

## 1. INTRODUCTION

### 1.1 Background

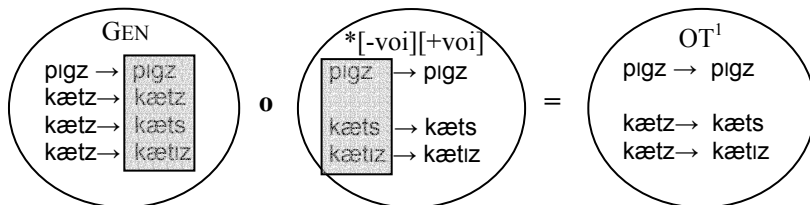
#### 1.1.1 Johnson's (1972) Observation

- The rewrite rules of classical generative phonology require only finite state power to model their effects.

#### 1.1.2 Finite State Optimality Theory

- Finite State Optimality Theory seeks to reconcile
  - the computational phonology characterization of phonological mappings as finite state mappings, i.e. regular relations (Johnson, 1972, Kaplan and Kay, 1980) and
  - the OT characterization of phonological mappings as the resolution of conflict between markedness and faithfulness constraints (Prince and Smolensky, 1993)
- Frank and Satta (1998) and Karttunen (1998):
  - Recent studies have identified limitations that can be imposed on OT that suffice to have it remain within the finite state realm.
  - Constraints are formalized as restrictions on outputs
  - Elimination of candidate outputs is accomplished via a variant of the composition operation on regular relations

(1) e.g. Suppose our 1<sup>st</sup> constraint is \*[-voi][+voi]:



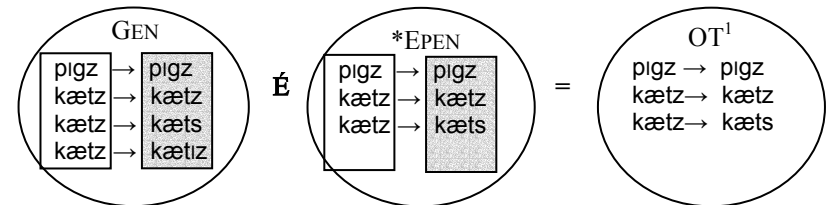
### 1.2 Incorporating Faithfulness

- Faithfulness constraints require simultaneous reference to the input and output sides of the phonological mapping.
- Under formalizations like the ones above, clever bookkeeping is needed to incorporate faithfulness constraints. (e.g. Eisner, 2002)

### 1.3 Our Proposal:

- A formalization of OT that allows transparent implementation of faithfulness constraints

(2) e.g. Suppose our 1<sup>st</sup> constraint is \*EPEN:



### 1.4 Apparent Difficulties:

- Deletion
- Epanthesis

## 2. A FORMAL MODEL OF OT – Variations on Frank and Satta (1998)

### Frank and Satta's Main Result

When ...

- GEN is a rational relation
- Constraints are regular, and
- Constraint evaluation is bounded

... the OT mapping will be a rational relation.

The model presented here will be subject to the same limitations.

### 2.1 What do constraints look like?

<sup>1</sup> Thanks are due to Paul Smolensky and Jason Eisner for very helpful discussion. The authors can be reached at [joan@cogsci.jhu.edu](mailto:joan@cogsci.jhu.edu) and [rfrank@jhu.edu](mailto:rfrank@jhu.edu). The first author is supported by a DoD NDSEG Fellowship.

(3) Interim definition of an optimality system:

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

$\Sigma$  is a finite alphabet,

$\text{GEN}$  is  $\Sigma^* \times \Sigma^*$ , and

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

- Each function  $c_i$  in  $C$  represents a constraint.

(4) Comparison of previous vs. proposed formalizations of constraints

Before	Now
$c_i$ assigns a degree of violation to a <i>string</i>	$c_i$ assigns a degree of violation to a <i>pair of strings</i>
Constraints are formalized as <i>sets of forms</i>	Constraints are formalized as <i>relations between input and output forms</i>
Being regular means . . .	Being regular means . . .
If we let $LC_{i,j}$ = all the <i>strings</i> in $\Sigma^*$ with exactly $j$ violations of $c_i$ , $LC_{i,j}$ is a regular <i>language</i>	If we let $RC_{i,j}$ = all the <i>string pairs</i> in $\Sigma^* \times \Sigma^*$ with exactly $j$ violations of $c_i$ , $RC_{i,j}$ is a regular <i>relation</i>

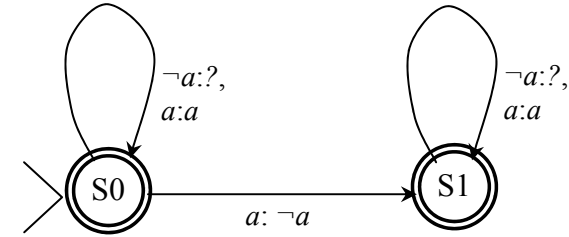
(5) Definition of a regular constraint

$c_i$  is regular if:

For each  $q \in \mathbb{N}$ ,  
the set  $\{(u,w) \mid (u,w) \in \Sigma^* \times \Sigma^*, c_i(u,w) = q\}$  is a regular relation.

e.g. If the constraint  $c_1$  penalizes unfaithful  $a$ 's, the finite state transducer corresponding to  $RC_{1,1}$  is drawn at the top of the next page. (? represents any alphabet symbol)

(6) Finite state transducer corresponding to  $RC_{\text{FAITH-IO}[a],1}$



- When constraint evaluation is bounded at  $k$ , its effects can be modeled using

$$RC_{i,0} \dots RC_{i,k-1}$$

This means we can recast each constraint as a series of binary constraints. For simplicity, we will assume binary constraints and drop the  $j$  subscript from now on.

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

### 2.2.1 What the Operation Must Do

$OT^i$  = the relation that results after constraints 1 through  $i$  have been applied.

(7) The optimality system mapping

$$OT^i = \begin{cases} [\text{GEN}] & \text{if } i = 0; \\ \text{an operation that considers} \\ \text{the input-output pairs for each input and} \\ \text{chooses (for each input) the pairs with outputs} \\ \text{least violating constraint } i \{OT^{i-1}\} & \text{if } i \geq 1; \end{cases}$$

(8) Comparison of previous vs. proposed method of constraint evaluation

Before	Now
Check if the output is in $LC_{i,j}$	Check if the input-output pair is in $RC_{i,j}$
Can use composition	Cannot use composition

However, intersection does allow simultaneous reference to input and output.

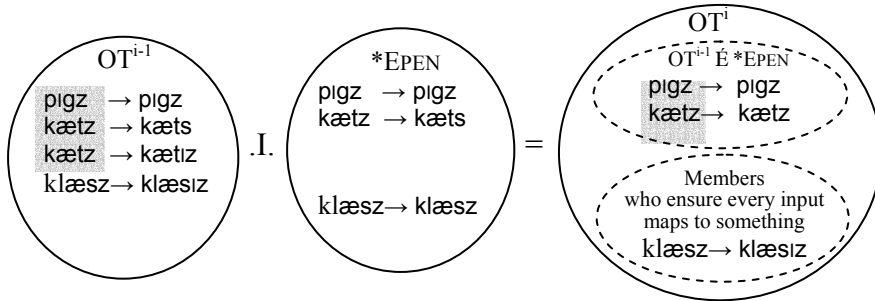
## 2.2.2 Lenient Intersection – variation on Karttunen (1998)

### The Intuition:

For each set of input-output pairs:

- Some member(s) are in  $Rc_i$ : Keep only those pairs
- No members are in  $Rc_i$ : Keep all pairs

(9) e.g. Suppose we have the following  $OT^{i-1}$  and our  $i^{\text{th}}$  constraint is \*EPEN:



### The Formalization:

- A combination of ordinary intersection and priority union (Kaplan, 1987)
- Priority union of  $Q$  and  $R$ :
  - denoted  $Q.P.R$
  - $Q$  union any pairs in  $R$  where the pair's input side is not an input in  $Q$

(10) Example of priority union (from Karttunen 1998)

$$Q = \{(a : x), (b : y)\}$$

$$R = \{(b : z), (c : w)\}$$

$$Q.P.R = \{(a : x), (b : y), (c : w)\}$$

(11) Definition of priority union

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

$Dom(Q)$  is the language that includes all the inputs of  $Q$

$Id(L)$  is the identity relation that carries every member of a regular language  $L$  into itself.

- Lenient Intersection of  $R$  and  $C$ :
  - denoted  $R.I.C$
  - First, intersect  $R$  with  $C$ .
  - Second, take the resulting relation  $(R \cap C)$  and priority union it with  $R$ .

(12a) Definition of lenient intersection, compact version

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

(12b) Definition of lenient intersection, spelled out version

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

When constraint evaluation is bounded, the map an OS induces can be recast using the lenient intersection operation.

(13) Definition of map induced by an optimality system with bounded constraint evaluation

$$OT^i = \begin{cases} [GEN] & \text{if } i = 0; \\ OT^{i-1}.I.Rc_{i,j} & \text{if } i \geq 1; \end{cases}$$

But . . .

## 3. THE PROBLEM WITH USING INTERSECTION

*Regular relations aren't closed under intersection.*

$$(14) \quad \begin{array}{ll} R1 = (a^n, b^n c^*) & \text{regular} \\ R2 = (a^n, b^* c^n) & \text{regular} \\ R1 \cap R2 = (a^n, b^n c^n) & \text{not regular} \end{array}$$

Ambiguous Correspondence:

In  $R1$ , the  $a$ 's correspond to the  $b$ 's.

In  $R2$ , the  $a$ 's correspond to the  $c$ 's.

In  $R3$ , the  $a$ 's correspond to both  $b$ 's and  $c$ 's.

## 4. THE SOLUTION

### 4.1 One more modification

(15) Definition of an optimality system

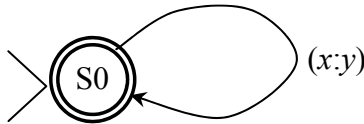
An optimality system (OS) is a three-tuple  $G = (\Sigma, \text{GEN}, C)$ , where

$\Sigma$  is a finite alphabet,

$\text{GEN} = \{(u,v) \mid u, v \in \Sigma^* \text{ and } |u| = |v|\}$

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

(16) When  $x, y \neq \epsilon$ , GEN is represented by the following transducer:



i.e.  $\text{GEN} \neq \Sigma^* \times \Sigma^*$

GEN is the largest subset of  $\Sigma^* \times \Sigma^*$  such that the input and output are the same length

- Pairs such as  $(x: \epsilon)$  and  $(\epsilon:y)$  are not transductions in GEN
- Constraint relations are also same length.<sup>2</sup>

By formalizing GEN and constraint relations in this way,

- Correspondence is no longer ambiguous  
*The  $i^{\text{th}}$  symbol in the input corresponds to the  $i^{\text{th}}$  symbol in the output*
- Using intersection will not exceed finite state power.  
*Same-length regular relations are closed under intersection. (Kaplan and Kay, 1980)*

#### 4.1.1 How Constraints Are Expressed

- Markedness Constraints expressed as:  
ex.  $*(?:a)$  or  $*(?:ab)$
- Faithfulness Constraints expressed as:  
ex.  $*(a:\neg a)$  or  $*(a:0)$  or  $*(0:a)$

## 5. INPUT-OUTPUT DIVERGENCES

<sup>2</sup> I think it is actually sufficient to require only GEN to be a same-length regular relation as long as constraint relations are regular. But I haven't proven this yet. (JCM)

### 5.1 Deletion

- Use the symbol  $0$  (Kaplan and Kay 1980)
- Like the empty string,  $\epsilon$ ,  $0$  has no pronunciation.
- Unlike  $\epsilon$ , however,  $0$  is a bona fide alphabet symbol.  
i.e. Supposing that  $a$  is also in  $\Sigma$ ,  
 $(a:0)$ ,  $(0:a)$ , and  $(0:0)$  will all be allowable transductions.  
e.g. instead of  $tends \rightarrow tenz$ ,  
use  $tends \rightarrow ten0z$

### 5.2 Epenthesis

- Why is introducing  $0$  insufficient for handling epenthesis?
- Every input-output pair associated with the same input form must compete.
- How to get the input-output pairs for  $a0b$  to compete with the input-output pairs for  $ab$ ?
- Modify the priority union operator so that it ignores  $0$ 's in the input. (idea due to Jason Eisner)

(17) Definition of priority union prime (Modification of (11))

$$Q . P' . R = Q \cup (Id(Range(Id(Dom(Q) \circ \text{ADD-OR-DELETE-0's}) \circ R))$$

Where ADD-OR-DELETE-0's inserts or deletes 0's:

(18) Definition of ADD-OR-DELETE-0's

$$\text{ADD-OR-DELETE-0's} = ((x:x) \mid (0:\epsilon) \mid (\epsilon:0))^*$$

Now we can also define a modified lenient intersection:

(19) Definition of lenient intersection prime (Modification of (12))

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

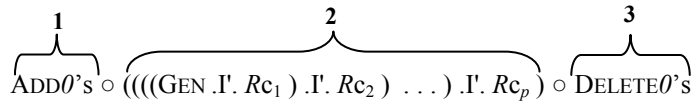
Now the input-output pairs where the input is:

klæss,  
k0l0æ0s0s, or  
k00læss (etc.)

... will compete with each other.

### 5.3 Putting it All Together

The system as a whole will look like this:



### Section 1

- ADD0's will take an underlying form and freely insert 0's

*ADD0's is not a same-length relation. However, we use composition, not intersection, to incorporate ADD0's, so we have not exceeded finite state power.*

### Section 2

- GEN will have multiple representations of each underlying form in its set of inputs.
- .I' will not distinguish between strings with the same non-0 sequence in the input.
- GEN .I' Rc<sub>i,j</sub> will eliminate input-output pairs which are non-optimal with respect to c<sub>i,j</sub>.

*This middle section (between the two composition symbols) is the relation that is crucially same-length, because relations that make up this middle section are combined using intersection.*

### Section 3

- Lastly, DELETE0's maps each 0 to the empty string.

*After composing OT<sup>p</sup> with DELETE0's, the relation is no longer same-length. However, we use composition, not intersection, in this last step, so the resulting relation is still rational.*

## 6. SUMMARY

We have presented a formalization of OT that allows transparent implementation of faithfulness constraints

- Constraints as restrictions on input-output pairs
- Constraint evaluation via intersection of regular relations
  - Closure of same-length regular relations under intersection
- GEN and constraint relations as same-length relations

## References

- 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*.
- 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*.
- 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.
- Frank, Robert and Giorgio Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307-315.
- Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. Mouton, The Hague.
- 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.
- 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://rucss.rutgers.edu/roa.html>.