Two proofs that classic Optimality Theory is expressively weaker than ordered rewrite rules

Brian Buccola
Evaluation 2
Submitted: December 15, 2012

Abstract

This paper provides the first mathematical demonstration that classic optimality–theoretic grammars, consisting of just two levels of representation (underlying and surface) and of just two types of constraints (markedness and basic faithfulness), are formally expressively weaker than ordered rewrite rules. Specifically, it is shown that there are sets of patterns which can be expressed by ordered rewrite rules but which in a two–level OT grammar require at least one faithfulness constraint that assigns a violation mark to certain pairs of input–output segments in correspondence, but only when they occur in certain contexts. Such a faithfulness constraint, however, which references segments other than the correspondents under consideration, is beyond the limits of what is standardly and intuitively considered classic OT. In addition, at least one such set of patterns (Canadian raising) is attested in natural language, thus making classic OT, as defined, and any more restrictive variant thereof, empirically inadequately expressive. The paper ends with a discussion of how classic OT might be extended slightly so that it becomes empirically adequately expressive while still being provably weaker than ordered rewrite rules.

1 Introduction

There are currently two major frameworks of generative phonology: (i) *rule–based serialism*, e.g., as laid out in *The Sound Patterns of English* (SPE; Chomsky and Halle, 1968), in which an input (underlying form) is mapped serially to intermediate outputs and finally to a terminal output (surface form) via ordered, context–sensitive rewrite rules; and (ii) *constraint–based parallelism*, e.g., Optimality Theory (OT; Prince and Smolensky, 2004), in which inputs are mapped directly to those outputs that are optimal with respect to some set of ranked (totally ordered) constraints, with no intermediate mappings (i.e., everything happens in parallel).[1]

This paper aims to answer the following two questions: (i) Are there input–output patterns that can be expressed in one framework but not in the other, and (ii) if so, are those patterns attested in natural language? The first is a formal question, the second empirical. Both questions, however, should be of interest to all phonologists. If it turns out that the rule–based serialism of SPE and the constraint–based parallelism of OT are expressively equivalent, then empirical coverage offers no reason to favor one framework over the other.

[1] In this paper we use the terms *input* and *output* as an informal convenience to refer to the mathematically functional nature of sets of ordered rules and sets of ranked constraints.

Alternatively, if one is expressively more powerful than the other, then the one with better empirical coverage is to be favored.[2]

There is a widespread intuition among phonologists that there is a version of OT which we might call *classic* (McCarthy, 2007), or traditional (Heinz, 2011b), or basic, and that this version of OT is inadequately expressive. Intuitively, classic OT consists of just two levels of representation (input/underlying and output/surface) and just two types of constraints. First, there are markedness constraints, which penalize output candidates containing certain sequences of segments, without reference to the input (i.e., the input is inconsequential to the assignment of violation marks). Second, there are faithfulness constraints, which penalize certain pairs of single input–output segments in correspondence, without reference to any other segments in the input or output (i.e., segments other than the two correspondents under consideration are inconsequential to the assignment of violation marks).[3]

The widespread claim is that there are certain *opaque* patterns which can be expressed by ordered rules but which are not expressible by such classic OT grammars. The patterns are called opaque because they are the result of phonological generalizations that are in some sense obscured on the surface and are only revealed by looking at intermediate levels of representation. Rewrite can express (in fact, they predict) opaque patterns because they can be ordered so that a later rule rewrites a phonological form in a way that obscures the application (or lack of application) of an earlier rule, and the opacity only becomes apparent once one considers the intermediate output of the earlier rule.

The intuition regarding classic OT is that (most types of) opaque patterns are inherently problematic: OT is surface–oriented, with only two levels of representation; thus, an appeal to intermediate forms is impossible. Moreover, authors like McCarthy (2007) and Baković (2007) have offered convincing informal demonstrations of OT's inability to handle a range of cases of opacity. Consequently, critics of OT have argued that OT is inherently misguided and that rule–based theories are to be preferred (Idsardi, 2000), while proponents of OT have moved to enrich classic OT either with some aspect of serialism, e.g., candidate chains (McCarthy, 2007), or with more complex constraint types, e.g., comparative markedness constraints that reference the input (McCarthy, 2002), or both.

However, to my knowledge, no attempt has yet been made to formally prove the claim that classic OT is expressively inadequate. More precisely, as Heinz (2011b) puts it, the question is still unanswered whether all patterns that can be expressed with ordered rewrite rules (the so–called *regular relations*) can

[2] As Heinz (2011a) puts it, "Which theory is unnecessarily more powerful than the other, and which theory is inadequately expressive? The most restrictive theory which is minimally adequately expressive is tacitly assumed to be the most desirable."

[3] This notion of classic OT therefore presupposes a correspondence relation between the segments of an input form and those of an output form, such that each input segment corresponds to at most one output segment, and vice versa. In the case of deletion and epenthesis, we could say either that a segment has no correspondent, or that its correspondent is the empty string/phonologically null segment.

also be expressed with an OT grammar consisting of "traditional" markedness and faithfulness constraints.

This paper sets out to formally answer that question. First, we define in precise, mathematical terms what is meant by a (classic) OT grammar. In particular, we specify what the allowable constraints are, taking care to stick as closely and inclusively as possible to phonologists' intuitions. To that end, we allow markedness constraints that penalize arbitrary sequences of segments, as well as faithfulness constraints that penalize arbitrary pairs of single input–output segments; and we disallow faithfulness constraints that reference segments other than the two correspondents under consideration. Since this version of classic OT includes many arbitrary, ad–hoc–looking constraints, including even antifaithfulness constraints, it can therefore be thought of as properly including most phonologists' intuitive notions of classic OT, and then some. Second, we demonstrate mathematically that there are sets of patterns, including a set of attested patterns (Canadian raising), which are expressible by ordered rewrite rules but which are not expressible by any such classic OT grammar, under any ranking of any set of constraints. Since this is a negative proof, and since classic OT is defined so inclusively, the proof applies as well to any more restrictive variant of classic OT, such as those that exclude antifaithfulness.

We start in section 2 by reviewing the basics of SPE–style rule ordering and four types of opacity. Then in section 3 we look at an example of counterbleeding on environment opacity in Polish. We observe that the patterns can easily be captured with ordered rules but that in OT we seem to need a faithfulness constraint with the following meaning: assign one violation mark for each occurrence of a certain pair of input–output segments in correspondence, *provided that* certain conditions regarding some *other* segment(s) in the input/output obtain. That is, the OT analysis seems intuitively to require a provisional sort of faithfulness constraint, i.e., an intuitively non–classic faithfulness constraint that references other segments, but we will not yet have proved it. This informal demonstration corresponds closely with phonologists' intuitions (alluded to above) that classic OT cannot handle this type of opacity without some enrichment of the OT architecture.

In section 4 we formalize the notion of an OT constraint as a function from input–output pairs to the natural numbers, represented by a weighted finite–state transducer (FST). We formally define a markedness constraint as any constraint that can be represented by an input–independent FST (either single–state or, as is generally the case, multistate), and we define a classic, non–provisional faithfulness constraint very broadly as any constraint that can be represented by an FST that is both input–dependent and single–state. A provisional faithfulness constraint, by contrast, is defined as a constraint that can only be represented by an FST that is both input–dependent and multistate. Informally, then, a provisional faithfulness constraint ends up resembling a hybrid faithfulness–markedness constraint, hence the intuition that such a constraint is definitively non–classic.

In sections 5 and 7 come the main results of the paper. We demonstrate mathematically that there is a set of patterns, based on the Polish data from

section 3, which can be described by rules ordered in a counterbleeding on environment relationship but which in OT requires at least one constraint that can only be represented by an FST that is both multistate and input–dependent. We then demonstrate that there is another set of patterns, based on data from Isthmus Nahuat (section 6), which can be described by rules ordered in a counterfeeding on environment relationship but which, again, in OT requires at least one constraint that can only be represented by an FST that is both multistate and input–dependent.

In section 8 we generalize the proof technique and sketch out how it can be successfully applied to several other cases of environment opacity, including Canadian raising. Importantly, the Canadian raising patterns to which the proof is applicable are fully attested. Thus, we establish that there are attested phonological patterns that are expressible by ordered rules but not by any classic OT grammar. We also see that the proof cannot be applied to two cases of focus opacity. We conjecture that focus opacity can in general be handled in OT with just markedness and non–provisional (albeit sometimes ad hoc) faithfulness constraints. Environment opacity, by contrast, seems in general to be provably beyond the expressivity of any version of classic OT that excludes provisional faithfulness.

Section 9 concludes with a review and a discussion of whether our definiton of classic OT can be slightly extended to include faithfulness constraints corresponding to input–dependent FSTs with two states or fewer, and to exclude those corresponding to input–dependent FSTs with more than two states. Doing so would capture all the examples of environment opacity discussed in this paper, thereby making OT expressively adequate, at least with respect to attested cases of opacity. What's more, we demonstrate that even this extended version of OT is expressively weaker, in a formal sense, than ordered rules. In other words, this slightly extended version of classic OT is both expressively adequate and more restrictive than ordered rules, thus making it a preferable framework.

2 Preliminaries

In this section we briefly review the basics of SPE–style rule ordering, the Kiparskian definition of opacity, and the intuitive notion of classic OT. No examples are given in this section because the remainder of the paper explores each type of opacity in detail.

In a rule–based theory of phonology like SPE (Chomsky and Halle, 1968), inputs are mapped to outputs via ordered, context–sensitive rewrite rules of the form

(1) $\alpha \rightarrow \beta \ / \ \gamma\_\delta$

read as, "α becomes (is rewritten as) β whenever α occurs immediately after γ and immediately before δ", where $\alpha, \beta, \gamma, \delta$ are either single symbols (phonological segments) or, as we shall assume, classes of symbols (feature bundles).

α is referred to as the *focus* of the rule, while γ and δ make up the *environment*. We call the focus and environment together the *input description*. Under a single such rule, then, an input $/\gamma\alpha\delta/$ is mapped to the output $[\gamma\beta\delta]$.[4]

Since rules simply map strings to strings, i.e., rules manipulate strings under certain context–sensitive conditions, it's possible for rules to *interact* in the following ways (Baković, 2011).

Definition 1. A rule $\mathcal{P}_1$ *feeds* a rule $\mathcal{P}_2$ if $\mathcal{P}_1$ creates additional inputs to $\mathcal{P}_2$. A rule $\mathcal{P}_1$ *bleeds* a rule $\mathcal{P}_2$ if $\mathcal{P}_1$ removes potential inputs to $\mathcal{P}_2$.

In addition, feeding and bleeding can both be further subcategorized (adapted from McCarthy, 1999).

Definition 2. $\mathcal{P}_1$ *feeds on* $\mathcal{P}_2$*'s focus* if $\mathcal{P}_1$ feeds $\mathcal{P}_2$ in such a way that the additional input to $\mathcal{P}_2$ created by $\mathcal{P}_1$ is $\mathcal{P}_2$'s focus. $\mathcal{P}_1$ *feeds on* $\mathcal{P}_2$*'s environment* if $\mathcal{P}_1$ feeds $\mathcal{P}_2$ in such a way that the additional input to $\mathcal{P}_2$ created by $\mathcal{P}_1$ is (part of) $\mathcal{P}_2$'s environment.

Definition 3. $\mathcal{P}_1$ *bleeds on* $\mathcal{P}_2$*'s focus* if $\mathcal{P}_1$ bleeds $\mathcal{P}_2$ in such a way that the potential input to $\mathcal{P}_2$ removed by $\mathcal{P}_1$ is $\mathcal{P}_2$'s focus. $\mathcal{P}_1$ *bleeds on* $\mathcal{P}_2$*'s environment* if $\mathcal{P}_1$ bleeds $\mathcal{P}_2$ in such a way that the potential input to $\mathcal{P}_2$ removed by $\mathcal{P}_1$ is (part of) $\mathcal{P}_2$'s environment.

By convention, when we say that $\mathcal{P}_1$ feeds/bleeds $\mathcal{P}_2$, we imply that $\mathcal{P}_1$ is ordered *before* $\mathcal{P}_2$, even though strictly speaking, feeding/bleeding, as defined, is independent of rule ordering. When a feeding/bleeding rule is ordered *after* the rule it feeds/bleeds, we use the following terms.

Definition 4. $\mathcal{P}_1$ *counterfeeds* $\mathcal{P}_2$ if $\mathcal{P}_1$ both feeds and is ordered *after* $\mathcal{P}_2$. $\mathcal{P}_1$ *counterbleeds* $\mathcal{P}_2$ if $\mathcal{P}_1$ both bleeds and is orderd *after* $\mathcal{P}_2$.

Kiparsky (1971, 1973) was the first to identify the phenomenon of *phonological opacity*, defining it as follows, where "process" can be construed simply as a rewrite rule.

Definition 5. A process $\mathcal{P}$ of the form $A \rightarrow B / \ C\_D$ is *opaque* to the extent that there are surface representations of the form:

a. A in the environment C_D; or

b. B derived by $\mathcal{P}$ in environments other than C_D.

The idea is that opaque phonological generalizations are (a) generalizations that appear not to hold true of a surface form (also known as *not surface true*), or (b) generalizations that are true of a surface form, but their application is hidden (so to speak) below the surface (*not surface apparent*).

[4]Symbols between forward slashes are inputs (underlying forms), and those between square brackets are terminal outputs (surface forms). We'll write non–terminal, i.e., intermediate, outputs with no brackets at all. These conventions also apply to OT inputs and output candidates.

These two types of opacity are typically associated with counterfeeding and counterbleeding rule ordering, respectively (Kager, 1999; McCarthy, 2007; Baković, 2011). In counterfeeding, a later rule creates additional inputs to an earlier rule, such that the earlier rule seems not to have applied to the surface form, even though it matches the rule's input description. In counterbleeding, a later rule removes part of the input description of an earlier rule, such that the earlier rule seems to have applied without satisfying its input description.

Thus, opacity is something that ordered rules express very well (and in fact predict).

For classic OT, however, opacity has been deeply problematic and has therefore driven many phonologists either to seek alternative frameworks or to enrich OT's basic architecture. Surprisingly, however, at least to my knowledge, no one has yet formally demonstrated that there are opaque patterns which classic OT, suitably and precisely defined, fails to express. In the next section, we explore a case of counterbleeding on environment opacity from Polish that will give insight into why phonologists have the strong intuition that classic OT grammars are inadequate, and in the sections thereafter we demonstrate mathematically that their intuition is in fact true.

First, however, we reiterate briefly from section 1 what is intuitively meant by a classic OT grammar. A classic OT grammar consists of just two levels of representation (input and output) and just two types of constraints: markedness, which penalize sequences of segments in the output, with no reference to the input, and faithfulness, which penalize pairs of single input–output segments in correspondence, with no reference to any other segments in the input or output.

3 Counterbleeding on environment opacity in Polish

Polish has the following three phonological patterns (Bethin, 1978; Kenstowicz and Kisseberth, 1979; Baković, 2011).[5]

(2)	**Polish data**	Glosses:
a.	/sol/ surfaces as [sul].	"rubble"
b.	/gruz/ surfaces as [grus].	"salt"
c.	/ʒwob/ surfaces as [ʒwup].	"crib"

[5]Some phonologists, e.g., Sanders (2003), have contended that these patterns are not productive in Polish and therefore do not constitute an actual case of opacity; that is, either the opaque generalizations are not existent/active on Polish because the patterns are simply lexicalized, or there is no evidence (e.g., from alternations) that the underlying forms given are in fact correct. Despite these worries, I have chosen Polish for expository reasons because the patterns are simple to understand. More importantly, however, even if the Polish patterns are unattested, their use in the proof in section 5 is still justified in the sense that the formal result—that there are patterns (attested or otherwise) which are expressible by ordered rules but not by classic OT—still holds. In addition, in section 8 it will be shown that the proof in section 5 can be applied to a wide range of cases of environment opacity, including Canadian Raising, which I believe is much more accepted as an actual case of opacity.

All three patterns can be captured by a rule $\mathcal{R}$ of raising ordered before a rule $\mathcal{D}$ of final–obstruent devoicing.[6]

(3) **Polish rules**

a. $\mathcal{R}$: [−low, +back] → [+high] / _[+voice, −nasal]

b. $\mathcal{D}$: [−sonorant] → [−voice] / _#

$\mathcal{R}$ maps /sol/ to sul, which $\mathcal{D}$ vacuously maps to [sul].[7] $\mathcal{R}$ vacuously maps /gruz/ to grus, which $\mathcal{D}$ maps to [grus]. Finally, $\mathcal{R}$ maps /zwob/ to zwub, which $\mathcal{D}$ maps to [zwup].

The two rules are in a counterbleeding relationship. Specifically, $\mathcal{D}$ counterbleeds on $\mathcal{R}$'s environment: $\mathcal{D}$ "removes" $\mathcal{R}$'s environment by changing it from voiced to voiceless. For example, the voiced segment /b/ in /ʒwob/, which matches $\mathcal{R}$'s environment and thus licenses the raising of /o/ to [u], is mapped by $\mathcal{D}$ to the voiceless segment [p], which does not match $\mathcal{R}$'s environment.

Another way to think about it is that if $\mathcal{D}$ were ordered *before* $\mathcal{R}$, then $\mathcal{D}$ *would* bleed on $\mathcal{R}$'s environment: $\mathcal{D}$ would map /ʒwob/ to ʒwop, which $\mathcal{R}$ would vacuously map to [ʒwop].

The rule $\mathcal{R}$ of raising in the pattern /ʒwob/ → [ʒwup] is considered opaque in the sense that it seems to have applied, yet its input description—specifically, there being a voiced obstruent environment—does not hold true of the terminal output. Only by considering the intermediate output ʒwub do we uncover the application of $\mathcal{R}$, with its input description matched.

Let's now try to capture these patterns in OT, using only classic markedness and faithfulness constraints, as defined informally above. For raising, we could posit a markedness constraint that penalizes outputs containing a mid back vowel followed immediately by a non–nasal, voiced consonant, which is ranked above a faithfulness constraint that penalizes mapping /o/ to [u]. For devoicing, we could posit a markedness constraint that penalizes word–final voiced obstruents, which is ranked above a faithfulness constraint that penalizes mapping a voiced segment to a voiceless one.

(4) **Polish constraints**

a. Raising
*[−low, −high, +back][+voice, −nasal] > IDENT–IO(high)

b. Devoicing
*[−sonorant, +voice]# > IDENT–IO(voice)

For readability, let's relabel the markedness constraint for raising as $^*\text{oC}_{\text{vce}}$, and let's relabel the markedness constraint for devoicing as $^*\text{C}_{\text{vce}}\#$.

Note that we don't know how either constraint in (4a) is ranked with respect to either constraint in (4b). However, to pursue an analysis we must choose

[6] The symbol # denotes a word boundary. Thus, rule $\mathcal{D}$ applies only to those obstruents occurring immediately before a word boundary, i.e., occurring word–finally.

[7] For our purposes, a *vacuous* rule application is defined broadly as a rule application that does not result in any change between the input and output.

some total ranking. One possibility is the following. (We'll see shortly that in fact no ranking of these four constraints that respects (4) can work.)

(5) **Polish hypothetical constraint ranking**
*oC$_{vce}$ > Ident–IO(high) > *C$_{vce}$# > Ident–IO(voice)

Below are two tableaux that verify that these four constraints and this ranking correctly make [sul] a more optimal output of /sol/ than [sol] and make [grus] a more optimal output of /gruz/ than [gruz].

(6) **OT analysis of** /sol/ → [sul]

/sol/	*oC$_{vce}$	Id(high)	*C$_{vce}$#	Id(vce)
a. sol	1			
→ b. sul$_{@}$		1		

(7) **OT analysis of** /gruz/ → [grus]

/gruz/	*oC$_{vce}$	Id(high)	*C$_{vce}$#	Id(vce)
a. gruz			1	
→ b. grus$_{@}$				1

Before we continue, there are two unconventional properties of these tableaux that require explanation. First, violations are given as natural numbers, rather than as a row of *'s, and cells with 0 violations are left blank. We do this to highlight the mathematically functional nature of a constraint. (It maps an input–output pair to a non–negative number of violations, which we discuss in detail in section 4.) Second, there is exactly one output candidate in each tableau with a subscript @ that denotes the actual, attested output for the given input. This annotation is an informal convenience that lets us easily determine whether a ranked constraint set picks out the right candidate or not.[8]

We now turn to the pattern /ʒwob/ → [ʒwup]. Below is a tableau consisting of our four constraints, ranked according to our hypothesis, and the four obvious output candidates that are in competition.

(8) **Attempted OT analysis of** /ʒwob/ → [ʒwup]

/ʒwob/	*oC$_{vce}$	Id(high)	*C$_{vce}$#	Id(vce)
a. ʒwob	1		1	
→ b. ʒwop				1
c. ʒwub		1	1	
d. ʒwup$_{@}$		1		1

Unfortunately, this set of constraints and this ranking favor the unattested output [ʒwop] over the attested output, [ʒwup]. Moreover, the violations in-

[8]Normally, when a ranked constraint set picks out an unattested form, that form is annotated with a skull, while the attested form is annotated with a sad face; and when a ranked constraint set picks out the attested form, there are no annotations. I prefer to always have an annotation for the attested form, rather than have to sometimes rely on the absence of one.

curred by [ʒwup] are a proper *super*set of those incurred by [ʒwop]; that is, [ʒwup] is *harmonically bounded* by [ʒwop]. Thus, no reranking of these constraints can possibly make the attested form [ʒwup] a more optimal output of /ʒwob/ than the unattested form [ʒwop]. We must add new constraints (and possibly discard some of four we posited) to make [ʒwup] optimal.

Intuitively, no markedness constraint seems to be of help. Presumably, outputs containing [p] word–finally or even the sequence [op] word–finally are unmarked in Polish, and the two rewrite rules would likewise predict such input–output pairs as, say, /ʒwop/ → [ʒwop].

Moreover, no classic faithfulness constraint jumps out as being helpful. The only differences between the input /ʒwob/ and the output candidates [ʒwup] and [ʒwop] are that /b/ is devoiced to [p] in both and that /o/ is raised to [u] in one. But both of these faithfulness violations are already captured by IDENT–IO(voice) and IDENT–IO(high), respectively.

We seem, rather, to require a faithfulness constraint we might call ANTIIDENT-IO(O) with the following definition: "Assign one violation mark for each occurrence of an output [o] that stands in correspondence with an input /o/, provided that the output [o] is followed immediately by a [p] (or any voiceless obstruent) that is in correspondence with an input /b/ (its voiced counterpart)."[9] For now, let's informally classify this constraint as a *provisional* faithfulness constraint: it's a faithfulness constraint on a pair of input–output segments $i \rightarrow o$ that applies only under conditions that reference input and output segments different from i and o.[10] Intuitively, this constraint is not part of the basic architecture of classic OT.

McCarthy (2007, pp. 25–26) discusses a very similar sort of constraint in an analysis of counter*feeding* on environment opacity in Bedouin Arabic.[11] He writes

> Constraints like this are necessarily embedded in a faithfulness theory that makes unattested and implausible typological predictions.

[9] ANTIIDENT–IO(O) is similar to the *local conjunction* of ANTIIDENT–IO(high) and IDENT–IO(voice), which we might write as [ANTIIDENT–IO(high) & IDENT–IO(voice)]$_\delta$ (Smolensky, 1993; Kager, 1999). Crucially, the domain δ of this local conjunction spans two different input segments and two different output segments, and it assigns a violation whenever both ANTIIDENT–IO(high) and IDENT–IO(voice) are violated within δ. ANTIIDENT–IO(O) is also similar to the *comparative markedness constraint* $_{\text{N}}{}^{*}$op, which assigns a violation for each "new" occurrence of [op] (McCarthy, 2002). Whichever way one writes this constraint, the point is that it's intuitively *not* part of classic, or basic OT, and yet the analysis seems to require it.

[10] One might actually prefer to classify this particular constraint as a provisional *anti*–faithfulness constraint since the pair of input–output segments that it targets consists of the same two segments, i.e., /o/→[o].

[11] The constraint he discusses has the following definition: "Assign a violation mark for every instance of a surface high vowel that stands in correspondence with an underlying low vowel, provided that this surface high vowel is followed in the next syllable by a vowel that has no underlying correspondent." We return to this constraint in section 8.

I assume, by the way, that by "constraints like this" McCarthy is referring to constraints like ANTIIDENT–IO(O) that have some provisional component in their meaning, i.e., faithfulness constraints that apply only in certain contexts.

and later characterizes them as an "undesirable enrichment of faithfulness theory".

Ideally, then, we would like to find an OT analysis of all the patterns in Polish using only markedness and non–provisional faithfulness constraints. Intuitively, however, we seem not to be able to. In the sections that follow, we shall mathematically demonstrate that this is so, and we shall prove a similar result for counterfeeding on environment opacity based on data from Isthmus Nahuat.[12]

First, however, we must formalize the notion of an OT constraint, which will then allow us to define formally the class of provisional faithfulness constraints.

4 Formalizing provisional faithfulness constraints

An OT constraint can be thought of simply as a function c from a set of input–output pairs to the set $\mathbb{N}$ of natural numbers (i.e., the non–negative integers). For example, suppose that we have a set I of inputs (underlying forms) and a set O of all possible output candidates for those inputs.[13] Suppose also that we have a function $U \subseteq I \times O$ of input–output pairs describing the unique attested output of each input. Then an OT constraint c is simply a function from $I \times O$ to $\mathbb{N}$ which assigns to each input–output pair $\langle i, o \rangle \in I \times O$ some non–negative number of violations. Ultimately, we want to find a ranking $>$ for a set C of constraints such that, for each $i \in I$, C and $>$ pick out the unique $o \in O$ such that $\langle i, o \rangle \in U$.

If, as formal objects, constraints are functions that map input–output pairs to natural numbers, then we need a way to write down, or represent, those functions. For example, we want to be able to say that the constraint IDENT–IO(voice) is a function c such that for each input i and output candidate o

$$c(\langle i, o \rangle) = \begin{cases} 0 & \text{if } \dots \\ 1 & \text{if } \dots \\ 2 & \text{if } \dots \\ \dots & \dots \end{cases}$$

Following Riggle (2004), among others, we shall represent an OT constraint as a weighted finite–state transducer (FST), which we'll call a finite–state OT constraint.[14] An FST is essentially an abstract machine that "reads" or

[12] Alternatives to classic OT, such as local conjunction and comparative markedness mentioned in footnote 9, as well as sympathy theory (McCarthy, 1999), output–output correspondence theory (Benua, 1997), stratal OT, etc., are all enrichments of OT that have been proposed, or at least employed, to handle opacity, among other phenomena. This paper, then, fills in a sort of lacuna in the literature—proving rigorously that *some* such enrichment is necessary.

[13] More precisely, let O be the generalized union of all the output candidate sets that GEN generates for each input $i \in I$.

[14] Note that this is not the only way to formally represent an OT constraint, but as we shall see, it provides a way to precisely and succinctly characterize the class of constraints we have been calling "provisional".

"processes" pairs of input–output strings, one input–output symbol pair at a time, transitioning from state to state and assigning 0 or 1 violations to each pair of input–output symbols. As we'll see shortly, each FST determines a unique function from the set of input–output pairs to $\mathbb{N}$. Thus, we can represent a constraint like IDENT–IO(voice) by any FST that corresponds to the function that does what we intuitively think IDENT–IO(voice) should do. Formally, we have the following definition (adapted from Riggle, 2004).

Definition 6. A *finite–state OT constraint* is a 5–tuple $\langle Q, \Sigma, \delta, q_0, F\rangle$, where:

1. Q is a nonempty set of states;
2. Σ is a set of symbols, e.g., IPA symbols representing phonological segments;
3. δ is a transition function from $Q \times \Sigma \times \Sigma$ to $\{0, 1\} \times Q$;
4. $q_0 \in Q$ is the unique start state;
5. $F \subseteq Q$ is the nonempty set of final states.

The transition function δ determines, for a given state, input symbol, and output symbol, which state to transition to and how many violations (0 or 1) to assign. That is, δ takes in a triple $\langle q_j, i, o\rangle$ consisting of some initial state $q_j \in Q$, an input symbol $i \in \Sigma$, and an output symbol $o \in \Sigma$ and returns a pair $\langle n, q_k\rangle$ consisting of some number of violations $n \in \{0, 1\}$ and some terminal state $q_k \in Q$. For readability, instead of writing a member of δ as $\langle\langle q_j, i, o\rangle, \langle n, q_k\rangle\rangle$, we will drop the inner angle brackets and just write $\langle q_j, i, o, n, q_k\rangle$.

Consider, for example, a language with the symbol set $\Sigma = \{\text{o}, \text{u}, \text{b}, \text{p}\}$, where each symbol corresponds to its obvious IPA counterpart. Then the constraint IDENT–IO(voice), which penalizes mapping /p/ to [b] or mapping /b/ to [p], and which penalizes no other mapping, can be represented by the FST **id(vce)** in (9), with the transition function given in (10). As a convention, we'll use SMALL CAPS to refer to OT faithfulness constraints and regular lowercase to refer to OT markedness constraints; **boldface** to refer to the FSTs that represent OT constraints; and regular lowercase with subscripted numbers to label the states of FSTs. Thus, IDENT–IO(voice) is an OT constraint; **id(vce)** is a finite–state OT constraint, namely an FST that represents IDENT–IO(voice); and id(vce)_0 is a state in **id(vce)**.

(9) **id(vce)** = $\langle\{\text{id(vce)}_0\}, \{\text{o, u, p, b}\}, \delta, \text{id(vce)}_0, \{\text{id(vce)}_0\}\rangle$

(10)
$$\begin{aligned}\delta = \{\ & \langle \text{id(vce)}_0, \text{p}, \text{p}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{o}, \text{p}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{p}, \text{b}, 1, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{o}, \text{b}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{p}, \text{o}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{o}, \text{o}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{p}, \text{u}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{o}, \text{u}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{b}, \text{p}, 1, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{u}, \text{p}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{b}, \text{b}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{u}, \text{b}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{b}, \text{o}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{u}, \text{o}, \text{o}, \text{id(vce)}_0\rangle,\\ & \langle \text{id(vce)}_0, \text{b}, \text{u}, \text{o}, \text{id(vce)}_0\rangle, && \langle \text{id(vce)}_0, \text{u}, \text{u}, \text{o}, \text{id(vce)}_0\rangle \quad \}\end{aligned}$$

id(vce) has just one state, id(vce)_0, so each transition in δ starts and ends with id(vce)_0, with only the input, output, and violation differing among transitions. Moreover, we see that there are

$$|\{\text{id(vce)}_0\}| \cdot |\Sigma| \cdot |\Sigma| = 1 \cdot 4 \cdot 4 = 16$$

possible state–input–output triples, each of which gets mapped by δ to a violation–state pair. Thus, δ has 16 members. Of these, the only two transitions that assign a violation of 1 are, as expected, those containing the input–output pair p→b or b→p.[15]

FSTs are conventionally visualized as in Figure 1. States are represented as circles; the start state has an arrow and the word "start" to its left; final states are double circles; and the transition function is represented as arcs (directed arrows) from state to state, with each arc labeled with an input, an output, and a violation number.

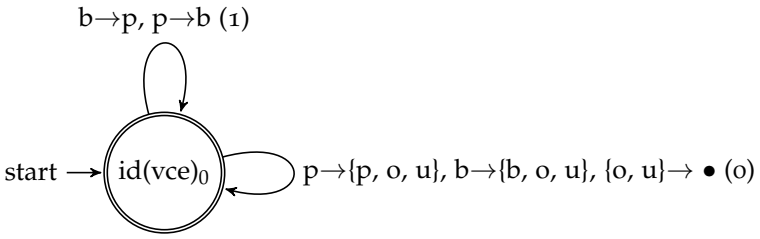

Figure 1: A visualization of **id(vce)**.

Note that the picture has been condensed in several ways:

1. The 16 arcs have been collapsed into just 2 arcs, one for those transitions of 0 violations and one for those transitions of 1 violation;
2. "p→b (1)" means that p is the input symbol, b the output symbol, and 1 the violation for that arc;
3. "b→p, p→b (1)" is shorthand for the two arcs labeled "b→p (1)" and "p→b (1)";
4. "p→{p, o, u} (0)" is shorthand for "p→p, p→o, p→u (0)"; and
5. "{o, u}→ • (0)" is shorthand for "o→{o, u, b, p}, u→{o, u, b, p} (0)". (• is therefore a wildcard metavariable that ranges over all of Σ; it will come in handy when we discuss markedness constraints.)

Suppose, for example, that we have the input string /po/ and the output string [bo]. Then, starting in state id(vce)_0, the machine processes the input–output pair p→b, assigns 1 violation, and loops back to state id(vce)_0; it then

[15] If one considers mapping, say, /p/ to [u] or /u/ to [p] to be a violation of IDENT–IO(voice), then the transition function can be modified accordingly: simply assign the pairs p→u and u→p a violation of 1. This issue does not concern us here, however.

processes the input–output pair o→o, assigns 0 violations, and loops back to state id(vce)$_0$. Since there are no more input–output pairs to process, and since the machine is in a final state, the process ends.[16] The total number of violations assigned is 1.

Consider now the input /bob/ and the output [pop]. Starting in state id(vce)$_0$, the machine processes the input–output pair b→p, assigns 1 violation, and loops back to id(vce)$_0$; it then processes the pair o→o, assigns 0 violations, and loops back to id(vce)$_0$; and finally, it processes the pair b→p, assigns 1 violation, and loops back to id(vce)$_0$. The process ends with the machine in a final state, and the total number of violations assigned is 2.

Now, technically, **id(vce)** is not a function that maps the input–output pair ⟨/po/, [bo]⟩ to 1 and the pair ⟨/bob/, [pop]⟩ to 2. However, with some reflection it should be obvious that **id(vce)** uniquely *determines* a function that does map ⟨/po/, [bo]⟩ to 1 and ⟨/bob/, [pop]⟩ to 2, among infinitely many other mappings, and this function is precisely the one we have in mind for IDENT–IO(voice).

id(vce) represents what we have informally been calling a classic input–output faithfulness constraint. Note that **id(vce)** has just one state and assigns violations based on pairs of single input–output segments, without reference to other parts of the input–output strings. Note also that a constraint like ANTIIDENT–IO(voice) can likewise be represented by a single–state FST, namely the one that only assigns a violation of 1 to the pairs p→p and b→b, and 0 to every other pair. In fact, within this formalism we can even devise an FST that assigns a violation of 1 to any arbitrary pair(s) input–output symbols, e.g., b→u, and 0 to the rest.

Next we turn to markedness constraints, which penalize sequences of segments in the output, regardless of the input. Consider a markedness constraint like

$$^*[-\text{low}, -\text{high}, +\text{back}][+\text{voice}, -\text{nasal}]$$

which penalizes any output containing a mid back vowel followed immediately by a voiced obstruent. Assuming again a symbol set $\Sigma = \{o, u, p, b\}$, this constraint can be relabeled as *ob, since [o] is the only mid back vowel, and [b] is the only voiced obstruent. Thus, *ob penalizes any output containing the string ob, regardless of the input. *ob can be represented by the FST ***ob** in (11) with the transition function in (12).

(11) $^*\mathbf{ob} = \langle\{^*\text{ob}_0, {}^*\text{ob}_1\}, \{o, u, p, b\}, \delta, {}^*\text{ob}_0, \{^*\text{ob}_0, {}^*\text{ob}_1\}\rangle$

[16] We may also say in this case that the input–output pair /po/ → [bo] is *accepted* by **id(vce)**. In a case where, after processing an input–output pair, a machine stops in a non–final state, we say that that pair is not accepted by the FST. However, in this paper we only consider FSTs whose states are all final states; thus, the notion of acceptance can be ignored here.

(12)
$$\begin{aligned}\delta = \{\quad & \langle {}^*\mathrm{ob}_0, \mathrm{p}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{p}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{p}, \mathrm{b}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{p}, \mathrm{b}, 1, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{p}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{p}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{p}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{p}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{b}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{b}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{b}, \mathrm{b}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{b}, \mathrm{b}, 1, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{b}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{b}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{b}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{b}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{o}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{o}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{o}, \mathrm{b}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{o}, \mathrm{b}, 1, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{o}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{o}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{o}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{o}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{u}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{u}, \mathrm{p}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{u}, \mathrm{b}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{u}, \mathrm{b}, 1, {}^*\mathrm{ob}_0\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{u}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{u}, \mathrm{o}, \mathrm{o}, {}^*\mathrm{ob}_1\rangle,\\ & \langle {}^*\mathrm{ob}_0, \mathrm{u}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle, && \langle {}^*\mathrm{ob}_1, \mathrm{u}, \mathrm{u}, \mathrm{o}, {}^*\mathrm{ob}_0\rangle \quad \}\end{aligned}$$

${}^*\mathbf{ob}$ is an FST with two states, ${}^*\mathrm{ob}_0$ and ${}^*\mathrm{ob}_1$. Thus, there are

$$|\{{}^*\mathrm{ob}_0, {}^*\mathrm{ob}_1\}| \cdot |\Sigma| \cdot |\Sigma| = 2 \cdot 4 \cdot 4 = 32$$

possible state–input–output triples, each of which gets mapped by δ to a violation–state pair. Thus, δ has 32 members. The only transitions that assign a violation of 1 are those that are both *from* state ${}^*\mathrm{ob}_1$ *and* with an input–output pair whose output symbol is b, and whose input symbol may be anything.

The visual representation of ${}^*\mathbf{ob}$ in Figure 2 provides an intuitive way to see why ${}^*\mathbf{ob}$ is constructed as it is. First, the only way to transition from ${}^*\mathrm{ob}_0$ to ${}^*\mathrm{ob}_1$ is to process an output o, for any input. Second, while in state ${}^*\mathrm{ob}_1$, processing further outputs o, for any input, leaves the machine in state ${}^*\mathrm{ob}_1$, and processing any non–o output, i.e., p, u, or b, for any input, takes the machine from ${}^*\mathrm{ob}_1$ back to ${}^*\mathrm{ob}_0$. Thus, we can think of state ${}^*\mathrm{ob}_1$ as representing having just processed at least one output o, for any input, and we can think of state ${}^*\mathrm{ob}_0$ as representing having just processed some non–o output, for any input.

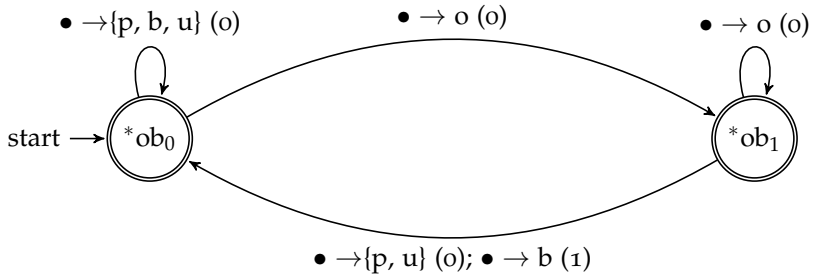

Figure 2: A visualization of ${}^*\mathbf{ob}$.

Next, as noted, the only way to incur a violation is to transition from $^*ob_1$ to $^*ob_0$ by processing an output b, for any input. We can think of this transition as representing having just processed an output b, for any input, after having just processed at least one output o, for any input. In other words, it represents having just processed an output ob, for any two corresponding input symbols. Thus, each occurrence of the substring ob in an output string will incur a violation, regardless of the input string, and that is exactly what the constraint *ob does.

In general, any markedness constraint that penalizes an output containing a sequence of more than one symbol will be represented by a multistate FST. For example, for a constraint like *abc, we want to penalize output strings like abc, but not zbc, azc, or abz. The FST must keep track of each output symbol by transitioning to a new state when necessary. In fact, even a markedness constraint like NoCoda requires two states. This constraint can be thought of as *C. (note the dot), where C is any consonant and . denotes a syllable boundary. Then, by adding the symbol . to the symbol set, we can devise an FST that assigns a violation to any output containing the sequence C. ; and since that sequence has length two, the FST will require two states, just like *ob. Only trivial markedness constraints like *F, for some feature (or set of features) F, or *a, for some single segment a, can be represented by single–state FSTs. The vast majority require at least two states.

Moreover, markedness constraints are represented by FSTs that are input–independent, in the sense that violations are assigned based solely on the output, not on the input. This is why in the visualization of ***ob** every arrow has the wildcard • to its left.

With these two generalizations in mind—that markedness constraints are represented by FSTs that are input–independent and either single– or (more generally) multistate—we can define a markedness constraint more precisely as follows.

Definition 7. A *markedness constraint* is any constraint that can be represented by a finite–state constraint $\langle Q, \Sigma, \delta, q_0, F\rangle$ such that, if $\langle q_j, \alpha, \beta, 1, q_k\rangle \in \delta$, then for any $\gamma \in \Sigma$ such that $\langle q_j, \gamma, \beta, n, q_k\rangle \in \delta$, it follows that $n = 1$. We call the FST representing a markedness constraint an *input–independent finite–state constraint*, or more simply a *finite–state markedness constraint*.

Faithfulness constraints, by contrast, are clearly input–dependent. Moreover, standard constraints like Ident–IO(voice) and Ident–IO(high), as well as non–standard constraints like AntiIdent–IO(voice) and AntiIdent–IO(high), are all clearly representable by single–state FSTs: they assign violations to pairs of single input–output symbols, with no need to transition to a second state.

However, the intuitively non–classic, provisional faithfulness constraint discussed above, AntiIdent–IO(O), cannot be represented by a single–state FST. It requires two states. Let's see why.

Consider again a language with the symbol set $\Sigma = \{o, u, p, b\}$, and recall the constraint AntiIdent–IO(O) we discussed for Polish with the following meaning: "Assign one violation mark to each occurrence of an output [o] that

stands in correspondence with an input /o/, provided that the output [o] is followed immediately by a [p] that is in correspondence with an input /b/." This constraint can be represented by the FST **id(o)** in (13) with the transition function in (14).

(13) **id(o)** = $\langle \{\text{id(o)}_0, \text{id(o)}_1\}, \{\text{o, u, p, b}\}, \delta, \text{id(o)}_0, \{\text{id(o)}_0, \text{id(o)}_1\}\rangle$

(14)
$$\begin{aligned}\delta = \{\quad & \langle \text{id(o)}_0, \text{p, p, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{p, p, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{p, b, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{p, b, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{p, o, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{p, o, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{p, u, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{p, u, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{b, p, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{b, p, 1, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{b, b, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{b, b, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{b, o, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{b, o, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{b, u, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{b, u, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{o, p, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{o, p, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{o, b, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{o, b, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{o, o, o, id(o)}_1\rangle, && \langle \text{id(o)}_1, \text{o, o, o, id(o)}_1\rangle,\\ & \langle \text{id(o)}_0, \text{o, u, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{o, u, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{u, p, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{u, p, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{u, b, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{u, b, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{u, o, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{u, o, o, id(o)}_0\rangle,\\ & \langle \text{id(o)}_0, \text{u, u, o, id(o)}_0\rangle, && \langle \text{id(o)}_1, \text{u, u, o, id(o)}_0\rangle \quad \}\end{aligned}$$

id(o) is an FST with two states, id(o)_0 and id(o)_1. Thus, there are

$$|\{\text{id(o)}_0, \text{id(o)}_1\}| \cdot |\Sigma| \cdot |\Sigma| = 2 \cdot 4 \cdot 4 = 32$$

possible state–input–output triples, each of which gets mapped by δ to a violation–state pair. Thus, as in the case of the finite–state markedness constraint ***ob** above, **id(o)** has 32 members. The only transition that assigns a violation of 1 is the one *from* state id(o)_1 *and* with an input–output pair whose input symbol is b and whose output symbol is p.

The visual representation in Figure 3 provides an intuitive way to see why **id(o)** is constructed as it is. First, the only way to transition from id(o)_0 to id(o)_1 is to process the input–output pair o→o. Second, while in state id(o)_1, processing further input–output pairs o→o leaves the machine in state id(o)_1, whereas processing any other input–output pair takes the machine from id(o)_1 back to id(o)_0. Thus, we can think of state id(o)_1 as representing having just processed at least one input–output pair o→o, and we can think of state id(o)_0 as representing having just processed some input–output pair other than o→o.

The only way to incur a violation is to transition from id(o)_1 to id(o)_0 by processing the input–output pair b→p. We can think of this transition as representing having just processed the input–output pair b→p, after having just processed the input–output pair o→o. In other words, it represents having just processed the input–output pair ob→op (o→o followed by b→p). Thus, any input string containing the substring ob, paired with any output string containg the substring op, such that the input symbol o corresponds to the

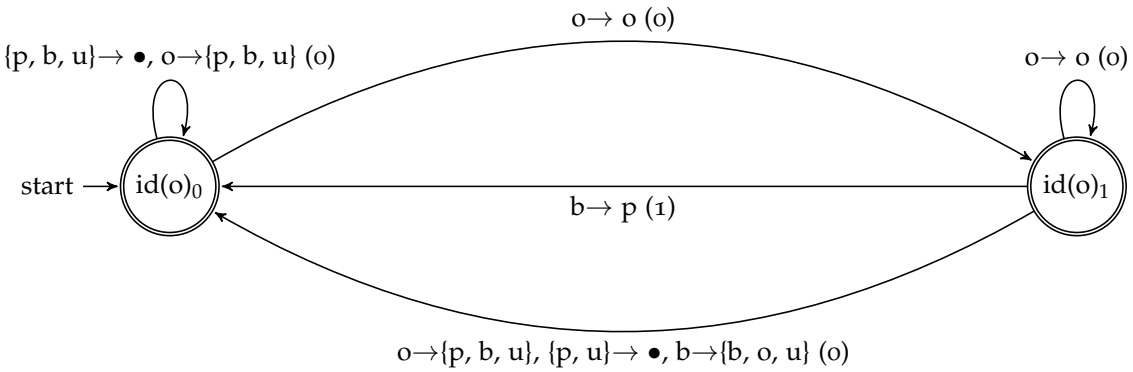

Figure 3: A visualization of **id(o)**, the FST representing IDENT–IO(O).

output symbol o, and the input symbol b corresponds to the output symbol p, will incur a violation. For example, the input–output pair $/\mathrm{ob}/ \rightarrow [\mathrm{op}]$ will incur one total violation, whereas $/\mathrm{op}/ \rightarrow [\mathrm{op}]$ will incur no violations.

Note that **id(o)** is both multistate, because it has two states (and cannot be reduced), and input–dependent, because going from state $\mathrm{id(o)}_1$ to $\mathrm{id(o)}_0$, the pair b→p incurs 1 violation, whereas the pair p→p incurs 0 violations. (This is also why it's not the case that each arrow has a • to its left.) Thus, in a sense, this constraint is a hybrid faithfulness–markedness constraint: it's input–dependent, like a faithfulness constraint, but it's multistate, sensitive to other parts of the output (and input) string, like most markedness constraints. This hybridity perhaps explains the intuition that such constraints are beyond classic OT.

We're now in a position to formally define the broad class of faithfulness constraints, as well as the class of provisional faithfulness constraints that are part of it. A definition of classic OT, which will exclude provisional faithfulness constraints, follows shortly.

Definition 8. A *faithfulness constraint* is any constraint that can be represented by a finite–state constraint $\langle Q, \Sigma, \delta, q_0, F\rangle$ such that $\langle q_j, \alpha, \beta, 1, q_k\rangle \in \delta$ and $\langle q_j, \gamma, \beta, 0, q_k\rangle \in \delta$, for some $\alpha, \beta, \gamma \in \Sigma$ and for some $q_j, q_k \in Q$. We call the FST representing a faithfulness constraint an *input–dependent finite–state constraint*, or more simply a *finite–state faithfulness constraint*

Definition 9. A *provisional faithfulness constraint* is any faithfulness constraint such that for any finite–state faithfulness constraint $\langle Q, \Sigma, \delta, q_0, F\rangle$ that represents it, $|Q| > 1$. We call the FST representing a provisional faithfulness constraint a *multistate faithfulness constraint*. We call any faithfulness constraint that is not a provisional faithfulness constraint a *non–provisional faithfulness constraint*, and we call the FST representing it a *single–state faithfulness constraint*.

A faithfulness constraint is therefore any constraint that is not markedness constraint. This includes both provisional and non–provisional constraints, as

well as antifaithfulness constraints and faithfulness constraints that penalize arbitrary pairs of input–output segments.

We now define a classic OT constraint as follows.

Definition 10. A *classic OT constraint* is any markedness or non–provisional faithfulness constraint. We call an OT grammar *classic* only if its constraint set includes just classic OT constraints.

The reason we include such arbitrary constraints in our definition of a classic OT constraint is the following: we will include any and all faithfulness constraints except provisional ones, and we will prove that even still, this broad version of classic OT cannot express (certain cases of) opacity. That is, we're being as generous as possible in what we consider to be possible constraints, excluding only provisional faithfulness constraints, but it still won't help.

Put another way, this version of classic OT is broad enough to encompass just about any other, more restricted version of classic OT. So if the proof that follows works for this broad version, then it will work for any more restricted version, too.

5 First proof: counterbleeding on environment

The claim we wish to prove is the following: There is a set of input–output patterns which can be expressed by ordered rules but which cannot be expressed by any classic OT grammar, as defined above. The claim can be stated more explicitly as follows.

Claim 1. There is a function U from inputs to outputs such that (i) there are rewrite rules that can be ordered so that for each input–output pair $\langle i, o\rangle \in U$, the rules map i to o; and (ii) there is no ranking $>$, for any set C of markedness and non–provisional faithfulness constraints, such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C, and $>$.

We shall break up this claim into several pieces, proving each piece as a lemma, and then ultimately proving the entire thing.

First, we simplify having to deal with "each set O of output candidates containing o" by proving that it suffices to examine the same, finite candidate set for each input–output pair, namely the set P of all attested outputs. Essentially, the idea is that if there is no ranking $>$ for any set C of constraints such that, for each input i, $>$ and C correctly pick out the attested output of i from P, then there must be no such ranking or constraint set that would do so for each input *i and* for each and every set O of output candidates containing the attested output of i. Intuitively, the reason is that P, being one instantiation of such a set O, is a counterexample.

Lemma 1. Let U be any function from inputs to outputs, and let P be any finite set of output candidates, such that for each $\langle i, o\rangle \in U$, $o \in P$. If there is

no ranking $>$ for any set C of markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in P with respect to i, C and $>$, then it follows that there is no ranking $>$ for any set C of markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, and for each set O of candidates containing o, o is the most optimal candidate in O with respect to i, C, and $>$.[17]

Proof. We prove by contradiction: assume the *if*–clause and the negation of the *then*–clause to derive a contradiction. Assume (ia) that P is a finite set of output candidates such that for each $\langle i, o\rangle \in U$, $o \in P$, and (ib) that there is no ranking $>$ for any set C of markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in P with respect to i, C, and $>$. Assume (ii) that there *is* a ranking $>$ for *some* set C of markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, and for each set O of candidates containing o, o is the most optimal candidate in O with respect to i, C, and $>$. Then, by assumptions (ia) and (ii), there *is* a ranking $>$ for *some* set C of markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in P with respect to i, C, and $>$. But this contradicts assumption (ib). □

Next, we simplify having to deal with "any set C of markedness and non–provisional faithfulness constraints". The problem is that the set of all possible markedness and non–provisional faithfulness constraints is infinitely large: they are functions from the set of all input–output pairs to the natural numbers, but there are infinitely many such functions. This in turn means that there are infinitely many subsets of the infinitely large set of all possible markedness and non–provisional faithfulness constraints.

It turns out that it suffices to consider only the infinitely large set C of all possible markedness and non–provisional faithfulness constraints. That is, we need not concern ourselves with the infinitely many subsets of C. Essentially, the idea is that, if the infinite set of all possible classic OT constraints cannot be ranked to pick out the attested output for each input, given any candidate set containing the attested output, then neither can any subset thereof; and the reason is that, if a subset could in fact be so ranked, then we could just rank all the rest of the infinitely many possible constraints below those of that subset (they would have no effect), and so the whole infinite set of possible constraints would likewise be able to be so ranked.

Lemma 2. Let U be a function from inputs to outputs, and let C be the set of all possible markedness and non–provisional faithfulness constraints, i.e., the set of all functions from the set of all input–output pairs to $\mathbb{N}$ that can be

[17] We do not actually need to require that C contain only markedness and non–provisional faithfulness constraints—C could range over any type of constraint, including all possible constraints—but since our claim is about classic OT constraints, I have kept the restriction on C in this lemma.

represented by a markedness or single–state faithulness constraint.[18] If there is no ranking $\succ$ for C such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C, and $\succ$, then it follows that there is no ranking $>$ for any set $C' \subseteq C$ such that for each $\langle i, o\rangle \in U$, and for each set O containing o, o is the most optimal candidate in O with respect to i, C', and $>$.

Proof. We prove by contradiction: assume the *if*–clause and the negation of the *then*–clause to derive a contradiction. Assume (i) that there is no ranking $\succ$ for the set C of all possible markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C, and $\succ$. Assume (ii) that there *is* a ranking $>$, for *some* set $C' \subseteq C$, such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C', and $>$. Clearly, $C' \neq C$; otherwise, we derive a contradiction: just let $\succ$ be the same ranking on C as the ranking $>$ on C', and we contradict assumpion (i). Thus, C' is a proper subset of C. Let $\succ$ be a ranking defined on C as follows: for each c_j, c_k such that $c_j \in C', c_k \in C - C'$, let $c_j \succ c_k$; for each $c_j, c_k \in C$ such that $c_j, c_k \in C'$, let $c_j \succ c_k$ iff $c_j > c_k$; and let $C - C'$ be ranked in any arbitrary way by $\succ$. In other words, C is ranked so that each member of C' outranks each member of $C - C'$; C' retains the ranking defined by $>$; and $C - C'$ is ranked in any arbitrary way. It follows that $\succ$ is a ranking for the set C of all possible markedness and non–provisional faithfulness constraints such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C, and $\succ$, because by assumption (ii) o is the most optimal candidate in O with respect to i, C', and $>$, and by construction $\succ$ and $>$ are the same ranking for C', and by construction each member of C' outranks each member of $C - C'$ with respect to $\succ$. But this contradicts assumption (i). □

To recap: to prove claim 1, it suffices to find a set U of input–output patterns and

1. to show that there is a set of rewrite rules that can be ordered so that for each $\langle i, o\rangle \in U$, the rules map i to o (Lemma 3 below);
2. to consider just a single, finite set O of output candidates such that for each $\langle i, o\rangle \in U$, $o \in O$ (Lemma 1);
3. to consider just the set C of all possible markedness and non–provisional faithfulness constraints (Lemma 2); and
4. to prove that there is no ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal output candidate in O with respect to i, C, and $>$ (Lemma 4 below).

[18] Again, for this lemma we do not actually need to limit C to only classic OT constraints, but we do so anyway because of the particular claim we wish to prove.

Taking a cue from the countebleeding on environment opacity in Polish, we shall prove claim 1 by letting U contain the "opaque" pattern /ʒwob/ → [ʒwup] and the hypothetical pattern /ʒwop/ → [ʒwop], and by letting O be the set containing the two output candidates [ʒwop] and [ʒwup].

One might immediately object to the use of a hypothetical pattern like /ʒwop/ → [ʒwop]. However, given the rewrite rules that we'll posit, we would expect similar patterns like /gop/ → [gop] and /bop/ → [bop], all of which would serve our purposes but complicate the proof exceedingly. The fact that Polish may not contain the input /ʒwop/ would be more of an accidental gap than anything having to do with opacity. More importantly, though, the proof that follows from this set of patterns should be considered a formal result: there are patterns (attested or otherwise) which are expressible by ordered rules by not by classic OT grammars. Whether those patterns occur in phonology is an important but separate, empirical question. We return to this issue later in the paper, once the reader is familiar with the proof technique, where we demonstrate that the proof can be applied to another set of patterns which are fully attested (Canadian raising).

For the sake of completeness, we prove as a lemma that our two patterns can be captured with ordered rules.

Lemma 3. Let $U = \{\langle$ʒwob, ʒwup$\rangle, \langle$ʒwop, ʒwop$\rangle\}$ be a function from inputs to outputs. Then there is a set of rewrite rules that can be ordered so that they map /ʒwob/ to [ʒwup] and /ʒwop/ to [ʒwop].

Proof. Let $\mathcal{R}$ be the raising rule discussed in section 3, let $\mathcal{D}$ be the devoicing rule, and let $\mathcal{R}$ be ordered before $\mathcal{D}$. Then $\mathcal{R}$ maps /ʒwob/ to ʒwub, which $\mathcal{D}$ maps to [ʒwup], and $\mathcal{R}$ vacuously maps /ʒwop/ to ʒwop, which $\mathcal{D}$ vacuously maps to [ʒwop]. □

Now, the final lemma which we must prove, and which together with all the previous lemmas proves Claim 1, is the following.

Lemma 4. Let $I = \{$ʒwob, ʒwup$\}$ be a set of inputs, let $O = \{$ʒwop, ʒwup$\}$ be a set of output candidates, let $U = \{\langle$ʒwob, ʒwup$\rangle, \langle$ʒwop, ʒwop$\rangle\}$ be a function from I to O, and let C be the set of all markedness and non–provisional faithfulness constraints, i.e., all functions from $I \times O$ to $\mathbb{N}$ that can be represented by a markedness or single–state faithfulness constraint. Then there is no ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in O with respect to i, C, and $>$.

Proof. We prove by contradiction: assume the negation and derive a contradiction. Assume that there is a ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in O with respect to i, C, and $>$. Since U has only two members, and since O has only two members, then it follows that $>$ is a ranking for C such that [ʒwup] is more optimal than [ʒwop] with respect to /ʒwob/, C, and $>$, and [ʒwop] is more optimal than [ʒwup] with respect to /ʒwop/, C, and $>$. But we also know that a candidate o_1 is more optimal than a candidate o_2 if and only if the highest ranked constraint on which o_1 and o_2

differ (are assigned a different number of violations) assigns fewer violations to o_1 than to o_2. It follows, then, that there is a ranking $>$ for C such that the highest ranked constraint on which [ʒwup] and [ʒwop] differ with respect to /ʒwob/, C, and $>$ assigns fewer violations to [ʒwup] than to [ʒwop]. Call this constraint c_j for the remainder of the proof. Likewise, the highest ranked constraint on which [ʒwup] and [ʒwop] differ with respect to /ʒwop/, C, and $>$ assigns fewer violations to [ʒwop] than to [ʒwup]. Call this constraint c_k for the remainder of the proof.

We can visualize these deductions in the form of two hypothetical tableaux t_1 and t_2. (We call them *–tableaux to distinguish them from a different type of tableau to come.)

(15) **Hypothetical *–tableau t_1**

	/ʒwob/	c_1	c_2	...	c_j	...
	a. ʒwop	?	?	?	?	?
→	b. ʒwup@	?	?	?	?	?

(16) **Hypothetical *–tableau t_2**

	/ʒwop/	c_1	c_2	...	c_k	...
→	a. ʒwop@	?	?	?	?	?
	b. ʒwup	?	?	?	?	?

The question marks are a informal indicator that we don't know how many violations each constraint assigns. We do know, however, that in t_1, each constraint to the left of c_j assigns the same number of violations to each candidate; c_j assigns [ʒwup] a smaller number of violations than to [ʒwop]; and the constraints to the right of c_j may assign any number of violations. The same can be said, *mutatis mutandis*, for t_2 as regards c_k. We do *not* know how c_j, c_k are ranked with respect to each other, and we do *not* know how many violations c_j assigns to [ʒwop] or [ʒwup] for the input /ʒwop/, and we do *not* know how many violations c_k assigns to [ʒwop] or [ʒwup] for the input /ʒwob/.

Ultimately, we want to deduce the unknown violation profiles of c_j and c_k and to show that at least one of c_j, c_k has a violation profile that can only be represented by a multistate faithfulness constraint, contra our assumption.

Now, informally, we can construct a set of two slightly different sort of tableaux—call them *wl*–tableaux, consisting of *wl*–constraints—that preserve information from the original two tableaux regarding whether a given constraint for a given tableau favors one candidate (the *winner*) over the other (the *loser*), or rather favors neither (both are default winners, so to speak), and discards such information as how *much* a winner wins by. For example, we can transform the two *–tableaux above to the following two *wl*–tableaux, where each c_i' in the new tableaux corresponds to c_i in the original tableaux, and so the set of *wl*–constraints retains the same ranking as the original constraints.[19]

[19]At first glance, it may seem like the *wl*–tableaux convey *more* information than the original tableaux: the original tableaux are full of question marks, whereas the *wl*–tableaux convey information about winners and losers. Recall, however, that the question marks are an informal

(17) **Hypothetical *wl*–tableau for t_1**

	/ʒwob/	c'_1	c'_2	…	c'_j	…
	a. ʒwop	w	w	w	l	?
→	b. ʒwup$_{@}$	w	w	w	w	?

(18) **Hypothetical *wl*–tableau for t_2**

	/ʒwop/	c'_1	c'_2	…	c'_k	…
→	a. ʒwop$_{@}$	w	w	w	w	?
	b. ʒwup	w	w	w	l	?

For the hypothetical *wl*–tableau for t_1, each constraint to the left of c'_j assigns w to both candidates; c'_j assigns w to [ʒwup] and l to [ʒwop]; and each constraint to the right of c_j may assign either w to both candidates, or w to one candidate and l to the other. No constraint in the *wl*–tableau for t_1 assigns l to both candidates because of how we stated the requirements on the construction of the tableau: intuitively, two l's would correspond to a constraint that favors neither candidate, but such a constraint is instead required to assign two w's. All of what has just been said may be said, *mutatis mutandis*, for the *wl*–tableau for t_2 with respect to c'_k. Moreover, we do *not* know how c'_j, c'_k are ranked with respect to each other, and we do *not* know what letter c'_j assigns to [ʒwop] or [ʒwup] for input /ʒwop/, and we do *not* know what letter c'_k assigns to [ʒwop] or [ʒwup] for input /ʒwob/.

What we have done, essentially, is define a function ϕ that maps a *–assigning constraint, i.e., a function from $I \times O$ to $\mathbb{N}$ that assigns to each input–output pair some non–negative number of violations, to a function from $I \times O$ to $\{w, l\}$ that assigns to each input–output pair either w or l. Let's define ϕ more formally, since it plays a crucial role in this proof and in this paper more generally.

Definition 11. Let I be any set of inputs (in our case, {ʒwob, ʒwop}), let O be any set of outputs (in our case, {ʒwup, ʒwop}), and let c be any function from $I \times O$ to $\mathbb{N}$. Then $\phi(c) =_{\text{df}} c'$ is the unique function from $I \times O$ to $\{w, l\}$ such that for each $i \in I$, and for each $o_j \in O$:

$$c'(\langle i, o_j \rangle) = \begin{cases} w & \text{if } c(\langle i, o_j \rangle) \leq c(\langle i, o_k \rangle), \quad \forall o_k \in O; \\ l & \text{otherwise, i.e., if } \exists o_k \in O : c(\langle i, o_j \rangle) > c(\langle i, o_k \rangle) \end{cases}$$

We call a function from $I \times O$ to $\mathbb{N}$ a **–constraint* (because traditionally it assigns some number of *'s), and we call a constraint $I \times O$ to $\{w, l\}$ a *wl–constraint*. There are infinitely many *–constraints because there are infinitely many functions from any non–empty set to $\mathbb{N}$. However, for a finite input

annotation indicating our ignorance about the precise numbers of violations. If we knew what the constraints were, then we would fill in the tableaux with numbers, and we would know not only the winners and losers, but also the precise number of violations. The *wl*–tableaux are really just a way to visualize the deductions we already made from the *–tableaux.

set and a finite set of output candidates, there are only finitely many *wl*–constraints, because there are only finitely many functions from a finite set to a finite set. In our case, there are

$$|\{w, l\}|^{|I \times O|} = 2^{2 \cdot 2} = 2^4 = 16$$

possible *wl*–constraints, in principle. Of course, as we noted, it's not possible for a *–constraint to get mapped by ϕ to a *wl*–constraint that assigns *l* to both candidates for a given input. So while the co–domain of ϕ has exactly 16 members, we already know that the range (or image) of ϕ has at most 9 members (16 minus the 7 that assign *l* to both candidates for some input).

An equivalent way of thinking about ϕ is that it induces a partition on *C* consisting of at most 16 non–overlapping classes of *–constraints: two *–constraints c_1, c_2 are in the same class if and only if $\phi(c_1) = \phi(c_2)$. Here are some examples of *possible* classes of *–constraints: the class of *–constraints which ϕ maps to the *wl*–constraint which assigns *w* to both candidates for both inputs; the class of *–constraints which ϕ maps to the *wl*–constraint which assigns *w* to both candidates for /ʒwob/ and which assigns *w* to [ʒwop] and *l* to [ʒwup] for /ʒwop/; the class of *–constraints which ϕ maps to the *wl*–constraint which assigns *l* to both candidates for both inputs; etc. There are 16 *possible* classes in total, but again, some of these 16 classes will turn out not to contain any *–constraints. For example, we already know that the last example is one such empty (more precisely, inexistent) class: by definition, there is no *–constraint *c* such that $\phi(c)$ assigns *l* to both candidates for both inputs.

Our goal now is to verify that c_j, c_k are in fact, as we have assumed, both *–constraints that can be represented by a markedness or single–state faithfulness constraint. We do so by first determining which *wl*–constraints $\phi(c_j)$ and $\phi(c_k)$ might possibly be and then verifying that in each case, at least one of these possible *wl*–constraints is the image of some *–constraint that can be represented by a markedness or single–state faithfulness constraint. If we find that at least one of $\phi(c_j)$ or $\phi(c_k)$ must be a *wl*–constraint that can only be the image of a provisional faithfulness constraint, then we will have succeeded in deriving a contradiction.

Here are two tables illustrating all 9 possible *wl*–constraints that $\phi(c_j)$ and $\phi(c_k)$ could be, once those 7 that assign *l* to both candidates for a given input are eliminated.

(19) **Possible *wl*–constraints for t_1**

/ʒwob/	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
a. ʒwop	*w*	*l*	*w*	*w*	*w*	*l*	*l*	*w*	*w*
b. ʒwup$_@$	*w*	*w*	*l*	*w*	*w*	*w*	*w*	*l*	*l*

(20) **Possible *wl*–constraints for t_2**

/ʒwop/	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
a. ʒwop$_@$	*w*	*w*	*w*	*l*	*w*	*l*	*w*	*l*	*w*
b. ʒwup	*w*	*w*	*w*	*w*	*l*	*w*	*l*	*w*	*l*

Since c_j by assumption assigns fewer violations to [ʒwup] than to [ʒwop] for input /ʒwob/, then $\phi(c_j)$ assigns w to [ʒwup] and l to [ʒwop] for input /ʒwob/. Thus, $\phi(c_j)$ can be either m_2, m_6, or m_7.

Since c_k by assumption assigns fewer violations to [ʒwop] than to [ʒwup] for input /ʒwop/, then $\phi(c_k)$ assigns w to [ʒwop] and l to [ʒwup] for input /ʒwop/. Thus, $\phi(c_k)$ can be either m_5, m_7, or m_9.

We see immediately that m_6 is the image of *–constraints like *op, *wop, etc., and that m_9 is the image of *–constraints like *up, *wup, etc., all of which can be represented by finite–state markedness constraints. However, it turns out to be impossible both that $\phi(c_j) = m_6$ and that $\phi(c_k) = m_9$.

Suppose for a moment that $\phi(c_j) = m_6$ and that $\phi(c_k) = m_9$. Then since $\phi(c_j) \neq \phi(c_k)$, it follows from the definition of ϕ as a function that $c_j \neq c_k$. Therefore, either c_j outranks c_k, or c_k outranks c_j. However, c_j cannot outrank c_k because we know from $\phi(c_j) = m_6$ that c_j assigns fewer violations to [ʒwup] than to [ʒwop] for /ʒwop/, which contradicts our earlier deduction that the highest ranked constraint on which [ʒwop] and [ʒwup] differ for /ʒwop/ assigns fewer violations to [ʒwop]; and c_k cannot outrank c_j for a similar reason. Thus, it cannot be the case that both $\phi(c_j) = m_6$ and $\phi(c_k) = m_9$. In other words, either $\phi(c_j) = m_2$ or $\phi(c_j) = m_7$, or else $\phi(c_k) = m_5$ or $\phi(c_k) = m_7$.

Some reflection reveals, however, that m_2, m_5, and m_7 are all the images, under ϕ, of constraints that can only be represented by a multistate faithfulness constraint.

Note first that none of m_2, m_5, or m_7 corresponds to any markedness constraint. The reason is that any markedness constraint c, being input–independent, would assign the same number of violations to [ʒwop] for both inputs /ʒwob/ and /ʒwop/ and would assign the same number of violations to [ʒwup] for both inputs /ʒwob/ and /ʒwop/. Thus, $\phi(c)$ would assign the same letter to [ʒwop] for both inputs /ʒwob/ and /ʒwop/ and would assign the same letter to [ʒwup] for both inputs /ʒwob/ and /ʒwop/. However, m_2, m_5, and m_7 all assign different letters to at least one of [ʒwop] and [ʒwup] for the inputs /ʒwob/ and /ʒwop/.

Consider now m_2, and suppose that $\phi(c_j) = m_2$. Then it must be that

$$c_j(\langle \text{ʒwob}, \text{ʒwop} \rangle) = c_j(\langle \text{ʒwob}, \text{ʒwup} \rangle) + a, \quad a > 0$$

because m_2 assigns w to [ʒwup] and l to [ʒwop] for /ʒwob/. Since the only difference between [ʒwop] and [ʒwup] is the third segment, it follows that the single–state FST representing c_j must assign a more violations to the pair o→o than to the pair o→u.[20] This should mean that

$$c_j(\langle \text{ʒwop}, \text{ʒwop} \rangle) = c_j(\langle \text{ʒwop}, \text{ʒwup} \rangle) + a, \quad a > 0$$

which should in turn mean that $\phi(c_j) = m_2$ assigns w to [ʒwup] and l to [ʒwop] for /ʒwop/. But m_2 does not. Therefore, it cannot be that $\phi(c_j) = m_2$. Rather, m_2

[20] Since our FSTs assign either 0 or 1 to any given pair of input–output symbols, it must be that $a = 1$. All other pairs may be assigned 0 or 1.

must correspond to provisional faithfulness constraint like AntiIdent–IO(O), discussed in sections 3 and 4.

Consider now m_5, and suppose that $\phi(c_k) = m_5$. Then it must be that

$$c_k(\langle \text{ʒwob}, \text{ʒwop} \rangle) = c_k(\langle \text{ʒwob}, \text{ʒwup} \rangle)$$

because m_5 assigns w to both [ʒwop] and [ʒwup] for /ʒwob/. Thus, the single–state FST representing c_k assigns exactly as many violations to the pair o→o as to the pair o→u. This should mean that

$$c_k(\langle \text{ʒwop}, \text{ʒwop} \rangle) = c_k(\langle \text{ʒwop}, \text{ʒwup} \rangle)$$

which should in turn mean that $\phi(c_k) = m_5$ assigns w to both [ʒwop] and [ʒwup] for /ʒwop/. But m_5 does not. Therefore, it cannot be that $\phi(c_k) = m_5$. Rather, m_5 must correspond to a provisional faithfulness constraint we might call Ident–IO(O) with the following definition: "Assign one violation mark for each occurrence of an output [u] in correspondence with an input /o/, provided that the output [u] is followed immediately by a [p] in correspondence with an input /p/." The FST representing this constraint would require two states, with the second state representing the mapping o→u, and the only violation being assigned on the transition from this second state back to the first for the pair p→p, but not for the pair b→p.

Since m_2 and m_5 both correspond to provisional faithfulness constraints, it must be that $\phi(c_j) = \phi(c_k) = m_7$. Again, it must be that

$$c_j(\langle \text{ʒwob}, \text{ʒwop} \rangle) = c_j(\langle \text{ʒwob}, \text{ʒwup} \rangle) + a, \quad a > 0$$

from which it should follow that

$$c_j(\langle \text{ʒwop}, \text{ʒwop} \rangle) = c_j(\langle \text{ʒwop}, \text{ʒwup} \rangle) + a, \quad a > 0$$

This should in turn mean that $\phi(c_j) = m_7$ assigns w to [ʒwup] and l to [ʒwop] for /ʒwop/. But m_7 does not. Therefore, it cannot be that $\phi(c_j) = m_7$. By similar reasoning, it cannot be that $\phi(c_k) = m_7$. Rather, m_7 must correpsond to a provisional faithfulness constraint we might call

$$[\text{Ident–IO(O)} \vee \text{AntiIdent–IO(O)}]$$

where Ident–IO(O) and AntiIdent–IO(O) are the provisional faithfulness constraints from above, and this disjunctive constraint assigns a violation whenever either (or both) of its disjuncts assigns a violation. As a disjunction of provisional faithfulness constraints, this faithfulness constraint is likewise necessarily provisional: to check whether the its conditions are met requires checking whether the conditions of each of its disjuncts are met, which in turn requires multiple states.

At last, we have reached our contradiction. At least one of c_j, c_k must be a provisional faithfulness constraint. But C, which contains c_j, c_k, contains only markedness and non–provisional faithfulness constraints. Thus, our

assumption must be wrong that there is a ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate with respect to i, C, and $>$.

□

We can now prove Claim 1 as the following theorem.

Theorem 1. There is a function U from inputs to outputs such that (i) there are rewrite rules that can be ordered so that for each input–output pair $\langle i, o\rangle \in U$, the rules map i to o; and (ii) there is no ranking $>$, for any set C of markedness and non–provisional faithfulness constraints, such that for each $\langle i, o\rangle \in U$, and for each set O of output candidates containing o, o is the most optimal candidate in O with respect to i, C, and $>$.

Proof. The proof follows from Lemmas 1, 2, 3, and 4. □

In the next section, we consider a case of counterfeeding on environment opacity in Isthmus Nahuat. Based on those patterns, we prove in section 7 that there is yet another set of patterns which can be expressed by ordered rules—this time ordered in a counterfeeding on environment relationship—but not by any classic OT grammar.

6 Counterfeeding on environment opacity in Isthmus Nahuat

In this section we discuss yet another set of patterns that would seem to satisfy Theorem 1. The patterns are from Isthmus Nahuat, and the difference between these patterns and those in Polish is that these can be captured by rules ordered in a counter*feeding* on environment relationship, not a counter*bleeding* on environment relationship.

Consider the following data from Isthmus Nahuat (Law, 1958; Kenstowicz and Kisseberth, 1979; Kager, 1999).

(21) **Isthmus Nahuat data** Glosses:

a. /támi/ surfaces as [tám]. "it ends"
b. /tájoːl/ surfaces as [tájːo̥l̥]. "shelled corn"
c. /ʃikɑkíli/ surfaces as [ʃikɑkíl]. "put it in it"

All three patterns can be captured by a rule $\mathcal{D}$ of word–final approximant devoicing ordered before a rule $\mathcal{A}$ of apocope (deletion of a word–final, unstressed vowel).

(22) **Isthmus Nahuat rules**

a. $\mathcal{D}$: [−syllabic, +sonorant, −nasal] → [−voice] / _#
b. $\mathcal{A}$: $\grave{\text{V}} \rightarrow \emptyset$ / _#

$\mathcal{D}$ vacuously maps /támi/ to támi, which $\mathcal{A}$ maps to [tám]. $\mathcal{D}$ maps /tájoːl/ to tájoːl̥, which $\mathcal{A}$ vacuously maps to [tájoːl̥]. Finally, $\mathcal{D}$ vacuously maps /ʃikɑkíli/ to ʃikɑkíli, which $\mathcal{D}$ maps to [ʃikɑkíl].

The two rules are in a counterfeeding relationship. Specifically, $\mathcal{A}$ counterfeeds on $\mathcal{D}$'s environment: $\mathcal{A}$ "adds" $\mathcal{D}$'s environment (right edge of the word) by deleting a segment following $\mathcal{D}$'s focus (an approximant).[21] For example, the last occurrence of /i/ in /ʃikɑkíli/, which does not match $\mathcal{D}$'s environment, is deleted by $\mathcal{A}$, thus creating a word–boundary and matching $\mathcal{D}$'s environment.

Another way to think about it is that if $\mathcal{A}$ were ordered *before* $\mathcal{D}$, then $\mathcal{A}$ *would* feed on $\mathcal{D}$'s environment: $\mathcal{A}$ would map /ʃikɑkíli/ to ʃikɑkíl, which $\mathcal{D}$ would map to [ʃikɑkíl̥].

The rule $\mathcal{D}$ of devoicing is considered opaque in the pattern /ʃikɑkíli/ → [ʃikɑkíl] in the sense that the input description for $\mathcal{D}$—namely, there being a word–final approximant—holds true of the surface form, so $\mathcal{D}$ seems not to have applied when it could have.

Let's try to capture these patterns in OT. For apocope, we could posit a markedness constraint that penalizes word–final (unstressed) vowels, which is ranked higher than a faithfulness constraint that penalizes vowel deletion (or deletion in general). For word–final approximant devoicing, we could posit a markedness constraint that penalizes word–final voiced approximants, which is ranked higher than a faithfulness constraint that penalizes devoicing approximants (or devoicing in general).

(23) **Isthmus Nahuat constraints**

a. Apocope
*V̀# > Max–IO

b. Devoicing
*[+sonorant, −nasal, +voice]# > Ident–IO(voice)

For readability, let's relabel the markedness constraint for devoicing as *l#.

Note that we don't know how either constraint in (23a) is ranked with respect to either constraint in (23b). However, to pursue an analysis we must choose some total ranking. One possibility is the following. (As with Polish, we'll see shortly that no ranking of these four constraints that respects (23) can work.)

(24) **Isthmus Nahuat hypothetical constraint ranking**
*V̀# > Max–IO > *l# > Ident–IO(voice)

Below are two tableaux that verify that these four constraints and this ranking correctly make [tám] a more optimal output of /támi/ than [támi] and make [tájoːl̥] a more optimal output of /tájoːl/ than [tájoːl].

(25) **OT analysis of** /támi/ → [tám]

	/támi/	*V̀#	Max	*l#	Id(vce)
	a. támi	1			
→	b. tám@		1		

[21] Note that not only does $\mathcal{A}$ counterfeed on $\mathcal{D}$'s environment, but $\mathcal{D}$ also feeds on $\mathcal{A}$'s environment.

(26) **OT analysis of** /tájoːl/ → [tájoːl̥]

	/tájoːl/	*V̀#	Max	*l#	Id(vce)
	a. tájoːl			1	
→	a. tájoːl̥@				1

We now turn to the pattern /ʃikakíli/ → [ʃikakíl̥]. Below is a tableau consisting of our four constraints, ranked according to our hypothesis, and the four obvious output candidates that are in competition.

(27) **Attempted OT analysis of** /ʃikakíli/ → [ʃikakíl]

	/ʃikakíli/	*V̀#	Max	*l#	Id(vce)
	a. ʃikakíli	1			
	b. ʃikakíl@		1	1	
→	c. ʃikakíl̥		1		1

As with Polish, this set of constraints and this ranking unfortunately favor an unattested form, namely [ʃikakíl̥]. Unlike with Polish, however, the violations incurred by the attested form, [ʃikakíl], are not a proper superset of those incurred by [ʃikakíl̥]. But it turns out that we're no better off. These two candidates only differ on the two constraints Ident–IO(voice) and *l#. But since we know that *l# outranks Ident–IO(voice), it's clear that, given just these four constraints, and no matter how the other two constraints are ranked with respect to Ident–IO(voice) or *l# or to each other, [ʃikakíl̥] is a more optimal output of /ʃikakíli/ than the attested output, [ʃikakíl].

Intuitively, no markedness constraint seems to be of help. Presumably, outputs containing the sequence [íl̥] word–finally are unmarked in this language, and the two rewrite rules would likewise predict such input–output pairs as, say, /ʃikakíl/ → [ʃikakíl̥].

Moreover, no non–provisional faithfulness constraint jumps out as being helpful. The only differences between the input /ʃikakíli/ and the two output candidates [ʃikakíl] and [ʃikakíl̥] are that the final /i/ is deleted in both and that /l/ is devoiced in one. But both of these faithfulness violations are already captured by Max–IO and Ident–IO(voice), respectively.

It seems, rather, that we require a faithfulness constraint we might call Ident–IO(L) with the following meaning: "Assign one violation mark for each occurrence of an output [l̥] in correspondence with an input /l/, provided that the input /l/ is followed immediately by an /i/ with no output correspondent (or whose output correspondent is the empty string/phonologically null segment)."[22] Ident–IO(L) is a provisional faithfulness constraint: the FST that represents it requires a second state to keep track of having processed the pair l→l̥. Thus, we would prefer to find an analysis without such a constraint. In the next section we prove, however, that no such analysis exists.

[22] Ident–IO(L) is similar to the *local conjunction* of Ident–IO(voice) and Max–IO, which we might write as [Ident–IO(voice) & Max–IO]$_\delta$ (Smolensky, 1993). Crucially, the domain δ of this local conjunction spans two different input segments and two different output segments, and it assigns a violation whenever both Ident–IO(voice) and Max–IO are violated within δ.

7 Second proof: counterfeeding on environment

We now prove Theorem 1 using patterns based on the data from Isthmus Nahuat. Of course, since Theorem 1 has already been proved, perhaps a better way of stating this section's result is that we prove the counterfeeding on environment analogs of Lemmas 3 and 4, which together with Lemmas 1 and 2 will then (re)prove Theorem 1.

We shall prove Theorem 1 by letting U be the set containing the "opaque" pattern /ʃikakíli/ → [ʃikakíl] and the hypothetical pattern /ʃikakíl/ → [ʃikakíl̥], and by letting O be the set containing the two output candidates [ʃikakíl] and [ʃikakíl̥].

Once again, we allay the worries of those who object to the use of a hypothetical pattern by noting that the pattern is entirely predicted by the two rewrite rules we just discussed and that we would expect similar patterns like /kíl/ → [kíl̥]. That Isthmus Nahuat may not contain the input /ʃikakíl/ would therefore be more of an accidental gap than anything having to do with opacity. In addition, the proof in this section should be considered another formal result: specifically, there is yet another set of patterns, different in nature from those of the first proof, which are expressible by ordered rewrite rules but not by classic OT grammars.

For the sake of completeness, we prove as a lemma that our two patterns can be captured with ordered rules.

Lemma 5. Let $U = \{\langle$ʃikakíli, ʃikakíl$\rangle, \langle$ʃikakíl, ʃikakíl̥$\rangle\}$ be a function from inputs to outputs. Then there is a set of rewrite rules that can be ordered so that they map /ʃikakíli/ to [ʃikakíl] and /ʃikakíl/ to [ʃikakíl̥].

Proof. Let $\mathcal{D}$ be the word–final approximant devoicing rule from section 6, let $\mathcal{A}$ be the apocope rule, and let $\mathcal{D}$ be ordered before $\mathcal{A}$. Then $\mathcal{D}$ vacuously maps /ʃikakíli/ to ʃikakíli, which $\mathcal{A}$ maps to [ʃikakíl], and $\mathcal{D}$ maps /ʃikakíl/ to ʃikakíl̥, which $\mathcal{A}$ vacuously maps to [ʃikakíl̥]. □

We now prove the counterfeeding on environment analog to Lemma 4. The proof will go much faster this time since it uses the same groundwork laid down for Lemma 4.

Lemma 6. Let $I = \{$ʃikakíli, ʃikakíl$\}$ be a set of inputs, let $O = \{$ʃikakíl, ʃikakíl̥$\}$ be a set of output candidates, let $U = \{\langle$ʃikakíli, ʃikakíl$\rangle, \langle$ʃikakíl, ʃikakíl̥$\rangle\}$ be a function from I to O, and let C be the set of all markedness and non–provisional faithfulness constraints, i.e., all functions from $I \times O$ to $\mathbb{N}$ that can be represented by markedness or single–state faithfulness constraints. Then there is no ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in O with respect to i, C, and $>$.

Proof. We prove by contradiction: assume the negation and derive a contradiction. Assume that there is a ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate in O with respect to i, C, and $>$. Since U has only two members, and since O has only two members, then it follows that

$>$ is a ranking for C such that [ʃikakíl] is more optimal than [ʃikakíl̥] with respect to /ʃikakíli/, C, and $>$, and [ʃikakíl̥] is more optimal than [ʃikakíl] with respect to /ʃikakíl/, C, and $>$. But we also know that a candidate o_1 is more optimal than a candidate o_2 if and only if the highest ranked constraint on which o_1 and o_2 differ (are assigned a different number of violations) assigns fewer violations to o_1 than to o_2. It follows, then, that $>$ is a ranking for C such that the highest ranked constraint on which [ʃikakíl] and [ʃikakíl̥] differ with respect to /ʃikakíli/, C, and $>$ assigns fewer violations to [ʃikakíl] than to [ʃikakíl̥]. Call this constraint c_j for the remainder of the proof. Likewise, the highest ranked constraint on which [ʃikakíl] and [ʃikakíl̥] differ with respect to /ʃikakíl/, C, and $>$ assigns fewer violations to [ʃikakíl̥] than to [ʃikakíl]. Call this constraint c_k for the remainder of the proof.

As before, we can visualize these deductions in the form of two hypothetical tableaux t_1 and t_2.

(28) **Hypothetical *–tableau t_1**

	/ʃikakíli/	c_1	c_2	…	c_j	…
→	a. ʃikakíl@	?	?	?	?	?
	b. ʃikakíl̥	?	?	?	?	?

(29) **Hypothetical *–tableau t_2**

	/ʃikakíl/	c_1	c_2	…	c_j	…
	a. ʃikakíl	?	?	?	?	?
→	b. ʃikakíl̥@	?	?	?	?	?

Recycling our function ϕ from Definition 11, we can map each $c_i \in C$ to $\phi(c_i)$, retain the ranking, and visualize the result with the following two hypothetical wl–tableaux.

(30) **Hypothetical wl–tableau for t_1**

	/ʃikakíli/	$\phi(c_1)$	$\phi(c_2)$	…	$\phi(c_j)$	…
→	a. ʃikakíl@	w	w	w	w	?
	b. ʃikakíl̥	w	w	w	l	?

(31) **Hypothetical wl–tableau for t_2**

	/ʃikakíl/	$\phi(c_1)$	$\phi(c_2)$	…	$\phi(c_j)$	…
	a. ʃikakíl	w	w	w	l	?
→	b. ʃikakíl̥@	w	w	w	w	?

Our goal now is, as before, to verify that c_j, c_k are in fact, as we have assumed, both *–constraints that can be represented by a markedness or single–state faithfulness constraint. We do so by first determining which wl–constraints $\phi(c_j)$ and $\phi(c_k)$ might possibly be and then verifying that in each case, at least one of these possible wl–constraints is the image of some *–constraint that can be represented by a single–state markedness or faithfulness constraint. If it turns out that at least one of $\phi(c_j), \phi(c_k)$ must be a wl–constraint that can only be the image of a provisional faithfulness constraint, then we will

have derived a contradiction.

Here again are two tables illustrating the 9 possible *wl*–constraints that $\phi(c_j)$ and $\phi(c_k)$ could be, once those 7 that assign *l* to both candidates for a given input are eliminated.

(32) **Possible *wl*–constraints for t_1**

/ʃikakíli/	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
a. ʃikakíl$_{@}$	*w*	*l*	*w*	*w*	*w*	*l*	*l*	*w*	*w*
b. ʃikakíl̥	*w*	*w*	*l*	*w*	*w*	*w*	*w*	*l*	*l*

(33) **Possible *wl*–constraints for t_2**

/ʃikakíl/	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
a. ʃikakíl	*w*	*w*	*w*	*l*	*w*	*l*	*w*	*l*	*w*
b. ʃikakíl̥$_{@}$	*w*	*w*	*w*	*w*	*l*	*w*	*l*	*w*	*l*

Since c_j by assumption assigns fewer violations to [ʃikakíl] than to [ʃikakíl̥] for input /ʃikakíli/, then $\phi(c_j)$ assigns *w* to [ʃikakíl] and *l* to [ʃikakíl̥] for input /ʃikakíli/. We do *not* know what letter c_j assigns to either candidate for input /ʃikakíl/. Thus, $\phi(c_j)$ can be either m_3, m_8, or m_9.

Since c_k by assumption assigns fewer violations to [ʃikakíl̥] than to [ʃikakíl] for input /ʃikakíl/, then $\phi(c_k)$ assigns *w* to [ʃikakíl̥] and *l* to [ʃikakíl] for input /ʃikakíl/. We do *not* know what letter c_k assigns to either candidate for input /ʃikakíli/. Thus, $\phi(c_k)$ could be either m_4, m_6, or m_8.

We see immediately that m_9 is the image of *–constraints like *íl̥, *kíl̥, etc., and that m_6 is the image of *–constraints like *íl, *kíl, etc., all of which can be represented by finite–state markedness constraints. However, as before, we can easily deduce that it's not the case that both $\phi(c_j) = m_9$ and $\phi(c_k) = m_6$. (We leave the details aside.) Thus, either $\phi(c_j) = m_3$ or $\phi(c_j) = m_8$, or else $\phi(c_k) = m_4$ or $\phi(c_k) = m_8$.

Some reflection reveals, however, that m_3, m_4, and m_8 are all the images, under ϕ, of constraints that can only be represented by a multistate faithfulness constraint.

Note first that none of m_3, m_4, or m_8 corresponds to any markedness constraint. The reason, as in the first proof, is that any markedness constraint *c*, being input–independent, would assign the same number of violations to [ʃikakíl] for both inputs /ʃikakíli/ and /ʃikakíl/ and would assign the same number of violations to [ʃikakíl̥] for both inputs /ʃikakíli/ and /ʃikakíl/. Thus, $\phi(c)$ would assign the same letter to [ʃikakíl] for both inputs /ʃikakíli/ and /ʃikakíl/ and would assign the same letter to [ʃikakíl̥] for both inputs /ʃikakíli/ and /ʃikakíl/. However, m_3, m_4, and m_8 all assign different letters to at least one of [ʃikakíl] and [ʃikaíl̥] for the inputs /ʃikakíli/ and /ʃikakíl/.

Consider now m_3, and suppose that $\phi(c_j) = m_3$. Then it must be that

$$c_j(\langle \text{ʃikakíli, ʃikakíl̥} \rangle) = c_j(\langle \text{ʃikakíli, ʃikakíl} \rangle) + a, \quad a > 0$$

because m_3 assigns *w* to [ʃikakíl] and *l* to [ʃikakíl̥] for /ʃikakíli/. Since the only difference between [ʃikakíl] and [ʃikakíl̥] is the final segment, it follows that

the single–state FST representing c_j assigns a more violations to the pair l→l̥ than to the pair l→l.[23] This should mean that

$$c_j(\langle \text{ʃikakíl}, \text{ʃikakíl̥}\rangle) = c_j(\langle \text{ʃikakíl}, \text{ʃikakíl}\rangle) + a, \quad a > 0$$

which should in turn mean that $\phi(c_j) = m_3$ assigns w to [ʃikakíl] and l to [ʃikakíl̥] for /ʃikakíl/. But m_3 does not. Therefore, it cannot be that $\phi(c_j) = m_3$. Rather, m_3 must correspond to a provisional faithfulness constraint like IDENT–IO(L), discussed in section 6.

Consider now m_4, and suppose that $\phi(c_k) = m_4$. Then it must be that

$$c_k(\langle \text{ʃikakíli}, \text{ʃikakíl}\rangle) = c_k(\langle \text{ʃikakíli}, \text{ʃikakíl̥}\rangle)$$

because m_4 assigns w to both [ʃikakíl] and [ʃikakíl̥] for /ʃikakíli/. Thus, the single–state FST representing c_k assigns exactly as many violations to the pair l→l as to the pair l→l̥. This should mean that

$$c_k(\langle \text{ʃikakíl}, \text{ʃikakíl}\rangle) = c_k(\langle \text{ʃikakíl}, \text{ʃikakíl̥}\rangle)$$

which should in turn mean that $\phi(c_k) = m_4$ assigns w to both [ʃikakíl] and [ʃikakíl̥] for /ʃikakíl/. But m_4 does not. Therefore, it cannot be that $\phi(c_k) = m_4$. Rather, m_4 must correspond to a provisional faithfulness constraint we might call ANTIIDENT–IO(L) with the following definition: "Assign one violation mark for each occurrence of an output [l] in correspondence with an input /l/, provided that neither /l/ nor [l] is followed by any segment." This is a provisional faithfulness constraint: the FST that represents it cannot simply assign a violation of 1 to each pair l→l but rather must first check that this is the last input–output symbol pair to process, and to do that requires an extra state.

Since m_3 and m_4 both correspond to provisional faithfulness constraints, it must be that $\phi(c_j) = \phi(c_k) = m_8$. Again, it must be that

$$c_j(\langle \text{ʃikakíli}, \text{ʃikakíl̥}\rangle) = c_j(\langle \text{ʃikakíli}, \text{ʃikakíl}\rangle) + a, \quad a > 0$$

from which it should follow that

$$c_j(\langle \text{ʃikakíl}, \text{ʃikakíl̥}\rangle) = c_j(\langle \text{ʃikakíl}, \text{ʃikakíl}\rangle) + a, \quad a > 0$$

This in turn should mean that $\phi(c_j) = m_8$ assigns w to [ʃikakíl] and l to [ʃikakíl̥] for /ʃikakíl/. But m_8 does not. Therefore, it cannot be that $\phi(c_j) = m_8$. By similar reasoning, it cannot be that $\phi(c_k) = m_8$. Rather, m_8 must correspond to a provisional faithfulness constraint we might call

$$[\text{IDENT–IO(L)} \vee \text{ANTIIDENT–IO(L)}]$$

where IDENT–IO(L) and ANTIIDENT–IO(L) are the provisional faithfulness constraints from above, and this disjunctive constraint assigns a violation

[23] As in the first proof, since our FSTs assign either 0 or 1 to each pair of input–output symbols, it follows that $a = 1$.

whenever either (or both) of its disjuncts assigns a violation. As a disjunction of provisional faithfulness constraints, this faithfulness constraint is likewise necessarily provisional.

We have now reached our contradiction. At least one of c_j, c_k must be a provisional faithfulness constraint. But C, which contains c_j, c_k, contains only markedness and non–provisional faithfulness constraints. Thus, our assumption must be wrong that there is a ranking $>$ for C such that for each $\langle i, o\rangle \in U$, o is the most optimal candidate with respect to i, C, and $>$.

□

The proof for the Isthmus Nahuat patterns is extremely similar to the proof for the Polish patterns. In the next section we generalize the proof, stripping it down to its essentials, and apply it to several other cases of opacity. We find that all the cases of environment opacity, whether counterfeeding or counterbleeding, are amenable to the proof technique, whereas the cases of focus opacity are not.

Summary. Up to this point, we have demonstrated informally that there are sets of attested patterns, one set from Polish and one set from Isthmus Nahuat, which can be expressed by ordered rewrite rules but which seem intuitively to be inexpressible by any classic OT grammar consisting of just markedness and non–provisional faithfulness constraints. Then, based on those two sets of patterns, we devised two slightly different sets of patterns which we proved are expressible by ordered rewrite rules but inexpressible by any classic OT grammar without at least one provisional faithfulness constraint. Thus, we have provided two formal results that classic OT, as we have defined it, as well as any more restricted version thereof, is expressively weaker than ordered rewrite rules. In the next section, we demonstrate (without giving full details) that the proof technique can be applied to several other sets of patterns, including Canadian raising, for which no hypothetical patterns are required. That is, the Canadian raising patterns are not only amenable to the proof and thus constitute yet another formal result that classic OT is expressively weaker than ordered rules, but they are also attested patterns, meaning that classic OT is provably inadequately expressive, from an empirical standpoint.

8 Generalizing the proof to other cases of opacity

There are several common elements that the two preceding proofs share, despite the fact that the Polish patterns are a case of counter*bleeding* on environment opacity and the Isthmus Nahuat patterns are a case of counter*feeding* on environment opacity.

First, in both cases, one of the two input–output patterns—call it $\langle i_1, o_1\rangle$—was the "opaque" pattern, while in the second input–output pattern—call it $\langle i_2, o_2\rangle$—the output o_2 was the output that i_1 *would* map to under the set of the most obvious classic OT constraints. For example, in the informal

demonstration for Polish (section 3), [ʒwop] was a more optimal output of /ʒwob/ than the attested form [ʒwup], regardless of constraint ranking, and /ʒwop/ → [ʒwop] was the hypothetical pattern used in the proof in section 5. In the informal demonstration for Isthmus Nahuat (section 6), [ʃikɑkíl̥] was a more optimal output of /ʃikɑkíli/ than the attested form [ʃikɑkíl], regardless of constraint ranking, and /ʃikɑkíl/ → [ʃikɑkíl̥] was the hypothetical pattern used in the proof in section 7.

Second, the crux of both proofs was to assume that there was a ranking $>$ for the set C of all markedness and non–provisional faithfulness constraints that would pair i_1 with o_1 and pair i_2 with o_2 and to then derive a contradiction. In a sense, then, we found a set of *contradictory*, or *inconsistent*, patterns in both cases.

These observations provide an easy way to devise new proofs for other cases of opacity.

1. Determine the opaque pattern $\langle i_1, o_1 \rangle$ and the two rules $\mathcal{P}_1, \mathcal{P}_2$ (and the order) that capture it.
2. Determine what the output of i_1 *would* be under the most obvious set of classic OT constraints. Call it o_2.[24]
3. Find an input (or construct a hypothetical one) which the rules (in their normal order) map to o_2. Call it i_2.
4. Consider the table of 16 different *wl*–constraints that includes the two inputs i_1, i_2 and the two output candidates o_1, o_2.
5. Show that the only *wl*–constraints that can work together to map each input to its unique output include at least one *wl*–constraint corresponding to a provisional faithfulness constraint.

Of course, we have noted several times that the table of all 16 possible *wl*–constraints can immediately be reduced to a table of 9. It turns out, in fact, that the table can be reduced to just 3. In both proofs, we only dealt with 5 total possibilities for $\phi(c_j), \phi(c_k)$, but it turned out in both proofs—and is true in general for these proofs—that 2 of the 5 *wl*–constraints correspond to markedness constraints but that c_j and c_k cannot both be one of these two constraints. That is, at least one of c_j, c_k must always correspond to one of the remaining 3 *wl*–constraints, which are illustrated in the generalized table below.

(34) **Reduced, generalized table of *wl*–constraints**

[24] If it turns out that $o_1 = o_2$, then it may be that that particular case of opacity is not problematic for OT.

$/i_1/$	m_1	m_2	m_3
$o_{1@}$	w	w	w
o_2	l	w	l
$/i_2/$			
o_1	w	l	l
$o_{2@}$	w	w	w

Intuitively, the reason for this reduction is that we need a constraint c_j that favors o_1 for i_1 and a constraint c_k that favors o_2 for i_2. There are several ways to do this.

1. Either c_j, c_k both favor o_1 for i_1 as well as o_2 for i_2, in which case $\phi(c_j) = \phi(c_k) = m_3$; or
2. only one of c_j, c_k favors o_1 for i_1 as well as o_2 for i_2, while the other favors just one of them, in which case one of c_j, c_k corresponds to m_3, while the other corresponds to m_1, or to m_2, or to one of the two wl–constraints not shown which correspond to markedness constraints[25]; or else
3. neither of c_j, c_k favor both o_1 for i_1 and o_2 for i_2, in which case $\phi(c_j) = m_1$ and $\phi(c_k) = m_2$.

The reader can check that in both the Polish proof and the Isthmus Nahuat proof, these are the three wl–constraints that we deduced must correspond to at least one of c_j, c_k. In addition, the provisional constraint corresponding to m_3 was a disjunction of the provisional constraints corresponding to m_1 and m_2.

Thus, in general, every such proof boils down to considering whether m_1, m_2, and m_3 correspond to provisional faithfulness constraints.[26] If so, then the two patterns are inconsistent and thus inexpressible in classic OT.

In the rest of this section, we look at a few more examples of opaque patterns. Our reduced, generalized table of wl–constraints will let us quickly see if we have an inconsistent set or not.

8.1 Other cases of counterbleeding on environment opacity

Consider the following patterns of Canadian raising.

(35) **Canadian raising data** Orthographic representations:

a. /raɪtər/ surfaces as [rʌɪɾər]. "writer"
b. /raɪdər/ surfaces as [raɪɾər]. "rider"

[25] In other words, if one of c_j, c_k corresponds to m_3, then the other could be a markedness constraint corresponding to one of the wl–constraints not shown in the table. In that case, the constraint corresponding to m_3 makes it so that the other constraint actually has no effect on the outcome of the evaluation: as we can see, m_3 alone already picks the right outputs.

[26] And since the constraints that map to m_3 are in general a disjunction of the constraints that map to m_1, m_2, it seems safe to say that if the constraints that map to m_1, m_2 are provisional faithfulness constraints, then so are the ones that map to m_3.

These two patterns can be captured by a rule $\mathcal{R}$ of raising ordered before a rule $\mathcal{F}$ of flapping: $\mathcal{R}$ vacuously maps /raɪdər/ to raɪdər, which $\mathcal{F}$ maps to [raɪɾər], and $\mathcal{R}$ maps /raɪtər/ to rʌɪtər, which $\mathcal{F}$ maps to [rʌɪɾər]. $\mathcal{F}$ counterbleeds on $\mathcal{R}$'s environment because if $\mathcal{F}$ were ordered *before* $\mathcal{R}$, then $\mathcal{F}$ would map /raɪtər/ to raɪɾər, which $\mathcal{R}$, its environment (/t/) having been bled by $\mathcal{F}$ (/t/ $\rightarrow$ [ɾ]), would map vacuously to [raɪɾər].

Though we won't demonstrate it, the obvious set of OT constraints is insufficient for analyzing these patterns. We get another proper superset dilemma, just like with the Polish pattern.

Fortunately for our proof, the output that /raɪtər/ *would* get mapped to under the set of obvious constraints, namely [raɪɾər], is in fact the attested output of /raɪdər/, so we need not come up with a hypothetical input. We can now throw these two inputs and output candidates into our reduced table of three *wl*–constraints.

(36) **Reduced table of *wl*–constraints for Canadian raising**

/raɪtər/	m_1	m_2	m_3
raɪɾər	l	w	l
rʌɪɾər@	w	w	w
/raɪdər/			
raɪɾər@	w	w	w
rʌɪɾər	w	l	l

Some reflection reveals that these two patterns are indeed inconsistent. First, none of m_1, m_2, or m_3 can correspond to a markedness constraint, for the letter assigments are different across both inputs, despite the output candidates being the same for both inputs. Second, none can correspond to a non–provisional faithfulness constraint either. The only difference between [raɪɾər] and [rʌɪɾər] is the second segment, which in both cases is in correspondence with /a/. Thus, if there were a *–constraint c corresponding to m_1, m_2, or m_3 that was representable by a single–state faithfulness constraint **c**, then **c** would assign, say, a violations to the pair a$\rightarrow$a and b violations to the pair a$\rightarrow$ʌ. If $a = b$, then $\phi(c)$ assigns w across the board. If $a > b$, then $\phi(c)$ assigns w to [rʌɪɾər] and l to [raɪɾər] for both /raɪtər/ and /raɪtər/. And if $b > a$, then $\phi(c)$ assigns w to [raɪɾər] and l to [rʌɪɾər] for both /raɪtər/ and /raɪtər/. Since these exhaust the possibilities of non–provisional faithfulness constraints, and since none of the letter assignments corresponds to m_1, m_2, or m_3, the latter *wl*–constraints cannot correspond to any non–provisional faithfulness constraints.

Thus, Canadian raising is yet another example of patterns which can be expressed by ordered rules but which in OT require a provisional faithfulness constraint. And in this case, both patterns are attested, so the empirical result that there are phonological patterns that are inexpressible by any classic OT grammar is established.

It's worth noting, by the way, that the interaction between raising and flapping in Canadian raising is just a slightly more complex version of the interaction between raising and devoicing in Polish. The difference is that in

Polish /b/ and /p/ both neutralize to [p], whereas in Canadian raising /d/ and /t/ both neutralize to a third segment, [ɾ]. If in Polish /b/ and /p/ neutralized to, say, [f], or if, say, /b/ and /f/ both neutralized to [p], then the patterns in both languages would be formally identical. The same would be true if in Canadian raising /d/ and /t/ both neutralized to [d].

Consider now one final example of counterbleeding on environment opacity, this time from Bedouin Arabic (McCarthy, 2007, p. 11).

(37) **Bedouin Arabic data 1** Glosses:

a. /ħaːkimiːn/ surfaces as [ħaːkʲmiːn] "ruling (masculine plural)"
b. /ħaːkmiːn/ surfaces as [ħaːkmiːn] (hypothetical)

These patterns can be captured by a rule $\mathcal{P}$ of palatalization ordered before a rule $\mathcal{D}$ of deletion: $\mathcal{P}$ maps /ħaːkimiːn/ to ħaːkʲimiːn, which $\mathcal{D}$ maps to [ħaːkʲmiːn], and $\mathcal{P}$ vacuoulsy maps /ħaːkmiːn/ to ħaːkmiːn, which $\mathcal{D}$ maps vacuously to [ħaːkmiːn]. $\mathcal{D}$ counterbleeds on $\mathcal{P}$'s environment because if $\mathcal{D}$ were ordered *before* $\mathcal{P}$, then $\mathcal{D}$ would map /ħaːkimiːn/ to ħaːkmiːn, which $\mathcal{P}$, its environment (/i/) having been bled by $\mathcal{D}$ (/i/ $\rightarrow \emptyset$), would vacuously map to [ħaːkmiːn].

Below is the reduced table of three *wl*–constraints for these two patterns.

(38) **Reduced table of *wl*–constraints for Bedouin Arabic 1**

/ħaːkimiːn/	m_1	m_2	m_3
ħaːkmiːn	*l*	*w*	*l*
ħaːkʲmiːn$_{@}$	*w*	*w*	*w*
/ħaːkmiːn/			
ħaːkmiːn$_{@}$	*w*	*w*	*w*
ħaːkʲmiːn	*w*	*l*	*l*

Once again, we clearly have an inconsistent set of patterns. (We leave the details aside.) Moreover, as with Polish, these patterns can be considered a simpler version of the Canadian raising patterns: /i/ and the empty string (denoted above by $\emptyset$) both, in some sense, get "neutralized" to the empty string, whereas in Canadian raising /t/ and /d/ get neutralized to a third segment, [ɾ].[27] If instead we had another vowel, say /u/, which like /i/ underwent deletion but which unlike /i/ did not trigger palatalization, then /i/ and /u/ would both get "neutralized" to the empty string, and the pattern would be identical to that of Canadian raising.

8.2 Other cases of counterfeeding on environment opacity

We now turn from counterbleeding to counterfeeding on environment opacity. Consider the following other data from Bedouin Arabic (McCarthy, 2007, p. 26).

[27]Implicit here is the assumption that the FST that would represent the constraints needed here is defined over a symbol set containing a symbol for the empty string, say $\emptyset$, so that deleting /i/ formally amounts to the input–output pair i$\rightarrow \emptyset$, hence the strange usage of "neutralize" for deletion above.

(39) **Bedouin Arabic data 2**

		Glosses:
a.	/gabr/ surfaces as [gabur]	"grave"
b.	/gabur/ surfaces as [gibur]	(hypothetical)

These two patterns can be captured by a rule $\mathcal{R}$ of raising /a/ to [i] in an open syllable ordered before a rule $\mathcal{E}$ of epenthesis: $\mathcal{R}$ vacuously maps /gabr/ to gabr, which $\mathcal{E}$ maps to [gabur], and $\mathcal{R}$ maps /gabur/ to gibur, which $\mathcal{E}$ vacuously maps to [gibur]. $\mathcal{E}$ feeds on $\mathcal{R}$'s environment because if $\mathcal{E}$ were ordered *before* $\mathcal{R}$, then $\mathcal{E}$ would map /gabr/ to gabur, which $\mathcal{R}$, its environment (open syllable) having been fed by $\mathcal{E}$ ($\emptyset \rightarrow$ [u]), would map to [gibur].

Below is the reduced table of three *wl*–constraints for these two patterns.

(40) **Reduced table of *wl*–constraints for Bedouin Arabic 2**

/gabr/	m_1	m_2	m_3
gabur$_{@}$	*w*	*w*	*w*
gibur	*l*	*w*	*l*
/gabur/			
gabur	*w*	*l*	*l*
gibur$_{@}$	*w*	*w*	*w*

Clearly, these patterns are inconsistent. Thus, Bedouin Arabic provides yet another case of counterfeeding on environment opacity which in OT requires a provisional faithfulness constraint.

Consider now one final case of counterfeeding on environment opacity, this time from Lomongo (Hulstaert, 1961; Kenstowicz and Kisseberth, 1979; Baković, 2011).

(41) **Lomongo data**

		Glosses:
a.	/obina/ surfaces as [oina].	"you (sg.) dance"
b.	/oina/ surfaces as [wina]	(hypothetical)

These two patterns can be captured by a rule $\mathcal{G}$ of changing pre–vocalic vowels to glides ordered before a rule $\mathcal{D}$ of intervocalic voiced obstruent deletion. $\mathcal{G}$ vacuously maps /obina/ to obina, which $\mathcal{D}$ maps to [oina], and $\mathcal{G}$ maps /oina/ to wina, which $\mathcal{D}$ vacuously maps to [wina]. $\mathcal{D}$ counterfeeds on $\mathcal{G}$'s environment because if $\mathcal{G}$ were ordered *before* $\mathcal{D}$, then $\mathcal{D}$ would map /obina/ to oina, which $\mathcal{G}$, its environment (a vowel) having been fed by $\mathcal{D}$ (/b/ $\rightarrow \emptyset$, pushing i next to o), would map to [wina].

Below is the reduced table of three *wl*–constraints for these two patterns.

(42) **Reduced table of *wl*–constraints for Lomongo**

/obina/	m_1	m_2	m_3
oina$_{@}$	*w*	*w*	*w*
wina	*l*	*w*	*l*
/oina/			
oina	*w*	*l*	*l*
wina$_{@}$	*w*	*w*	*w*

Once again, we have an inconsistent set.

8.3 Counterbleeding and counterfeeding on focus opacity

In this last subsection we briefly explore two cases of *focus* (not environment) opacity. In both cases, unlike with Polish, Isthmus Nahuat, and all the other cases just reviewed, the obvious OT analysis can be salvaged with a non–provisional faithfulness constraint. Thus, the proof technique is inapplicable because these cases of opacity do not give rise to an inconsistent set of patterns (attested or hypothetical). We conjecture that focus opacity is unlike environment opacity, in that it can in fact be handled in OT with only markedness and non–provisional faithfulness constraints. This is somewhat surprising given that Baković (2011) has argued that counterbleeding on focus opacity (but not counterfeeding on focus opacity) is just as problematic for OT as counterbleeding and counterfeeding on environment opacity.

Consider the following data from Western Basque (de Rijk, 1970; Kirchner, 1996; Kager, 1999; Baković, 2011).

(43) **Western Basque data** Glosses:

a. /semee/ surfaces as [semie]. "son"
b. /alabaa/ surfaces as [alabea]. "daughter"

Both patterns can be captured by a rule $\mathcal{R}_{m\to h}$ of mid–to–high raising ordered before a rule $\mathcal{R}_{l\to m}$ of low–to–mid raising.

(44) **Western Basque rules**

a. $\mathcal{R}_{m\to h}$: [−low] → [+high] / _V
b. $\mathcal{R}_{l\to m}$: [+low] → [−low] / _V

$\mathcal{R}_{m\to h}$ maps /semee/ to semie, which $\mathcal{R}_{l\to m}$ maps vacuously to [semie], and $\mathcal{R}_{m\to h}$ vacuously maps /alabaa/ to alabaa, which $\mathcal{R}_{l\to m}$ maps to [alabea].

The two rules are in a counterfeeding on focus relationship because if $\mathcal{R}_{l\to m}$ were ordered *before* $\mathcal{R}_{m\to h}$, then $\mathcal{R}_{l\to m}$ would map /alabaa/ to alabea, which $\mathcal{R}_{m\to h}$, its focus (/e/) having been fed by $\mathcal{R}_{l\to m}$ (/a/ → [e]), would map to [alabia].

Let's try to come up with an OT analysis. For low–to–mid raising, we could posit a markedness constraint that penalizes outputs containing a low vowel followed immediately by another vowel, which is ranked higher than a constraint that penalizes changing a low vowel to a mid vowel. For mid–to–high raising, we could posit a markedness constraint that penalizes outputs containing a mid vowel followed immediately by another vowel, which is ranked higher than a faithfulness constraint that penalizes changing mid vowels to high vowels.

(45) **Western Basque constraints**

a. Low–to–mid raising
*[+low]V > IDENT–IO(low)

b. Mid–to–high raising
*[−low, −high]V > IDENT–IO(high)

Let's relabel *[+low]V as *aV, and let's relabel *[−low, −high]V as *eV.

Note that we don't know how either constraint in (45a) is ranked with respect to either constraint in (45b). However, let's arbitrarily choose the following total ranking.

(46) **Western Basque hypothetical constraint ranking**
*aV > IDENT–IO(low) > *eV > IDENT–IO(high)

We will not demonstrate here that these constraints and this ranking correctly make [semie] a more optimal output of /semee/ than [semee]. Instead, we'll skip right to the tableau for the input /alabaa/.

(47) **Attempted OT analysis of** /alabaa/ → [alabea]

	/alabaa/	*aV	ID(low)	*eV	ID(high)
	a. alabaa	1			
	b. alabea@		1	1	
→	c. alabia		1		1

This set of constraints and this ranking favor the unattested output [alabia] over the attested output [alabea]. Moreover, the only constraints on which [alabea] and [alabia] differ on are *eV and IDENT–IO(high). However, we know that *eV outranks IDENT–IO(high), so given just these four constraints, and no matter how the other two constraints are ranked with respect to *eV and IDENT–IO(high) or to each other, [alabia] more optimal than the attested output, [alabea].

However, unlike with Polish and Isthmus Nahuat, there *is* a non–provisional faithfulness constraint we can add to salvage this analysis: IDENT–IO(A), which penalizes each occurrence of an output [i] in correspondence with an input /a/. In other words, it penalizes mapping /a/ to [i], but not /a/ to [e]; thus, it is representable by a single–state faithfulness constraint that assigns 1 to the pair a→i and 0 to every other pair, including a→e. Simply rank IDENT–IO(A) above the other four constraints, and [alabea] becomes the clear winner. (Moreover, it seems reasonable to assume that /a/ in fact never surfaces as [i], so we could just make IDENT–IO(A) undominated.)

As our final example of opacity, consider now the following data from certain dialects of Low German (Kiparsky, 1968; Kenstowicz and Kisseberth, 1979; Baković, 2011).

(48) **Low German data** Glosses:
a. /taːgə/ surfaces as [taːɣə]. "day"
b. /haʊz/ surfaces as [haʊs]. "house"
c. /taːg/ surfaces as [taːx]. "day"

All three patterns can be captured by a rule $\mathcal{S}$ of voiced obstruent spirantization ordered before a rule $\mathcal{D}$ of word–final obstruent devoicing.

(49) **Low German rules**

a. $\mathcal{S}$: [−sonorant, +voice] → [+continuant] / V_

b. $\mathcal{D}$: [−sonorant] → [−voice] / _#

$\mathcal{S}$ maps /ta:gə/ to ta:ɣə, which $\mathcal{D}$ maps vacuously to [ta:ɣə]. $\mathcal{S}$ vacuously maps /haʊz/ to haʊz, which $\mathcal{D}$ maps to [haʊs]. Finally, $\mathcal{S}$ maps /ta:g/ to ta:ɣ, which $\mathcal{D}$ maps to [ta:x].

$\mathcal{D}$ counterbleeds on $\mathcal{S}$'s focus because if $\mathcal{D}$ were ordered *before* $\mathcal{S}$, then $\mathcal{D}$ would map /ta:g/ to ta:k, which $\mathcal{S}$, its focus (a voiced obstruent) having been bled by $\mathcal{D}$ (/g/ → [k]), would vacuously map to [ta:k].

Let's try to analyze these patterns in OT. For spirantization, we could posit a markedness constraint that penalizes outputs containing a vowel followed immediately by a voiced stop, which is ranked above a faithfulness constraint that penalizes changing stops to fricatives. For devoicing, we could posit a constraint that penalizes outputs containing word–final voiced obstruents, which is ranked above a faithfulness constraint that penalizes changing voiced obstruents to voiceless ones.

(50) **Low German constraints**

a. Spirantization
*V[−sonorant, +voice, −continuant] > IDENT–IO(continuant)

b. Devoicing
*[−sonorant, +voice]# > IDENT–IO(voice)

Let's relabel the markedness constraint for spirantization as $^{*}\text{VC}_{\text{vce, –cont}}$, and let's relabel the markedness constraint for devoicing as $^{*}\text{C}_{\text{vce}}\#$. We'll arbitrarily choose the following ranking.

(51) **Low German hypothetical ranking**
$^{*}\text{VC}_{\text{vce, –cont}}$ > IDENT–IO(continuant) > $^{*}\text{C}_{\text{vce}}\#$ > IDENT–IO(voice)

As before, we'll skip right to the tableau for the input /ta:g/.

(52) **Attempted OT analysis of** /ta:g/ → [ta:x]

	/ta:g/	$^{*}\text{VC}_{\text{vce, –cont}}$	ID(cont)	$^{*}\text{C}_{\text{vce}}\#$	ID(vce)
	a. ta:g	1		1	
	b. ta:ɣ		1	1	
→	c. ta:k				1
	d. ta:x@		1		1

These four constraints and this ranking favor the unattested output [ta:k] over the attested output, [ta:x]. Moreover, the only constraint on which [ta:k] and [ta:x] differ is IDENT–IO(cont), which favors the unattested output [ta:k]. In other words, as we saw in Polish, the violations assigned to the attested output, [ta:x], by these four constraints are a proper superset of those assigned to the unattested output [ta:k]; that is, [ta:x] is harmonically bounded by [ta:k]. It follows that no ranking of these constraints can possibly make [ta:x] more

optimal than [taːk].

Unlike with Polish (and Isthmus Nahuat), however, we can salvage this analysis by adding a constraint we might call IDENT–IO(G) which penalizes mapping /g/ to [k] but which does not penalize mapping /g/ to [ɣ] or mapping /g/ to [x]. As exotic and ad hoc as the constraint may seem, it is a non–provisional faithfulness constraint: it can be represented by a single–state FST that only assigns a violation of 1 for the input–output pair g→k, and assigns 0 to every other input–output pair, including g→x.[28]

One caveat is in order: whereas in Western Basque it seemed reasonable to assume that /a/ never surfaces as [i], it's possible that in Low German /g/ is not spirantized after, say, [ŋ], so that, for example, /taŋg/ surfaces as [taŋk]. In this case, we *cannot* simply make IDENT–IO(G) undominated. However, it might suffice to rank a markedness constraint like *ŋx higher than IDENT–IO(G), assuming that there are no attested output forms containing the sequence [ŋx]. Other constraints and rankings may also be necessary. Crucially, however, the difference between Low German and Polish is that with Low German we are in principle able to find at least one non–provisional faithfulness constraint to salvage the analysis, however ad hoc it may be, whereas in Polish no such constraint is available, not even in principle.

More generally, the point is that that even if counterbleeding on focus ultimately requires some kind of provisional faithfulness constraint (or worse), it's not immediately clear that this is so. And the same could be said of counterfeeding on focus opacity.

Summary. In this section we generalized the proof technique employed in sections 5 and 7, stripping it down to the essentials, namely the table of three *wl*–constraints that allow us to determine whether a given set of patterns is inconsistent or not. We then demonstrated in this way that several other patterns of environment opacity are provably inexpressible with any classic OT grammar, including Canadian raising, for which no hypothetical pattern was posited. Thus, not only do there seem to be general classes of patterns (environment opacity) that are provably inexpressible by any classic OT grammar, but at least one set of such patterns is fully attested, making classic OT expressively inadequate in an empirical sense. We also demonstrated that two cases of focus opacity seem to be expressible by a classic OT grammar, albeit with ad–hoc–looking constraints. We conjecture, therefore, that environment opacity is in general not expressible by classic OT grammars, whereas focus opacity is. This would make sense given that classic OT is defined here as

[28]The reason that IDENT–IO(G) may seem exotic or ad hoc is twofold. First, there is likely no independent motivation in Low German or elsewhere for such a constraint. Second, there is no obvious phonetic (or featural) motivation for such a constraint: the reason that g→k is penalized cannot be that this language disprefers devoicing velar stops, because then we would expect this constraint to also penalize g→x, which crucially it does not. Rather, we have to conclude that this language disprefers devoicing velar stops *unless* the stop is also spirantized, which is unusual.

That being said, while IDENT–IO(G) is an unusual constraint, it is within the formal limits of what we have defined as a classic OT constraint, unlike, say, the provisional faithfulness constraint IDENT–IO(O) that was necessary for Polish.

excluding only provisional faithfulness constraints, which reference segments other than the two correspondents under consideration, whereas focus opacity revolves around a series of structural changes to just one segment.

9 Conclusion

In this paper we wished to answer the following two questions: (i) Are there patterns which are expressible by ordered rewrite rules but which are not expressible by classic OT, suitably and precisely defined, and (ii) if so, are those patterns attested in natural language?

We discussed the widespread intuition among phonologists that classic OT consists of just two levels of representation (input and output) and just two types of constraints: markedness, which penalize output candidates containing certain sequences of segments, without reference to the input, and faithfulness, which penalize certain pairs of single input–output segments, without reference to any other segments in the input or output. We also discussed other faithfulness constraints like antifaithfulness constraints and constraints that penalize arbitrary input–output segments and the intuition among some phonologists that these are non–classic, or at least ad hoc if used in an analysis.

We then demonstrated informally, by way of two simple analyses of Polish and of Isthmus Nahuat, why it is that opacity is intuitively problematic for classic OT. Specifically, the most obvious constraints lead to an unattested form being optimal, and no markedness or faithfulness constraint seems to be of help.

Then we formalized OT constraints as finite–state transducers. We defined a classic, or non–provisional, faithfulness constraint very broadly to include any constraint that is representable by an FST that is both input–dependent and single–state. The foresight behind this generous definition was that we would prove that even by allowing antifaithfulness and arbitrary single–segment faithfulness, classic OT grammars are still unable to express certain patterns that are expressible by ordered rules.

Using hypothetical patterns based on data from Polish and Isthmus Nahuat, we then proved that there are patterns which can be expressed by rules ordered in a counterbleeding on environment relationship but which cannot be expressed by any classic OT grammar, and that there are patterns which can be expressed by rules ordered in a counterfeeding on environment relationship but which cannot be expressed by any classic OT grammar. In other words, we formally answered the first question: classic OT, as defined, as well as any more restricted version thereof, is expressively weaker than ordered rewrite rules.

Finally, we generalized the proof and applied it (though not completely, due to space limitations) to several other cases of environment opacity. In particular, we showed that two attested patterns of Canadian raising are expressible by ordered rules but not by any classic OT grammar. In other words, we answered the second question: there are attested phonological

patterns which are expressible by ordered rules but which are not expressible by any classic OT grammar. We also saw that two cases of focus opacity seem to be expressible by classic OT grammars, albeit with constraints that look ad hoc. We conjecture, therefore, that only environment opacity, not focus opacity, is in the general case problematic for OT. This would make sense, since focus opacity concerns structural changes to a single segment, meaning that constraints that reference other segments, i.e., provisional faithfulness constraints, are unnecessary for analysis, thus placing focus opacity within the expressive power of our broad version of classic OT.

We end this paper with the following question: What if our definition of classic OT were extended to include provisional faithfulness constraints that are representable by input–dependent FSTs with exactly two states, and to exclude those provisional constraints that must be represented by input–dependent FSTs with more than two states? All the provisional faithfulness constraints we have discussed—IDENT–IO(O) and ANTIIDENT–IO(O) in Polish, IDENT–IO(L) and ANTIIDENT–IO(L) in Isthmus Nahuat, as well as those for the other languages we did not discuss—require no more than two states. This is because in all those cases, the generalizations behind the opacity are localized to just two neighboring loci, e.g., /o/ ([o, u]) and /b/ ([b, p]) in Polish and /l/ ([l, l̰]) and /i/ ([i], ∅) in Isthmus Nahuat. Thus, this slightly extended version of OT would be able to express the entire range of cases of opacity discussed in this paper, and perhaps in all of phonology. What's more, this version of classic OT would, I conjecture, still be expressively weaker than ordered rewrite rules: we can imagine cases of opacity that are expressible by rules but which in OT require provisional faithfulness constraints that can only be represented by FSTs with, say, three states. For example, we could imagine the following three rules.

(53) **Super opacity rules**

a. o → u / d_b
b. d → b / _o, u
c. b → p / _#

These rules would map an input like /bob/ to [bop], they would map an input like /dop/ to [bop], and they would map an input like /dob/ to [bup]. The latter pattern is "super" opaque in the sense that all three rules have non–vacuously applied, and not one but two rules have been obscured by counterbleeding. Now consider the reduced table of *wl*–constraints for these three patterns. (Since we're now considering three patterns instead of two, there are seven *wl*–constraints to consider instead of three.)

(54) **Reduced table of *wl*–constraints for super opacity**

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
/dob/							
bop	*l*	*w*	*w*	*l*	*l*	*w*	*l*
bup@	*w*	*w*	*w*	*w*	*w*	*w*	*w*
/dop/							
bop@	*w*	*w*	*w*	*w*	*w*	*w*	*w*
bup	*w*	*l*	*w*	*l*	*w*	*l*	*l*
/bob/							
bop@	*w*	*w*	*w*	*w*	*w*	*w*	*w*
bup	*w*	*w*	*l*	*w*	*l*	*l*	*l*

With some reflection, it should be clear that each of these *wl*–constraints corresponds to a provisional faithfulness constraint that can only be represented by finite–state faithfulness constraint with more than two states.

For example, m_1 cannot correspond to a faithfulness constraint that is represented by a single–state FST that penalizes only d→b, or only o→o, or only b→p; nor even by a two–state FST that penalizes only do→bo, or only ob→op. Rather, it must correspond to a faithfulness constraint that can only be represented by a three–state FST that penalizes dob→bop. We leave the rest of the proof for future research.

Presumably, such patterns do not occur in phonology. However, as we can see, they are expressible by ordered rules, but not by this extended version of classic OT. This result therefore suggests that, at least with regard to empirical expressivity of opacity, OT is to be preferred over rules.

References

Baković, Eric. 2007. A revised typology of opaque generalizations. *Phonology* 24:217–259.

Baković, Eric. 2011. Opacity and ordering. In *The handbook of phonological theory*, ed. John A. Goldsmith, Jason Riggle, and Alan C. L. Yu. Blackwell Publishers, second edition.

Benua, Laura. 1997. Transderivational identity: Phonological relations between words. Doctoral Dissertation, University of Massachusetts, Amherst. URL roa.rutgers.edu/files/259-0498/roa-259-benua-2.pdf.

Bethin, Christina Y. 1978. Phonological rules in the nominative singular and genitive plural of the Slavic substantive declension. Doctoral Dissertation, University of Illinois, Champaign–Urbana.

Chomsky, Noam, and Morris Halle. 1968. *The sound pattern of English*. The MIT Press.

Heinz, Jeffrey. 2011a. Computational phonology—part I: Foundations. *Language and Linguistics Compass* 5:140–152.

Heinz, Jeffrey. 2011b. Computational phonology—part II: Grammars, learning, and the future. *Language and Linguistics Compass* 5:153–168.

Hulstaert, Gustaaf. 1961. *Grammaire du lomongo*. Musée royal de l'Afrique centrale.

Idsardi, William J. 2000. Clarifying opacity. *The Linguistic Review* 17:337–350.

Kager, René. 1999. *Optimality Theory*. Cambridge University Press.

Kenstowicz, Michael, and Charles Kisseberth. 1979. *Generative phonology: Description and theory*. Academic Press.

Kiparsky, Paul. 1968. Linguistic universals and language change. In *Universals in linguistic theory*, ed. Emmon W. Bach and Robert T. Harms, 170–202. Holt, Reinhart, & Winston.

Kiparsky, Paul. 1971. Historical linguistics. In *A survey of linguistic science*, ed. William Orr Dingwall. University of Maryland Linguistics Program, College Park.

Kiparsky, Paul. 1973. Abstractness, opacity, and global rules. In *Three dimensions of linguistic theory*, ed. Osamu Fujimura and Donald L. Smith. TEC.

Kirchner, Robert. 1996. Synchronic chain shifts in Optimality Theory. *Linguistic Inquiry* 27:341–350.

Law, Howard. 1958. Morphological structure of Isthmus Nahuat. *International Journal of American Languages* 24:108–129.

McCarthy, John. 1999. Sympathy and phonological opacity. *Phonology* 16:331–399.

McCarthy, John. 2002. Comparative markedness (long version). In *Papers in Optimality Theory II*, ed. Angela Carpenter, Andries Coetzee, and Paul de Lacy, volume 26 of *University of Massachusetts Occasional Papers in Linguistics*. GLSA Publications. URL http://works.bepress.com/john_j_mccarthy/99.

McCarthy, John. 2007. *Hidden generalizations: Phonological opacity in Optimality Theory*. Equinox Publishing Ltd.

Prince, Alan, and Paul Smolensky. 2004. *Optimality Theory: Constraint interaction in Generative Grammar*. Blackwell Publishers.

Riggle, Jason. 2004. Generation recognition and learning in finite state Optimality Theory. Doctoral Dissertation, UCLA.

de Rijk, Rudolph. 1970. Vowel interaction in Bizcayan Basque. *Fontes Linguae Vasconum* 2:149–167.

Sanders, Nathan. 2003. Opacity and sound change in the Polish lexicon. Doctoral Dissertation, UCSC.

Smolensky, Paul. 1993. Harmony, markedness, and phonological activity. URL http://roa.rutgers.edu/files/87-0000/87-0000-SMOLENSKY-0-0.PDF, handout of keynote address, Rutgers Optimality Workshop 1, New Brunswick, October 1993.