Proceedings of the 18th International Conference on Computational Linguistics
(COLING 2000), Saarbrücken, Germany, August 2000. [Minor edits for clarity.]

Directional Constraint Evaluation in Optimality Theory*

Jason Eisner
Department of Computer Science / University of Rochester
Rochester, NY 14607-0226 (U.S.A.) / jason@cs.rochester.edu

Weighted finite-state constraints that can *count* unboundedly many violations make Optimality Theory more powerful than finite-state transduction (Frank and Satta, 1998). This result is empirically and computationally awkward. We propose replacing these unbounded constraints, as well as non-finite-state Generalized Alignment constraints, with a new class of finite-state *directional constraints*. We give linguistic applications, results on generative power, and algorithms to compile grammars into transducers.

1 Introduction

Optimality Theory is a grammar framework that directly expresses constraints on phonological forms. Roughly, the grammar prefers forms that violate each constraint as little as possible.

Most constraints used in practice describe disfavored *local* configurations in the phonological form (Eisner, 1997a). It is therefore possible for a given form to offend a single constraint *at several locations* in the form. (For example, a constraint against syllable codas will be offended by every syllable that has a coda.)

When comparing forms, then, how do we aggregate a form's multiple local offenses into an overall **violation level**?

A constraint could answer this question in at least three ways, the third being our proposal:

- **Unbounded evaluation** (Prince and Smolensky, 1993). A form's violation level is given by the number of offenses. Forms with fewer offenses are preferred.
- **Bounded evaluation** (Frank and Satta, 1998; Karttunen, 1998). A form's violation level is $\min(k, \text{number of offenses})$ for some k. This is like unbounded evaluation except that the constraint does not distinguish among forms with $\geq k$ offenses.[1]
- **Directional evaluation.** A form's violation level considers the location of offenses, not their total number. Under **left-to-right** evaluation, the constraint prefers forms whose offenses are as late as possible. To compare two forms, it aligns them (according to their common underlying representation), and scans them in parallel from left to right, until reaching an underlying location where one form has more offenses than the other ("sudden death"). **Right-to-left** evaluation is similar. The number of offenses (and amount of surface material) per underlying location is unbounded.

§2 of this paper gives linguistic and computational motivation for the proposal. §3 formalizes the idea and shows that *composing a transducer with a directional constraint yields a transducer.* Thus directional constraints, like bounded ones, keep OT within the class of regular relations. (But we also show them to be more expressive.)

2 Motivation

2.1 Intuitions

Recall that OT's constraint ranking mechanism is an answer to the question: How can a grammar evaluate a form by aggregating its violations of several constraints? Above we asked the same question at a finer scale: How can a *constraint* evaluate a form by aggregating its offenses at several locations? Figure 1 illustrates that our answer is just constraint ranking redux.

Directional evaluation strictly ranks the importance of the locations within a form, e.g., from left to right. This exemplifies OT's "do only when necessary" strategy: the constraint prefers to *postpone* offenses until they become strictly necessary toward the right of the form, even at the cost of having more of them.

One might think from Figure 1 that each directional constraint could be decomposed into several binary or other bounded constraints, yielding a grammar using only bounded constraints. However, no single such grammar is general enough to handle all inputs: the number of constraints needed for the decomposition corresponds to the length (i.e., the number of locations) of the underlying representation.

* I am grateful to the 3 anonymous referees for feedback.

[1] Note that $k = 1$ gives "binary" constraints that can be described simply as languages. Any k-bounded constraint can easily be simulated by k binary constraints.

	(a)	C_1	NoCoda	(b)	C_1	σ_1	σ_2	σ_3	σ_4	(c)	C_1	σ_4	σ_3	σ_2	σ_1
ban.to.di.bo		*!	*		*!	*					*!				*
ban.ton.di.bo	☞		**			*	*!			☞				*	*
ban.to.dim.bon			***!	☞		*		*	*			*!	*		*
ban.ton.dim.bon			***!*			*	*!	*	*			*!	*	*	*

Figure 1: Directional evaluation as subconstraint ranking. All candidates have 4 syllables; we simplify here by regarding these as the locations. C_1 is some high-ranked constraint that eliminates *ban.to.di.bo*; NoCoda is offended by syllable codas. (a) Traditional unbounded evaluation of NoCoda. (b) Left-to-right evaluation of NoCoda, shown as if it were split into 4 constraints evaluating the syllables separately. (c) Right-to-left.

2.2 Iterative and floating phenomena

The main empirical motivation for directionally evaluated constraints is the existence of "iterative" phenomena such as metrical footing. (Derivational theories described these with procedures that scanned a form from one end to the other and modified it; see (Johnson, 1972).)

For most other phenomena, directional constraints are indistinguishable from traditional unbounded constraints. Usually, the candidates with the fewest offenses are still the ones that survive. (Since their competitors offend at exactly the same locations, and more.) This is precisely because most phonology is local: satisfying a constraint at one location does not usually block satisfying it at another.

Distinguishing cases, like the artificial Fig. 1—where the constraint can only *trade* offenses at one location for offenses at another—arise only under special conditions involving nonlocal phenomena. Just as directional evaluation would predict, such a forced tradeoff is always resolved (to our knowledge) by placing offenses as late, or as early, as higher constraints allow:

- *Prosodic groupings* force each segment or syllable to choose which constituent (if any) to associate with. So-called left-to-right directional syllabification (Mester and Padgett, 1994) will syllabify /CVCCCV/ greedily as CVC.C<u>V</u>.CV rather than CV.C<u>V</u>C.CV, postponing epenthetic material until as late as possible. Similarly, left-to-right binary footing (Hayes, 1995) prefers $(\sigma\sigma)(\sigma\sigma)\sigma$ over $\sigma(\sigma\sigma)(\sigma\sigma)$ or $(\sigma\sigma)\sigma(\sigma\sigma)$, postponing unfooted syllables.
- *Floating lexical material* must surface somewhere in the form. Floating features (e.g., tone) tend to dock at the leftmost or rightmost available site, postponing the appearance of these marked features. Infixed morphemes tend to be infixed as little as possible (McCarthy and Prince, 1995), postponing the appearance of an affix edge or other affix material within the stem.[2]
- *Floating non-lexical material* must also appear somewhere. If a high-ranked constraint, Culminativity, requires that a primary stress mark appear on each word, then a directional constraint against primary stress will not only prevent additional marks but also push the single mark to the first or last available syllable—the traditional "End Rule" (Prince, 1983).
- *Harmony* must decide how far to spread features, and *OCP effects* such as Grassman's Law must decide which copies of a feature to eliminate. Again, directional evaluation seems to capture the facts.

2.3 Why not Generalized Alignment?

In OT, following a remark by Robert Kirchner, it has been traditional to analyze such phenomena using highly non-local **Generalized Alignment** (GA) constraints (McCarthy and Prince, 1993). For example, left-to-right footing is favored by Align-Left$_\sigma$(Foot, Stem), which requires every foot to be left-aligned with a morphological stem. Not only does each misaligned foot offend the constraint, but the seriousness of its offense is given by the

[2] "Available site" and "possible" amount of infixation are defined here by higher-ranked constraints. These might restrict the allowed tone-bearing units and the allowed CV shape after infixation, but do not fully determine where the floating material will surface.

A referee asks why codas do not float (to postpone NoCoda offenses). For the same reason they do not delete: "moving" an underlying coda violates local deletion *and* insertion constraints, so is even worse! Floating tones require special mechanisms for *non-local* faithfulness: Gen or a constraint must ensure that the anchored tone sequence resembles the underlying floating one, which may be represented locationlessly, on an auxiliary input tape (or, if bounded, as an input prefix).

number of syllables by which it is misaligned. These numbers are summed over all offending feet to obtain the violation level. For example, $[\sigma(\sigma\sigma)(\sigma\sigma)\sigma(\sigma\sigma)]_{Stem}$ has 1+3+6=10 violations, and $[\sigma\sigma\sigma\sigma(\sigma\sigma)(\sigma\sigma)]_{Stem}$ is equally bad at 4+6=10 violations. Shifting feet leftward or eliminating them reduces the violation level.

Stemberger (1996) argued that GA constraints were too powerful. Ellison (1995) showed that no single finite-state unbounded constraint could define the same violation levels as a GA constraint. Eisner (1997a) showed more strongly that since GA can be made to center a floating tone on a phrase,[3] no *hierarchy* of finite-state unbounded constraints could even define the same *optimal candidates* as a GA constraint. Thus GA cannot be simulated in Ellison's (1994) finite-state framework (§3.2).

For this reason, as well as the awkwardness and non-locality of evaluating GA offenses, we propose to replace GA with directional constraints. Directional constraints appear to more directly capture the observed phenomena.

We do note that another, trickier possibility is to eliminate GA in favor of ordinary *unbounded* constraints that are indifferent to the location of offenses. Ellison (1994) noted that GA constraints that evaluated the placement of only one element (e.g., primary stress) could be replaced by simpler NoIntervening constraints. Eisner (1997b) gives a GA-free treatment of the metrical stress typology of (Hayes, 1995).

2.4 Generative power

It has recently been proposed that for computational reasons, OT should eliminate not only GA but *all* unbounded constraints (Frank and Satta, 1998; Karttunen, 1998). As with GA, we offer the less extreme approach of replacing them with directional constraints instead.

Recall that a phonological grammar, as usually conceived, is a description of permissible (UR, SR) pairs.[4] It has long been believed that naturally occurring phonological grammars are *regular relations* (Johnson, 1972; Kaplan and Kay, 1994). This means that they can be implemented as **finite-state transducers (FSTs)** that accept exactly the grammatical pairs. FSTs are immensely useful in performing many relevant tasks rapidly: generation (obtaining all possible SRs for a UR), comprehension (conversely), characterizing the set of forms on which two grammars (even one derivational and one OT grammar) would differ, etc. Moreover, FSTs can be applied in parallel to regular sets of forms. For example, one can obtain a weighted set of possible SRs (a phoneme lattice) from a speech recognizer, pass it through the inverted transducer, intersect the resulting weighted set of URs with the lexicon, and then extract the best surviving URs.

Ellison (1994) and Eisner (1997a) frame OT within this tradition, by modeling Gen and the constraints as weighted finite-state machines (see §3.2). But although those papers showed how to generate the set of SRs for a *single* given UR, they did not compile the OT grammar into an FST, or obtain the other benefits thereof.

In fact, Frank and Satta (1998) showed that such compilation is impossible in the general case of unbounded constraints. To see why, consider the grammar Max, Dep, Harmony[height] $\gg$ Ident-IO[height]. This grammar insists on height harmony among surface vowels, but dislikes changes from the UR. The result is the unattested phenomenon of "majority assimilation" (Baković, 1999; Lombardi, 1999): a UR with more high vowels than low will surface with all vowels high, and vice-versa. So OT may *compare unbounded counts* in a way that an FST cannot and phonology does not.

This suggests that OT with unbounded constraints is too powerful. Hence Frank and Satta (1998) and Karttunen (1998) propose using only bounded constraints. They show this reduces OT's power to finite-state transduction.

The worry is that bounded constraints may not be expressive *enough*. A 2-bounded version of NoCoda would not distinguish among the final three forms in Figure 1: it is agnostic when the input forces multiple codas in all candidates.

To be sure, a k-bounded approximation may work well for large k.[5] But its automaton (§3.2) will typically have k times as many states as the unbounded original, since it unrolls loops: the

[3] This is indeed too powerful: centering is unattested.

[4] UR = underlying representation, SR = surface repn.

[5] In particular, in the case of phonological generation, an output of this approximate grammar is guaranteed correct unless it achieves k violations for some k-bounded constraint. One can then raise k, recompile the grammar, and try again. But k may grow quite large for long inputs like phonological phrases.

state must keep track of the offense count. Intersecting many such large constraints can produce very large FSTs—while still failing to capture simple generalizations, e.g., that *all* codas are dispreferred.

In §3, we will show that directional constraints are more powerful than bounded constraints, as they *can* express such generalizations—yet they keep us within the world of regular relations and FSTs.

2.5 Related Work

Walther (1999), working with intersective constraints, defines a similar notion of **Bounded Local Optimization (BLO)**. Trommer (1998; 1999) applies a variant of Walther's idea to OT. The motivation in both cases is linguistic.

We sketch how our idea differs via 3 examples:

UR	uuuuu	uu	uuu	uuuuu
candidate X	vvvbb	vv	vbb	vvvbb
candidate Y	vvbaa	vvvvbaa		vzbaa

Consider $*b$, a left-to-right constraint that is offended by each instance of b. On our proposal, candidate X wins in each column, because Y always offends $*b$ first, at position 3 in the UR.

But under BLO, this offense is not fatal. Y can survive $*b$ by inserting epenthetic material (column 2: Y wins by postponing b *relative to the SR*), or by changing v to z (column 3: Y ties X, since $vv \neq vz$ and BLO merely requires the cheapest choice *given the surface output so far*). In the same way, NoCoda under BLO would trigger many changes unrelated to codas. Our definition avoids these apparent inconveniences.

Walther and Trommer do not consider the expressive power of BLO (cf. §3.3) or whether grammars can be compiled into UR-to-SR FSTs (our main result; see discussion in §3.4).

3 Formal Results

3.1 Definition of OT

An **OT grammar** is a pair $(\mathsf{Gen}, \vec{C})$ where

- the **candidate generator** Gen is a relation that maps each **input** string (in Σ^*) to a nonempty set of candidate **outputs**;
- the **hierarchy** $\vec{C} = (C_1, C_2, \ldots)$ is a finite tuple of **constraint** functions that evaluate outputs.

We write $\vec{C}(\delta)$ for the tuple $(C_1(\delta), C_2(\delta), \ldots)$.

Given a UR, σ, as input, the grammar admits as its SRs all the outputs δ such that $\vec{C}(\delta)$ is lexicographically minimal in $\{\vec{C}(\delta) : \delta \in \mathsf{Gen}(\sigma)\}$.

The values taken by C_i are called its **violation levels**. Conventionally these have been natural numbers, but any ordered set will do.

Our **directional constraints** require the following innovations. Each input σ is a string as usual, but the outputs are not strings. Rather, each candidate $\delta \in \mathsf{Gen}(\sigma)$ is a tuple of $|\sigma| + 1$ strings. We write $\overline{\delta}$ for the concatenation of these strings (the "real" SR). So δ specifies an *alignment* of $\overline{\delta} \in \Delta^*$ with $\sigma \in \Sigma^*$. Directional constraint C_i maps δ to a violation level—itself a length-$(|\sigma|+1)$ tuple of natural numbers that counts the offenses per position in underlying σ. Violation levels are compared lexicographically.

3.2 Finite-state assumptions

We now confine our attention to finite-state OT grammars, following (Ellison, 1994; Tesar, 1995; Eisner, 1997a; Frank and Satta, 1998; Karttunen, 1998). $\mathsf{Gen} \subseteq \Sigma^* \times \Delta^*$ is a regular relation,[6] and may be implemented as an unweighted FST. Each constraint is implemented[7] as a possibly nondeterministic, weighted finite-state automaton (**WFSA**) that accepts Δ^* and whose arcs are weighted with natural numbers.

An FST, T, is a finite-state automaton in which each arc is labeled with a string pair $\alpha:\gamma$. Without loss of generality, we require $|\alpha| \leq 1$. This lets us define an **aligned transduction** that maps strings to tuples: If $\sigma = a_1 \cdots a_n$, we define $T(\sigma)$ as the set of $(n+1)$-tuples $\delta = (\delta_0, \delta_1, \ldots \delta_n)$ such that T has a path transducing $\sigma : \overline{\delta}$ along which $\delta_0 \cdots \delta_{i-1}$ is the complete output before a_i is read from the input.

We now describe how to evaluate $C(\delta)$ where C is a WFSA. Consider the path in C that accepts $\overline{\delta}$.[8] In (un)bounded evaluation, $C(\delta)$ is the total weight of this path. In left-to-right evaluation, $C(\delta)$ is the $n+1$ tuple giving the respective total weights of the subpaths that consume $\delta_0, \ldots \delta_n$. In right-to-left evaluation, $C(\delta)$ is the reverse of the previous tuple.[9]

[6] Ellison required only that $\mathsf{Gen}(\sigma)$ be regular $(\forall\sigma)$.

[7] Space prevents giving the equivalent characterization as a locally weighted language (Walther, 1999).

[8] If there are multiple accepting paths (nondeterminism), take the one that gives the least value of $C(\delta)$.

[9] This is equivalent to $C^R(\delta_n^R, \ldots, \delta_0^R)$ where ^{R} denotes reversal of the automaton or string as appropriate.

3.3 Expressive power

Thanks to Gen, finite-state OT can trivially implement any regular input-output relation with no constraints at all! And §3.4 below shows that whether we allow directional or bounded constraints does not affect this generative power.

But in another sense, directional constraints are strictly more expressive than bounded ones. If Gen is fixed, then any hierarchy of bounded constraints can be simulated by some hierarchy of directional constraints[10]—but *not* vice-versa.

Indeed, we show even more strongly that directional constraints cannot always be simulated even by unbounded constraints.[11] Define $*b$ as in §2.5. This ranks the set $(a|b)^n$ in lexicographic order, so it makes 2^n distinctions. Let Gen be the regular relation

$$(a{:}a|b{:}b)^*(c{:}a(a{:}a|b{:}b)^* \mid c{:}b(a{:}a|b{:}b|a{:}b|b{:}a)^*)$$

We claim that the grammar $(\mathsf{Gen}, *b)$ is not equivalent to $(\mathsf{Gen}, C_1, \ldots, C_s)$ for any bounded or unbounded constraints $C_1, \ldots C_s$. There exists k such that for all $(n, \delta \in \Delta^n)$, each $C_i(\delta) < kn$.[12] So candidates δ of length n have at most $(kn)^s$ different violation profiles $\vec{C}(\delta)$. Choose n such that $2^n > (kn)^s$. Then the set of 2^n strings $(a|b)^n$ must contain two distinct strings, $\delta = x_1 \cdots x_n$ and $\delta' = y_1 \cdots y_n$, with $\vec{C}(\delta) = \vec{C}(\delta')$. Let i be minimal such that $x_i \neq y_i$, and without loss of generality assume $x_i = a, y_i = b$. Put $\sigma = x_1 \cdots x_{i-1} c x_{i+1} \cdots x_n$. Now $\delta, \delta' \in \mathsf{Gen}(\sigma)$ and δ is lexicographically minimum in $\mathsf{Gen}(\sigma)$. So the grammar $(\mathsf{Gen}, *b)$ maps σ to δ only, whereas $(\mathsf{Gen}, \vec{C})$ cannot distinguish between δ and δ', so it maps σ to neither or both.

3.4 Grammar compilation: OT = FST

It is trivial to translate an arbitrary FST grammar into OT: let Gen be the FST, and $\vec{C} = ()$. The rest of this section shows, conversely, how to compile a finite-state OT grammar $(\mathsf{Gen}, \vec{C})$ into an FST, provided that the grammar uses only bounded and/or directional constraints.

[10] How? By using states to count, a bounded constraint's WFSA can be transformed so that all the weight of each path falls on its final arc. This defines the same optimal candidates, even when interpreted directionally.

[11] Nor vice-versa, since only unbounded constraints can implement non-regular relations (§2.4,§3.4).

[12] Eliminate ϵ from the constraints' WDFAs (regard as outputless WFSTs, use §3.4.4) so δ-reading paths have length n. Take k to exceed all arc weights in the result.

3.4.1 The outer loop of compilation

Let $T_0 = \mathsf{Gen}$. For $i > 0$, we will construct an FST T_i that implements the partial grammar $(\mathsf{Gen}, C_1, C_2, \ldots C_i)$. We construct T_i from T_{i-1} and C_i only: $T_i(x)$ contains the forms $y \in T_{i-1}(x)$ for which $C_i(y)$ is minimal.

If C_i is k-bounded, we use the construction of (Frank and Satta, 1998; Karttunen, 1998).

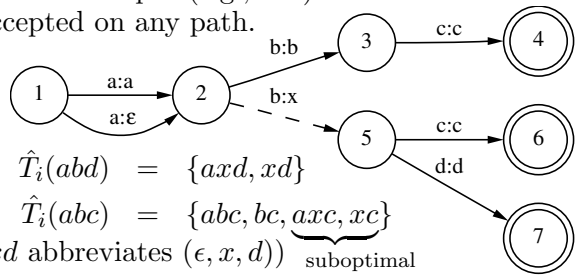
If C_i is a left-to-right constraint, we compose T_{i-1} with the WFSA that represents C_i, obtaining a *weighted* finite-state transducer (**WFST**), $\hat{T}_i$. This transducer may be regarded as assigning a C_i-violation level (an $(|\sigma| + 1)$-tuple) to each $\sigma : \delta$ it accepts. We must now prune away the suboptimal candidates: using the DBP algorithm below, we construct a new unweighted FST T_i that transduces $\sigma : \delta$ iff the weighted $\hat{T}_i$ can transduce $\sigma : \delta$ as cheaply as any $\sigma : \delta'$.

If C_i is right-to-left, we do just the same, except DBP is used to construct T_i^R from $\hat{T}_i^R$.

3.4.2 Directional Best Paths: The idea

All that remains is to give the construction of T_i from $\hat{T}_i$, which we call **Directional Best Paths (DBP)**. Recall standard best-paths or shortest-paths algorithms that pare a WFSA down to its paths of minimum *total* weight (Dijkstra, 1959; Ellison, 1994). Our greedier version does not sum along paths; it always immediately takes the lightest "available" arc. But:

Crucially, available arcs are defined *relative* to the input string, because we must retain one or more optimal output candidates for *each* input. So availability requires "lookahead": we must take a heavier arc ($b : x$ below) just when the rest of the input (e.g., *abd*) cannot otherwise be accepted on any path.



$\hat{T}_i(abd) = \{axd, xd\}$

$\hat{T}_i(abc) = \{abc, bc, \underbrace{axc, xc}_{\text{suboptimal}}\}$

(xd abbreviates (ϵ, x, d))

On this example, DBP would simply make state 6 non-final (disallowing the $b : x$ arc for input *abc* but not *abd*); but often it must add states!

This input sensitivity is what lets us compile a hierarchy of directional constraints, once and for all, into an single FST that can find the optimal output for *any* of the infinitely many possible inputs. We saw in §2.4 why this is so desirable. By

contrast, Ellison's (1994) best-paths construction for unbounded constraints, and previously proposed constructions for directional-style constraints (see §2.5) only find the optimal output for a single input, or at best a finite lexicon.

3.4.3 Dir. Best Paths: A special case

§3.2 restricted our FSTs such that for every arc label $\alpha : \gamma$, $|\alpha| \leq 1$. In this section we construct T_i from $\hat{T}_i$ under the stronger assumption that $|\alpha| = 1$, i.e., $\hat{T}_i$ is ϵ-free on the input side.

If Q is the stateset of $\hat{T}_i$, then let the stateset of T_i be $\{[q; R; S] : R \subseteq S \subseteq Q, q \in S - R\}$. This has size $|Q| \cdot 3^{|Q|-1}$. However, most of these states are typically unreachable from the start state. Lazy "on-the-fly" construction techniques (Mohri, 1997) can be used to avoid allocating states or arcs until they are discovered during exploration from the start state.

For $\sigma \in \Sigma^*, q \in Q$, define $V(\sigma, q)$ as the minimum "cost" (a $|\sigma|$-tuple of arc weights) of any σ-reading path from $\hat{T}_i$'s start state q_0 to q.

The start state of T_i is $[q_0; \emptyset; \{q_0\}]$. The intent is that T_i have a path from its start state to $[q; R; S]$ that alignedly transduces $\sigma : \delta$[13] iff

- $\hat{T}_i$ has a q_0 to q, $\sigma : \delta$ path of cost $V(\sigma, q)$;
- $R = \{q' \in Q : V(\sigma, q') < V(\sigma, q)\}$; and
- $S = \{q' \in Q : V(\sigma, q') \leq V(\sigma, q)\}$.

So as T_i reads σ, it "follows" $\hat{T}_i$'s *cheapest* σ-reading paths to q, while calculating R, to which yet cheaper (but perhaps dead-end) paths exist.

Let $[q; R; S]$ be a final state (in T_i) iff q is final and no $q' \in R$ is final (in $\hat{T}_i$). So an accepting path in $\hat{T}_i$ survives into T_i iff there is no lower-cost accepting path in $\hat{T}_i$ for the same input.

The arcs from $[q; R; S]$ correspond to arcs from q. For each arc from q to q' labeled $a : \gamma$ and with weight W, add an unweighted $a : \gamma$ arc from $[q; R; S]$ to $[q'; R'; S']$, provided that the latter state exists (i.e., unless $q' \in R'$, indicating that there is a cheaper path to q'). Here R' is the set of states that are either reachable from R by a (single) a-reading arc, or reachable from S by an a-reading arc of weight $< W$. S' is the union of R' and all states reachable from S by an a-reading arc of weight W.

3.4.4 Dir. Best Paths: The general case

To apply the above construction, we must first transform $\hat{T}_i$ so it is ϵ-free on the input side. Of course input ϵ's are crucial if Gen is to be allowed to insert unbounded amounts of surface material (to be pruned back by the constraints).[14] To eliminate ϵ's while still allowing unbounded insertion, we are forced to introduce FST arc labels of the form $a : \Gamma$ where Γ is actually a regular set of strings, represented as an FSA or regular expression. Following ϵ-elimination, we can apply the construction of §3.4.3 to get T_i, and finally convert T_i back to a normal transducer by expanding each $a : \Gamma$ into a subgraph.

When we eliminate an arc labeled $\epsilon : \gamma$, we must push γ and the arc's weight back onto a previous non-ϵ arc (but no further; contrast (Mohri, 1997)). The resulting machine will implement the same aligned transduction as $\hat{T}_i$ but more transparently: in the notation of §3.2, the arc reading a_i will transduce it directly to δ_i.[15]

Concretely, suppose $\hat{T}_i$ can get from state q to q'' via a path of *total* weight W that begins with $a : \gamma_1$ on its first arc followed by $\epsilon : \gamma_2$, $\epsilon : \gamma_3$, ... on its remaining arcs. We would like to substitute an arc from q to q'' with label $a : \gamma_1\gamma_2\gamma_3\ldots$ and weight W. But there may be infinitely many such q–q'' paths, of varying weight, so we actually write $a : \Gamma$, where Γ describes just those q–q'' paths with minimum W.

The exact procedure is as follows. Let G be the possibly disconnected subgraph of $\hat{T}_i$ formed by ϵ-reading arcs. Run an all-pairs shortest-paths algorithm[16] on G. This finds, for each state pair (q', q'') connected by an ϵ-reading path, the subgraph $G_{q',q''}$ of G formed by the minimum-weight ϵ-reading paths from q' to q'', as well as the common weight $W_{q',q''}$ of these paths. So for each arc in $\hat{T}_i$ from q to q', with weight W and label $a : \gamma$, we now add an arc to $\hat{T}_i$ from q to q'' with weight $W + W_{q',q''}$ and label $a : \gamma G_{q',q''}(\epsilon)$. ($G(\epsilon)$ denotes the regular language to which G transduces ϵ.) Having done this, we can delete all ϵ-reading arcs.

The modified ϵ-free $\hat{T}_i$ is equivalent to the

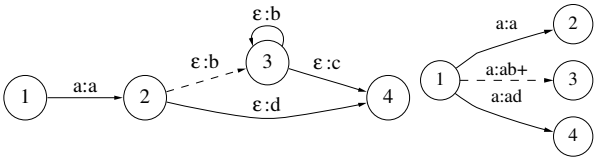
[13] δ is a tuple of $|\sigma|+1$ strings, but $\delta_0 = \epsilon$ by ϵ-freeness.

[14] As is conventional. Besides epenthetic material, Gen often introduces copious prosodic structure.

[15] That arc is labeled $a_i : \Gamma$ where $\delta_i \in \Gamma$. But what is a_0? A special symbol $E \in \Sigma$ that we introduce so that δ_0 can be pushed back onto it: Before ϵ-elimination, we modify $\hat{T}_i$ by giving it a new start state, connected to the old start state with an arc $E : \epsilon$. After ϵ-elimination, we apply DBP and replace E with ϵ in the result T_i.

[16] Cormen *et al.* (1990) cite several, including fast algorithms for when edge weights are small integers.

original except for eliminating some of the suboptimal subpaths. Here is a graph fragment before and after ϵ-elimination:



Note: Right-to-left evaluation applies DBP to $\hat{T}_i^R$, so consistency with our previous definitions means it must push ϵ's forward, not backward.

4 Conclusions

This paper has proposed a new notion in OT: "directional evaluation," where underlying locations are strictly ranked by their importance.

Traditional finite-state OT constraints have enough power to compare arbitrarily high counts; Generalized Alignment is even worse. Directional constraints seem to capture the pros of these constraints: they appropriately militate against every instance of a disfavored configuration in a candidate form, no matter how many, and they naturally capture iterative and edgemost effects. Yet they do not have the excess power: we have shown that an grammar of directional and/or bounded constraints can be compiled into a finite-state transducer. That is both empirically and computationally desirable.

The most obvious future work comes from linguistics. *Can* directional constraints do all the work of unbounded and GA constraints? How do they change the style of analysis? (E.g., directional versions of markedness constraints pin down the locations of marked objects, leaving lower-ranked constraints no say.) Finally, directional constraints can be variously formulated (is *CLUSTER offended at the start, or end, of each cluster? or of its enclosing syllable?). So what conventions or restrictions should apply?

References

Eric Baković. 1999. Assimilation to the unmarked. Ms., Rutgers Optimality Archive ROA-340.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Jason Eisner. 1997a. Efficient generation in primitive Optimality Theory. In *Proc. of the 35th Annual ACL and 8th EACL*, Madrid, July, 313–320.

Jason Eisner. 1997b. FOOTFORM decomposed: Using primitive constraints in OT. In Benjamin Bruening, editor, *Proc. of SCIL VIII*, MIT Working Papers in Linguistics 31, Cambridge, MA.

T. Mark Ellison. 1994. Phonological derivation in optimality theory. In *Proc. of COLING*.

T. Mark Ellison. 1995. OT, finite-state representations and procedurality. In *Proc. of the Conference on Formal Grammar*, Barcelona.

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

Bruce Hayes. 1995. *Metrical Stress Theory: Principles and Case Studies*. U. of Chicago Press.

C. Douglas Johnson. 1972. *Formal Aspects of Phonological Description*. Mouton.

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

Lauri Karttunen. 1998. The proper treatment of optimality in computational phonology. In *Proc. of FSMNLP'98*, 1–12, Bilkent U., Ankara, Turkey.

Linda Lombardi. 1999. Positional faithfulness and voicing assimilation in Optimality Theory. *Natural Language and Linguistic Theory*, 17:267–302.

John McCarthy and Alan Prince. 1993. Generalized alignment. In Geert Booij and Jaap van Marle, editors, *Yearbook of Morphology*, 79–153. Kluwer.

John McCarthy and Alan Prince. 1995. Faithfulness and reduplicative identity. In Jill Beckman *et al.*, editor, *Papers in Optimality Theory*, 259–384. U. of Massachusetts, Amherst: GLSA.

Armin Mester and Jaye Padgett. 1994. Directional syllabification in Generalized Alignment. Phonology at Santa Cruz 3, October.

Mehryar Mohri. 1997. Finite-state transducers in language & speech processing. *Comp. Ling.* 23(2).

Alan Prince and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Ms., Rutgers U. and U. of Colorado.

Alan Prince. 1983. Relating to the grid. *Linguistic Inquiry*, 14:19–100.

J. P. Stemberger. 1996. The scope of the theory: Where does beyond lie? In L. McNair, K. Singer, L. M. Dobrin, and M. M. Aucoin, eds. *Papers from the Parasession on Theory and Data in Linguistics, CLS 23*, 139–164. Chicago Linguistic Society.

Bruce Tesar. 1995. *Computational Optimality Theory*. Ph.D. thesis, U. of Colorado, Boulder.

Jochen Trommer. 1998. Optimal morphology. In T. Mark Ellison, editor, *Proc. of the 4th ACL SIGPHON Workshop*, Quebec, July.

Jochen Trommer. 1999. Mende tone patterns revisited: Tone mapping as local constraint evaluation. In *Linguistics in Potsdam Working Papers*.

Markus Walther. 1999. One-level prosodic morphology. Arbeiten zur Linguistik 1, U. of Marburg.