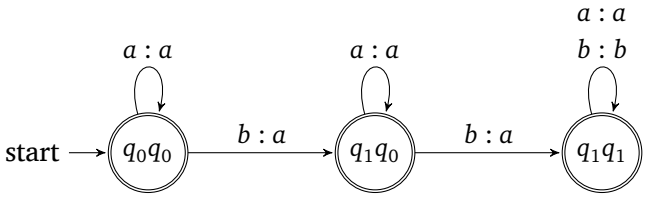
During the first run of (43), the FST is in state q_0 for three time steps, and then state q_1 for two time steps. During the second run, (43) is in state q_0 for four time steps, and then state q_1 for one time step. The behavior of (43) when it is applied twice to some input can be described by inspecting the *columns* of table (45). These columns describe, for each time step, the state of the FST during each of its iterations. Based on this, we may construct an FST that simulates two iterations of (43) by taking each state to represent a column of (45). The run of such a transducer, shown in (46), resembles table (45), except that the "Input 2" row is omitted and the "Run 1" and "Run 2" rows are merged.

(46) Combining the two runs of (45)

Time:	0		1		2		3		4
Input:		a		a		b		b	
Run:	q_0 q_0		q_0 q_0		q_0 q_0		q_1 q_0		q_1 q_1
Output:		a		a		a		a	

Whenever the transitions $p \xrightarrow{x:y} q$ and $r \xrightarrow{y:z} s$ occur in (43), an FST whose behavior is described by (46) has the transition $pr \xrightarrow{x:z} qs$. For example, between time steps 2 and 3, (45) shows that (43) undergoes the transition $q_0 \xrightarrow{b:a} q_1$ during its first run and $q_0 \xrightarrow{a:a} q_0$ during its second run. Accordingly, (46) shows that the 2-fold iteration of (43) undergoes the transition $q_0q_0 \xrightarrow{b:a} q_1q_0$ from time step 2 to time step 3. By combining all possible transitions of (43) in this way, we obtain the FST shown below.

(47) The composition of (43) with itself



The three states of (47) represent the three possible values that might appear in a column of table (45). The FST is in state q_0q_0 when it has not seen any bs. The FST enters state q_1q_0 after seeing the first b. This represents the fact that (43) enters state q_1 on upon seeing b in its first iteration. However, because the b is changed to an a, (43) does not enter state q_1 in its second iteration until the second b of the original input is seen. When this happens, (47) enters state q_1q_1. It is clear that the behavior of (47) is to change the first two bs of its input to as.

Note that the state q_0q_1 does not appear in (47). This is because (43) can only be in state q_0 at time t if it has not emitted any bs, so the output of the first run cannot contain any bs before time t. However, (43) can only be in state q_1 if it has seen at least one b. Since the input to the second run, which is the output of the first run, does not contain any bs before time t, (43) cannot enter state q_1 during its second run until after time t.

The ideas discussed in Example 42 are formalized as follows.

Definition 48. An FST $\langle Q, \Sigma, \Gamma, I, F, \to\rangle$ is *same-length* if for every $q, r \in Q$, $q \xrightarrow{a:b} r$ implies that $|a| = |b| = 1$.

Definition 49. Let $T = \langle Q, \Sigma, \Sigma, I, F, \to\rangle$ be a same-length FST. For each $n > 1$, the *n-fold column transducer for* T is defined as the transducer $T^n := \langle Q^n, \Sigma, \Sigma, I^n, F^n, \to_n\rangle$, where for each $q_1q_2\dots q_n, r_1r_2\dots r_n \in Q^n$,

$$q_1q_2\dots q_n \xrightarrow{a:b}_n r_1r_2\dots r_n$$

holds if and only if there exist $a_0, a_1, \dots, a_n \in \Sigma$ such that $a_0 = a$, $a_n = b$, and for each $i \in \{1, 2, \dots, m\}$, $q_i \xrightarrow{a_{i-1}:a_i} r_i$.

For each n, $[T_n]$ is the n-fold composition of $[T]$ with itself. A transducer for $[T]^+$ is created by taking the union of all the T_ns.

Definition 50. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. The *column transducer for* T is defined as the transducer $T^+ := \langle Q^+, \Sigma, \Sigma, I^+, F^+, \rightarrow_+ \rangle$, where

$$\rightarrow_+ = \rightarrow \cup \Big(\bigcup_{n=2}^{\infty} \rightarrow_n \Big).$$

Since T^+ has infinitely many states, it is not an FST. The next subsection shows how we can attempt to reduce the size of the state set by taking its quotient under an equivalence relation. An FST equivalent to T^+ is obtained if the quotient is finite.

6.1.2 Quotient transducers

The technique for merging states is based on eliminating repetitions of *copying states* – states whose behavior is to copy the input.

Definition 51. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. A state q is *left-copying* if for every start state $q_0 \in I$, $q_0 \xrightarrow{a:b}_* q$ implies $a = b$. A state q is *right-copying* if for every accept state $q_f \in F$, $q \xrightarrow{a:b}_* q_f$ implies $a = b$. A state q is *non-copying* if it is neither left-copying nor right-copying. A state q is *copying* if it is not non-copying.

For a state q to be left-copying means that the only way to reach q is for the transducer to copy its input. For q to be right-copying means that once q has been reached, the only possible action of the transducer is to copy its input. The equivalence relation on the states of T^+ is defined by merging any two columns that are identical except for repetitions of copying states of T.

Definition 52. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. Define the equivalence relation $\simeq_T$ over Q^+ as follows. We say that $q \simeq_T r$ if and only if we can write

$$q = q_1^{p_1} q_2^{p_2} \dots q_m^{p_m}$$
$$r = q_1^{r_1} q_2^{r_2} \dots q_m^{r_m},$$

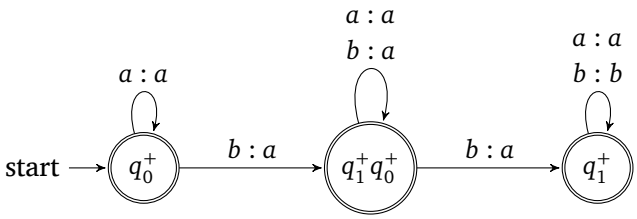
where $p_i = r_i$ whenever q_i is non-copying and for each j, $p_j > 0$ and $r_j > 0$. The *quotient transducer for* T is defined as the transducer $T_{\simeq} := \langle Q^+/\simeq, \Sigma, \Sigma, I/\simeq, F/\simeq, \Rightarrow \rangle$, where $[q]_{\simeq_T} \overset{a:b}{\Longrightarrow} [r]_{\simeq_T}$ if and only if $q \xrightarrow{a:b}_+ r$.

Since $\simeq$ does not distinguish between multiple repetitions of the same copying state, a canonical notation for equivalence classes of $\simeq$ can be defined by replacing $q_i^{p_i}$ and $q_i^{r_i}$ with q_i^+ whenever q_i is a

copying state. For example, if q is copying but p and r are not, then the column $p^8q^5r^9$ belongs to the equivalence class $p^8q^+r^9$.

Example 53. Consider again the FST (43). The state q_0 is left-copying, since the only transition reaching q_0 copies an a to the output stream. The state q_1 is right-copying, since the unique transition from q_1 copies an a or a b to the output stream. In (47), we saw that the columns of length 2 for the column transducer of (43) are q_0q_0, q_1q_0, and q_1q_1. In general, columns obtained by iterating (43) are of the form $q_1^*q_0^*$. Since q_1 is right-copying and q_0 is left-copying, the quotient transducer for (43), shown below, contains three states: q_0^+, $q_1^+q_0^+$, and q_1^+.

(54) Quotient transducer for (43)



It is easy to see that the transducer above has the same behavior as (44), the transducer computing the transitive closure of (43).

Abdulla *et al.* (2002) prove that merging states in this way does not change the behavior of T^+ under the condition that no reachable column in Q^+ contains a substring of the form pq, where $p \neq q$ and p and q are both left-copying or both right-copying. They ensure that this condition is fulfilled by requiring that T be deterministic. Abdulla *et al.* (2003) relaxes the assumption of determinism by introducing an algorithm that makes the portion of T containing only the left-copying states deterministic, and the portion of T containing only the right-copying states reverse-deterministic. FSTs preprocessed in this manner are called *bideterministic*, and any same-length FST can be bideterminized. Once an FST has been bideterminized, it can be safely used to construct a quotient transducer without changing the behavior of the column transducer.

Theorem 55 (Abdulla *et al.* 2002, 2003)**.** *If T is same-length and bideterministic, then* $[T_{\simeq}] = [T^+] = [T]^+$.

The algorithm for constructing $T_{\simeq}$ from an FST T is as follows. First, T is made bideterministic. Then, the algorithm constructs T^n for n increasingly large, while taking the quotient of the state set

after each iteration. The algorithm terminates when $T^n = T^{n+1}$ after columns have been merged based on $\simeq$.

Algorithm 56: Constructing $T_{\simeq}$ from T, Abdulla *et al.* (2002, 2003)

1: **procedure** CLOSURE(T)
2: Initialize $T_{\simeq} \leftarrow T$, but with each state q replaced with $[q]_{\simeq}$.
3: Make $T_{\simeq}$ bideterministic.
4: **do**
5: Set $T'_{\simeq} \leftarrow T_{\simeq}$.
6: **for** each transition $[q]_{\simeq} \overset{a:b}{\Longrightarrow} [r]_{\simeq}$ of $T_{\simeq}$ and $s \xrightarrow{b:c} t$ of T **do**
7: Add the transition $[qs]_{\simeq} \overset{a:c}{\Longrightarrow} [rt]_{\simeq}$ to $T_{\simeq}$.
8: Remove unreachable states from $T_{\simeq}$.
9: **while** $T_{\simeq} \neq T'_{\simeq}$
10: **return** $T_{\simeq}$.

Each iteration of the algorithm discovers equivalence classes of $T_{\simeq}$ with increasingly longer canonical names. If the length of the canonical name of a reachable state in $T_{\simeq}$ is bounded by some $n \in \mathbb{N}$, then after the nth iteration all reachable states of $T_{\simeq}$ will have been discovered, and the exit condition for the while-loop on line 9 will be satisfied. This gives us the termination condition for the algorithm.

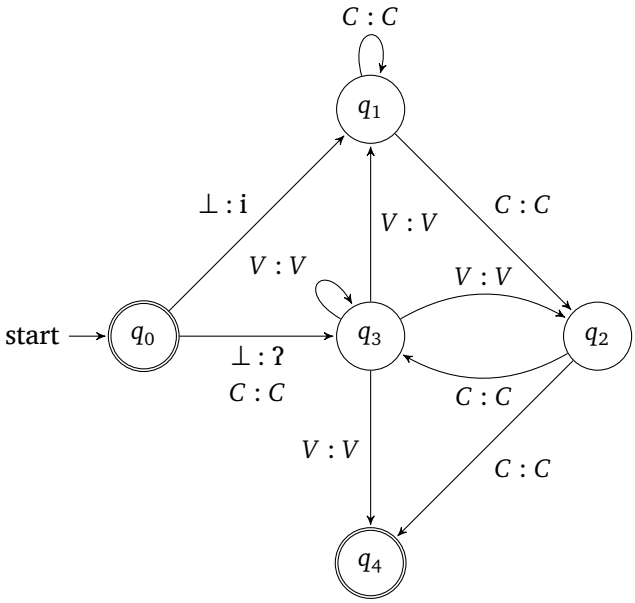
Theorem 57 (Abdulla *et al.* 2002, 2003)**.** *Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. If the following conditions are met, then $[T]^+$ is rational.*

- *There is a bound on the number of non-copying states from Q appearing in the reachable states of T^+.*
- *There is a bound on the number of alternations between left-copying and right-copying states from Q appearing in the reachable states of T^+.*

Example 58. Let G be the HS grammar described at the beginning of Section 4. Recall that this grammar describes a process of Classical Arabic wherein an SR receives the prefix [ʔi] if the UR begins with a consonant cluster. The behavior of $\mathcal{H}_G$ is as follows. If the input begins with a consonant cluster, then $\mathcal{H}_G$ adds an i to the beginning. If the input begins with a vowel, then $\mathcal{H}_G$ adds a ʔ to the beginning. For example, the SR [ʔifʕal] for the UR /fʕal/ "do!" is derived as follows: fʕal $\mathcal{H}_G$ ifʕal $\mathcal{H}_G$ ʔifʕal $\mathcal{H}_G$ ʔifʕal. An FST T with this behavior is shown below. To ensure that T is same-length, the symbol $\perp$

represents λ, and insertions are represented as substitutions of $\perp$ for other symbols. Let us assume that for each state q, T has the transition $q \xrightarrow{\perp:\perp} q$.

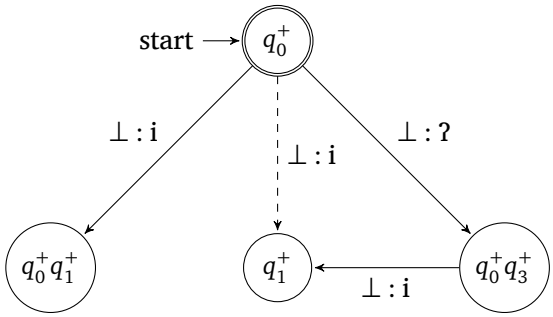
(59) An FST for $\mathcal{H}_G$ (C is a consonant, V is a vowel)



The start state q_0 is a left-copying state, while q_1, q_2, q_3, and q_4 are right-copying states. Observe that the sub-automaton containing only state q_0 is deterministic, while the sub-automaton excluding state q_0 is reverse-deterministic. Therefore, T is already bideterministic.

Let us apply Abdulla *et al.*'s algorithm to T. The equivalence class q_0^+ is the start state of $T_{\simeq}$. Bideterminism guarantees that no equivalence class of $T_{\simeq}$ has a canonical name containing pq where $p \neq q$ and p and q are both left-copying or both right-copying. Therefore, after the first iteration, the new states added to $T_{\simeq}$ are $q_0^+q_1^+$, $q_0^+q_2^+$, $q_0^+q_3^+$, $q_0^+q_4^+$, $q_1^+q_0^+$, $q_2^+q_0^+$, $q_3^+q_0^+$, and $q_4^+q_0^+$. Starting from the start state q_0^+, the new transitions $q_0^+ \overset{\perp:\mathrm{i}}{\Longrightarrow} q_0^+q_1^+$ and $q_0^+ \overset{\perp:c}{\Longrightarrow} q_0^+q_3^+$, with $c = \text{ʔ}$, are added to $T_{\simeq}$ based on the condition in line 6. From $q_0^+q_3^+$, the transition $q_0^+q_3^+ \overset{\perp:\mathrm{i}}{\Longrightarrow} q_1^+$ is added. No other new transitions begin at one of the currently reachable states, so the first iteration terminates here. The new transitions are shown below.

(60) Transitions added during the first iteration (the dashed arrow represents an existing transition)



In the second iteration, the new states added are $q_0^+q_3^+q_0^+$ and $q_0^+q_1^+q_0^+$. Note that (60) has no transitions originating from $q_0^+q_1^+$, and the sole transition originating from $q_0^+q_3^+$ emits a vowel as output. However, the no transition T from q_0 reads a vowel as input, so no new transitions are added. Since $T_{\simeq}$ has not changed during the second iteration, the algorithm terminates, returning an FST with the transitions in both (59) and (60). Observe that the behavior of this FST is to either simulate (59) or to add ʔi before a consonant cluster – exactly the behavior of $[T]^+$.

6.2 *Matching deletion and copying-state alternations*

Theorem 57 identifies two conditions under which the transitive closure $[T]^+$ of a same-length rational relation $[T]$ is rational. This subsection applies Theorem 57 to FSTs computing $\mathcal{H}_G$, developing some intuition on when the transitive closure of $\mathcal{H}_G$ is rational. Observe first that Theorem 57 is simplified in the case of FSTs implementing a single change because the first condition is automatically satisfied.

Proposition 61. *Suppose T is a same-length FST such that $x\ [T]\ y$ if and only if $x \mapsto y$ is a single change. Then, every reachable state of T is left-copying or right-copying.*

Proof. Suppose q is a reachable non-copying state of T. Then, there exist a starting state q_0 and an accepting state q_f such that

- $q_0 \xrightarrow{a:b}_* q$ with $a \neq b$ and
- $q \xrightarrow{c:d}_* q_f$ with $c \neq d$.

Thus, $ac\,[T]\,bd$. However, since $a \neq b$ and $c \neq d$, at least two symbols of ac must differ from their counterparts in bd, so $ac \mapsto bd$ cannot be a change. Thus, T cannot have any reachable non-copying states. □

The remaining condition of Theorem 57 is a bound on the number of alternations between left-copying and right-copying states in the reachable columns of T^+. Suppose T is a same-length FST implementing a single change. If T enters a left-copying state at time t, then it has copied its input between time steps 0 and t, so the single change that T implements occurs after time t. Similarly, if T enters a right-copying state at time t, then the single change occurs before time t, since T can only copy its input after time t. This means that a bound on the number of alternations between left-copying and right-copying states corresponds to a bound on the number of times T can make a change before time t followed by a change after time t and *vice versa*, for each time step t.

Example 62. Define the *matching substitution* function g as follows.

$$g\left(\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}\right) := \begin{cases}\left(a^2b^2\right)^{m-n}\left(\bot^4\right)^{n+1}a^2ca^2\left(\bot^4\right)^{n+1}, & m \geq n \\ \left(\bot^4\right)^{m+1}a^2ca^2\left(\bot^4\right)^{m+1}\left(b^2a^2\right)^{n-m}, & m \leq n\end{cases}$$

This function is like the matching deletion function, except that instead of deleting symbols adjacent to the c, g replaces them with $\bot$. Let G be the HS grammar for the matching deletion function constructed in Section 5, and let T be a same-length FST that implements $\mathcal{H}_G$, except that deleted symbols are replaced with $\bot$.

On input $\left(a^2b^2\right)^{m+1}a^2ca^2\left(b^2a^2\right)^{n+1}$, T reads the symbol c between time steps $t = 4(m+1)+2 = 4m+6$ and $t+1 = 4m+7$. For each a^2b^2 unit and b^2a^2 unit that is changed to $\bot^4$, T makes two substitutions after time t and two substitutions before time t. Thus, at time t, $T_{\simeq}$ is in the state $\left(q_L^+q_R^+\right)^k$, where q_L is left-copying, q_R is right-copying, and k is the total number of a^2b^2 and b^2a^2 units deleted. Since k can be arbitrarily large for large values of m and n, there is no bound on the number of alternations between right-copying and left-copying states in the reachable columns of T^+, so T does not fulfill the termination conditions for Abdulla *et al.*'s (2002) algorithm.

As Example 62 shows, the kind of single-change behavior that is incompatible with Theorem 57 – the kind that alternates between making changes before and after a certain position in the string – is exactly the kind of behavior for $\mathcal{H}_G$ that is used to implement the matching deletion function. This justifies the claim that non-rational mappings in HS with strictly local markedness constraints consist of coordinated changes occurring on opposite sides of some center marker. The following example illustrates how deletion makes such behavior possible.

Example 63. Recall that, in Section 5, the markedness constraint $*baca$ was used to ensure that deletions occurred in the correct order. By specifying a set of banned substrings containing the c, $*baca$ is able to enforce a dependency between the choice of which deletion is carried out and the material that exists on each side of the c.

Now, consider the matching substitution function. It is straightforward to modify the grammar from Section 5 for the matching deletion function so that substitions of a or b for $\perp$ are performed in place of deletions. However, as more and more symbols are changed to $\perp$, the a^2b^2 and b^2a^2 units closest to the c become arbitrarily far away from each other. This suggests that no finite set of banned substrings can enforce a dependency between the two sides of the c, since for long inputs the number of $\perp$s adjacent to the c will exceed the maximum length of a banned substring attempting to enforce the dependency. In other words, deletion makes coordinated changes possible by making the dependency between the deletions *local*.

7 CONCLUSION

In the preceding sections, we have seen that HS can generate a non-rational mapping by performing deletions that occur on opposite, alternating sides of some center marker. These deletions create context-free nesting dependencies between various portions of the deleted material. Although SL constraints by definition can only enforce dependencies across a bounded distance, we saw that deletion was able to extend the reach of SL constraints by moving far-away material into their domains of influence.

The matching deletion function defined in Section 5 differs qualitatively from the majority-rules deletion mappings presented in

Section 3. Majority-rules deletion reflects the ability of standard OT to perform *global* optimization. Because standard OT does not assume GEN to be limited to one change at a time, faithfulness constraints have the ability to measure the degree to which SRs differ from URs. Thus, in the example from Section 3, MAX is responsible for determining which symbol should be deleted from the input string. This determination itself is not rational, since FSTs cannot distinguish between large numbers of as and bs. It is not obvious whether or not HS can implement majority-rules deletion using SL constraints because the limited nature of GEN strips faithfulness constraints of the ability to count the number of symbols deleted across the string, so that both markedness constraints and faithfulness constraints are limited to a local domain of influence. However, as Lamont (2018a,b) shows, majority-rules mappings are possible in HS if markedness constraints are given global scope, compensating for the limited power of faithfulness constraints.

While the majority-rules mapping reflects the global nature of optimization in standard OT, the matching deletion function reflects the derivational nature of HS and the propensity of HS derivations to converge to a fixed point.[12] The constraints $*baca$ and STOP are able to control the deletion process powered by $*ab$ and $*caa$ by exploiting the fact that intermediate strings can encode the previous action of the grammar. Thus, despite the limited power of constraints in HS, complex computations are still made possible if state is encoded into the intermediate strings produced in a derivation. This kind of technique has been used in the HS phonology literature to derive certain complex patterns. For example, McCarthy (2008) gives an HS derivation of *rhythmic syncope*, a mapping in which every second vowel is deleted, by first organizing phonemes into two-syllable feet, then assigning stress to the first syllable in each foot, and then deleting the unstressed vowels. This cascade of processes serves to mark up the UR with syllable boundaries, foot boundaries, and stress markers, so that it is possible for an SL constraint to determine by inspection whether or not a vowel belongs to an even-numbered position within the string. While McCarthy presents this analysis of rhythmic syncope as an ad-

[12] Moreton (1999, 2004) shows that derivations driven by OT-style ranked constraint systems must *always* converge to a fixed point.

vantage of HS over standard OT, the construction of Section 5 shows that encoding state in intermediate strings can enable computations that are too complex for phonology.

While this paper has shown that HS is not rational, I have left open the question of finding a suitable limitation of HS that would eliminate the possibility of generating non-rational mappings. Using Abdulla *et al.*'s algorithm on Hao's (2017) model would provide a method to construct an FST for an HS grammar, although there is no guarantee that the algorithm would terminate. Thus, the approach to finite-state OT studied here is similar to Riggle's (2004) OTCA, which does not impose any *a priori* restrictions on standard OT, but also does not have a guarantee of termination. One possibility for a finite-state restriction might be to replace recursive calls to the grammar with a bounded cascade of distinct phonological processes, in the style of McCarthy's implementation of rhythmic syncope. I leave the development of such ideas for future work.

ACKNOWLEDGEMENTS

I would like to thank Ryan Bennett and Robert Frank for suggesting the line of inquiry that led to this paper and for their extensive discussions on this topic. Andrew Lamont, Dustin Bowers, Nicholas Hathaway, and the audience at FSMNLP provided additional useful feedback. Finally, I would like to thank Frank Drewes and the anonymous reviewers for their rigorous evaluation of this work and their guidance in its development.

REFERENCES

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso (2002), Regular Model Checking Made Simple and Effcient, in Luboš Brim, Mojmír Křetínský, Antonín Kučera, and Petr Jančar, editors, *CONCUR 2002—Concurrency Theory: 13th International Conference, Brno, Czech Republic, August 20–23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pp. 116–131, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45694-0, doi:10.1007/3-540-45694-5_9.

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso (2003), Algorithmic Improvements in Regular Model Checking, in Warren A. Hunt and Fabio Somenzi, editors, *Computer Aided Verification: 15th*

International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003. Proceedings, volume 2725 of *Lecture Notes in Computer Science*, pp. 236–248, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45069-6, doi:10.1007/978-3-540-45069-6_25.

James K. Baker (1975), The DRAGON System—An Overview, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, ISSN 0096-3518, doi:10.1109/TASSP.1975.1162650.

Eric Baković (1999), Assimilation to the Unmarked, in Jim Alexander, Alexis Dimitriadis, Na-Rae Han, Elsi Kaiser, Michelle Minnick Fox, Christine Moisset, and Alexander Williams, editors, *Proceedings of the 23rd Annual Penn Linguistics Colloquium*, volume 6.1 of *University of Pennsylvania Working Papers in Linguistics*, pp. 1–16, Penn Graduate Linguistics Society, Philadelphia, PA, USA.

Eric Baković (2000), *Harmony, Dominance and Control*, PhD dissertation, Rutgers University, New Brunswick, NJ, USA.

Robert C. Berwick (1984), Strong Generative Capacity, Weak Generative Capacity, and Modern Linguistic Theories, *Computational Linguistics*, 10(3-4):189–202, ISSN 0891-2017.

Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili (2000), Regular Model Checking, in E. Allen Emerson and Aravinda Prasad Sistla, editors, *Computer Aided Verification: 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pp. 403–418, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45047-4, doi:10.1007/10722167_31.

Joan Chen-Main and Robert Frank (2003), Implementing Faithfulness Constraints in a Finite State Model of Optimality Theory, in Pádraig Cunningham, Tim Fernando, and Carl Vogel, editors, *Proceedings of the 14th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 28–33, Trinity College Dublin, Dublin, Ireland.

Noam Chomsky (1959), On Certain Formal Properties of Grammars, *Information and Control*, 2(2):137–167, ISSN 0019-9958, doi:10.1016/S0019-9958(59)90362-6.

Noam Chomsky and Morris Halle (1968), *The Sound Pattern of English*, Harper & Row, New York, NY, USA, 1 edition, ISBN 978-0-06-041276-0.

Dennis Dams, Yassine Lakhnech, and Martin Steffen (2001a), Iterating Transducers, Technical Report TR-ST-01-03, University of Kiel Institut für Informatik und praktische Mathematik, Kiel, Germany.

Dennis Dams, Yassine Lakhnech, and Martin Steffen (2001b), Iterating Transducers, in Gérard Berry, Hubert Comon, and Alain Finkel, editors, *13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*,

volume 2102 of *Lecture Notes in Computer Science*, pp. 286–297, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-44585-2, doi:10.1007/3-540-44585-4_27.

Dennis DAMS, Yassine LAKHNECH, and Martin STEFFEN (2002), Iterating Transducers, *The Journal of Logic and Algebraic Programming*, 52–53:109–127, ISSN 1567-8326, doi:10.1016/S1567-8326(02)00025-5.

Matt EDLEFSEN, Dylan LEEMAN, Nathan MYERS, Nathaniel SMITH, Molly VISSCHER, and David WELLCOME (2008), Deciding Strictly Local (SL) Languages, in Jon BREITENBUCHER, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pp. 66–73, College of Wooster, Wooster, OH, USA.

Jason EISNER (2000), Directional Constraint Evaluation in Optimality Theory, in *Proceedings of the 18th Conference on Computational Linguistics*, volume 1, pp. 257–263, Association for Computational Linguistics, Saarbrücken, Germany, ISBN 1-55860-717-X, doi:10.3115/990820.990858.

Jason EISNER (2002), Comprehension and Compilation in Optimality Theory, in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 56–63, Association for Computational Linguistics, Philadelphia, PA, USA, doi:10.3115/1073083.1073095.

Emily ELFNER (2009), Syllabification and Stress-Epenthesis Interactions in Harmonic Serialism, *Rutgers Optimality Archive*, ROA-1047.

Emily ELFNER (2016), Stress-Epenthesis Interactions in Harmonic Serialism, in John J. MCCARTHY, Joe PATER, Vieri SAMEK-LODOVICI, and Armin MESTER, editors, *Harmonic Grammar and Harmonic Serialism*, Advances in Optimality Theory, pp. 261–300, Equinox Publishing, Sheffield, United Kingdom, ISBN 978-1-84553-149-2.

T. Mark ELLISON (1994), Phonological Derivation in Optimality Theory, in *Proceedings of the 15th Conference on Computational Linguistics*, volume 2, pp. 1007–1013, Association for Computational Linguistics, Kyoto, Japan, doi:10.3115/991250.991312.

Robert FRANK and Giorgio SATTA (1998), Optimality Theory and the Generative Complexity of Constraint Violability, *Computational Linguistics*, 24(2):307–315, ISSN 0891-2017.

Dale GERDEMANN and Mans HULDEN (2012), Practical Finite State Optimality Theory, in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pp. 10–19, Association for Computational Linguistics, San Sebastián, Spain.

Dale GERDEMANN and Gertjan VAN NOORD (2000), Approximation and Exactness in Finite State Optimality Theory, in Jason EISNER, Lauri KARTTUNEN, and Alain THÉRIAULT, editors, *Finite-State Phonlogy: Proceedings*

of the Fifth Workshop on the ACL Special Interest Group in Computational Phonology, pp. 34–45, Association for Computational Linguistics, Luxembourg City, Luxembourg.

Kenneth HALE (1973), Deep-Surface Canonical Disparities in Relation to Analysis and Change: An Australian Example, in Thomas A. SEBEOK, editor, *Current Trends in Linguistics*, volume 8: Linguistics in Oceania, pp. 401–458, Mouton, The Hague, Netherlands.

Yiding HAO (2017), Harmonic Serialism and Finite-State Optimality Theory, in Frank DREWES, editor, *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing*, pp. 20–29, Association for Computational Linguistics, Umeå, Sweden, doi:10.18653/v1/W17-4003.

Jeffrey HEINZ (2009), On the Role of Locality in Learning Stress Patterns, *Phonology*, 26(2):303–351, ISSN 0952-6757, doi:10.1017/S0952675709990145.

Jeffrey HEINZ (2014), Culminativity Times Harmony Equals Unbounded Stress, in Harry VAN DER HULST, editor, *Word Stress: Theoretical and Typological Issues*, pp. 255–275, Cambridge University Press, Cambridge, United Kingdom, ISBN 978-1-107-03951-3, doi:10.1017/CBO9781139600408.012.

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*, pp. 58–64, Association for Computational Linguistics, Portland, OR, USA.

Jeffrey Nicholas HEINZ (2007), *Inductive Learning of Phonotactic Patterns*, PhD dissertation, University of California, Los Angeles, Los Angeles, CA, USA.

Patrick W. HOHEPA (1967), *A Profile Generative Grammar of Maori*, number 20 in Indiana University Publications in Anthropology and Linguistics, Waverly Press, Baltimore, MD, USA.

Frederick JELINEK (1976), Continuous Speech Recognition by Statistical Methods, *Proceedings of the IEEE*, 64(4):532–556, ISSN 0018-9219, doi:10.1109/PROC.1976.10159.

C. Douglas JOHNSON (1970), *Formal Aspects of Phonological Description*, PhD dissertation, University of California, Berkeley, Berkeley, CA, USA.

C. Douglas JOHNSON (1972), *Formal Aspects of Phonological Description*, Mouton, The Hague, Netherlands.

Bengt JONSSON and Marcus NILSSON (2000), Transitive Closures of Regular Relations for Verifying Infinite-State Systems, in Susanne GRAF and Michael SCHWARTZBACH, editors, *6th International Conference, TACAS 2000, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25–April 2, 2000, Proceedings*, volume 1785 of *Lecture Notes in Computer Science*, pp. 220–235, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-46419-8, doi:10.1007/3-540-46419-0_16.

Ronald M. Kaplan and Martin Kay (1994), Regular Models of Phonological Rule Systems, *Computational Linguistics*, 20(3):331–378, ISSN 0891-2017.

Lauri Karttunen (1998), The Proper Treatment of Optimality in Computational Phonology, in *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pp. 1–12, Association for Computational Linguistics, Ankara, Turkey.

Regine Lai (2015), Learnable vs. Unlearnable Harmony Patterns, *Linguistic Inquiry*, 46(3):425–451, ISSN 0024-3892, doi:10.1162/LING_a_00188.

Yeeking Regine Lai (2012), *Domain Specificity in Learning Phonology*, PhD dissertation, University of Delaware, Newark, DE, USA.

Andrew Lamont (2018a), Precedence Is Pathological, conference presentation at Phonology in the Northeast, Cambridge, MA, USA.

Andrew Lamont (2018b), Precedence Is Pathological: The Problem of Alphabetical Sorting, conference presentation at the West Coast Conference on Formal Linguistics, Los Angeles, CA, USA.

Bruce T. Lowerre (1976), *The HARPY Speech Recognition System*, PhD dissertation, Carnegie-Mellon University, Pittsburgh, PA, USA.

John J. McCarthy (2007), *Hidden Generalizations: Phonological Opacity in Optimality Theory*, Advances in Optimality Theory, Equinox Publishing, Sheffield, United Kingdom, ISBN 978-1-84553-051-8.

John J. McCarthy (2008), The Serial Interaction of Stress and Syncope, *Natural Language & Linguistic Theory*, 26(3):499–546, ISSN 1573-0859, doi:10.1007/s11049-008-9051-3.

John J. McCarthy (2009), Harmony in Harmonic Serialism, *Rutgers Optimality Archive*, ROA-1009.

John J. McCarthy (2010), An Introduction to Harmonic Serialism, *Language and Linguistics Compass*, 4(10):1001–1018, ISSN 1749-818X, doi:10.1111/j.1749-818X.2010.00240.x.

John J. McCarthy and Alan Prince (1994), The Emergence of the Unmarked: Optimality in Prosodic Morphology, in Mercè Gonzàlez, editor, *Proceedings of the North East Linguistics Society 24*, pp. 333–379, GLSA Publications, Amherst, MA, USA.

John J. McCarthy and Alan Prince (1995), Faithfulness and Reduplicative Identity, *University of Massachusetts Occasional Papers in Linguistics*, 18: Papers in Optimality Theory:249–384.

Kevin McMullin and Gunnar Ólafur Hansson (2016), Long-Distance Phonotactics as Tier-Based Strictly 2-Local Languages, in *Proceedings of the 2014 Annual Meeting on Phonology*, Proceedings of the Annual Meetings on Phonology, pp. 13–24, Linguistic Society of America, Cambridge, MA, USA, doi:10.3765/amp.v2i0.3750.

Kevin James McMullin (2016), *Tier-Based Locality in Long-Distance Phonotactics: Learnability and Typology*, Ph.D. thesis, University of British Columbia, Vancouver, Canada.

Robert McNaughton and Seymour A. Papert (1971), *Counter-Free Automata*, number 65 in Research Monograph, MIT Press, Cambridge, MA, USA, ISBN 978-0-262-13076-9.

Elliott Moreton (1999), Non-Computable Functions in Optimality Theory, *Rutgers Optimality Archive*, ROA-364.

Elliott Moreton (2004), Non-Computable Functions in Optimality Theory, in John J. McCarthy, editor, *Optimality Theory in Phonology: A Reader*, pp. 141–164, Blackwell Publishing, Oxford, United Kingdom, ISBN 978-0-470-75617-1, doi:10.1002/9780470756171.ch6.

Joe Pater (2009), Weighted Constraints in Generative Linguistics, *Cognitive Science*, 33(6):999–1035, ISSN 1551-6709, doi:10.1111/j.1551-6709.2009.01047.x.

Christopher Potts, Joe Pater, Karen Jesney, Rajesh Bhatt, and Michael Becker (2010), Harmonic Grammar with Linear Programming: From Linear Systems to Linguistic Typology, *Phonology*, 27(1):77–117, ISSN 1469-8188, doi:10.1017/S0952675710000047.

Alan Prince (2002), Arguing Optimality, *Rutgers Optimality Archive*, ROA-562.

Alan Prince (2003), Arguing Optimality, *University of Massachusetts Occasional Papers in Linguistics*, 26.

Alan Prince and Paul Smolensky (1993), Optimality Theory: Constraint Interaction in Generative Grammar, Technical Report 2, Rutgers University, New Brunswick, NJ, USA.

Alan Prince and Paul Smolensky (2004), *Optimality Theory: Constraint Interaction in Generative Grammar*, Blackwell Publishing, Malden, MA, USA, ISBN 978-1-4051-1932-0.

Kathryn Pruitt (2008), Iterative Foot Optimization and Locality in Stress Systems, *Rutgers Optimality Archive*, ROA-999.

Kathryn Ringler Pruitt (2012), *Stress in Harmonic Serialism*, PhD dissertation, University of Massachusetts Amherst, Amherst, MA, USA.

Jason Alan Riggle (2004), *Generation, Recognition, and Learning in Finite State Optimality Theory*, PhD dissertation, University of California, Los Angeles, Los Angeles, CA, USA.

James Rogers and Geoffrey K. Pullum (2011), Aural Pattern Recognition Experiments and the Subregular Hierarchy, *Journal of Logic, Language and Information*, 20(3):329–342, ISSN 1572-9583, doi:10.1007/s10849-011-9140-2.

Vieri Samek-Lodovici and Alan Prince (1999), Optima, *Rutgers Optimality Archive*, ROA-363.

Vieri SAMEK-LODOVICI and Alan PRINCE (2002), The Fundamental Properties of Harmonic Bounding, Technical Report TR-71, Rutgers Center for Cognitive Science, Piscataway, NJ, USA.

Michael SIPSER (2013), *Introduction to the Theory of Computation*, Cengage Learning, Boston, MA, USA, 3 edition, ISBN 978-1-133-18781-3.

Rachel WALKER (2008), Gradualness and Fell-Swoop Derivations, conference presentation at the UCSC Graduate Alumni Conference, Santa Cruz, CA, USA.

Rachel WALKER (2010), Nonmyopic Harmony and the Nature of Derivations, *Linguistic Inquiry*, 41(1):169–179, ISSN 00243892, doi:10.1162/ling.2010.41.1.169.