

Formalizing Optimality Theory: Complexity

Rajesh Bhatt

1 The Overall Logic

- SPE phonology can be modelled as a finite-state transducer (see Johnson (1972), Kaplan and Kay (1994), and Karttunen (1993)).
- OT as it stands cannot be modelled as a finite-state transducer (see Frank and Satta (1998)).
- Thus if OT is taken to be modelling exactly the dependencies handled by SPE phonology, then OT would have more complexity than is needed.
- But if certain restrictions are imposed on the nature of constraints, OT can be modelled as a finite-state transducer (see Frank and Satta (1998) and Karttunen (1998)).

2 The Relevance of Finite State Transducers

2.1 An Introduction to FSTs

Finite State Transducers (FST) can be thought of as paired finite state automata (FSA). They are sometimes also called Mealy's Automata.

- (1) a. Finite State Automata:
 $M = (K, \Sigma, \Delta, s, F)$ is a finite automata,
 K is the set of states, Σ is a finite alphabet, s is the start state,
 $F \subseteq K$ is the set of final states,
 $\Delta \subseteq K \times (\Sigma \cup \{\epsilon\}) \times K$ is the state transition relation.
- b. Finite State Transducers:
 $M = (K, \Sigma_1, \Sigma_2, \Delta, s, F)$ is a finite state transducer,
 K is the set of states, Σ is a finite alphabet, s is the start state,
 $F \subseteq K$ is the set of final states,
 $\Delta \subseteq K \times (\Sigma \cup \{\epsilon\}) \times \Sigma^* \times K$ is the transition relation.

Based on Δ , we can define two functions δ and d .

$$\delta : K \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^K,$$

the state transition function is defined as follows:

$$\delta(q, a) = \{q' : \exists w' \in \Sigma^* [(q, a, w', q') \in \Delta]\}.$$

$$d : K \times (\Sigma \cup \{\epsilon\}) \times K \rightarrow 2^{\Sigma^*},$$

the emission function is defined as follows:

$$d(q, a, q') = \{w' \in \Sigma^* : (q, a, w', q') \in \Delta\}.$$

Unlike FSAs which basically only say yes or no/accept or reject, FST produce a string as an output.

- (2) some very simple examples:
 - a. Upper Case to Lower Case
 - b. Extra Space Removal
 - c. Coke Can Machine

Another less procedural way of thinking about FSTs is to think of them as describing relations between strings. This is going to come in handy when we use them to represent a relation between Underlying Representations and Surface Representations.

Consider the relation between strings $R \subseteq (\Sigma_1^* \times \Sigma_2^*)$.

We will call R a **rational relation** if there is an FST M s.t.

$$\forall w_1 \in \Sigma_1^*, w_2 \in \Sigma_2^* [(w_1, w_2) \in R \rightarrow \exists q \in F [(s, w_1, q) \in \delta^* \wedge (s, w_1, w_2, q) \in d^*]]$$

(δ^* and d^* are the transitive closures of δ and d respectively.)

With this definition in hand, we can talk interchangeably about FSTs and Rational Relations.

For any rational relation R , we can define the following functions:

- Left Projection of R :
 $\text{Left}(R) = \{u \in \Sigma_1^* : \exists v \in \Sigma_2^* [(u, v) \in R]\}$
- Right Projection of R :
 $\text{Right}(R) = \{v \in \Sigma_2^* : \exists u \in \Sigma_1^* [(u, v) \in R]\}$

It can be shown that for any rational relation R , both $\text{Left}(R)$ and $\text{Right}(R)$ are regular languages.

(3) Some Properties of FSTs/Rational Relations:

a. Closed under Union, Concatenation, and Kleene Star

b. Closed under Composition

$$R_3 = R_1 \circ R_2 = \{(u, w) : \exists v[(u, v) \in R_1 \wedge (v, w) \in R_2]\}$$

c. Not Closed under Intersection and Complementation

$$(R_1 = \{(a^i b^j, c^i) : i, j \geq 0\}, R_2 = \{(a^i b^j, c^j) : i, j \geq 0\},$$

$$R_1, R_2 \text{ are rational relations, } R_1 \cap R_2 = \{(a^k b^k, c^k) : k \geq 0\} \text{ is not.})$$

2.2 FSTs are Good for You

- The math behind FSTs is quite well understood. This makes them a useful formal model.

- They are extremely efficient especially in their subsequential and p -subsequential variants and have been used for a whole bunch of purposes in natural language processing (cf. Mohri (1996) and Mohri (1997)):

- Representation of very large dictionaries
- Finite-state approximations of Syntax (Mohri and Nederhof (2001))
- Speech Processing
- Compilation of Morphological and Phonological Rules (Kaplan and Kay (1994))

One important aspect of transducer is that the amount of time they take to process a string is proportional to the length of the string and not to the size of the transducer. This makes them very efficient.

For a very useful implementation, see Beesley and Karttunen (2003).

2.3 FSTs and Phonology

Phonological derivations can be seen as a relation with the following form:
 $\text{PHON} = \{(\text{UR}, \text{SR}) : \text{UR is a possible underlying representation and SR is a possible surface representation for this UR}\}.$

The substantive claim underlying Johnson (1972), Kaplan and Kay (1994) and Karttunen (1993) is that PHON (as conceived within SPE style phonology) is a rational relation i.e. the UR/SR relationship can be represented by a transducer.

SPE style phonology consisted of a series of ordered rewrite rules. Johnson (1972) showed that each of these rewrite rules could be modelled by a finite state transducer.

We can look at PHON as the composition of a whole bunch of FSTs corresponding to each rewrite rule. Then since FSTs are closed under composition, it follows that PHON can be represented by an FST i.e. PHON is a rational relation.

SPE style phonology originally had rewrite rules that could not be modelled by FSTs. But in terms of the work they did, they could be replaced by rewrite rules that could be modelled by FSTs. It seems that based on Johnson (1972)'s observation, the more permissive rules were abandoned in favor of empirically adequate but more restrictive rewrite rules that could be modelled by FSTs (J. McCarthy p.c.).

3 OT with FSTs

3.1 Frank and Satta (1998)'s Formalization

- (4) An **Optimality System** is a triple $G = (\Sigma, \text{GEN}, C)$,
where Σ is a finite alphabet,

GEN is a function from Σ^* to 2^{Σ^*} ,
(if w is a well-formed UR, then $\text{GEN}(w)$ is the non-empty set of all associated SRs.)

$C = \langle c_1, \dots, c_p \rangle, p \geq 1$, is an ordered sequence of total functions from Σ^* to N

Each function c in C when applied to a candidate yields the 'degree of violation' of the constraint by the candidate.

Applied to a set of candidates S , in general we want the subset of S that violates c to the least degree i.e. the subset which gets the lowest score from c :

$$(5) \quad \operatorname{argmin}_c(S) = \{w : w \in S, c(w) = \min(\{c(w') : w' \in S\})\}$$

Now we can define the function defined by an OT grammar:

$$(6) \quad OT_G^i(w) =$$

a. if $i = 0$:

$$\operatorname{GEN}(w),$$

b. if $i \geq 1$ and $\operatorname{argmin}_{c_i}(OT^{i-1}_G(w)) = OT^{i-1}_G(w)$:

$$OT^{i-1}_G(w)$$

c. if $i \geq 1$ and $\operatorname{argmin}_{c_i}(OT^{i-1}_G(w)) \neq OT^{i-1}_G(w)$:

$$\operatorname{argmin}_{c_i}(OT^{i-1}_G(w))$$

OT_G^p is the optimality function associated with G .

3.2 The Question of Complexity

(7) What is the generative capacity of optimality functions?

(8) Consider the functions f, g, h from Σ^* to $\{0, 1\}$.

a. f maps all strings of the form a^* to 1, everything else to 0.

f is regular.

b. g maps all strings of the form $a^n b^n$ to 1, everything else to 0.

g is context-free, not regular.

c. h maps all strings of the form $a^n b^n c^n$ to 1, everything else to 0.

h is context-sensitive, not context-free, and not regular.

But OT functions apply to strings (URs) and yield sets of optimal strings (SRs). How do we reason about their generative power?

We could talk about how complex $L(G)$ is.

$$(9) \quad L(G) = \bigcup_{w \in \Sigma^*} (OT_G(w))$$

But this characterization is linguistically not very interesting (as it stands) because it does not show the (UR,SR) pairing.

Alternatively, we could see if we can model an OT grammar by an FST. If we can, then we know that the OT grammar can be handled with finite state tools and we also retain the (UR,SR) pairing.

Two important assumptions:

- GEN is a rational relation.
- Each OT constraint is regular.
(in the sense of f in (8). More precisely, we say that an OT constraint is regular if for each $k \in N$, the set $\{w : w \in \Sigma^* \wedge c(w) = k\}$ is regular. It seems that almost all OT constraints are regular. Frank and Satta (1998) consider the case of the ONS(ET) constraint.)

These assumptions are important because if we allow GEN or any OT constraint to be non-rational, then the complexity question becomes less interesting. The complexity question then follows directly from the design of GEN/the relevant OT constraint and not from the architectural properties of an OT grammar.

3.3 OT as a Rational Relation

The main result:

- (10) Optimality Systems can be implemented via FSTs as long each constraint has a finite co-domain i.e. there is an upper bound to the number of *'s that any candidate may receive.
- (11) Restricting Relations to Languages:
 - a. $\text{Lrst}(R, L) = \{(u, v) : (u, v) \in R \wedge u \in L\}$
(the **Left Restriction** of R to L)
 - b. $\text{Rrst}(R, L) = \{(u, v) : (u, v) \in R \wedge v \in L\}$
(the **Right Restriction** of R to L)

- (12) If R is a rational relation and L is a regular language, then $\text{Lrst}(R, L)$ and $\text{Rrst}(R, L)$ are both rational relations.

(The proof follows from the fact that for any regular language L , there is a rational relation $\{(w, w) : w \in L\}$. Composing this relation with R on the left yields $\text{Lrst}(R, L)$ and composing it on the right yields $\text{Rrst}(R, L)$. Since rational relations are closed under composition, it follows that $\text{Lrst}(R, L)$ and $\text{Rrst}(R, L)$ are also both rational relations.)

To simplify the proof, we first restrict ourselves to constraints that have a co-domain of size 2:

- (13) a. For each $c_i \in C$, c_i is a total function from Σ^* to $\{0, 1\}$.
 b. Both $L(c_i) = \{w : c_i(w) = 0\}$ and its complement are regular.

The definition of OT^i can now be simplified as follows:

- (14) $OT^i(w) =$
 a. if $i = 0$:

$$\text{GEN}(w),$$

 b. if $i \geq 1$ and $OT^{i-1}(w) \cap L(c_i) = \phi$:

$$OT^{i-1}(w)$$

 c. if $i \geq 1$ and $OT^{i-1}(w) \cap L(c_i) \neq \phi$:

$$OT^{i-1}(w) \cap L(c_i)$$

OT_G^p is the optimality function associated with G .

- (15) **Lemma:** Let $G = (\Sigma, \text{GEN}, C)$ be an OS such that GEN is a rational relation, each constraint in C is regular and has a co-domain of size two. Then OT_G is a rational relation.

(16) Proof by Induction:

For $i = 0$, $OT^0 = \text{GEN}(w)$ is representable by GEN, which is a rational relation by definition.

Let us assume that for $i - 1$, OT^{i-1} is representable by a rational relation i.e. there is a rational relation R such $OT^{i-1}(w) = [R](w)$.

a. Consider $R_1 = \text{Rrst}(R, L(c_i))$.

Since R is a rational relation and $L(c_i)$ is regular (by definition), R_1 is a rational relation (see 12).

R_1 associates a UR to the set of SRs that are optimal up to constraint c_{i-1} and that also satisfy constraint c_i .

b. Now consider $L_1 = \text{Left}(R_1)$.

This is the set of URs that are assigned an SR by R_1 . Since R_1 is a rational relation, L_1 is a regular language.

c. Since L_1 is regular, \bar{L}_1 is also regular.

Each UR in \bar{L}_1 is such that no associated SR is both optimal upto c_{i-1} and satisfies c_i .

d. Finally consider $R_2 = \text{Lrst}(R, \bar{L}_1)$.

R_2 covers cases that are optimal up to c_{i-1} but which do not satisfy c_i .

The rational relation $R = R_1 \cup R_2$ represents OT^i .

(17) Extending to constraints with finite co-domains:

Basic Idea: any constraint with a finite co-domain can be simulated by a finite number of constraints with a co-domain of size 2.

Let the original constraint c have co-domain $\{1, \dots, k\}$, $k > 1$. Then we introduce the new constraints c_i for $1 \leq i \leq k$ s.t.

- $c_i(w) = 0$, if $c(w) < i$
- $c_i(w) = 1$, if $c(w) \geq i$

(it can be shown that each c_i is regular.)

With these constraints, we can make exactly the distinctions we made with c .

As pointed out by Jason Eisner, the ordering between these constraints is not significant since the following relationship holds:

$$L(c_1) \subset L(c_2) \subset \dots \subset L(c_k)$$

(18) **Theorem:** Let $G = (\Sigma, \text{GEN}, C)$ be an OS such that GEN is a rational relation and each constraint in C is regular and has a finite co-domain. Then OT_G is a rational relation.

3.4 Markus Hiller's Counterexample

(19) $G = (\Sigma, \text{GEN}, \{c\})$

a. $\Sigma = \{a, b\}$

b. $\text{GEN} = \{(a^m b^n, a^m b^n) : m, n \in N\} \cup \{(a^m b^n, b^m a^n) : m, n \in N\}$

c. $c(w) = \#_a(w)$

• GEN is a rational relation.

• c is a regular function:

for any $k \in N$, $\{w : c(w) = k\} = \{a^k b^i : i \geq 0\} \cup \{b^i a^k : i \geq 0\}$, which is regular.

• The relation R corresponding to OT_G is:

$$R = \{(a^m b^n, a^m b^n) : m < n\} \cup$$

$$\{(a^m b^n, b^m a^n) : m > n\} \cup$$

$$\{(a^m b^n, a^m b^n) : m = n\} \cup \{(a^m b^n, b^m a^n) : m = n\}$$

• Next consider the regular language $L = \{a^m b^n : m, n \in N\}$ and the right restriction of R to L :

$$\text{Rrst}(R, L) = \{(a^m b^n, a^m b^n) : m \leq n\}$$

$\text{Left}(\text{Rrst}(R, L)) = \{a^m b^n : m \leq n\}$, not regular.

Hence $\text{Rrst}(R, L)$ can't be a rational relation, and working back R can't be one either.

3.5 A more direct conversion to FSTs: Karttunen (1998)

• Frank and Satta (1998) treat GEN and constraints differently: GEN is a rational relation, while the constraints are regular functions.

• But their formalization allows constraints to 'count', and Karttunen (1998) presents a formalization where the constraints are themselves rational relations and there is no 'counting' happening.

(20) a. NoCoda: an FST that only allows strings without Codas to pass through.

b. HaveOnset: an FST that only allows strings with an Onset before a Nucleus to pass through.

c. Parse: an FST that does not allow through strings with unparsed segments.

Of course, this formalization only makes a distinction between a violating string and a non-violating string. If we want to make further distinctions, we can add additional constraints in the following fashion:

- (21) a. NoCoda₁: an FST that only allows strings with one Coda or less to pass through.
 b. NoCoda₂: an FST that only allows strings with two Codas or less to pass through.
 c. ...
 d. NoCoda_n: an FST that only allows strings with n Codas or less to pass through.

Note that there is no real ‘counting’ happening here: just pattern matching.

The ranking between members of this family of constraints does not have any impact on the output.

- (22) Putting it all together 1: Ordinary Composition:

Given that GEN, c_1, \dots, c_p are all rational relations, it seems reasonable to try to define OT_G as the composition of these relations:

$$OT_G = \text{GEN} \circ c_1 \circ \dots \circ c_p$$

But this definition actually yields an SPE-style system in that it does not permit any constraint violation - even low ranked constraints have to be satisfied.

- (23) Putting it all together 2: Lenient Composition:

We try to satisfy a low ranked constraint but only when satisfying it and all higher ranked constraints doesn’t lead to an empty set. If it does, we only consider the output of the higher ranked constraints.

- a. Priority Union, \cup_P : combines two relations, but if they conflict, priority is given to the first relation.

$$Q \cup_P R = Q \cup Lrst(Left(Q), R)$$

take everything from Q, and those elements from R which do not conflict with elements from Q.

(Rational Relations are closed under Priority Union.)

- b. Lenient Composition, $.O.$:

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

first try to satisfy the new constraint, but if it doesn’t work, let the string through.

(Rational Relations are closed under Lenient Composition.)

$$OT_G = \text{GEN} .O.c_1.O \dots .O.c_p$$

OT_G is a rational relation.

- The difference between Frank and Satta (1998) and Karttunen (1998) is ultimately a matter of presentation.
 - Frank and Satta (1998) start with constraints that can count but unboundedly large numbers, but show that the Rational Relation property only holds with constraints that have finite co-domains.
 - Karttunen (1998) starts with constraints that are already Rational Relations/FSTs i.e. they cannot count unboundedly large numbers. He then shows that such a system always yield an OT_G that is a Rational Relation.

4 Last Words

- Karttunen (1993): Attested phonological processes mediating between UR and SR can be modelled by a finite-state transducer.
- Karttunen's point holds independently of whether the relation between UR and SR is best characterized in terms of a sequence of rewrite rules or an OT optimization.
- Thus if Karttunen's empirical argument is correct and the goal of OT is to characterize the UR/SR mapping, then Frank and Satta (1998) have shown that OT has more formal power than is needed.

Frank and Satta (1998) end with the following suggestion:

Constraints should be limited in the number of distinctions they can make in levels of violation. We suspect that following this regimen will necessitate a shift in the type of optimization which is carried out in OT, from global optimization over arbitrarily large representations to local optimization over structural domains of bounded complexity (where only a bounded number of violations can possibly occur).

References

- Beesley, K. R., and L. Karttunen (2003) *Finite-State Morphology: Xerox Tools And Techniques*, Cambridge University Press, Cambridge, England.
- Frank, R., and G. Satta (1998) “Optimality Theory and the Generative Complexity of Constraint Violability,” *Computational Linguistics* 24:2, 307–315.
- Johnson, C. D. (1972) *Formal Aspects of Phonological Description*, Mouton, The Hague.
- Kaplan, R. M., and M. Kay (1994) “Regular Models of Phonological Rule Systems,” *Computational Linguistics* 20:3, 331–378. Written in 1980.
- Karttunen, L. (1993) “Finite-State Constraints,” in J. Goldsmith, ed., *The Last Phonological Rule*, University of Chicago Press, 173–194.
- Karttunen, L. (1998) “The Proper Treatment of Optimality in Computational Phonology,” in *Proceedings of International Workshop on Finite-State Methods in Natural Language Processing*, Bilkent University, Ankara, Turkey, 1–12.
- Mohri, M. (1996) “On some applications of Finite State Automata Theory to Natural Language Processing,” *Natural Language Engineering* 2:1, 1–20.
- Mohri, M. (1997) “Finite-State Transducers in Language and Speech Processing,” *Computational Linguistics* 23:2, 269–312.
- Mohri, M., and M.-J. Nederhof (2001) “Regular Approximation of Context-Free Grammars through Transformation,” chapter 9 in J.-C. Junqua and G. van Noord, eds., *Robustness in Language and Speech Technology*, Kluwer Academic Publishers, The Netherlands, 153–163.