

# Implementing Faithfulness Constraints in a Finite State Model of Optimality Theory

Joan Chen-Main<sup>1</sup> and Robert Frank<sup>1</sup>

**Abstract.** Optimality Theory is a linguistic framework that characterizes surface forms in languages as the resolution of constraint conflict. Work in finite state phonology based on the rewrite rule framework that preceded Optimality Theory suggests phonological mappings fall in the class of rational relations. Previous work has shown that these two characterizations of phonological mappings are reconcilable, subject to certain restrictions. In previous models, the constraints are formalized as referring only to the output side of the mapping. This paper presents a model that optimizes over input-output pairs. This is important for modeling the family of faithfulness constraints, constraints which evaluate the relationship between input and output. The idea is based on the closure of same-length regular relations over intersection and the notion of “lenient intersection,” a combination of ordinary intersection and priority union.

## 1 INTRODUCTION

For those who view the mind as a sort of computer, it is interesting to ask what kind of computer the mind is. The question is ambitious enough to warrant asking about the subcomponents of the mind and the particular subcomponent that this paper is interested in is the subcomponent that computes language. To shed light on what kind of computer is necessary for computing language, it is interesting to ask whether a correlation between natural languages and formal languages exists. Traditionally, phonological mappings in natural languages were explained using ordered sets of rewrite rules and the success of finite state models of many phonological rule systems was taken to suggest that phonological mappings fell within the class of rational relations. The adoption of a new framework, Optimality Theory (OT) (Prince and Smolensky 1993), brought about work on how the mappings allowed by OT could be modeled using finite state means. This paper considers a means for integrating faithfulness constraints, a key part of the Optimality Theory (OT) framework, into a previously proposed model of OT that did not explicitly allow for faithfulness constraints.

The central idea of OT is that surface forms in natural languages are the outcome of competing constraints. The constraints fall into two overarching families, well-formedness constraints and faithfulness constraints. Well-formedness constraints specify what a well-formed surface form should be like. These are also called markedness constraints, because they penalize marked forms. Faithfulness constraints require that information in the input be realized in the surface form. While well-formedness constraints do compete with each other, perhaps the primary competition is characterized as well-formedness constraints standing in conflict with faithfulness constraints.

Within the tradition of modeling GEN and the constraints as weighted finite-state machines (Ellison 1994, Eisner 1997 – as cited by Stabler, 1999), faithfulness constraints can be straightforwardly treated in the same manner as markedness constraints, because evaluation of a candidate against a particular constraint is coded into the system by hand. As pointed out by Eisner (2000), however, this implementation of OT can only generate the set of surface forms for a single underlying form. In contrast, Frank and Satta (1998) and Karttunen (1998) conceive of the optimality system as a system that maps any and every input string to its optimal output form(s). Both papers demonstrate that the OT mapping may be computed with a finite state transducer as long as the system only distinguishes between a bounded number of constraint violations. Their formalizations are both done completely within a finite state calculus and both rely on composition of relations.<sup>1</sup> The system first maps each input to a set of outputs, creating candidate input-output pairs. The job of each constraint is to consider each set of pairs that share the same input and eliminate pairs that are not optimal. That is, among pairs with the same input, the constraint will allow only pairs that minimally violate it to pass onto the next constraint evaluation unless no pairs satisfy the constraint at hand. Thus, when no candidate pairs for a particular input satisfy the constraint at hand, all candidates with the minimal number of violations will be preserved for the next evaluation. When all the constraints have applied, only the optimal mappings should be left.

At first glance, it might seem that a variation on intersection would be the appropriate operation for this conditional elimination, but Frank and Satta and Karttunen are forced to use composition because rational relations are not closed under intersection. Because of this, their construction only allows the constraints to evaluate the output side of the mappings. Reference to only the output side is sufficient for markedness constraints, but faithfulness constraints require simultaneous reference to the underlying material, the input side of the transduction. One way to handle this would be to include all the information from the input and the output in the string that undergoes evaluation. For example, the candidate pair (*digz*, *digz*) could be represented as (*ddlggz*), where an underlined segment represents the underlying form. It is (*ddlggz*) that will undergo evaluation. This has been suggested explicitly in a paper by Eisner (2002), though the idea may have also appeared in Eisner’s earlier work. An alternative version of this approach is to formalize constraints such that both the input and output are considered during evaluation. We may do so using a finite state calculus by requiring the OT mapping to be a same-length rational relation. Because same-length rational relations are closed under

---

<sup>1</sup> Department of Cognitive Science, Johns Hopkins University, Baltimore, MD, USA

<sup>1</sup> Perhaps it is more accurate to say that 1) Frank and Satta use a special type of relation-language intersection, right restriction, that refers only to the output side of a relation and that 2) the right restriction operation can be recast, as shown by Karttunen, as a variation on composition.

intersection, this allows us to use a variation on the intersection operation.

Phonological processes, however, include deletion and epenthesis (insertion) of segments, suggesting that phonological mappings are not same length. We will also provide a solution for this in this paper.

Section 2 both lays out the proposed model and provides some background work on which this work is based. Section 3 describes how the model works and addresses deletion and epenthesis. Section 4 contains concluding remarks.

## 2 A MODEL OF OT AND OPERATIONS FOR CONSTRAINT EVALUATION: OLD IDEAS AND VARIATIONS ON OLD IDEAS

This section lays out the pieces that we will use.

### 2.1 A model of OT – Variations on Frank and Satta (1998)

The definition below is a modification of Frank and Satta's formalization of an OT system:

Definition

An optimality system (OS) is a four-tuple  $G = (\Sigma, \Pi, \text{GEN}, C)$ , where

$\Sigma$  is a finite alphabet,

$\Pi$  is the set of pairs  $(x:y)$  where  $(x:y) \in \mathbb{R}[\Sigma \times \Sigma]$

$\text{GEN} = \Pi^*$ , and

$C = \langle c_1, \dots, c_p \rangle$ , where  $p \geq 1$ , is an ordered sequence of total functions from  $\Pi^*$  to  $\mathbb{N}$ .

Definition of an optimality system

As in Frank and Satta's model, we may think of members of the domain of  $\text{GEN}$  as input strings, and members of the range of  $\text{GEN}$  as candidate outputs.  $\text{GEN}$  may map each input string to multiple candidate outputs.

Unlike Frank and Satta, however, this definition treats each transduction pair of one input symbol and the output symbol it maps to as a unit.  $\Pi$  is the set of permitted pairs and  $\text{GEN}$  is the result of concatenating any number of permitted pairs. Thus,  $\text{GEN}$  is  $[\Sigma \times \Sigma]^*$ , which is a subset of  $\Sigma^* \times \Sigma^*$ . The strings in the former are automatically "same length" for input and output, but this is not so for the latter. This distinction is important in later discussion]

With faithfulness constraints in mind, we would like to allow constraints to refer to both the input and output strings. In the definition above, each function  $c_i$  in  $C$  represents a constraint. When  $c_i$  is applied to a member of  $\Pi^*$ , the function assigns the pair of strings a non-negative integer. This number is the "degree of violation," i.e. the number of violations of the  $i$ th constraint that the particular string pair incurs. The co-domain of  $c_i$  is the set of possible numbers of violations  $c_i$  can assign. In typical OT research, the co-domain of  $c_i$  has an unbounded number of members. In an OS that incorporates the bounded evaluation proposal (Frank and Satta, 1998; Karttunen, 1998) mentioned above, the co-domain of each constraint is finite.

This definition of constraints is slightly different from the definition used by Frank and Satta. In the definition above, constraints map string pairs to number of violations. In Frank and

Satta's model, constraints map single strings to number of violations. In later comparison, it will be useful to be able to refer to Frank and Satta's conception of constraints, so let us review it here.

$M = \langle m_1, \dots, m_p \rangle$ , where  $p \geq 1$ , is an ordered sequence of total functions from  $\Sigma^*$  to  $\mathbb{N}$ .

Frank and Satta's definition of constraints<sup>2</sup>

Of the candidates received for evaluation by  $m_i$ , the candidates assigned the lowest number by  $m_i$ , i.e. which minimally violate  $m_i$ , are the candidates that pass on for evaluation against the next constraint.  $\text{OT}^i$  will denote the relation that results after constraints 1 through  $i$  have been applied. In the case that every candidate in  $\text{OT}^{i-1}$  violates  $m_i$  to the same degree, then  $\text{OT}^i$  will be the same as  $\text{OT}^{(i-1)}$ . To facilitate reference to the set that passes on, Frank and Satta define a set called  $\text{argmin}_c\{S\}$ , where  $S$  is another set and  $c$  is a function which assigns members of  $S$  a value.  $\text{argmin}_c\{S\}$  is the subset of  $S$  that is assigned the lowest value by function  $c$ . Because the model has been modified, the set that passes on must be defined slightly differently. Below, we define  $\text{argmin}_c\{F(x)\}$ , where  $F$  is some function,  $F(x)$  is a set, and  $c$  is a function which assigns values to pairs  $(x, y)$  with  $y \in F(x)$ .

$$\text{argmin}_c\{F(x)\} = \{y \mid y \in F(x), \quad c(x, y) = \min\{c(x, y') \mid y' \in F(x)\}\}$$

Definition of argmin

Frank and Satta also provide a definition for the map that an OS induces:

$$\text{OT}_G^i(w) = \begin{cases} [\text{GEN}](w) & \text{if } i = 0; \\ \text{argmin}_{c_i}\{\text{OT}_G^{i-1}(w)\} & \text{if } i \geq 1; \end{cases}$$

Definition of map induced by an optimality system

Function  $\text{OT}_G^p$  is called the optimality function associated with a particular triple  $G$ . We will follow Frank and Satta's convention of dropping the subscript when there is no ambiguity. This is the same definition we will use, but our mapping uses the modified definition of  $\text{argmin}_c\{F(x)\}$ .

### 2.2 Requirements for Rationality – from Frank and Satta (1998)

Frank and Satta's main result includes three requirements for a the mapping of an OS to be rational.

Let  $G = (\Sigma, \text{GEN}, M)$  be an OS such that  $\text{GEN}$  is a rational relation and each constraint in  $M$  is regular and has a finite co-domain. Then  $\text{OT}_G$  is a rational relation.

Frank and Satta's (1998) main result

<sup>2</sup> Frank and Satta called the set of constraints  $C$  and the members  $c_i$ . We call the set  $M$ , because all the members are markedness constraints. Also, we want to reserve  $C$  to refer to a set containing both faithfulness and markedness constraints.

First, GEN must be rational. Allowing GEN to be beyond the power of finite state transducers would ensure that  $OT^0$  would not be rational. In the model presented here, we have a rational GEN by definition. Additionally, the constraints in  $M$  are assumed to be regular in that each  $m$  satisfies the following requirement: For each  $q \in \mathbb{N}$ , the set  $\{w \mid w \in \Sigma^*, m_i(w) = q\}$  is a regular language. For example, all strings in  $\Sigma^*$  that had exactly two violations of  $m$  would constitute a regular language. Let us call this the constraint language  $Lm_{i,2}$ . The  $i$  denotes the particular constraint and the 2 denotes number of violations. Frank and Satta note that nearly all the constraints proposed in the OT literature are regular in this sense. Lastly, constraint evaluation must be bounded, as mentioned above. This is the condition that each constraint have a finite co-domain.

### 2.3 Lenient Composition – from Karttunen (1998)

Karttunen defines an operation called lenient composition and uses this to formalize OT. Against the backdrop of requirements for rationality, the idea is the same as that in Frank and Satta. Karttunen encapsulates constraint application as a finite state operation.

Lenient composition is a combination of ordinary composition and priority union, an operator introduced by Kaplan (1987). Given two relations,  $Q$  and  $R$ , the priority union of  $Q$  and  $R$ , denoted  $Q.P.R$ , is  $Q$  together with any pairs in  $R$  where the pair's input side is not an input in  $Q$ . Karttunen provides the following illustration:

$$\begin{aligned} Q &= \{(a : x), (b : y)\} \\ R &= \{(b : z), (c : w)\} \\ Q.P.R &= \{(a : x), (b : y), (c : w)\} \end{aligned}$$

Example of priority union

$Q$  maps  $a$  to  $x$  and  $b$  to  $y$ .  $R$  maps  $b$  to  $z$  and  $c$  to  $w$ .  $Q.P.R$  maps  $a$  to  $x$ ,  $b$  to  $y$ , and  $c$  to  $w$ .  $Q.P.R$  does not map  $b$  to  $z$ , because  $b$  is an input in  $Q$ .

A definition of the priority union operator  $.P.$  is given below.  $Dom(Q)$  is the language that includes all the inputs of  $Q$  and  $Id(L)$  is the identity relation that carries every member of a regular language  $L$  into itself.<sup>3</sup>

$$Q.P.R = Q \cup (\overline{Id(Dom(Q))} \circ R)$$

Definition of priority union

When two relations  $R$  and  $C$  are leniently composed, we first compose  $R$  with  $C$ . We then take the resulting relation  $R \circ C$  and priority union it with  $R$ . Lenient composition, denoted  $.O.$ , is defined below.

$$R.O.C = (R \circ C).P.R$$

Definition of lenient composition, compact version

$$R.O.C = (R \circ C) \cup (\overline{Id(Dom(R \circ C))} \circ R)$$

Definition of lenient composition, spelled out version

To see how lenient composition fits into Karttunen's model of OT, let us think of  $R$  as GEN and  $C$  as the identity relation on the language which satisfies the first constraint. Since GEN is rational and  $C$  is rational, the composition of GEN and  $C$  is also rational.  $GEN \circ C$  will include all string pairs whose output satisfies constraint 1. The inputs in GEN that do not map to any output which satisfies constraint 1 will not be mapped to anything by  $GEN \circ C$ . These inputs, however, will be mapped to some output(s) by the relation on the right side of the union symbol.  $Dom(GEN \circ C)$  refers to the domain of  $GEN \circ C$ . We know  $Dom(GEN \circ C)$  is a regular language because the domain of a rational relation is a regular language. We also know that  $Dom(GEN \circ C)$ , i.e. the strings which do not get mapped to anything in  $GEN \circ C$ , is also a regular language because regular languages are closed under complementation. Since an identity relation on a regular language yields a rational relation,  $Id(Dom(GEN \circ C))$  is rational. When we compose this relation with GEN, we will get all the string pairs in GEN whose inputs did not map to any output satisfying constraint 1. Because  $Id(Dom(GEN \circ C))$  and GEN are both rational, their composition yields a rational relation. Both relations on either side of the union symbol are rational and rational relations are closed under union. Therefore, the entire expression on the right of the equals sign is a rational relation. During the next step, we can think of  $R$  as the relation we get after applying constraint 1 to GEN and  $C$  as the identity relation on the language which satisfies the second constraint. As described here, this assumes that each constraint can be violated at most once. However, if we have bounded constraint evaluation, we can recast a constraint such as  $*a$  to be  $*(one\ a)$ ,  $*(two\ a's)$ ,  $*(three\ a's)$ , etc.

When constraint evaluation is bounded, Frank and Satta's definition of the map an OS induces can be recast using Karttunen's lenient composition operation.

$$OT_G^i(w) = \begin{cases} [GEN](w) & \text{if } i = 0; \\ OT_G^{i-1}.O.Id(Lm_{i,j}) & \text{if } i \geq 1; \\ \text{where } 0 \leq j \leq k \end{cases}$$

Definition of map induced by an optimality system with constraint evaluation bounded at  $k$

When two relations  $R1$  and  $R2$  are composed, it is the output of  $R1$  that becomes the input of  $R2$ . The input of  $R1$  associated with that output of  $R1$  does not play a role in determining whether or not the output of  $R1$  will map to anything in  $R2$ . In this sense, composition only allows reference to the output side of the transduction, which is well suited for markedness constraints.

## 3 OT AS A SAME LENGTH RATIONAL RELATION

### 3.1 What does Gen look like?

In the definition of an OS given above (section 2.1), GEN is defined as  $\Pi^*$ . Recall that  $\Pi$  consists of pairs of symbols. The pairs in  $\Pi$  contain all the alphabet symbols, but they do not contain

<sup>3</sup> Karttunen gives his definition in the Xerox calculus. The definition here is the translation into standard notation.

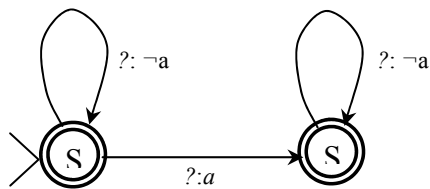
$\varepsilon$ . Since pairs such as  $(x: \varepsilon)$  and  $(\varepsilon: y)$  are not members of  $\Pi$ , GEN is a same-length relation.<sup>4</sup> From the definition given for GEN, we also know GEN is a rational relation.

Same-length rational relations are the subset of the class of rational relations that consist of string pairs  $(x, y)$  such that the length of  $x$  is the same as the length of  $y$ . Kaplan and Kay (1994) discuss the properties of such relations. Because they are a subclass of rational relations, all properties of rational relations, such as closure under union, are also properties of same-length rational relations. Same-length rational relations, however, have properties that are not properties of rational relations in general. What is relevant for this paper is that while rational relations are not closed under intersection, same-length rational relations are.

We also know, however, that due to epenthesis and deletion, pairs of underlying and surface forms in phonology are not always the same length. To allow for epenthesis and deletion, we can use the symbol  $\theta$ , described by Kaplan and Kay (1994) as having been used in two-level systems which model derivational phonological rule systems. Like the empty string,  $\varepsilon$ ,  $\theta$  has no pronunciation. Unlike  $\varepsilon$ , however,  $\theta$  is a bona fide alphabet symbol. Therefore, supposing that  $a$  is also in  $\Sigma$ ,  $(a:\theta)$ ,  $(\theta:a)$ , and  $(\theta:\theta)$  will all be members of  $\Pi$ . This means we will use a representation such as  $(\text{tends}, \text{ten}\theta z)$  instead of  $(\text{tends}, \text{ten}z)$ .

### 3.2 What do the constraint relations look like?

In section 2.2, constraints in  $M$  were described as regular in that each  $m$  divided up  $\Sigma^*$  into regular languages by number of violations. Since the constraints in  $C$  map string pairs to number of violations, we will have to modify our notion of regular constraints. Constraints in  $C$  are assumed to be regular in that each  $c$  satisfies the following requirement: For each  $q \in \mathbb{N}$ , the set  $\{(u, w) \mid (u, w) \in \Pi^*, c_i(u, w) = q\}$  is a regular relation. Just as all strings in  $\Sigma^*$  that had exactly  $j$  violations of  $m$  would constitute a regular language, so all string pairs in  $\Pi^*$  that had exactly  $j$  violations of  $c_i$  would constitute a regular relation. Let us call this the constraint relation  $Rc_{i,j}$ . The  $i$  denotes the particular constraint and the  $j$  denotes number of violations. For example, if the constraint  $c_1$  penalizes  $a$ 's in the output, the finite state transducer corresponding to  $Rc_{1,1}$ , the relation whose output strings have at most one  $a$ , is drawn below.  $?$  means any alphabet symbol.



Finite state transducer corresponding to  $Rc_{* (? : a), 1}$

Because the constraint penalizes pairs, it is more accurate to think of  $c_1$  as  $*(?, a)$  rather than  $*a$ . Note that when  $c_i$  and  $m_i$  are markedness constraints penalizing the same forms, we can describe this transducer as accepting any string pair when the input is a member of  $\Sigma^*$  and the output is a member of  $Lm_{i,j}$ . The relation

<sup>4</sup> We could allow  $(\varepsilon:\varepsilon)$  to be a member of  $\Pi$  without changing the same-length property of GEN, though this does not have any linguistic relevance. Disallowing  $(x: \varepsilon)$  and  $(\varepsilon: y)$  as members of  $\Pi$  is the crucial condition.

it accepts is equivalent to the right restriction of  $\Sigma^* \times \Sigma^*$  and  $Lm_{i,j}$ . That is,

$$Rc_{i,j} = \text{Rrst}(\Sigma^* \times \Sigma^*, Lm_{i,j}).$$

It follows that the range of the constraint relation for  $c_i$  is the same as the constraint language for the constraint  $m_i$ . That is,

$$\text{Range}(Rc_{i,j}) = Lm_{i,j}.$$

### 3.3 What operation do we use for constraint evaluation?

As its name suggests, the definition of lenient intersection is based on the definition of lenient composition. Lenient intersection is a combination of ordinary intersection and priority union. When two relations  $R$  and  $C$  are leniently intersected, we first intersect  $R$  with  $C$ . We then take the resulting relation  $(R \cap C)$  and priority union it with  $R$ . Lenient intersection, denoted  $.I.$ , is defined below.

$$R.I.C = (R \cap C) .P. R$$

Definition of lenient intersection, compact version

$$R.I.C = (R \cap C) \cup (\text{Id}(\text{Dom}(R \cap C)) \circ R)$$

Definition of lenient intersection, spelled out version

Using the closure properties of same-length rational relations and regular languages, we can show that same-length rational relations are closed under lenient intersection. If  $R$  and  $C$  are same-length rational relations, then  $R \cap C$  is a same-length rational relation as well. This means that  $\text{Dom}(R \cap C)$  is a regular language. This in turn means that  $\text{Id}(\text{Dom}(R \cap C))$  is a same-length rational relation. The identity relation on that language,  $\text{Id}(\text{Dom}(R \cap C))$ , is a same-length rational relation. Composing two same-length rational relations,  $\text{Id}(\text{Dom}(R \cap C)) \circ R$ , yields another same-length rational relation. Since both relations on either side of the union symbol are same-length rational relations and same-length rational relations are closed under union, the entire expression on the right of the equals sign is a same-length rational relation.

Lenient intersection will play a role that is very similar to the role played by lenient composition in the old system.

Having discussed the relationship between markedness constraints as previously formulated and markedness constraints as formulated here in section 3.2, let us check that the two formulations are consistent. That is, let us check that

$$\text{OT} .O. \text{Id}(Lm_{i,j}) = \text{OT} .I. Rc_{i,j}.$$

Let us first describe the members of  $\text{OT} .O. \text{Id}(Lm_{i,j})$ .

From the definition of lenient composition, we can rewrite this relation.

$$\text{OT} .O. \text{Id}(Lm_{i,j}) = [\text{OT} \circ \text{Id}(Lm_{i,j})] .P. \text{OT}$$

The relation on the left of the priority union operator is the composition of OT with  $\text{Id}(Lm_{i,j})$ . This composition will yield a subset of OT such that each member pair's output is a member of  $Lm_{i,j}$ . This is the same as the right restriction of OT to  $Lm_{i,j}$ .

$$\text{OT} .O. \text{Id}(Lm_{i,j}) = [\text{Rrst}(\text{OT}, Lm_{i,j})] .P. \text{OT} \quad (\spadesuit)$$

Let us now turn to the members of OT .I.  $Rc_{i,j}$ .

From the definition of lenient intersection, we can rewrite this relation.

$$OT .I. Rc_{i,j} = [OT \cap Rc_{i,j}] .P. OT$$

From the discussion above, we have seen that  $Rc_{i,j}$  can be written as the right restriction of  $\Sigma^* \times \Sigma^*$  and  $Lm_{i,j}$ .

$$OT .I. Rc_{i,j} = [OT \cap Rrst(\Sigma^* \times \Sigma^*, Lm_{i,j})] .P. OT$$

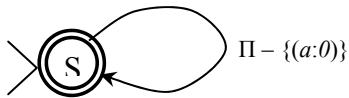
The relation on the left of the priority union operator is the intersection of OT and  $Rrst(\Sigma^* \times \Sigma^*, Lm_{i,j})$ . The input side of OT, i.e.  $Dom(OT)$ , is a subset of  $\Sigma^* \times \Sigma^*$ . This means that the input side of the intersected relations will be a subset of the input side of OT. The output side of the intersected relations will be members of the output side of OT that are also members of  $Lm_{i,j}$ . Thus, the result of the intersection will be pairs such that the input is a member of  $Dom(OT)$  and the output is a member of  $Lm_{i,j}$ . This is the same as the right restriction of OT to  $Lm_{i,j}$ .

$$OT .I. Rc_{i,j} = [Rrst(OT, Lm_{i,j})] .P. OT$$

By transitivity of equality, using ( $\blacklozenge$ ),

$$OT .O. Id(Lm_{i,j}) = OT .I. Rc_{i,j}.$$

As this paper has continually emphasized, constraints penalize pairs. This allows faithfulness constraints to be implemented in the same manner as markedness constraints. Suppose a constraint specifies that an underlying  $a$  must not be deleted:  $MAX-a-IO$ . We can think of this as  $*(a:0)$



Finite state transducer corresponding to  $Rc_{*(a:0),0}$

### 3.4 The EPEN Relation

Although we have a means, via using  $0$ , for allowing epenthesis and deletion and keeping the relation same-length, there is a remaining problem. A pair such as  $(aab, aab)$  must compete against pairs such as  $(aa0b, aa0b)$  or  $(a00a0b, a00a0b)$ . Because  $0$  is treated as a real symbol, however,  $aab$ ,  $aa0b$ , and  $a00a0b$  are all different inputs. This will disallow their outputs ought to compete against one another. We can avoid this problem by forcing all inputs to be of the same form in that there are the same number of  $0$ 's between each alphabet symbol corresponding to phonological content. We can think of these input  $0$ 's as empty input slots.

This move is justifiable by appealing to Tesar's (1995, p21) arguments that the constraints must ban cycles of epenthesis if optimality is to be well-defined. If it were the case that epenthesis eventually always increased the harmony of a form, then there would be no optimal output form, because one could always increase the harmony by epenthesizing one more segment. If epenthesis eventually had no effect on the harmony of a form, then an infinite number of optimal descriptions exist since a new optimal form could be created by taking any optimal form and

adding epenthetic segments. At some point, then, epenthesis must begin to decrease the harmony of a form. I assume this is the effect of having constraints in the DEP-IO family and that an OS which includes DEP-IO constraints, there must be a bound on the number of consecutive epenthetic segments.

The number of  $0$ 's between the non- $0$ 's, then, will be the bound on the number of consecutive epenthetic segments. Let us call the bound  $q$ . EPEN will be the identity relation on strings that have  $q$   $0$ 's after each non- $0$ .

$$EPEN = (x_1x_2\dots x_n, 0^qx_10^qx_20^q\dots x_n0^q) \text{ where } x \neq 0 \text{ and } x \in \Sigma$$

Definition of EPEN

### 3.4 Putting the pieces together

The system as a whole will look like this:

$$EPEN \circ ((\dots((\dots((GEN .I. Rc_{1,1}) .I. Rc_{1,2}) \dots) .I. Rc_{1,k}) \dots) .I. Rc_{p,k}) \circ DELETE0's$$

where  $k$  is the bound on constraint evaluation and  $p$  is the number of constraints in  $C$ .

EPEN will pad every input with  $0$ 's to be of the form we want,  $0^q(x0^q)^*$ . GEN will overgenerate, but composing EPEN with GEN will leave only one input representation for each abstract underlying language form. Lastly, we can have a relation that maps each  $0$  to the empty string. After composing EPEN with  $OT^p$  and the resulting relation with  $DELETE0's$ , the relation is no longer same-length, but it is still rational, keeping the mapping within finite state power. That the final mapping is not same-length does not matter since we know that rational relations are closed under composition.

## 4 CONCLUDING REMARKS

This paper presents a model of a system that induces an Optimality Theory mapping that is a modification of the Frank and Satta (1998) model. Specifically, we have changed  $OT^i$ , the mapping after the constraints 1 through  $i$  have applied to GEN, defined constraints to evaluate string pairs, and added the EPEN relation to eliminate the problem created by multiple representations of the same input. When we subject the system to the same requirements for rationality put forth by Frank and Satta (1998), we can answer the question "Can we model the resolution of markedness and faithfulness constraint conflict using finite state means?" with a qualified "Yes." Because this answer is subject to certain restrictions, it cannot provide definitive evidence to bear on the larger questions posed in the introduction. The qualified "yes" does, however, provide some support for the conjecture that a correlation between natural languages and formal languages exists and the conjecture that the subcomponent of the mind that computes phonological mappings need not be more powerful than a finite state machine.

## ACKNOWLEDGEMENTS

Valuable comments from Paul Smolensky in the development of this work are gratefully acknowledged. The first author is supported by a National Defense Science and Engineering Graduate Fellowship.

## References

Eisner, Jason. 2000. Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18<sup>th</sup> International Conference on Computational Linguistics*, 257-263.

Eisner, Jason. 2002. Comprehension and Compilation in Optimality Theory. In *Proceedings of the 40<sup>th</sup> Annual Meeting of the Association for Computational Linguistics*.

Frank, Robert and Giorgio Satta. 1990. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307-315.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331-378. Written in 1980.

Karttunen, Lauri. 1998. The proper treatment of Optimality Theory in computational phonology. In *Finite-state Methods in Natural Language Processing*, 1-12, Ankara.

Prince, Alan and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Manuscript, Rutgers University and University of Colorado, Boulder.

Tesar, Bruce. 1995. Computational Optimality Theory. Doctoral dissertation, Department of Computer Science. University of Colorado, Boulder. Also available at the Rutgers Optimality Archive ROA-90, <http://ruccs.rutgers.edu/roa.html>.

*As cited in Karttunen, Lauri. 1998.:*

Kaplan, Ronald. 1987. Three seductions of computational psycholinguistics. In P. Whitelock, M. M. Wood, H. L. Somers, R. Johnson, and P. Bennett, editors, *Linguistic theory and Computer Applications*. Academic Press, New York. Reprinted in *Formal Issues in Lexical-Functional Grammar*. University of Chicago Press, 1996.

*As cited in Stabler, E. 1999. Notes on computational phonology. Manuscript, UCLA.:*

Eisner, Jason. 1997. Efficient generation in Primitive Optimality Theory. In *Proceedings of the 35<sup>th</sup> Annual Meeting of the Association for Computational Linguistics*.

Ellison, Mark T. 1994. Phonological derivation in optimality theory. In *Proceedings of the 15<sup>th</sup> International Conference on Computational Linguistics*, 1007-1013. Also available at the Edinburgh Computational Phonology Archive.