Appeared in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pp. 56-63, 2002.

Comprehension and Compilation in Optimality Theory*

Jason Eisner
Department of Computer Science
Johns Hopkins University
Baltimore, MD, USA 21218-2691
jason@cs.jhu.edu

Abstract

This paper ties up some loose ends in finite-state Optimality Theory. First, it discusses how to perform comprehension under Optimality Theory grammars consisting of finite-state constraints. Comprehension has not been much studied in OT; we show that unlike production, it does not always yield a regular set, making finite-state methods inapplicable. However, after giving a suitably flexible presentation of OT, we show carefully how to treat comprehension under recent variants of OT in which grammars can be compiled into finite-state transducers. We then unify these variants, showing that compilation is possible if all components of the grammar are regular relations, *including the harmony ordering on scored candidates*. A side benefit of our construction is a far simpler implementation of directional OT (Eisner, 2000).

1 Introduction

To produce language is to convert utterances from their underlying ("deep") form to a surface form. Optimality Theory or OT (Prince and Smolensky, 1993) proposes to describe phonological production as an optimization process. For an underlying x, a speaker purportedly chooses the surface form z so as to maximize the **harmony** of the pair (x, z). Broadly speaking, (x, z) is harmonic if z is "easy" to pronounce and "similar" to x. But the precise harmony measure depends on the language; according to OT, it can be specified by a grammar of ranked desiderata known as constraints.

According to OT, then, production maps each underlying form to its best possible surface pronunciation. It is akin to the function that maps each child x to his or her most flattering outfit z. Different children look best in different clothes, and for an oddly shaped child x, even the best conceivable outfit z may be an awkward compromise between style and fit—that is, between ease of pronunciation and similarity to x.

Language *comprehension* is production in reverse. In OT, it maps each outfit z to the set of children x for whom that outfit is optimal, i.e., is at least as flattering as any other outfit z':

$$\begin{aligned} \textsc{produce}(x) &= \{z : (\nexists z')\ (x, z') > (x, z)\} \\ \textsc{comprehend}(z) &= \{x : z \in \textsc{produce}(x)\} \\ &= \{x : (\nexists z')\ (x, z') > (x, z)\} \end{aligned}$$

In general z and z' may range over infinitely many possible pronunciations. While the formulas above are almost identical, comprehension is in a sense more complex because it varies both the underlying and surface forms. While $\textsc{produce}(x)$ considers all pairs (x, z'), $\textsc{comprehend}(z)$ must *for each* x consider all pairs (x, z'). Of course, this nested definition does not preclude computational shortcuts.

This paper has three modest goals:

1. To show that OT comprehension does in fact present a computational problem that production does not. Even when the OT grammar is required to be finite-state, so that production can be performed with finite-state techniques, comprehension *cannot* in general be performed with finite-state techniques.
2. To consider recent constructions that cut through this problem (Frank and Satta, 1998; Karttunen, 1998; Eisner, 2000; Gerdemann and van Noord, 2000). By *altering* or *approximating* the OT formalism—that is, by hook or by crook—these constructions manage to compile OT grammars into finite-state transducers. Transducers may readily be inverted to do comprehension as easily as production. We carefully lay out how to use them for comprehension in realistic circumstances (in the presence of correspondence theory, lexical constraints, hearer uncertainty, and phonetic postprocessing).
3. To give a unified treatment in the extended finite-state calculus of the constructions referenced above. This clarifies their meaning and makes them easy to implement. For example, we obtain a transparent algebraic version of Eisner's (2000) unbearably technical automaton construction for his proposed formalism of "directional OT."

*Thanks to Kie Zuraw for asking about comprehension; to Ron Kaplan for demanding an algebraic construction before he believed directional OT was finite-state; and to others whose questions convinced me that this paper deserved to be written.

The treatment shows that all the constructions emerge directly from a generalized presentation of OT, in which the crucial fact is that the harmony ordering on scored candidates is a regular relation.

2 Previous Work on Comprehension

Work focusing on OT comprehension—or even mentioning it—has been surprisingly sparse. While the recent constructions mentioned in §1 can easily be applied to the comprehension problem, as we will explain, they were motivated primarily by a desire to pare back OT's generative power to that of previous rewrite-rule formalisms (Johnson, 1972).

Fosler (1996) noted the existence of the OT comprehension task and speculated that it might succumb to heuristic search. Smolensky (1996) proposed to solve it by optimizing the *underlying* form,

$$\text{COMPREHEND}(z) \stackrel{?}{=} \{x : (\nexists x')\ (x', z) > (x, z)\}$$

Hale and Reiss (1998) pointed out in response that any comprehension-by-optimization strategy would have to arrange for multiple optima: after all, phonological comprehension is a one-to-many mapping (since phonological production is many-to-one).[1]

The correctness of Smolensky's proposal (i.e., whether it really computes COMPREHEND) depends on the particular harmony measure. It can be made to work, multiple optima and all, if the harmony measure is constructed with both production and comprehension in mind. Indeed, for any phonology, it is trivial to design a harmony measure that both production and comprehension optimize. (Just define the harmony of (x, z) to be 1 or 0 according to whether the mapping $x \mapsto z$ is in the language!) But we are really only interested in harmony measures that are defined by OT-style grammars (rankings of "simple" constraints). In this case Smolensky's proposal can be unworkable. In particular, §4 will show that a *finite-state* production grammar in classical OT need not be invertible by *any* finite-state comprehension grammar.

[1] Hale & Reiss's criticism may be specific to phonology and syntax. For some phenomena in semantics, pragmatics, and even morphology, Blutner (1999) argues for a *one-to-one* form-meaning mapping in which marked forms express marked meanings. He deliberately uses **bidirectional optimization** to rule out many-to-one cases: roughly speaking, an (x, z) pair is grammatical for him only if z is optimal given x *and* vice-versa.

3 A General Presentation of OT

This section (graphically summarized in Fig. 1) lays out a generalized version of OT's theory of production, introducing some notational and representational conventions that may be useful to others and will be important below. In particular, all objects are represented as strings, or as functions that map strings to strings. This will enable us to use finite-state techniques later.

The **underlying form** x and **surface form** z are represented as strings. We often refer to these strings as **input** and **output**. Following Eisner (1997), each **candidate** (x, z) is also represented as a string y.

The notation (x, z) that we have been using so far for candidates is actually misleading, since in fact the candidates y that are compared encode more than just x and z. They also encode a particular alignment or **correspondence** between x and z. For example, if $x = \underline{\texttt{abdip}}$ and $z = \texttt{a[di][bu]}$, then a typical candidate would be encoded

$$y = \underline{\texttt{a}}\texttt{a}\underline{\texttt{b}}\texttt{0[}\underline{\texttt{d}}\texttt{d}\underline{\texttt{i}}\texttt{i][}\underline{\texttt{p}}\texttt{b}\underline{\texttt{0}}\texttt{u]}$$

which specifies that $\underline{\texttt{a}}$ corresponds to a, $\underline{\texttt{b}}$ was deleted (has no surface correspondent), voiceless $\underline{\texttt{p}}$ surfaces as voiced b, etc. The harmony of y might depend on this alignment as well as on x and z (just as an outfit might fit worse when worn backwards).

Because we are distinguishing underlying and surface material by using disjoint alphabets $\Sigma = \{\underline{\texttt{a}}, \underline{\texttt{b}}, \ldots\}$ and $\Delta = \{\texttt{[}, \texttt{]}, \texttt{a}, \texttt{b}, \ldots\}$,[2] it is easy to extract the underlying and surface forms (x and z) from y.

Although the above example assumes that x and z are simple strings of phonemes and brackets, nothing herein depends on that assumption. Autosegmental representations too can be encoded as strings (Eisner, 1997).

In general, an OT grammar consists of 4 components: a constraint ranking, a harmony ordering, and generating and pronouncing functions. The constraint ranking is the language-specific part of the grammar; the other components are often supposed to be universal across languages.

The **generating function** GEN maps any $x \in \Sigma^*$ to the (nonempty) set of candidates y whose underlying form is x. In other words, GEN just inserts

[2] An alternative would be to distinguish them by odd and even positions in the string.

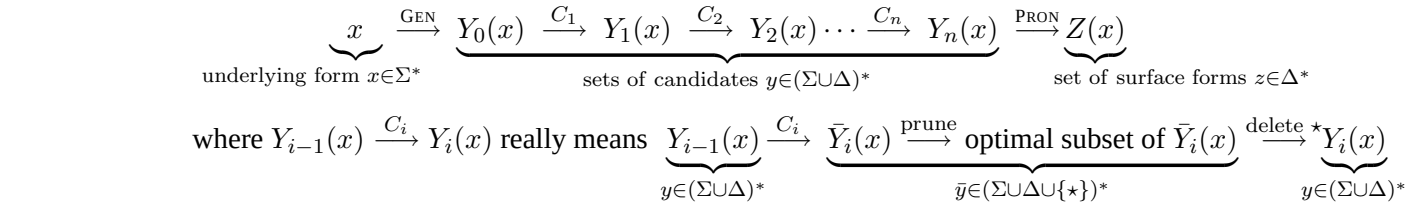

Figure 1: This paper's view of OT production. In the second line, C_i inserts $\star$'s into candidates; then the candidates with suboptimal starrings are pruned away, and finally the $\star$'s are removed from the survivors.

arbitrary substrings from Δ^* amongst the characters of x, subject to any restrictions on what constitutes a legitimate candidate y.[3] (Legitimacy might for instance demand that y's surface material z have matched, non-nested left and right brackets, or even that z be similar to x in terms of edit distance.)

A **constraint ranking** is simply a sequence $C_1, C_2, \ldots C_n$ of **constraints**. Let us take each C_i to be a function that **scores** candidates y by annotating them with violation marks $\star$. For example, a NODELETE constraint would map $y =$ a a b 0 c 0 [d d i i] [p b 0 u] to $\bar{y} =$NODELETE$(y) =$ a a b $\star$ 0 c $\star$ 0 [d d i i] [p b 0 u], inserting a $\star$ after each underlying phoneme that does not correspond to any surface phoneme. This unconventional formulation is needed for new approaches that care about the exact *location* of the $\star$'s. In traditional OT only the *number* of $\star$'s is important, although the locations are sometimes shown for readability.

Finally, OT requires a **harmony ordering** $\succ$ on scored candidates $\bar{y} \in (\Sigma \cup \Delta \cup \{\star\})^*$. In traditional OT, $\bar{y}$ is most harmonic when it contains the fewest $\star$'s. For example, among candidates scored by NODELETE, the most harmonic ones are the ones with the fewest deletions; many candidates may tie for this honor. §6 considers other harmony orderings, a possibility recognized by Prince and Smolensky (1993) ($\succ$ corresponds to their H-EVAL). In general $\succ$ may be a *partial* order: two competing candidates may be equally harmonic or incomparable (in which case both can survive), and candidates with different underlying forms never compete at all.

Production under such a grammar is a matter of **successive filtering** by the constraints $C_1, \ldots C_n$. Given an underlying form x, let

$$Y_0(x) = \text{GEN}(x) \quad (1)$$

$$Y_i(x) = \{y \in Y_{i-1}(x) : (\nexists y' \in Y_{i-1}(x))\ C_i(y') \succ C_i(y)\} \quad (2)$$

The set of optimal candidates is now $Y_n(x)$. Extracting z from each $y \in Y_n(x)$ gives the set $Z(x)$ or PRODUCE(x) of acceptable surface forms:

$$Z(x) = \{\text{PRON}(y) : y \in Y_n(x)\} \subseteq \Delta^* \quad (3)$$

PRON denotes the simple **pronunciation function** that extracts z from y. It is the counterpart to GEN: just as GEN fleshes out $x \in \Sigma^*$ into y by inserting symbols of Δ, PRON slims y down to $z \in \Delta^*$ by removing symbols of Σ.

Notice that $Y_n \subseteq Y_{n-1} \subseteq \ldots \subseteq Y_0$. The only candidates $y \in Y_{i-1}$ that survive filtering by C_i are the ones that C_i considers most harmonic.

The above notation is general enough to handle some of the important variations of OT, such as Paradigm Uniformity and Sympathy Theory. In particular, one can define GEN so that each candidate y encodes not just an alignment between x and z, but an alignment among x, z, and some other strings that are neither underlying nor surface. These other strings may represent the surface forms for other members of the same morphological paradigm, or intermediate throwaway candidates to which z is sympathetic. Production still optimizes y, which means that it simultaneously optimizes z and the other strings.

4 Comprehension in Finite-State OT

This section assumes OT's traditional harmony ordering, in which the candidates that survive filtering by C_i are the ones into which C_i inserts fewest $\star$'s.

Much computational work on OT has been conducted within a finite-state framework (Ellison, 1994), in keeping with a tradition of finite-state phonology (Johnson, 1972; Kaplan and Kay, 1994).[4]

[3] It is never really *necessary* for GEN to enforce such restrictions, since they can equally well be enforced by the top-ranked constraint C_1 (see below).

[4] The tradition already included (inviolable) phonological

Finite-state OT is a restriction of the formalism discussed above. It specifically assumes that GEN, $C_1, \ldots C_n$, and PRON are all regular relations, meaning that they can be described by finite-state transducers. GEN is a nondeterministic transducer that maps each x to multiple candidates y. The other transducers map each y to a single $\bar{y}$ or z.

These finite-state assumptions were proposed (in a different and slightly weaker form) by Ellison (1994). Their empirical adequacy has been defended by Eisner (1997).

In addition to having the right kind of power linguistically, regular relations are closed under various relevant operations and allow (efficient) parallel processing of regular sets of strings. Ellison (1994) exploited such properties to give a production algorithm for finite-state OT. Given x and a finite-state OT grammar, he used finite-state operations to construct the set $Y_n(x)$ of optimal candidates, represented as a finite-state automaton.

Ellison's construction demonstrates that Y_n is always a regular set. Since PRON is regular, it follows that PRODUCE$(x) = Z(x)$ is also a regular set.

We now show that COMPREHEND(z), in constrast, need *not* be a regular set. Let $\Sigma = \{\underline{\mathsf{a}}, \underline{\mathsf{b}}\}$, $\Delta = \{\mathsf{[}, \mathsf{]}, \mathsf{a}, \mathsf{b}, \ldots\}$ and suppose that GEN allows candidates like the ones in §3, in which parts of the string may be bracketed between [and]. The crucial grammar consists of two finite-state constraints. C_2 penalizes $\underline{\mathsf{a}}$'s that fall between brackets (by inserting $\star$ next to each one) and also penalizes $\underline{\mathsf{b}}$'s that fall outside of brackets. It is dominated by C_1, which penalizes brackets that do not fall at either edge of the string. Note that this grammar is completely permissive as to the number and location of surface characters other than brackets.

If x contains more $\underline{\mathsf{a}}$'s than $\underline{\mathsf{b}}$'s, then PRODUCE(x) is the set $\hat{\Delta}^*$ of *all* unbracketed surface forms, where $\hat{\Delta}$ is Δ minus the bracket symbols. If x contains fewer $\underline{\mathsf{a}}$'s than $\underline{\mathsf{b}}$'s, then PRODUCE$(x) = \mathsf{[}\hat{\Delta}^*\mathsf{]}$. And if $\underline{\mathsf{a}}$'s and $\underline{\mathsf{b}}$'s appear equally often in x, then PRODUCE(x) is the union of the two sets.

Thus, while the x-to-z mapping is not a regular relation under this grammar, at least PRODUCE(x) is a regular set for each x—just as finite-state OT guarantees. But for any unbracketed $z \in \hat{\Delta}^*$, such as $z = \mathsf{abc}$, COMPREHEND(z) is *not* regular: it is the set of underlying strings with # of $\underline{\mathsf{a}}$'s $\geq$ # of $\underline{\mathsf{b}}$'s.

This result seems to eliminate any hope of handling OT comprehension in a finite-state framework. It is interesting to note that both OT and current speech recognition systems construct finite-state models of production and define comprehension as the inverse of production. Speech recognizers *do* correctly implement comprehension via finite-state optimization (Pereira and Riley, 1997). But this is impossible in OT because OT has a more complicated production model. (In speech recognizers, the most probable phonetic or phonological surface form is not presumed to have suppressed its competitors.)

One might try to salvage the situation by barring constraints like C_1 or C_2 from the theory as linguistically implausible. Unfortunately this is unlikely to succeed. Primitive OT (Eisner, 1997) already restricts OT to something like a bare minimum of constraints, allowing just two simple constraint families that are widely used by practitioners of OT. Yet even these primitive constraints retain enough power to simulate any finite-state constraint. In any case, C_1 and C_2 themselves are fairly similar to "domain" constraints used to describe tone systems (Cole and Kisseberth, 1994). While C_2 is somewhat odd in that it penalizes two distinct configurations at once, one would obtain the same effect by combining three separately plausible constraints: C_2 requires $\underline{\mathsf{a}}$'s between brackets (i.e., in a tone domain) to receive surface high tones, C_3 requires $\underline{\mathsf{b}}$'s outside brackets to receive surface high tones, and C_4 penalizes all surface high tones.[5]

Another obvious if unsatisfying hack would impose heuristic limits on the length of x, for example by allowing the comprehension system to return the approximation COMPREHEND$(z) \cap \{x : |x| \leq 2 \cdot |z|\}$. This set is finite and hence regular, so per-

constraints, notably Koskenniemi's (1983) two-level model, which like OT used finite-state constraints on candidates y that encoded an alignment between underlying x and surface z.

[5] Since the surface tones indicate the total number of $\underline{\mathsf{a}}$'s and $\underline{\mathsf{b}}$'s in the underlying form, COMPREHEND(z) is actually a finite set in this version, hence regular. But the non-regularity argument does go through if the tonal information in z is not available to the comprehension system (as when reading text without diacritics); we cover this case in §5. (One can assume that some lower-ranked constraints require a special suffix before], so that the bracket information need not be directly available to the comprehension system either.)

haps it can be produced by some finite-state method, although the automaton to describe the set might be large in some cases.

Recent efforts to force OT into a fully finite-state mold are more promising. As we will see, they identify the problem as the harmony ordering $\succ$, rather than the space of constraints or the potential infinitude of the answer set.

5 Regular-Relation Comprehension

Since COMPREHEND(z) need not be a regular set in traditional OT, a corollary is that COMPREHEND and its inverse PRODUCE are not regular relations. That much was previously shown by Markus Hiller and Paul Smolensky (Frank and Satta, 1998), using similar examples.

However, at least *some* OT grammars ought to describe regular relations. It has long been hypothesized that all human phonologies are regular relations, at least if one omits reduplication, and this is necessarily true of phonologies that were successfully described with pre-OT formalisms (Johnson, 1972; Koskenniemi, 1983).

Regular relations are important for us because they are computationally tractable. Any regular relation can be implemented as a finite-state transducer T, which can be inverted and used for comprehension as well as production. PRODUCE$(x) = T(x) = \mathsf{range}(x \circ T)$, and COMPREHEND$(z) = T^{-1}(z) = \mathsf{domain}(T \circ z)$.

We are therefore interested in compiling OT grammars into finite-state transducers—by hook or by crook. §6 discusses how; but first let us see how such compilation is useful in realistic situations.

Any *practical* comprehension strategy must recognize that the hearer does not really perceive the entire surface form. After all, the surface form contains phonetically invisible material (e.g., syllable and foot boundaries) and makes phonetically imperceptible distinctions (e.g., two copies of a tone versus one doubly linked copy). How to comprehend in this case?

The solution is to modify PRON to "go all the way"—to delete not only underlying material but also phonetically invisible material. Indeed, PRON can also be made to perform any purely phonetic processing. Each output z of PRODUCE is now not a phonological surface form but a string of phonemes or spectrogram segments. So long as PRON is a regular relation (perhaps a nondeterministic or probabilistic one that takes phonetic variation into account), we will still be able to construct T and use it for production and comprehension as above.[6]

How about the lexicon? When the phonology can be represented as a transducer, COMPREHEND(z) is a regular set. It contains all inputs x that could have produced output z. In practice, many of these inputs are not in the lexicon, nor are they possible novel words. One should restrict to inputs that appear in the lexicon (also a regular set) by intersecting COMPREHEND(z) with the lexicon. For novel words this intersection will be empty; but one can find the possible underlying forms of the novel word, for learning's sake, by intersecting COMPREHEND(z) with a larger (infinite) regular set representing all forms satisfying the language's lexical constraints.

There is an alternative treatment of the lexicon. GEN can be extended "backwards" to incorporate morphology just as PRON was extended "forwards" to incorporate phonetics. On this view, the input x is a sequence of abstract morphemes, and GEN performs morphological preprocessing to turn x into possible candidates y. GEN looks up each abstract morpheme's phonological string $\in \Sigma^*$ from the lexicon,[7] then combines these phonological strings by concatenation or template merger, then nondeterministically inserts surface material from Δ^*. Such a GEN can plausibly be built up (by composition) as a regular relation from abstract morpheme sequences to phonological candidates. This regularity, as for PRON, is all that is required.

Representing a phonology as a transducer T has additional virtues. T can be applied efficiently to any input string x, whereas Ellison (1994) or Eisner (1997) requires a fresh automaton construction for each x. A nice trick is to build T without

[6] Pereira and Riley (1997) build a speech recognizer by composing a probabilistic finite-state language model, a finite-state pronouncing dictionary, and a probabilistic finite-state acoustic model. These three components correspond precisely to the input to GEN, the traditional OT grammar, and PRON, so we are simply suggesting the same thing in different terminology.

[7] Nondeterministically in the case of phonologically conditioned allomorphs: INDEFINITE APPLE $\mapsto$ {*ən*æ*pl*, *ən*æ*pl*} $\subseteq \Sigma^*$. This yields competing candidates that differ even in their underlying phonological material.

PRON and apply it to all conceivable x's in parallel, yielding the complete set of all optimal candidates $Y_n(\Sigma^*) = \bigcup_{x \in \Sigma^*} Y_n(x)$. If Y and Y' denote the sets of optimal candidates under two grammars, then $(Y \cap \neg Y') \cup (Y' \cap \neg Y)$ yields the candidates that are optimal under only one grammar. Applying GEN^{-1} or PRON to this set finds the regular set of underlying or surface forms that the two grammars would treat differently; one can then look for empirical cases in this set, in order to distinguish between the two grammars.

6 Theorem on Compiling OT

Why are OT phonologies not always regular relations? The trouble is that inputs may be arbitrarily long, and so may accrue arbitrarily large numbers of violations. Traditional OT (§4) is supposed to distinguish all such numbers. Consider syllabification in English, which prefers to syllabify the long input $\underline{\texttt{bi}}\,\underbrace{\underline{\texttt{bambam}}\ldots\underline{\texttt{bam}}}_{k \text{ copies}}$ as `[bi][bam][bam]`$\ldots$`[bam]` (with k codas) rather than `[bib][am][bam]`$\ldots$`[bam]` (with $k+1$ codas). NOCODA must therefore distinguish annotated candidates $\bar{y}$ with k $\star$'s (which are optimal) from those with $k+1$ $\star$'s (which are not). It requires a $(\geq k+2)$-state automaton to make this distinction by looking only at the $\star$'s in $\bar{y}$. And if k can be arbitrarily large, then no *finite*-state automaton will handle all cases.

Thus, constraints like NOCODA do not allow an upper bound on k for all $x \in \Sigma^*$. Of course, the minimal number of violations k of a constraint is fixed *given* the underlying form x, which is useful in production.[8] But comprehension is less fortunate: we *cannot* bound k given only the surface form z. In the grammar of §4, COMPREHEND(`abc`) included underlying forms whose optimal candidates had arbitrarily large numbers of violations k.

Now, in most cases, the *effect* of an OT grammar can be achieved without actually counting anything. (This is to be expected since rewrite-rule grammars were previously written for the same phonologies, and they did not use counting!) This is possible despite the above arguments because for some grammars, the distinction between optimal and suboptimal $\bar{y}$ can be made by looking at the non-$\star$ symbols in $\bar{y}$ rather than trying to count the $\star$'s. In our NOCODA example, a surface substring such as `...ib`$\star$`][a...` might signal that $\bar{y}$ is suboptimal because it contains an "unnecessary" coda. Of course, the validity of this conclusion depends on the grammar and specifically the constraints $C_1, \ldots C_{i-1}$ ranked above NOCODA, since whether that coda is really unnecessary depends on whether $\bar{Y}_{i-1}$ also contains the competing candidate `...i][ba...` with fewer codas.

But as we have seen, some OT grammars do have effects that overstep the finite-state boundary (§4). Recent efforts to treat OT with transducers have therefore tried to remove counting from the formalism. We now unify such efforts by showing that they all modify the harmony ordering $\succ$.

§4 described finite-state OT grammars as ones where GEN, PRON, and the constraints are regular relations. We claim that *if the harmony ordering $\succ$ is also a regular relation on strings of $(\Sigma \cup \Delta \cup \{\star\})^*$, then the entire grammar (*PRODUCE*) is also regular*.

We require harmony orderings to be **compatible** with GEN: an ordering must treat $\bar{y}', \bar{y}$ as incomparable (neither is $\succ$ the other) if they were produced from different underlying forms.[9]

To make the notation readable let us denote the $\succ$ relation by the letter H. Thus, a transducer for H accepts the pair $(\bar{y}', \bar{y})$ if and only if $\bar{y}' \succ \bar{y}$.

The construction is inductive. $Y_0 = \text{GEN}$ is regular by assumption. If Y_{i-1} is regular, then so is Y_i since (as we will show)

$$Y_i = (\bar{Y}_i \circ \neg \mathsf{range}(\bar{Y}_i \circ H)) \circ D \qquad (4)$$

where $\bar{Y}_i \stackrel{\text{def}}{=} Y_{i-1} \circ C_i$ and maps x to the set of starred candidates that C_i will prune; $\neg$ denotes the complement of a regular language; and D is a transducer that removes all $\star$'s. Therefore PRODUCE $= Y_n \circ$ PRON is regular as claimed.

[8] Ellison (1994) was able to construct PRODUCE(x) from x. One can even build a transducer for PRODUCE that is correct on all inputs that can achieve $\leq K$ violations and returns $\emptyset$ on other inputs (signalling that the transducer needs to be recompiled with increased K). Simply use the construction of (Frank and Satta, 1998; Karttunen, 1998), composed with a hard constraint that the answer must have $\leq K$ violations.

[9] For example, the harmony ordering of traditional OT is $\{(\bar{y}', \bar{y}) : \bar{y}'$ has the same underlying form as, but contains fewer $\star$'s than, $\bar{y}\}$. If we were allowed to drop the same-underlying-form condition then the ordering would become regular, and then our claim would falsely imply that all traditional finite-state OT grammars were regular relations.

It remains to derive (4). Equation (2) implies

$$C_i(Y_i(x)) = \{\bar{y} \in \bar{Y}_i(x) : (\nexists \bar{y}' \in \bar{Y}_i(x))\ \bar{y}' \succ \bar{y}\} \quad (5)$$
$$= \bar{Y}_i(x) - \{\bar{y} : (\exists \bar{y}' \in \bar{Y}_i(x))\ \bar{y}' \succ \bar{y}\} \quad (6)$$
$$= \bar{Y}_i(x) - H(\bar{Y}_i(x)) \quad (7)$$

One can read $H(\bar{Y}_i(x))$ as "starred candidates that are worse than other starred candidates," i.e., suboptimal. The set difference (7) leaves only the optimal candidates. We now see

$$(x, \bar{y}) \in Y_i \circ C_i \ \Leftrightarrow\ \bar{y} \in C_i(Y_i(x)) \quad (8)$$
$$\Leftrightarrow\ \bar{y} \in \bar{Y}_i(x), \bar{y} \notin H(\bar{Y}_i(x)) \quad \text{[by (7)]} \quad (9)$$
$$\Leftrightarrow\ \bar{y} \in \bar{Y}_i(x), (\nexists z)\bar{y} \in H(\bar{Y}_i(z)) \ \text{[see below]} \quad (10)$$
$$\Leftrightarrow\ (x, \bar{y}) \in \bar{Y}_i, \bar{y} \notin \mathsf{range}(\bar{Y}_i \circ H) \quad (11)$$
$$\Leftrightarrow\ (x, \bar{y}) \in \bar{Y}_i \circ \neg\mathsf{range}(\bar{Y}_i \circ H) \quad (12)$$
$$\text{therefore } Y_i \circ C_i = \bar{Y}_i \circ \neg\mathsf{range}(\bar{Y}_i \circ H) \quad (13)$$

and composing both sides with D yields (4). To justify (9) $\Leftrightarrow$ (10) we must show when $\bar{y} \in \bar{Y}_i(x)$ that $\bar{y} \in H(\bar{Y}_i(x)) \Leftrightarrow (\exists z)\bar{y} \in H(\bar{Y}_i(z))$. For the $\Rightarrow$ direction, just take $z = x$. For $\Leftarrow$, $\bar{y} \in H(\bar{Y}_i(z))$ means that $(\exists \bar{y}' \in \bar{Y}_i(z))\bar{y}' \succ \bar{y}$; but then $x = z$ (giving $\bar{y} \in H(\bar{Y}_i(x))$), since if not, our compatibility requirement on H would have made $\bar{y}' \in \bar{Y}_i(z)$ incomparable with $\bar{y} \in \bar{Y}_i(x)$.

Extending the pretty notation of (Karttunen, 1998), we may use (4) to define a left-associative **generalized optimality operator** $\mathtt{oo}_H$:

$$Y\ \mathtt{oo}_H\ C \stackrel{\text{def}}{=} (Y \circ C \circ \neg\mathsf{range}(Y \circ C \circ H)) \circ D \quad (14)$$

Then for any regular OT grammar, PRODUCE =

$$\text{GEN}\ \mathtt{oo}_H\ C_1\ \mathtt{oo}_H\ C_2 \cdots\ \mathtt{oo}_H\ C_n \circ \text{PRON}$$

and can be inverted to get COMPREHEND. More generally, different constraints can usefully be applied with different H's (Eisner, 2000).

The algebraic construction above is inspired by a version that Gerdemann and van Noord (2000) give for a particular variant of OT. Their regular expressions can be used to implement it, simply replacing their `add_violation` by our H.

Typically, H ignores surface characters when comparing starred candidates. So H can be written as $\mathtt{elim}(\Delta) \circ G \circ \mathtt{elim}(\Delta)^{-1}$ where $\mathtt{elim}(\Delta)$ is a transducer that removes all characters of Δ. To satisfy the compatibility requirement on H, G should be a subset of the relation $(\Sigma|\star|(\epsilon:\star)|(\star:\epsilon))^*$.[10]

[10] This transducer regexp says to map any symbol in $\Sigma \cup \{\star\}$ to itself, or insert or delete $\star$—and then repeat.

We now summarize the main proposals from the literature (see §1), propose operator names, and cast them in the general framework.

• Y `o` C: Inviolable constraint (Koskenniemi, 1983; Bird, 1995), implemented by composition.

• Y `o+` C: Counting constraint (Prince and Smolensky, 1993): more violations is more disharmonic. No finite-state implementation possible.

• Y `oo` C: Binary approximation (Karttunen, 1998; Frank and Satta, 1998). All candidates with any violations are equally disharmonic. Implemented by $G = (\Sigma^*(\epsilon:\star)\Sigma^*)^+$, which relates underlying forms without violations to the same forms with violations.

• $Y\ \mathtt{oo}_3\ C$: 3-bounded approximation (Karttunen, 1998; Frank and Satta, 1998). Like `o+` , but all candidates with ≥ 3 violations are equally disharmonic. G is most easily described with a transducer that keeps count of the input and output $\star$'s so far, on a scale of 0, 1, 2, ≥ 3. Final states are those whose output count exceeds their input count on this scale.

• Y `o⊂` C: Matching or subset approximation (Gerdemann and van Noord, 2000). A candidate is more disharmonic than another if it has stars in all the same locations and some more besides.[11] Here $G = ((\Sigma|\star)^*(\epsilon:\star)(\Sigma|\star)^*)^+$.

• Y `o>` C: Left-to-right directional evaluation (Eisner, 2000). A candidate is more disharmonic than another if in the leftmost position where they differ (ignoring surface characters), it has a $\star$. This revises OT's "do only when necessary" mantra to "do only when necessary and then as late as possible" (even if delaying $\star$'s means suffering *more* of them later). Here $G = (\Sigma|\star)^*(\epsilon:\star)^+(\Sigma(\star^* \times \star^*))^*$. Unlike the other proposals, here two forms can both be optimal only if they have exactly the same pattern of violations with respect to their underlying material.

• Y `<o` C: Right-to-left directional evaluation. "Do only when necessary and then as *early* as possible." Here G is the reverse of the G used in `o>` .

The novelty of the matching and directional proposals is their attention to *where* the violations fall. Eisner's directional proposal (`o>`, `<o`) is the only

[11] Many candidates are incomparable under this ordering, so Gerdemann and van Noord also showed how to weaken the notation of "same location" in order to approximate `o+` better.

(a) $x =$ bantodibo

[ban][to][di][bo]
[ban][ton][di][bo]
[ban][to][dim][bon]
[ban][ton][dim][bon]

(b)

NoCoda
ban⋆todibo
ban⋆to⋆dibo
ban⋆todi⋆bo⋆
ban⋆to⋆di⋆bo⋆

(c)

	C_1	NoCoda
	*!	*
☞		**
		***!
		***!*

(d)

	C_1	σ_1	σ_2	σ_3	σ_4
	*!	*			
		*	*!		
☞		*		*	*
		*	*!	*	*

Figure 2: Counting vs. directionality. [Adapted from (Eisner, 2000).] C_1 is some high-ranked constraint that kills the most faithful candidate; NoCoda dislikes syllable codas. (a) Surface material of the candidates. (b) Scored candidates for G to compare. Surface characters but not $\star$'s have been removed by `elim`(Δ). (c) In traditional evaluation `o+` , G counts the $\star$'s. (d) Directional evaluation `o>` gets a different result, as if NoCoda were split into 4 constraints evaluating the syllables separately. More accurately, it is as if NoCoda were split into one constraint per underlying letter, counting the number of $\star$'s right after that letter.

one defended on linguistic as well as computational grounds. He argues that violation counting (`o+`) is a bug in OT rather than a feature worth approximating, since it predicts unattested phenomena such as "majority assimilation" (Baković, 1999; Lombardi, 1999). Conversely, he argues that comparing violations directionally is not a hack but a desirable feature, since it naturally predicts "iterative phenomena" whose description in traditional OT (via Generalized Alignment) is awkward from both a linguistic and a computational point of view. Fig. 2 contrasts the traditional and directional harmony orderings.

Eisner (2000) also proved that `o>` was a regular operator for directional H, by making use of a rather different insight, but that machine-level construction was highly technical. The new algebraic construction is simple and can be implemented with a few regular expressions, as for any other H.

7 Conclusion

See the itemized points in §1 for a detailed summary. In general, this paper has laid out a clear, general framework for finite-state OT systems, and used it to obtain positive and negative results about the understudied problem of comprehension. Perhaps these results will have some bearing on the development of realistic learning algorithms.

The paper has also established sufficient conditions for a finite-state OT grammar to compile into a finite-state transducer. It should be easy to imagine new variants of OT that meet these conditions.

References

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

Steven Bird. 1995. *Computational Phonology: A Constraint-Based Approach*. Cambridge.

Reinhard Blutner. 1999. Some aspects of optimality in natural language interpretation. In *Papers on Optimality Theoretic Semantics*. Utrecht.

J. Cole and C. Kisseberth. 1994. An optimal domains theory of harmony. *Studies in the Linguistic Sciences*, 24(2).

Jason Eisner. 1997. Efficient generation in primitive Optimality Theory. In *Proc. of ACL/EACL*.

Jason Eisner. 2000. Directional constraint evaluation in Optimality Theory. In *Proc. of COLING*.

T. Mark Ellison. 1994. Phonological derivation in Optimality Theory. In *Proc. of COLING*

J. Eric Fosler. 1996. On reversing the generation process in Optimality Theory. *Proc. of ACL Student Session*.

R. Frank and G. Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307–315.

D. Gerdemann and G. van Noord. 2000. Approximation and exactness in finite-state Optimality Theory. In *Proc. of ACL SIGPHON Workshop*.

Mark Hale and Charles Reiss. 1998. Formal and empirical arguments concerning phonological acquisition. *Linguistic Inquiry*, 29:656–683.

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

R. Kaplan and M. Kay. 1994. Regular models of phonological rule systems. *Comp. Ling.*, 20(3).

L. Karttunen. 1998. The proper treatment of optimality in computational phonology. In *Proc. of FSMNLP*.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model for word-form recognition and production. Publication 11, Dept. of General Linguistics, University of Helsinki.

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

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*. MIT Press.

A. Prince and P. Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Ms., Rutgers and U. of Colorado (Boulder).

Paul Smolensky. 1996. On the comprehension/production dilemma in child language. *Linguistic Inquiry*, 27:720–731.