REVIEW OF TESAR'S (2013) "OUTPUT-DRIVEN PHONOLOGY: THEORY AND LEARNING"

GIORGIO MAGRI

[To appear in *Phonology*]

Contents

1. Introduction

In his recent bid for the presidency of the Association for Mathematics of Language, Makoto Kanazawa writes: "The scientific study of language is such a large and important field that it's strange that so little mathematical research is being carried out. By comparison, the latest issue of the *Journal of Economic Theory* has 14 original articles, and all but one of them are mathematical papers in the sense of containing theorems and proofs. And the title of the journal is not *Journal of Mathematical Economics*, which is actually a separate journal."[1] That linguistics lags behind economics is not surprising: making more money is a stronger drive for change and progress than better understanding the language faculty. But Tesar's book shows that the time might be ripe for change and progress in our field as well. Through a formally sophisticated *analytical* approach (as opposed to a purely *simulation-based*

Date: November 4, 2014.
I would like to thank Bruce Tesar and Andrew Nevins for detailed comments.

[1] See http://electron.cs.earlham.edu/pipermail/mol/2014-January/000479.html

approach), the book provides a beautiful example of the interplay between learnability and structural assumptions on the typological space. It thus shows that computational phonology has become a mature subfield of generative linguistics.

Tesar's book formalizes the notion of *output-drivenness* and explores its phonological meaning, its OT characterization, and its learnability implications. Informally, a phonological grammar is *output-driven* provided "disparities between input and output are only introduced to the extent necessary to satisfy output restrictions" (p. 1). Chapter 2 formalizes this intuition into a notion of output-drivenness defined independently of the phonological formalism assumed (SPE, OT, HG, ...). The phonological underpinning of this notion is illustrated by showing that non-output-drivenness unifies a number of phonological phenomena such as chain shifts, derived environment effects, and opacity. Chapter 3 zooms in on OT and provides a sharp characterization of output-drivenness for OT grammars. It shows that output-drivenness in OT depends on the nature of the faithfulness constraints and provides a characterization of the faithfulness constraints which guarantee output-drivenness. Chapter 4 then surveys and systematizes a number of OT approaches to non-output-driven phonological phenomena through the formal characterization obtained in chapter 3. Chapters 7 and 8 explore the implications of output-drivenness for OT learnability, focusing on the problem of learning simultaneously a grammar and a lexicon of underlying forms from paradigmatic information. Chapter 1 provides an informal overview, chapters 5 and 6 review selected prior work on OT learnability, and chapter 9 wraps up.

Tesar's book makes for a difficult reading. This review aims at offering a reader's guide, focused on the learnability implications of output-drivenness (which are plausibly less accessible to the readership of the journal than its phonological implications). I thus start in section 2 right away with a review of the learning problem tackled by Tesar and I pause to discuss two notions of *efficient* learning, a loser and a more demanding one. In section 3, I review Tesar's notion of output-drivenness, tailored to the rather restrictive representational framework considered in the statement of the learning problem. In the rest of the review, I tackle the following question: what can output-drivenness do for the learner? I address this question from two perspectives. In section 4, I focus on time complexity: does output-drivenness help the learner to tackle the learning problem faster? In section 5, I focus on correctness: does output-drivenness help the learner to better succeed in solving the learning problem? In order to address these questions, I compare Tesar's *Output Driven Learner* (ODL) with Merchant's (2008) state-of-the-art *Contrast Pair and Ranking Information* (CPR) learner, which is only briefly described in Tesar's book. For completeness, CPR is reviewed in appendix A, available in the Online Supplementary Materials.

CPR and ODL share the same architecture, namely both maintain partial lexical and grammatical (ranking) information, which are incremented iteratively by boosting one type of partial information against the other. Furthermore, the subroutines used by ODL are a speeded-up version of those used by CPR, as explained in appendix B, available in the Online Supplementary Materials. Section 4 thus quantifies the speed-up afforded by output-drivenness by comparing the efficiency of CPR's and ODL's subroutines, using both the generous and the demanding measure introduced in section 2. CPR and ODL use two types of subroutines: for the extraction of ranking information and for the extraction of lexical information. Furthermore, these subroutines can be applied to two types of data: either to a single entry in the paradigm or to a pair of entries which share a morpheme (so called *contrast pair*) and are thus able to display morphophonemic alternations. Output-drivenness affords a huge speed-up of the subroutine which extracts ranking information from a single paradigmatic entry: while Merchant's original subroutine is optimal but (plausibly) inefficient even relative to the looser notion of efficiency, Tesar's restatement under the assumption of output-drivenness is efficient even relative to the more demanding notion. Furthermore, output-drivenness affords a substantial speed-up of Merchant's subroutine for the extraction of lexical information from a single paradigmatic entry: while the original subroutine is only efficient relative to the looser notion of efficiency, Tesar's restatement under the assumption of output-drivenness is efficient even relative to the more demanding notion. Unfortunately, output-drivenness is not helpful when these subroutines are applied to contrast pairs: Merchant's subroutine for the extraction of ranking information from contrast pairs remains inefficient and the subroutine for the extraction of lexical information from contrast pairs remains efficient only relative to the loose notion of efficiency.

Section 5 turns from time complexity to correctness. It is easy to formally characterize a class of cases where Merchant's CPR is provably bound to fail. The trouble stems from the fact that CPR is excessively prudent: whenever a paradigm is consistent with a feature being set underlyingly to two different values, CPR refuses to make a choice. If that happens for two features whose values are correlated, CPR inevitably falls short of success. This is, as far as I can tell, the core weakness of CPR's approach. And output-drivenness does not help. Indeed, since Tesar's ODL uses subroutines which are (faster but) equivalent to CPR's subroutines, it inherits CPR's excessive prudence. The crucial question of whether any such problematic case is actually attested in natural language phonology is left open at this stage. Yet, this interplay between the typology of what is attested and the formal restrictions on what is learnable speaks of the relevance of this line of research for phonological theory.

2. The problem of learning languages from paradigms

This section introduces the learning problem tackled in Tesar's book, namely the problem of learning a language from a paradigm within an assigned typology of languages. Subsection 2.1 defines the typological space; subsections 2.2 and 2.3 introduce the notions of *language* and *paradigm*; and subsection 2.4 puts the pieces together into an explicit statement of the learning problem. Subsection 2.5 simplifies the learning problem through some additional rather restrictive representational assumptions. And subsection 2.6 discusses two notions (a more generous and a more demanding one) of efficient learning.

2.1. **Typology**

Let $\mathcal{R}$ and $\mathcal{S}$ be finite sets of *underlying roots* and *underlying suffixes*, respectively. They are assumed to be disjoint, so that it is always possible to tell whether a given underlying morpheme is a root or a suffix. A generic underlying root (suffix) is denoted by a capital (small) boldfaced letter from the beginning of the alphabet: $\mathbf{A}, \mathbf{B}, \dots$ (and $\mathbf{a}, \mathbf{b}, \dots$). To illustrate, assume that the underlying roots consist of the consonant /p/ plus a vowel which is either short or long and either stressed or unstressed, as (1a); and that the underlying suffixes have the different consonant /k/ to keep them distinct, as in (1b).

(1) a. $\mathcal{R}$ = {/pa/ /paː/ /pá/ /páː/}
 b. $\mathcal{S}$ = {/ka/ /kaː/ /ká/ /káː/}

Let $\mathbb{R}$ and $\mathbb{S}$ be the sets of *meanings* for underlying roots and suffixes, respectively. For convenience, they are assumed to be disjoint and to contain exactly one root (suffix) meaning for each underlying root (suffix). A generic root meaning is denoted by M and a generic suffix meaning by m. An example is provided in (2).

(2) a. $\mathbb{R}$ = {BOY, GIRL, DOG, CAT}
 b. $\mathbb{S}$ = {nom, acc, gen, dat}

The set of *underlying forms* consists of all concatenations of a root and a suffix. A generic underlying form is thus denoted by $\mathbf{Aa}, \mathbf{Bb}, \dots$. Since the sets $\mathcal{R}$ and $\mathcal{S}$ of underlying roots and suffixes are disjoint, the root and the suffix in any given underlying concatenation can always be pulled apart. In our example, the set of underlying forms thus consists of the sixteen concatenations of the four underlying roots with the four underlying suffixes, listed in (3).

(3)
$$\left\{\begin{array}{llll} /\text{paka}/ & /\text{pakaː}/ & /\text{paká}/ & /\text{pakáː}/ \\ /\text{paːka}/ & /\text{paːkaː}/ & /\text{paːká}/ & /\text{paːkáː}/ \\ /\text{páka}/ & /\text{pákaː}/ & /\text{páká}/ & /\text{pákáː}/ \\ /\text{páːka}/ & /\text{páːkaː}/ & /\text{páːká}/ & /\text{páːkáː}/ \end{array}\right\}$$

A *generating function Gen* maps an underlying concatenation $\mathbf{Aa}$ of an underlying root $\mathbf{A}$ and an underlying suffix $\mathbf{a}$ to a set of *candidate surface forms*. We assume that the set of surface forms is some subset of the set of root/suffix concatenations. Thus again, the root and the suffix in any given surface concatenation can always be pulled apart. A concatenation is denoted with letters from the end of the alphabet when it is construed as a surface form: $\dots \mathbf{Xx}, \mathbf{Yy}, \mathbf{Zz}$. Furthermore, we assume that all underlying forms share the same candidate set. In our example, we could thus define the candidate set shared by all underlying concatenations as the subset (4) of the eight root/suffix concatenations which have exactly one stress.

(4)
$$\left\{\begin{array}{llll} & & [\text{paká}] & [\text{pakáː}] \\ & & [\text{paːká}] & [\text{paːkáː}] \\ [\text{páka}] & [\text{pákaː}] & & \\ [\text{páːka}] & [\text{páːkaː}] & & \end{array}\right\}$$

Finally, $\mathcal{C}$ is a finite set of *phonological constraints*. Each constraint takes an underlying concatenation $\mathbf{Aa}$ and a surface candidate concatenation $\mathbf{Xx}$ and returns the corresponding number of violations. To illustrate, consider the four markedness constraints (5a) together with the two identity faithfulness constraints (5b) relative to the two relevant features [STRESS] and [LENGTH]. The two consonants *p* and *k* play no role in this example (apart from distinguishing roots from suffixes): each morpheme can be identified with its vowel.

(5) a.

MAINLEFT:	assigns a violation to forms whose initial syllable is unstressed
MAINRIGHT:	assigns a violation to forms whose final syllable is unstressed
NOLONG:	assigns a violation to each long vowel
WSP:	assigns a violation to each long vowel which is unstressed

 b. IDENT[STRESS], IDENT[LENGTH]

These ingredients ($\mathcal{R}, \mathcal{S}, \mathbb{R}, \mathbb{S}, Gen, \mathcal{C}$) define a typology of languages, as explained in the next subsection. The example defined in (1)-(5) has become known in the literature (since Tesar 2006) as the *Paka typology.*

2.2. **Languages**

A *lexicon* assigns an underlying root or suffix morpheme to each root or suffix meaning. A generic lexicon is denoted by *Lex*. Thus, the identities $Lex(\mathsf{M}) = \mathbf{A}$ and $Lex(\mathsf{m}) = \mathbf{a}$ say that the root meaning M corresponds to the underlying root morpheme $\mathbf{A}$ and that the suffix meaning m corresponds to the underlying suffix morpheme $\mathbf{a}$ according to the lexicon *Lex*. For convenience, we assume lexicons to be one-to-one (no ambiguity). An example of lexicon for our running example is described in table format in (6). It says that the root meaning BOY corresponds to the underlying root /pa/, the suffix meaning nom to the underlying suffix /ka/, and so on.

(6)
$$Lex = \begin{array}{c} \\ \\ \end{array}\left[\begin{array}{cccccccc} \textsf{BOY} & \textsf{GIRL} & \textsf{DOG} & \textsf{CAT} & \textsf{nom} & \textsf{acc} & \textsf{gen} & \textsf{dat} \\ /\text{pa}/ & /\text{paː}/ & /\text{pá}/ & /\text{páː}/ & /\text{ka}/ & /\text{kaː}/ & /\text{ká}/ & /\text{káː}/ \end{array}\right]$$

Any ranking $\gg$ of the constraint set defines the corresponding OT grammar, denoted by $G_{\gg}$. The identity $G_{\gg}(\mathbf{Aa}) = \mathbf{Xx}$ thus says that the grammar $G_{\gg}$ corresponding to the ranking $\gg$ maps the underlying root/suffix concatenation **Aa** to the surface root/suffix concatenation **Xx**. For instance, the ranking (7a) of the constraint set (5) induces the grammar which maps the sixteen underlying concatenations (3) as in (7b).[2] The surface realization of a root meaning such as BOY of course depends on the suffix it combines with. The meaning combination BOYnom corresponds by (6) to the underlying concatenation /paka/ which surfaces as [páka] according to (7b). And the meaning combination BOYgen corresponds to the underlying concatenation /paká/ which surfaces as [paká]. The stress feature of the root morpheme BOY thus *alternates* depending on the suffix it combines with.

(7) a. WSP $\gg$ IDENT[STRESS] $\gg$ ML $\gg$ MR $\gg$ IDENT[LENGTH] $\gg$ NOLONG

b.
$$\left[\begin{array}{cccc} (/\text{paka}/, [\text{páka}]) & (/\text{pakaː}/, [\text{páka}]) & (/\text{paká}/, [\text{paká}]) & (/\text{pakáː}/, [\text{pakáː}]) \\ (/\text{paːka}/, [\text{páːka}]) & (/\text{paːkaː}/, [\text{páːka}]) & (/\text{paːká}/, [\text{paká}]) & (/\text{paːkáː}/, [\text{pakáː}]) \\ (/\text{páka}/, [\text{páka}]) & (/\text{pákaː}/, [\text{páka}]) & (/\text{páká}/, [\text{páka}]) & (/\text{pákáː}/, [\text{páka}]) \\ (/\text{páːka}/, [\text{páːka}]) & (/\text{páːkaː}/, [\text{páːka}]) & (/\text{páːká}/, [\text{páːka}]) & (/\text{páːkáː}/, [\text{páːka}]) \end{array}\right]$$

A *language* is a pair $(Lex, \gg)$ of a lexicon and an OT grammar, or equivalently the corresponding constraint ranking $\gg$. This definition captures the intuition that languages such as English, Italian or Latin all consist of a specific lexicon and a specific phonological grammar. The ingredients $(\mathcal{R}, \mathcal{S}, \mathbb{R}, \mathbb{S}, Gen, \mathcal{C})$ described in subsection 2.1 thus define a *typology of languages*, namely the typology of all pairs $(Lex, \gg)$ of a lexicon Lex together with the OT grammar corresponding to a constraint ranking $\gg$.

2.3. Paradigms

A *paradigm* specifies the surface realization of certain combinations of a root and a suffix meaning. A generic paradigm is denoted by Π. Thus, the identity $\Pi(\mathsf{Mm}) = \mathbf{Xx}$ says that the meaning combination Mm of the root meaning M and the suffix meaning m is realized as the surface concatenation **Xx** according to the paradigm Π. This definition captures the usual notion of paradigm, such as the Latin paradigm in (8). Root meanings are listed by row, suffix meanings are listed by column, the topmost and leftmost entry [aqua] provides the surface realization of the meaning combination WATERnom.

(8)
$$\begin{array}{c|cccc} & \textsf{nom} & \textsf{acc} & \textsf{gen} & \textsf{dat} \\ \textsf{WATER} & [\text{aqua}] & [\text{aquam}] & [\text{aquae}] & [\text{aquae}] \\ \textsf{ROSE} & [\text{rosa}] & [\text{rosam}] & [\text{rosae}] & [\text{rosae}] \end{array}$$

A paradigm Π is *consistent with* a language $(Lex, \gg)$ provided it satisfies the crucial condition (9). The lexicon Lex maps the root meaning M to some underlying root morpheme $\mathbf{A} = Lex(\mathsf{M})$ and the suffix meaning m to some underlying suffix morpheme $\mathbf{a} = Lex(\mathsf{m})$. We can then concatenate this root **A** with this suffix **a** and feed the resulting concatenation **Aa** to the OT grammar $G_{\gg}$ corresponding to the ranking $\gg$, which will return a certain surface concatenation **Xx**. Condition (9) says that the surface concatenation **Xx** thus obtained is exactly the one assigned by the paradigm Π to the meaning combination Mm we started from.

(9) $$\Pi(\mathsf{Mm}) = G_{\gg}\big(\underbrace{Lex(\mathsf{M})}_{\mathbf{A}}\underbrace{Lex(\mathsf{m})}_{\mathbf{a}}\big)$$

To illustrate, the Latin language is consistent with the paradigm (8) because its lexicon maps the root meaning WATER to /akw-/ and the suffix meaning nom to /-a/ and its phonology allows the resulting underlying concatenation /akwa/ to surface faithfully, as in the topmost and leftmost entry in (8). To give another example, the paradigm (10) is consistent with the language $(Lex, \gg)$ corresponding to the lexicon Lex in (6) and the grammar $G_{\gg}$ in (7). In fact, the root meaning BOY corresponds to the underlying root /pa/ according to the lexicon Lex, the suffix meaning nom corresponds to the underlying form /ka/, and the resulting underlying concatenation /paka/ surfaces as [páka] according to the grammar $G_{\gg}$, as in the topmost and leftmost entry of the paradigm (10).

(10)
$$\begin{array}{c|cccc} & \textsf{nom} & \textsf{acc} & \textsf{gen} & \textsf{dat} \\ \textsf{BOY} & [\text{páka}] & [\text{páka}] & [\text{paká}] & [\text{pakáː}] \\ \textsf{GIRL} & [\text{páːka}] & [\text{páːka}] & [\text{paká}] & [\text{pakáː}] \\ \textsf{DOG} & [\text{páka}] & [\text{páka}] & [\text{páka}] & [\text{páka}] \\ \textsf{CAT} & [\text{páːka}] & [\text{páːka}] & [\text{páːka}] & [\text{páːka}] \end{array}$$

[2]This grammar can be described in words as follows. Stress is lexiconsl: if an underlying form has exactly one stress, it is preserved no matter whether it is initial or final (this follows from the fact that IDENT[STRESS] outranks both MAINLEFT and MAINRIGHT). Otherwise, default stress is initial: if an underlying form has no stress or two stresses, it surfaces with initial stress (this follows from the fact that MAINLEFT outranks MAINRIGHT). Finally, long vowels are only allowed in the stressed position: an underlying long vowel surfaces long only if it is stressed (this follows from the fact that IDENT[LONG] is sandwiched between WSP and NOLONG).

A paradigm is, essentially, a collection of surface concatenations $\mathbf{Xx}$, $\mathbf{Yy}$, … annotated with the corresponding root and suffix meanings. These meanings reveal whether two different surface concatenations $\mathbf{Xx}$ and $\mathbf{Yy}$ are the surface realizations of two meaning combinations Mm and $\mathsf{M}\widehat{\mathsf{m}}$ which share, say, the same root meaning M. If the surface forms $\mathbf{X}$ and $\mathbf{Y}$ are different, then we know that the underlying form (whatever it is) assigned by the lexicon to the shared root meaning M undergoes some alternation when it surfaces in the context of one or both of the two suffixes m and $\widehat{\mathsf{m}}$. Here is an example. We have noted above that stress of the morpheme corresponding to the root meaning BOY alternates according to the grammar (7) depending on whether that root is concatenated with nom or dat. The paradigm (10) encodes this alternation, because the first and third entry in the top row show that the root surfaces with a different stress in the two concatenations. In other words, a paradigm is a way to encode information on the alternations enforced by the corresponding grammar without revealing the underlying forms assigned by the corresponding lexicon.

2.4. **The learning problem**

A child acquiring English, Italian, or Latin is exposed to a sequence of surface concatenations $\mathbf{Xx}$, $\mathbf{Yy}$, … Because of the assumption that surface roots and suffixes belong to disjoint sets of morphemes, the learner can parse any surface concatenation $\mathbf{Xx}$ into its root $\mathbf{X}$ and its suffix $\mathbf{x}$. Assume that the context of utterance provides the child with enough cues to reconstruct the root meaning M corresponding to the surface root $\mathbf{X}$ and the suffix meaning m corresponding to the surface suffix $\mathbf{x}$. The child thus effectively has access to pairs $(\mathsf{Mm}, \mathbf{Xx})$ of a root/suffix meaning combination Mm and its surface realization $\mathbf{Xx}$. The collection of these pairs is a paradigm consistent with the native phonology. These considerations motivate the problem of learning languages from paradigms, explicitly stated in (11).

(11) THE PROBLEM OF LEARNING LANGUAGES FROM PARADIGMS:
a. *Given*: i. the specifications $(\mathcal{R}, \mathcal{S}, \mathbb{R}, \mathbb{S}, Gen, \mathcal{C})$ which define a typology of languages;
ii. a paradigm Π consistent with some language in the typology.
b. *Find:* a language $(Lex, \gg)$ in the typology consistent with the paradigm Π according to condition (9).

From the generative perspective, the learner has full knowledge of the underlying typology of languages, as stated in (11a.i). As motivated above, the data fed to the learner consist of some paradigm generated by some language in the typology, as stated in (11a.ii). Equivalently, the learner is provided with information concerning the alternations without being revealed the underlying forms. The target language is among the possibly many languages which are consistent with the paradigm. As a first stab at the problem of reconstructing the target language, the learner thus looks for a language consistent with the input paradigm, as stated in (11b).[3] To illustrate, a specific instance of the learning problem (11) is provided in (12). The language $(Lex, \gg)$ consisting of the lexicon *Lex* in (6) and the ranking $\gg$ in (7) is a solution of this specific instance of the problem of learning languages from paradigms.

(12) a. *Given*: i. the Paka typological specifications (1)-(5);
ii. the paradigm (10).
b. *Find:* a language $(Lex, \gg)$ in the Paka typology consistent with the paradigm according to condition (9).

2.5. **Some simplifying assumptions**

To start from the simplest possible case, Tesar tackles the learning problem (11) under the restrictive assumptions (13) on the underlying typology. Morphemes are assumed to have the simplest possible shape, namely to consist of a single segment. The phonology is assumed to be feature-based. And the features are assumed to be as simple as possible.

(13) a. Each root and each suffix morpheme consists of a single segment.
b. These root and suffix segments are distinguished by a certain set Φ of phonological features.
c. These features are binary, total, and independent (namely, all value combinations are instantiated).[4]

To illustrate, note that the Paka typology defined in (1)-(5) satisfies these simplifying assumptions (13). In fact, underlying root and suffix morphemes effectively consist of a single segment, namely the vowel (the consonant serves no other purpose than to distinguish roots from suffixes). And the set Φ of relevant features consists of the two features [LENGTH] and [STRESS], which are both binary and total and furthermore independent (all four combinations of values are instantiated).

[3]Since there might be many languages besides the target one which are consistent with the input paradigm, the current statement (11b) of the learning goal is rather loose. A natural way to strengthen the learning goal is to add a *restrictiveness* (or *subset*) *condition*: the language returned by the learner is required to be not only consistent with the paradigm, but also to be one of the most restrictive such languages (according to a properly defined notion of restrictiveness for languages). Tesar discusses these refinements of the learning problem in his chapter 8, but that discussion falls outside of the scope of this review.

[4]For instance, the features [HIGH] and [LOW] are not independent, because the combination [+LOW, +HIGH] is impossible.

2.6. **How to measure the efficiency of a learner**

Certain instances of the learning problem (11) are "larger" than others: they have more features, more morphemes, more constraints, a larger input paradigm, and so on. Larger instances require the learner to "do more work", namely to perform a larger number of steps. The *time complexity* of a learner is the rate with which the number of steps made by the learner grows with the size of the instances. If it grows slowly (i.e., polynomially), then the learner is *efficient* and can tackle instances of the problem which have a realistic size. Whether a learner qualifies as efficient depends of course on how the size of the instances of the problem is measured. In this subsection, I consider two different measures of the size of the instances of the problem (11) of learning languages from paradigms, which lead to a generous and a (more) demanding notion of efficiency.

2.6.1. *A generous definition of efficiency*

According to one approach, a learner should be allowed enough time to scan the morpheme sets $\mathcal{R}$ and $\mathcal{S}$ to find a good underlying morpheme for each meaning and thus assemble a lexicon. The number of steps which a learner is allowed to make should thus depend on the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes, leading to the definition (14).[5]

(14) An algorithm for the problem (11) of learning languages from paradigms is *efficient* provided the number of steps grows slowly with the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes (and the number $|\mathcal{C}|$ of constraints).

Letting the permitted number of steps depend on the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes as in (14) affords the learner with enough time to inspect the input paradigm Π. Furthermore, it affords the learner with enough time to scan any candidate set, which is crucial given that no currently available OT ranking algorithm succeeds without scanning all the candidates (in the worst case). On the other hand, letting the permitted number of steps depend on the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes does not make the problem trivial. In fact, the number of lexicons is factorial in the number of morphemes (because lexicons are bijections between meanings and an equal number of morphemes). Thus, allowing the learner the time to scan all the morphemes does *not* afford it enough time to exhaustively scan the typology of languages: brute force search is not efficient.

2.6.2. *A more demanding definition of efficiency*

In subsection 2.5, the learning setting was simplified by assuming that the underlying root and suffix morphemes consist of single segments which are distinguished by a certain set Φ of total, binary and independent features. Within this feature-based framework, Tesar suggests that the number of steps that a learner is allowed to make should depend on the number $|\Phi|$ of features rather than on the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes, leading to the definition (15). The difference between the two formulations (14) and (15) is substantial, because the number of morphemes grows exponentially in the number of features. Tesar's notion (15) of efficiency is thus more demanding than the one in (14).[6]

(15) An algorithm for the problem (11) of learning languages from paradigms is *efficient* provided the number of steps grows slowly with the number $|\Phi|$ of features, as well as the number $|\mathcal{C}|$ of constraints, the number of entries in the paradigm Π, and the size of the largest candidate set.

Tesar's definition (15) must crucially allow the number of steps made by the learner to also depend on the number of entries in the input paradigm Π. In fact, the input paradigm Π could consist of a full $4^{|\Phi|}$ entries, in which case a learner could not even inspect the paradigm in time polynomial in the number $|\Phi|$ of features. Furthermore, Tesar's definition (15) must crucially allow the number of steps to also depend on the size of the largest candidate set. In fact, a candidate set could consist of a full $4^{|\Phi|}$ entries, in which case the learner could not inspect the candidate set in time polynomial in the number $|\Phi|$ of features and thus it would not be able to tackle even a basic ranking problem (because, as recalled above, no currently available OT ranking algorithm succeeds without scanning all the candidates in the worst case).[7]

3. Output-drivenness

This section introduces the notion of output-drivenness. The presentation is tailored to the restrictive representational framework described in the statement of the learning problem in section 2. It thus does no justice to the theory of output-drivenness developed in chapters 2 and 3 of Tesar's book for much broader framework.

[5]Analogously, in order to learn a constraint ranking, the learner should be allowed enough time to scan the constraint set $\mathcal{C}$ and the number of steps should thus be allowed to grow (polynomially!) with the size $|\mathcal{C}|$ of the constraint set.

[6]Here is another way to appreciate this point. The cardinality of the typology of all possible lexicons was expressed above as $(|\mathcal{R}| + |\mathcal{S}|)!$ in terms of the number $|\mathcal{R}| + |\mathcal{S}|$ of morphemes. This becomes $(4^{|\Phi|})!$ in terms of the number $|\Phi|$ of features, because $|\mathcal{R}| + |\mathcal{S}| = 4^{|\Phi|}$. In other words, measuring the typology of lexicons against the number of features as required by (15), makes it "doubly" super-polynomial, because of both the exponential and the factorial. Measuring the typology of lexicons against the number of morphemes as required by (14) effectively "discounts" one of the two super-polynomial dependencies, namely the exponential one (so that the problem is made easier), although crucially not the factorial one (so that the problem is not made trivial).

[7]Consider the instance (12) of the problem of learning languages from paradigms, based on the Paka typology defined in (1)-(5). The size of this instance of the problem is the same, no matter whether it is measured according to (14) or (15). This equivalence is due to the fact that the input paradigm is large (it has an entry for each meaning combination) and the candidate sets are large as well (they consist of half of the concatenations). In any such case with a large paradigm or large candidate sets, the tight dependence of (15) on the number of features (rather than on the number of morphemes) is loosened up by the simultaneous dependence on the size of the paradigm and of the candidate sets, making the two definitions (14) and (15) equivalent.

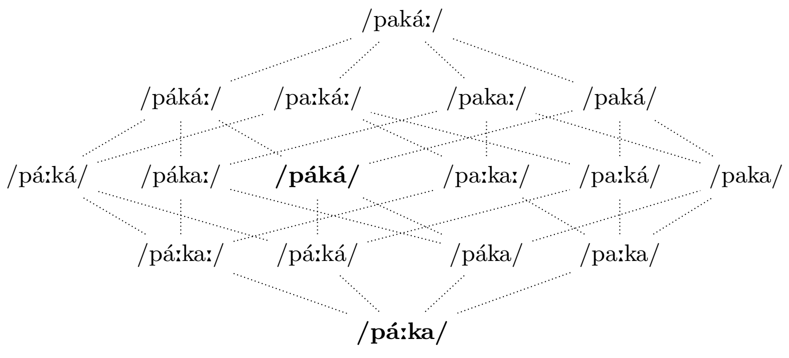
3.1. **Similarity**

Recall the representational framework we are assuming. Underlying forms $\mathbf{Aa}, \mathbf{Bb}, \dots$ and their surface candidates $\mathbf{Xx}, \mathbf{Yy}, \dots$ are concatenations of roots $\mathbf{A}, \mathbf{B}, \dots, \mathbf{X}, \mathbf{Y}, \dots$ and suffixes $\mathbf{a}, \mathbf{b}, \dots \mathbf{x}, \mathbf{y}, \dots$. To start from the simplest possible case, we have assumed in (13) that each root and each suffix consists of a single segment. That these segments are characterized in terms of a certain set Φ of phonological features. And that these features are all total and binary. Within this representational framework, condition (16) formalizes the intuition that an underlying concatenation **Aa** is *less similar* to a surface concatenation **Xx** than another underling concatenation **Bb** is.[8]

(16) If the underlying root segment **B** (the underlying suffix segment **b**) differs from the surface root segment **X** (the surface suffix segment **x**) for the value of some feature φ in the feature set Φ, then the underlying root segment **A** (the underlying suffix segment **a**) differs from the surface root segment **X** (the surface suffix segment **x**) for the value of that feature φ as well.

To illustrate, consider again the Paka typology defined in (1)-(5). The corresponding feature set Φ consists of the two features [LENGTH] and [STRESS] (which are total and binary). The underlying form **Aa** = /páːka/ is less similar to the surface form **Xx** = [pakáː] than the underlying form **Bb** = /páká/ is. In fact, condition (16) holds for the roots, because the more similar underlying root **B** = /pá/ differs from the surface root **X** = [pa] relative to the feature [STRESS] and the less similar root **A** = /páː/ differs relative to that feature as well (besides also differing relative to the other feature [LENGTH]). Condition (16) holds for the suffixes as well, because the more similar underlying suffix **b** = /ká/ differs from the surface suffix **x** = [káː] relative to the feature [LENGTH] and the less similar suffix **a** = /ka/ differs relative to that feature as well (besides also differing relative to the other feature [STRESS]). By reasoning analogously, the sixteen underlying forms (3) of the Paka typology can be ordered as in (17) according to their similarity to the surface form **Xx** = [pakáː] (more similar forms sit above less similar ones).

(17)



The underlying form /pakáː/ sits at the top of the lattice, because it is of course the most similar underlying form to the surface form [pakáː] which induces the lattice. The underlying form /páːka/ sits at the bottom of the lattice, because both its root and suffix differ relative to both features [LENGTH] and [STRESS] from the surface form [pakáː] which induces the lattice.

3.2. **Output-drivenness**

Output-drivenness captures the following intuition: any discrepancy between an underlying and a surface concatenation is *driven* by the goal of making the surface (or *output*) concatenation fit the phonotactics. This intuition can be elaborated as follows. Suppose that a grammar G maps a certain underlying concatenation **Aa** to a certain surface concatenation **Xx**. Intuitively, this means that **Xx** is phonotactically licit and that **Aa** is not too dissimilar from **Xx** according to that grammar. Consider another underlying concatenation **Bb** which is more similar to **Xx** than **Aa** is. An output-driven grammar G then also maps **Bb** to **Xx**, since the phonotactic status of **Xx** does not depend on the underlying form and furthermore **Bb** is even more similar to **Xx**.

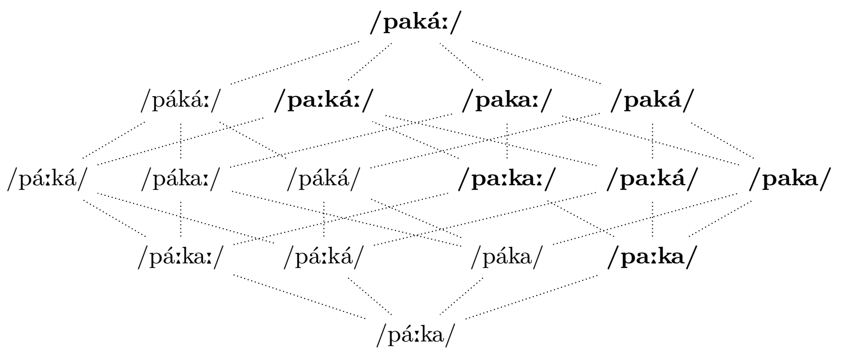
(18) A grammar G is *output-driven* provides it satisfies the following condition: if G maps an underlying concatenation **Aa** to a surface concatenation **Xx** and if another underlying concatenation **Bb** is more similar to **Xx** than **Aa** is, then G also maps **Bb** to **Xx**.

Consider again the relative similarity of the sixteen underlying forms (3) of the Paka typology to the surface form **Xx** = [pakáː], represented with the lattice (17). Suppose that an output-driven grammar maps the underlying form **Aa** = /paːka/ to that surface form **Xx** = [pakáː]. Then it must also map to **Xx** = [pakáː] any underlying form boldfaced in (19), as they are all more similar to **Xx** = [pakáː] than **Aa** = /paːka/ is. Output-drivenness thus

[8]More precisely, condition (16) formalizes the intuition that the underlying concatenation **Aa** is at most as similar to the surface concatenation **Xx** as the underling concatenation **Bb** is. I will allow myself some sloppiness, and say "less similar than" instead of "at most as similar as". Following Tesar, I always use **Aa** for the underlying concatenation which is less similar to the surface concatenation **Xx** and **Bb** for the one which is more similar. As a mnemonic, think about the alphabetic order: "b" is closer than "a" to the end of the alphabet, where "x" sits.

substantially limits the "freedom" of a grammar. We will see below in subsection 3.4 that each OT grammar in the Paka typology (1)-(5) is guaranteed to satisfy the output-drivenness condition (18).

(19)



Chain shifts and derived environment effects yield phonological mappings which are not output-driven.

3.3. (Eventual)-Idempotency

Assume that the grammar G maps some underlying concatenation $\mathbf{Aa}$ to some surface concatenation $\mathbf{Xx}$. This means in particular that the surface concatenation $\mathbf{Xx}$ is phonotactically licit according to this grammar G. Furthermore, condition (16) guarantees that $\mathbf{Xx}$ (construed as an underlying concatenation) is more similar to itself (construed as a surface concatenation) than any other underlying concatenation $\mathbf{Aa}$. The output-drivenness condition (18) with $\mathbf{Xx}$ in place of $\mathbf{Bb}$ thus ensures that G maps any phonotactically licit form $\mathbf{Xx}$ (construed as an underlying concatenation) to itself (construed as a surface concatenation). The latter condition says that the grammar G is *idempotent*, a property which plays a crucial role in the literature on the acquisition of phonotactics (Prince and Tesar 2004, Hayes 2004). In other words, output-drivenness entails idempotency (the reverse is false). In conclusion, output-drivenness is stronger than idempotency which is in turn stronger than Moreton's (2004) *eventual idempotency.*

3.4. Output-drivenness for OT grammars

The definition (18) of output-drivenness is independent of the framework (SPE, OT, HG, ...) used to define phonological grammars. In chapter 3, Tesar then focuses on OT and addresses the following question: which constraint sets yield typologies of OT grammars which are all output-drivenness? In order to address this question, consider an OT grammar which maps the less similar underlying concatenation $\mathbf{Aa}$ to the surface concatenation $\mathbf{Xx}$. This means that the mapping $(\mathbf{Aa}, \mathbf{Xx})$ beats the mapping $(\mathbf{Aa}, \mathbf{Yy})$ according to the constraint ranking corresponding to that grammar, for any other surface concatenation $\mathbf{Yy}$. And this means in turn that one constraint which prefers the winner mapping $(\mathbf{Aa}, \mathbf{Xx})$ over the loser mapping $(\mathbf{Aa}, \mathbf{Yy})$ is ranked above every constraint which instead prefers $(\mathbf{Aa}, \mathbf{Yy})$ over $(\mathbf{Aa}, \mathbf{Xx})$, as depicted in (20).

(20)
$$\begin{array}{cl} C & \text{a constraint which prefers } (\mathbf{Aa}, \mathbf{Xx}) \text{ over } (\mathbf{Aa}, \mathbf{Yy}) \\ | & \\ C_1, C_2, \ldots & \text{all the constraints which prefer } (\mathbf{Aa}, \mathbf{Yy}) \text{ over } (\mathbf{Aa}, \mathbf{Xx}) \end{array}$$

Consider another underlying concatenation $\mathbf{Bb}$ which is more similar to $\mathbf{Xx}$ than $\mathbf{Aa}$ is and must therefore be mapped to $\mathbf{Xx}$ as well in order for output-drivenness to hold. If the constraint C top ranked in (20) is a markedness constraint M, then it does not care whether the underlying concatenation is $\mathbf{Aa}$ or $\mathbf{Bb}$. The fact that M prefers the mapping $(\mathbf{Aa}, \mathbf{Xx})$ over the mapping $(\mathbf{Aa}, \mathbf{Yy})$ thus entails that it also prefers $(\mathbf{Bb}, \mathbf{Xx})$ over $(\mathbf{Bb}, \mathbf{Yy})$. If instead this constraint C is a faithfulness constraint F, this entailment holds provided F satisfies condition (21). The diagram (20) can thus be updated through the boxed condition in (22).

(21) *Output-drivenness faithfulness condition I.* Assume that the underlying form $\mathbf{Bb}$ is more similar to the surface form $\mathbf{Xx}$ than the underlying form $\mathbf{Aa}$ is. If a faithfulness constraint F prefers the mapping $(\mathbf{Aa}, \mathbf{Xx})$ to the mapping $(\mathbf{Aa}, \mathbf{Yy})$, then it also prefers the mapping $(\mathbf{Bb}, \mathbf{Xx})$ to the mapping $(\mathbf{Bb}, \mathbf{Yy})$.

(22)
$$\begin{array}{cl} C & \text{a constraint which } \boxed{\text{prefers } (\mathbf{Bb}, \mathbf{Xx}) \text{ over } (\mathbf{Bb}, \mathbf{Yy})} \\ | & \\ C_1, C_2, \ldots & \text{all the constraints which prefer } (\mathbf{Aa}, \mathbf{Yy}) \text{ over } (\mathbf{Aa}, \mathbf{Xx}) \end{array}$$

Consider a constraint which incorrectly prefers the mapping $(\mathbf{Bb}, \mathbf{Yy})$ over the mapping $(\mathbf{Bb}, \mathbf{Xx})$. If it is a markedness constraint M, then again it also prefers $(\mathbf{Aa}, \mathbf{Yy})$ over $(\mathbf{Aa}, \mathbf{Xx})$, namely it is one of the constraints $C_1, C_2, \ldots$ ranked at the bottom of (22). If instead it is a faithfulness constraint F, that same conclusion holds provided the faithfulness constraint F satisfies condition (23). The diagram (22) can thus be updated through the boxed condition in (24). The ranking configuration in (24) ensures that the ranking considered ranks a constraint which prefers $(\mathbf{Bb}, \mathbf{Xx})$ over $(\mathbf{Bb}, \mathbf{Yy})$ above every constraint which instead prefers $(\mathbf{Bb}, \mathbf{Yy})$ over $(\mathbf{Bb}, \mathbf{Xx})$, so that $(\mathbf{Bb}, \mathbf{Xx})$ wins over $(\mathbf{Bb}, \mathbf{Yy})$. Since this conclusion holds for any surface concatenation $\mathbf{Yy}$, the grammar considered indeed maps the more similar underlying form $\mathbf{Bb}$ to $\mathbf{Xx}$, as desired.

(23) *Output-drivenness faithfulness condition II.* Assume that the underlying form $\mathbf{Bb}$ is more similar to the surface form $\mathbf{Xx}$ than the underlying form $\mathbf{Aa}$ is. If a faithfulness constraint F prefers the mapping $(\mathbf{Bb}, \mathbf{Yy})$ to the mapping $(\mathbf{Bb}, \mathbf{Xx})$, then it also prefers the mapping $(\mathbf{Aa}, \mathbf{Yy})$ to the mapping $(\mathbf{Aa}, \mathbf{Xx})$.

(24)
$$\begin{array}{rl} C & \text{a constraint which prefers } (\mathbf{Bb}, \mathbf{Xx}) \text{ over } (\mathbf{Bb}, \mathbf{Yy}) \\ | & \\ C_1, C_2, \ldots & \text{all the constraints which } \boxed{\text{prefer } (\mathbf{Bb}, \mathbf{Yy}) \text{ over } (\mathbf{Bb}, \mathbf{Xx})} \end{array}$$

Let me take stock. Suppose that every faithfulness constraint F satisfies the two conditions (21) and (23) whenever the underlying concatenation $\mathbf{Bb}$ is more similar to the surface concatenation $\mathbf{Xx}$ than the underlying concatenation $\mathbf{Aa}$ is. Then, any grammar corresponding to any ranking of the constraint set is output-driven, no matter what the markedness constraints look like. Tesar shows in particular that Ident-type faithfulness constraints (together with Max and Dep) satisfy these two crucial conditions (21) and (23). Thus for example, every grammar in the OT typology defined by the Paka specifications (1)-(5) is output-driven, because the set of faithfulness constraints (5b) only consists of the two identity constraints Ident[stress] and Ident[length]. Other faithfulness constraints (including Dep[V], Max[C], the local conjunction of two Ident-type faithfulness constraints, and positional faithfulness constraints) are shown by Tesar not to satisfy these two conditions (21) and (23). They have indeed been used in the OT phonological literature to capture non-output-driven phenomena such as chain shifts and derived environment effects (see Tesar's chapter 4 for discussion and references).

4. What can output-drivenness do for the learner? the perspective of time complexity

Section 2 has reviewed the problem of learning languages from paradigms. Tesar tackles this problem starting from the premise that, "for language learning to be possible, the linguistic theory must have some kind of non-trivial structure [. . .] that can be exploited by a learner" (p. 18). Section 3 has reviewed a structural condition that could be imposed on the typology explored by the learner, namely output-drivenness. Unfortunately, this is not a typological universal: several attested phonological phenomena are not output-driven and various faithfulness constraints indeed give rise to a non-output-driven behavior. Tesar contends nonetheless that "the definition of output-driven maps is a first cut at identifying that structure" which makes learning possible. These considerations motivate the following question: what can output-drivenness do for the learner? As anticipated in section 1, this section (together with the next one) address this question by comparing Tesar's *Output Driven Learner* (ODL) with Merchant's (2008) state-of-the-art *Contrast Pair and Ranking Information* (CPR) learner. Subsection 4.1 outlines the architecture shared by CPR and ODL, based on the idea of maintaining partial lexical and ranking information which is incremented iteratively. Subsection 4.2 (complemented by appendix A, available as Online Supplementary Materials) introduces CPR's subroutines for extracting lexical and ranking information. These subroutines are guaranteed to never make an error, but are too slow to be efficient. Subsection 4.3 (complemented by appendix B, available as Online Supplementary Materials) then introduces ODL's subroutines as a sped-up reformulation of CPR's subroutines through output-drivenness. The implications of this sped-up for efficiency are discussed. Finally, subsections 4.4 and 4.5 illustrate two of ODL's sped-up subroutines in more detail, underscoring the role played by output-drivenness in their analyses.

4.1. **The common architecture of CPR and ODL**

4.1.1. *Partial lexical information*

As stated in (13), we are assuming that root and suffix morphemes consist of single segments which are distinguished by a certain set Φ of phonological features. A lexicon can thus describe the underlying segment it assigns to a certain meaning by specifying the value of each feature in that segment. To illustrate, consider again the Paka typology defined in (1)-(5). As noted above, its root and suffix morphemes effectively consist of single segments (the vowels) which are distinguished by the two features [length] and [stress]. The lexicon (6) can thus be specified in terms of feature values as in (25). The two entries "−" in the leftmost column say that the underlying segment corresponding to the root meaning BOY is [−long] and [−stress], namely it is /pa/, as indeed prescribed by (6).

(25)
$$Lex = \begin{array}{c} \begin{array}{cccccccc} \text{BOY} & \text{GIRL} & \text{DOG} & \text{CAT} & \text{nom} & \text{acc} & \text{gen} & \text{dat} \end{array} \\ \left[\begin{array}{cccccccc} - & + & - & + & - & + & - & + \\ - & - & + & + & - & - & + & + \end{array}\right] \end{array} \begin{array}{l} \\ \text{[LENGTH]} \\ \text{[STRESS]} \end{array}$$

A *partial* lexicon leaves some features unset for some root or suffix segments. An extreme case is that of an *empty* lexicon, which does not specify the value of any feature for any meaning. At the other extreme, a lexicon which specifies the value of each feature for each meaning is *total*. From now on, I will use the term "lexicon" and the symbol *Lex* to refer to a lexicon which can be either partial or total. To illustrate, the lexicon (25) is total while the lexicon (26) is partial, for instance because it fails at specifying the value of the feature [length] for the root meaning GIRL.

(26)
$$
\begin{array}{rcccccccccl}
 & & \mathsf{BOY} & \mathsf{GIRL} & \mathsf{DOG} & \mathsf{CAT} & \mathsf{nom} & \mathsf{acc} & \mathsf{gen} & \mathsf{dat} & \\
\multirow{2}{*}{$Lex =$} & \Big[& - & & - & + & & & - & & \Big] \ [\textsc{length}] \\
 & \Big[& - & - & & + & - & & + & + & \Big] \ [\textsc{stress}]
\end{array}
$$

Two lexicons are *consistent* provided they agree where both are specified: if a certain feature is set for a certain meaning by both lexicons, then it is set to the same value. To illustrate, the two lexicons in (25) and (26) are consistent. Finally, an underlying form **Aa** for a meaning combination Mm is *consistent* with a lexicon *Lex* provided it agrees with any feature specified by the lexicon: if a feature is set by the lexicon *Lex* for the root meaning M (for the suffix meaning m) to some value, the root morpheme **A** (the suffix morpheme **a**) has that value for that feature. To illustrate, the partial lexicon (26) is consistent with both underlying forms /paká/ and /paːká/ for the meaning combination $\mathsf{GIRLgen}$, as the lexicon does not set the feature [LENGTH] for the root meaning GIRL.

4.1.2. *Partial grammatical/ranking information*

An OT grammar is a collection of mappings (**Aa**, **Xx**) which specify that an underlying form **Aa** is mapped to a surface form **Xx**. A partial grammar specifies the surface realization only for some underlying concatenations. An extreme case is that of an *empty* grammar, which does not specify the surface realization of any underlying concatenation. At the other extreme, a grammar which provides a mapping for each underlying concatenation is *total*. From now on, I will use the term "grammar" and the symbol G to refer to a grammar which can be either partial or total. To illustrate, consider again the Paka typology defined in (1)-(5). The grammar (7) is total, because it provides the surface realization of each of the sixteen underlying concatenations (3) of the Paka typology. The grammar (27) is instead partial, because it fails at providing the surface realization for the underlying concatenations /pakaː/, /páká/, etcetera.

(27)
$$
\left[\begin{array}{cccc}
\text{(/paka/, [páka])} & & \text{(/paká/, [paká])} & \\
\text{(/paːka/, [páːka])} & \text{(/paːkaː/, [páːka])} & \text{(/paːká/, [paká])} & \text{(/paːkáː/, [pakáː])} \\
 & \text{(/pákaː/, [páka])} & & \text{(/pákáː/, [páka])} \\
\text{(/páːka/, [páːka])} & \text{(/páːkaː/, [páːka])} & \text{(/páːká/, [páːka])} &
\end{array}\right]
$$

Two grammars are *consistent* provided there exists at least a ranking which is consistent with both. This entails in particular that the two grammars agree where both are specified: if they both provide a surface realization for a certain underlying concatenation, then they provide the same surface realization. To illustrate, the grammar (7) and the partial grammar (27) just considered are consistent. Finally, a mapping (**Aa**, **Xx**) of an underlying concatenation **Aa** to a surface concatenation **Xx** is *consistent* with a grammar G provided there exists at least a ranking which is consistent with both, namely maps **Aa** to **Xx** and furthermore maps the underlying concatenation **Bb** to the surface concatenation **Yy** for every mapping (**Bb**, **Yy**) prescribed by the grammar G. To illustrate, the partial grammar (27) is consistent with the mapping (/pakaː/, [páka]) because both the mapping and that partial grammar are consistent with the ranking (7a).

4.1.3. *Iteratively boosting partial lexical and partial ranking information*

The learning problem (11) asks for a language consistent with a given paradigm. A language consists of a lexicon and an OT grammar, or equivalently the corresponding ranking. A natural approach to this learning problem is to maintain partial lexical information and partial grammatical information and to increment them iteratively by "boosting" one type of partial information with the other. This idea is formalized in the algorithmic scheme (28). The learner reads the input paradigm Π which is provided with an instance of the learning problem (line 1). The learner maintains lexical information in the form of a current lexicon *Lex* and grammatical information in the form of a current grammar G. They are initialized to the empty lexicon and the empty grammar (line 2). They are then updated by boosting one against the other. On the one hand, the learner tries to extract new grammatical information (namely, to add new mappings to its current grammar G) using the lexical information encoded in its current (possibly partial) lexicon *Lex* (line 4). On the other hand, the learner tries to extract new lexical information (namely, to set the value of some feature still unspecified by the current lexicon *Lex*) using the ranking information encoded in the current (possibly partial) grammar G (line 5). These two steps are repeated one after the other until no new lexical or grammatical information can be extracted from the input paradigm (line 6). The algorithm then returns its current lexicon *Lex* and its current grammar G (line 7). The returned lexicon *Lex* and the returned grammar G can of course be partial. The learner is deemed *correct* as long as any total lexicon Lex_{total} consistent with the returned lexicon *Lex* and any total grammar G_{total} consistent with the returned grammar G yield a language $(Lex_{\text{total}}, G_{\text{total}})$ which is consistent with the input paradigm Π, and thus solves the instance of the problem the learner has been trained on.

(28) THE ALGORITHMIC CORE SHARED BY CPR AND ODL:

```
1: Require: a paradigm $\Pi$
2: Initialize: set $G$ equal to the empty grammar and $Lex$ equal to the empty lexicon
3: repeat
4:     extend the current partial grammar $G$ using the current (possibly partial) lexicon $Lex$
5:     extend the current partial lexicon $Lex$ using the current (possibly partial) grammar $G$
6: until neither the current lexicon $Lex$ nor the current grammar $G$ can be further extended
```

```
7: Return: the current lexicon $Lex$ and the current grammar $G$
```

Both Merchant's (2008) CPR and Tesar's ODL subscribe to this idea of iteratively boosting partial lexical and grammatical information against each other. The algorithmic scheme (28) thus distills their shared architecture. I now turn to the subroutines used by CPR and ODL at lines 4 and 5 for extracting ranking and lexical information.

4.2. **CPR's slow subroutines**

CPR's subroutines are based on three core ideas, which go under the heading of *join*, *inconsistency detection*, and *contrast pairs*. This subsection offers a quick sketch of these three ideas (complemented by appendix A, available as Online Supplementary Materials). Merchant develops a new subroutine for extracting grammatical/ranking information. The intuitive idea is as follows (see appendix A.3 for details). The partial lexical information encoded in the current lexicon *Lex* reduces the range of possible underlying forms for a certain meaning combination Mm to those consistent with *Lex*. The subroutine thus considers all mappings of all consistent underlying forms to the surface form of Mm provided by the paradigm. And it then extracts any ranking information which is shared across all such mappings. Shared ranking information is extracted through a novel technique based on the notion of *join*. The technique is provably optimal, in the sense that it extracts as much ranking information as possible.

CPR's subroutine for extracting lexical information is based on the classical method of *inconsistency detection* (Kager 1999, Tesar et al. 2003, and Tesar 2006, among many others). The intuitive idea is as follows (see appendix A.5 for details). A certain feature is set to a certain value for a certain meaning combination Mm by detecting the fact that the opposite value is inconsistent with the current ranking information. In the sense that there is no ranking consistent with the current ranking information which is able to map an underlying form for Mm with that opposite value to the surface form for Mm provided by the paradigm. Both the subroutine for extracting ranking information through the join and the subroutine for extracting lexical information through inconsistency detection come with correctness guarantees: if the current partial lexical and ranking information are correct, the extended lexical information and the extended ranking information returned by these subroutines are correct as well.

These two subroutines for extracting ranking and lexical information can be applied not only to a *single entry* in the paradigm corresponding to a single meaning combination Mm but also to a *contrast pair*, namely a pair of meaning combinations which share either the root meaning (namely, a pair such as Mm and $\mathsf{M}\widehat{\mathsf{m}}$) or the suffix meaning (namely, a pair such as Mm and $\widehat{\mathsf{M}}\mathsf{m}$). Various authors (including Alderete et al. 2005 and Merchant and Tesar 2008) have pointed out that the subroutine for extracting lexical information through inconsistency detection extracts more information when applied to contrast pairs than when applied to individual paradigmatic entries. The intuitive idea is as follows (see appendix A.6 for details). Contrast pairs (not single entries) are indeed the smallest unit of data able to display morphophonemic alternations. A learner which extracts lexical information by only looking at a single entry at the time is thus effectively ignoring the alternations encoded in the paradigm and is therefore bound to fail. This intuition carries over to Merchant's new subroutine for extracting ranking information, which is indeed boosted by being applied to contrast pairs rather than to single paradigmatic entries (see appendix A.4 for details).

The crux of CPR's subroutines is their time complexity. Recall from subsection 2.6 that the time complexity of an algorithm for learning languages from paradigms can be measured either in terms of the number of morphemes (leading to a laxer notion of efficiency) or in terms of the number of features (leading to a more demanding notion of efficiency). The time complexity of CPR's subroutines is summarized in (29).

(29) Time complexity of CPR's subroutines:

	From a single entry	**From a contrast pair**
Extraction of ranking info.	unknown, but probably exponential in the # of morphemes	unknown, but probably exponential in the # of morphemes
Extraction of lexical info.	exponential in the # of features, but polynomial in the # of morphemes	exponential in the # of features, but polynomial in the # of morphemes

Merchant's subroutine for extracting ranking information based on the join relies on the operation of *fusional closure*. Unfortunately, there are no results (I am aware of) on the efficient computation of the fusional closure. The time complexity of Merchant's subroutine for extracting ranking information is thus unknown. But the subroutine is most likely not efficient, no matter how efficiency is defined and no matter whether the subroutine is applied to a single entry of the paradigm or to a contrast pair, as summarized in the first row of the table (29). The efficiency of the subroutine for extracting lexical information based on inconsistency detection depends on how its time complexity is measured. In fact, the subroutine is efficient relative to the laxer measure of time complexity (in terms of the number of morphemes) but not relative to the more demanding measure of time complexity (in terms of the number of features), no matter whether the subroutine is applied to a single entry in the paradigm or to a contrast pair, as summarized in the second row of the table (29).

4.3. **ODL's sped-up subroutines through output-drivenness**

Output-drivenness now enters the scene. The most spectacular implication of output-drivenness is a huge speed-up of Merchant's subroutine for extracting ranking information from a single entry in the paradigm. That subroutine requires the computation of the join and is thus (most plausibly) inefficient, no matter how efficiency is defined, as recalled in the top left entry of table (29). Output-drivenness circumvents the computation of the join, leading to a restatement of the subroutine whose time complexity grows only very slowly (namely linearly) with the number of features, as stated in the top left entry of table (30). In other words, the subroutine becomes efficient. And efficiency holds even relative to the strictest of the two measures of time complexity introduced in subsection 2.6.

(30) Time complexity of ODL's subroutines:

	From a single entry	**From a contrast pair**
Extraction of ranking info.	linear in the # of features	unknown, but probably exponential in the # of morphemes
Extraction of lexical info.	linear in the # of features	exponential in the # of features, but polynomial in the # of morphemes

Tesar's subroutine for extracting ranking information from a single meaning concatenation using output-drivenness is presented in subsection 4.4, together with an analysis of its time complexity and correctness which highlights the role of output-drivenness. Appendix B.1 explains the equivalence between Tesar's and Merchant's subroutines.

Output-drivenness also provides a substantial speed-up of the subroutine for extracting lexical information from a single entry in the paradigm through inconsistency detection. As recalled in the bottom left entry of table (29), that subroutine is only efficient relative to the laxer measure of time complexity (in terms of the number of morphemes), not in terms of the stricter measure (in terms of the number of features). Output-drivenness affords a restatement of this subroutine whose time complexity grows only very slowly (namely linearly) with the number of features, as stated in the bottom left entry of table (30). In other words, the subroutine becomes efficient also relative to the strict measure of time complexity. Tesar's subroutine for extracting lexical information from a single meaning concatenation using output-drivenness is presented in subsection 4.5, together with an analysis of its time complexity and correctness which highlights the role of output-drivenness. Appendix B.2 explains the connection between Tesar's subroutine and inconsistency detection.

Tesar discusses in detail how output-drivenness can be used to reformulate the subroutine for extracting lexical information from contrast pairs through inconsistency detection. This discussion is a bit more technical, and therefore relegated to appendix B.3. The upshot of this discussion is the following. Output-drivenness does not improve the time complexity of the subroutine, which remains inefficient relative to the stricter measure of time complexity (in terms of the number of features), as stated in the bottom right entry of table (30). Output-drivenness can nonetheless be used to construct a variant of this subroutine which is indeed faster, although weaker.

Merchant's subroutine for extracting ranking information requires the time-consuming computation of the join, no matter whether the subroutine is applied to a single entry of the paradigm or to a contrast pair. As recalled above, output-drivenness affords a huge reduction of time complexity when the subroutine is applied to a single entry, because output-drivenness effectively allows to circumvent the computation of a join. Tesar does not discuss the case where Merchant's subroutine for extracting ranking information is applied to a contrast pair. In this case, it seems to me that there is no way around the join, as explained in appendix B.4. If that conclusion is correct, the time complexity cannot be reduced through output-drivenness and the subroutine remains inefficient, as stated in the top right entry of table (30).

A comparison between tables (29) and (30) shows that output-drivenness has a substantial effect on time complexity when lexical and ranking information is extracted from a single entry in the paradigm. But that output-drivenness unfortunately falls short (or shorter) when lexical and ranking information is extracted from a contrast pair. Table (30) furthermore raises the following question: does the learner really need the full power afforded by Merchant's subroutine for extracting ranking information from contrast pairs? Or can the learner content itself with the ranking information that can be extracted from a single entry at the time? This question is (to the best of my knowledge) currently open.[9] And it seems to me a crucial question for the prospects of output-drivenness. If it turns out that the ranking information extracted from single entries suffices, then we can dismiss the bad news in the top right entry of table (30). And conclude that output-drivenness is indeed the structural assumption needed by the learner in order to achieve efficiency. If instead the additional ranking information provided by contrast pairs is needed, then output-drivenness will plausibly have to be supplemented with additional structural assumptions, able to yield a significant improvement in the top right entry of table (30).

4.4. **Extracting ranking information using output-drivenness**

4.4.1. *Description of the subroutine*

The subroutine Extract Ranking Information (ERI) defined in (31) takes as input some ranking information in the form of a partial grammar G, some lexical information in the form of a (possibly partial) lexicon Lex, and a paradigm Π (line 1). And it extends that ranking information by adding to the partial grammar G an additional

[9]Merchant does not compare a learner who extracts ranking information only from single entries to a learner (like CPR) which extracts it from contrast pairs. The question of whether ranking information needs to be extracted from contrast pairs as well is thus left open by Merchant's work.

mapping as follows. The subroutine considers a certain meaning combination Mm (line 2). It reads off the paradigm Π the surface realization **Xx** of that meaning combination Mm (line 3). It then constructs the underlying concatenation **Bb** as follows (line 4). If a feature is set by the input lexicon *Lex* to a certain value for the root meaning M (for the suffix meaning m), then the underlying root **B** (the underlying suffix **b**) has that value for that feature; otherwise, the underlying root **B** (the underlying suffix **b**) has the same value for that feature as the surface root **X** (the surface suffix **x**). The subroutine then updates the current partial grammar G by adding the mapping $(\mathbf{Bb}, \mathbf{Xx})$ of the underlying concatenation **Bb** just constructed to the surface realization **Xx** provided by the paradigm (line 5). The grammar thus updated is then returned (line 6).

(31) Extract Ranking Information (ERI)

```
1: Require: a partial grammar $G$, a (possibly partial) lexicon $Lex$, a paradigm $\Pi$
2: Require: a meaning combination Mm in the paradigm $\Pi$
3: read the surface realization $\mathbf{Xx}$ of the meaning combination Mm off the paradigm $\Pi$
4: construct the underlying form $\mathbf{Bb}$ for Mm which is consistent with $Lex$ and otherwise identical to $\mathbf{Xx}$
5: add to the partial grammar $G$ the mapping $(\mathbf{Bb}, \mathbf{Xx})$
6: Return: the updated partial grammar $G$
```

4.4.2. *Analysis of the subroutine*

The time complexity of ERI is controlled by the time required to execute line 4, which in turn grows only linearly with the number of features used to define the root and suffix segments. The subroutine is therefore efficient even relative to the more demanding notion of efficiency introduced in subsection 2.6.2. The following straightforward lemma uses output-drivenness to show that ERI is also correct.

Lemma 1. *Assume that the underlying typology is output-driven. Then, the subroutine* ERI *defined in (31)* preserves consistency*, in the following sense. Assume that the input paradigm* Π *is consistent with some target lexicon and some target grammar, that the input grammar G is consistent with that target grammar, and that the input lexicon Lex is consistent with that target lexicon. Then, also the updated grammar returned by ERI is consistent with that target grammar.*

Proof. Let **Aa** be the underlying concatenation posited by the target lexicon for the meaning combination Mm. Let **Bb** be the underlying form constructed by ERI at line 4. Since **Bb** is consistent with the input partial lexicon *Lex* and since the latter is consistent with the target lexicon, then **Bb** is consistent with the target lexicon. This means that the two underlying forms **Aa** and **Bb** agree for every feature value set by the input lexicon *Lex*. Furthermore **Bb** and the surface form **Xx** agree for every remaining feature value. Thus, the underlying form **Bb** constructed by ERI is more similar to the surface form **Xx** than the underlying form **Aa** posited by the target lexicon. Since the target grammar is by hypothesis output-driven and it maps the less similar underlying form **Aa** to **Xx**, then it also maps the more similar underlying form **Bb** to **Xx**. In other words, the target grammar is consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ of the underlying form **Bb** to the surface form **Xx**. Since the target grammar is by hypothesis also consistent with the input partial grammar G, then it is consistent with the updated partial grammar returned by ERI. □

Appendix B.1 develops further the analysis of ERI. It shows that ERI is equivalent to Merchant's (2008) subroutine for the extraction of ranking information through the notion of join (Tesar does not point out this equivalence). Crucially, Merchant's subroutine is optimal (namely, it extracts as much ranking information as possible) although unfortunately not efficient. Output-drivenness thus allows ERI to inherit the optimality of Merchant's subroutine while affording a huge gain in time complexity.

4.4.3. *First example*

Consider the instance (12) of the problem of learning languages from paradigms, whereby the underlying typology is the *Paka* typology defined in (1)-(5) and the paradigm is the one in (10), generated by the lexicon (6) and the OT grammar (7). Suppose that the subroutine ERI is fed the empty input grammar G and the empty lexicon *Lex* (line 1). And that it considers the meaning combination BOYnom (line 2). ERI then reads off the paradigm (10) the surface realization $\mathbf{Xx}$ = [páka] of that meaning combination (line 3). And it constructs the underlying form **Bb** which is identical to that surface form **Xx** but for the features set in the input lexicon *Lex* (line 4). Since *Lex* is empty, ERI constructs the completely faithful underlying form $\mathbf{Bb}$ = /páka/. ERI then adds to the empty grammar G the mapping $(\mathbf{Bb}, \mathbf{Xx})$ = (/páka/, [páka]) of the underlying form /páka/ faithfully to [páka]. After running ERI iteratively on all meaning combinations, it finally returns the partial grammar in (32a), which contains the four forms which appear in the paradigm faithfully mapped to themselves. This partial grammar is equivalent to the ranking conditions in (32b).

(32) a. $\Big[\big(\text{/páka/, [páka]}\big) \quad \big(\text{/paká/, [paká]}\big) \quad \big(\text{/pakáː/, [pakáː]}\big) \quad \big(\text{/páːka/, [páːka]}\big) \Big]$

b.
$$\begin{tikzcd}
 & \textsc{Ident[Stress]} \arrow[dl, dash] \arrow[dr, dash] & & \textsc{Ident[Length]} \arrow[d, dash] \\
\textsc{MainLeft} & & \textsc{MainRight} & \textsc{NoLong}
\end{tikzcd}$$

The application of ERI with an empty input lexicon *Lex* is also called *phonotactic learning*: since the lexicon is empty, the learner is extracting ranking information from the simple fact that the surface forms [páka], [paká], [paká:], and [pá:ka] appear in the paradigm, namely are all phonotactically licit (see subsection 5.6.2 of Tesar's book).

4.4.4. *Second example*

Suppose now that ERI is fed the partial grammar G in (32) and the partial lexicon *Lex* in (33) which sets a couple of values for the feature [LENGTH] (line 1). And that it considers the meaning combination DOGdat (line 2). ERI then reads off the paradigm (10) the surface realization $\mathbf{Xx}$ = [páka] of that meaning combination (line 3). And it constructs the underlying form $\mathbf{Bb}$ = /páka:/ (line 4) which is identical to the surface form $\mathbf{Xx}$ = [páka] but for the features set in the current lexicon for the root DOG (none) and the suffix dat (namely [LENGTH], which has been set to +).

(33)
$$Lex = \begin{array}{cccccccccc} \mathsf{BOY} & \mathsf{GIRL} & \mathsf{DOG} & \mathsf{CAT} & \mathsf{nom} & \mathsf{acc} & \mathsf{gen} & \mathsf{dat} & \\ - & & & & & & & + & \text{[LENGTH]} \\ & & & & & & & & \text{[STRESS]} \end{array}$$

ERI then adds to the empty grammar G the mapping $(\mathbf{Bb}, \mathbf{Xx})$ = (/páka:/, [páka]) of the underlying form /páka:/ just constructed to the surface form [páka] (line 6) and returns the updated grammar (34a). This partial grammar is equivalent to the ranking conditions in (34b).

(34) a. $\Big[$(/páka/, [páka]) (/paká/, [paká]) (/paká:/, [paká:]) (/pá:ka/, [pá:ka]) (/páka:/, [páka])$\Big]$

b.
$$\begin{tikzcd} & \textsc{Ident[Stress]} \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \textsc{WSP} \arrow[d, dash] \\ \textsc{MainLeft} & \textsc{MainRight} & \textsc{Ident[Length]} \arrow[d, dash] \\ & & \textsc{NoLong} \end{tikzcd}$$

This example illustrates how ERI is able to extract new non-phonotactic ranking information from a lexicon such as (33) which is just partial and thus sets only a few feature values.

4.5. **Extracting lexical information using output-drivenness**

4.5.1. *Description of the subroutine*

The subroutine EXTRACT LEXICAL INFORMATION (ELI) defined in (35) takes as input some lexical information in the form of a partial lexicon *Lex*, some ranking information in the form of a (possibly partial) grammar G, and a paradigm Π (line 1). The subroutine considers a certain meaning combination Mm (line 2). And it focuses on some feature φ which is not set by the input lexicon *Lex* (line 3). For concreteness, I am assuming here that the feature φ is unset for the root meaning M (the case where it is unset for the suffix meaning m is completely analogous). The goal of the subroutine is to extend the input partial lexicon *Lex* by trying to set the feature φ for the root meaning M. The subroutine reads off the paradigm Π the surface realization $\mathbf{Xx}$ of that meaning combination Mm (line 4). And it considers the underlying concatenation $\mathbf{Bb}$ which is consistent with the input lexicon *Lex* and is otherwise identical to the surface concatenation $\mathbf{Xx}$, but for the fact that the underlying root $\mathbf{B}$ and the surface root $\mathbf{X}$ have the opposite value for the feature φ (line 5). If the input grammar G is not consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ of this underlying concatenation $\mathbf{Bb}$ to the surface concatenation $\mathbf{Xx}$ (line 6), then the subroutine concludes that it is not possible to set the feature φ for the root meaning M to the opposite value than the one it surfaces with in $\mathbf{X}$. The subroutine thus updates the lexicon by setting the feature φ for the root meaning M equal to the value it surfaces with in $\mathbf{X}$ (line 7). The lexicon thus updated is then returned (line 9).

(35) EXTRACT LEXICAL INFORMATION (ELI)

```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a meaning combination Mm in the paradigm $\Pi$
3: Require: a feature $\varphi$ unset by $Lex$ for either M or m — for concreteness, say it is unset for M
4: read the surface realization $\mathbf{Xx}$ of the meaning combination Mm off the paradigm $\Pi$
5: construct the underlying concatenation $\mathbf{Bb}$ as follows:
     • $\mathbf{Bb}$ is consistent with the lexicon $Lex$
     • $\mathbf{B}$ has the opposite value of $\mathbf{X}$ for the feature $\varphi$
     • $\mathbf{Bb}$ agrees with the surface realization $\mathbf{Xx}$ for any other feature which is unset by $Lex$
6: if the partial grammar $G$ is inconsistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ then
7:     update the lexicon $Lex$ by setting the feature $\varphi$ for the root M equal to the value it has in $\mathbf{X}$
8: end if
9: Return: the updated lexicon $Lex$
```

4.5.2. *Analysis of the subroutine*

The time complexity of the subroutine ELI is controlled by the time required to execute line 4, which in turn grows only linearly with the number of features used to define the root and suffix segments. The subroutine is therefore efficient even relative to the more demanding notion of efficiency introduced in subsection 2.6.2. The following straightforward lemma uses output-drivenness to show that ELI is also correct.

Lemma 2. *Assume that the underlying typology is output-driven. Then, the subroutine* ELI *defined in (35)* preserves consistency*, in the following sense. Assume that the input paradigm* Π *is consistent with some target lexicon and some target grammar, that the input grammar* G *is consistent with that target grammar, and that the input lexicon* Lex *is consistent with that target lexicon. Then, also the updated lexicon returned by ELI is consistent with that target lexicon.*

Proof. Suppose by contradiction that ELI does not preserve consistency. This means that the target lexicon posits an underlying concatenation $\mathbf{Aa}$ for the meaning combination Mm whose underlying root $\mathbf{A}$ has a value for the feature φ which is different from the value set by ELI. This means in turn that the target underlying form $\mathbf{Aa}$ and the underlying form $\mathbf{Bb}$ constructed by ELI at line 4 agree on the value of the feature φ for the root meaning M, as they both set it to the opposite value of the surface root $\mathbf{X}$. Furthermore, $\mathbf{Bb}$ and $\mathbf{Aa}$ agree on every feature set by Lex (because $\mathbf{Bb}$ is consistent with Lex which is turn consistent with the target lexicon). Finally, $\mathbf{Bb}$ agrees with $\mathbf{Xx}$ on every other feature. The underlying concatenation $\mathbf{Bb}$ is thus more similar to the surface concatenation $\mathbf{Xx}$ than the underlying concatenation $\mathbf{Aa}$ posited by the target lexicon. Since the target grammar is by hypothesis output-driven and it maps the less similar underlying form $\mathbf{Aa}$ to $\mathbf{Xx}$, then it also maps the more similar underlying form $\mathbf{Bb}$ to $\mathbf{Xx}$. In other words, the target ranking is consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ of the underlying form $\mathbf{Bb}$ to the surface form $\mathbf{Xx}$. The target ranking is furthermore also consistent with the input partial grammar G (because of the hypothesis that the latter is consistent with the target grammar). Since the target ranking is consistent with both G and the mapping $(\mathbf{Bb}, \mathbf{Xx})$, they are consistent, contradicting the if-clause in line 6. □

Appendix B.2 develops further the analysis of ELI. It shows that it is equivalent to the classical strategy of *inconsistency detection*. Crucially, the time complexity of inconsistency detection is exponential in the number of features. Output-drivenness thus affords a substantial reduction of time complexity.

4.5.3. *Example*

Consider again the instance (12) of the problem of learning languages from paradigms. Suppose that ELI is fed the empty lexicon Lex and the input grammar G in (32) obtained above through phonotactic learning (line 1). That it considers the meaning combination BOYnom (line 2). And that it focuses on the feature $\varphi = [\textsc{length}]$ which is not set by the input lexicon Lex for the root meaning BOY (line 3). ERI then reads off the paradigm (10) the surface realization $\mathbf{Xx}$ = [páka] of that meaning combination (line 4). The target feature $\varphi = [\textsc{length}]$ surfaces with the value $-$ in the surface root $\mathbf{X}$ = [pá]. ELI thus tries to rule out the hypothesis that this feature $\varphi = [\textsc{length}]$ is set equal to the opposite value + in the underlying root for BOY. To this end, ELI constructs the underlying concatenation $\mathbf{Bb}$ which is consistent with Lex and is otherwise identical to the surface realization $\mathbf{Xx}$, but for the feature $\varphi = [\textsc{length}]$ for the root BOY, which is set to the value + (line 5). Since Lex is empty, this is the underlying concatenation $\mathbf{Bb}$ = /pá:ka/. ELI then checks whether the input partial grammar G is consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ = (/pá:ka/, [páka]) of the underlying concatenation /pá:ka/ just constructed to the surface concatenation [páka] (line 6). Consistency fails. In fact, this mapping requires in particular the markedness constraint NoLong to be ranked above the faithfulness constraint Ident[Length], in order for (/pá:ka/, [páka]) to win over (/pá:ka/, [pá:ka]). And this ranking condition is inconsistent with the ranking conditions (32b) expressed by the input grammar G. ELI thus updates the current empty lexicon by setting the feature [length] for the root meaning BOY equal to the value $-$ its surfaces with in [páka], so that the updated lexicon looks like (36).

(36)
$$\begin{array}{rcl} & \begin{array}{cccccccc}\mathsf{BOY}&\mathsf{GIRL}&\mathsf{DOG}&\mathsf{CAT}&\mathsf{nom}&\mathsf{acc}&\mathsf{gen}&\mathsf{dat}\end{array} & \\ Lex = & \left[\begin{array}{cccccccc} - & & & & & & & \\ & & & & & & & \end{array}\right] & \begin{array}{l}[\textsc{length}]\\ [\textsc{stress}]\end{array}\end{array}$$

This step can be summarized as follows. The paradigm contains forms with a long vowel. This means that long vowels are phonotactically licit (at least when stressed). Phonotactic learning has been able to learn that, namely to learn that Ident[Length] needs to be ranked above NoLong, as in (32). An underlying (stressed) long vowel thus cannot be shortened, hence the inconsistency of the mapping (/pá:ka/, [páka]) considered by ELI.

5. What can output-drivenness do for the learner? the perspective of correctness

Suppose that ODL[10] is trained on an input paradigm which is consistent with some target language $(Lex^{\text{tar}}, \gg^{\text{tar}})$ in the underlying typology. ODL initializes its current lexicon Lex to the empty lexicon and its current grammar G to the empty grammar. The initial lexicon Lex, being empty, is trivially consistent with the target lexicon Lex^{tar}. Furthermore, the initial grammar G, being empty, is trivially consistent with the (grammar corresponding to the)

[10]Throughout this section, I focus on ODL. But these considerations trivially extend to CPR, since the two algorithms share the same architecture and use equivalent subroutines, as reviewed in the preceding section.

target ranking $\gg^{\text{tar}}$. ODL then applies the subroutines ERI, ELI, and their restatements in terms of contrast pairs, one after the other, for a number of iterations. The analyses of these subroutines provided above in subsections 4.4 and 4.5 (as well as in appendix B.3) show that each of these subroutines preserves consistency: if the lexicon and the grammar provided as input to these subroutines are consistent with the target lexicon Lex^{tar} and the target ranking $\gg^{\text{tar}}$, then the updated lexicon and the updated grammar returned by these subroutines are also consistent with the target lexicon Lex^{tar} and the target ranking $\gg^{\text{tar}}$. In conclusion, the current lexicon and the current grammar entertained by ODL are initially consistent with $(Lex^{\text{tar}}, \gg^{\text{tar}})$ and stay consistent throughout, leading to the following result.

Theorem 1 (Consistency of ODL)**.** *Suppose that the underlying typology is output-driven. Let* $(Lex^{\text{tar}}, \gg^{\text{tar}})$ *be any language consistent with the paradigm that ODL is trained on. Then, the final lexicon learned by ODL is consistent with* Lex^{tar} *and the final grammar learned by ODL is consistent with* $\gg^{\text{tar}}$*.*

This theorem is half good news and half bad news. The good news is that ODL is *prudent*: theorem 1 ensures that ODL is always consistent with some target grammar. And prudence guarantees safe choices: ODL never sets a feature incorrectly in the current lexicon or adds an incorrect mapping to the current grammar. The bad news is that ODL is *too prudent*: theorem 1 says that ODL is not satisfied with being consistent with *one* target language but actually insists on being consistent with *all* of them. And excessive prudence hinders success: it is easy to formally single out cases where ODL is bound to fail because of its excessive prudence. The rest of this section elaborates on this point.

Tesar shows that when ODL is trained on the instance (12) of the problem of learning languages from paradigms, the lexicon returned by ODL fails to set the value of the feature [LENGTH] for the two suffixes nom and acc. That is not big deal, since those feature values are *not contrastive*. In other words, there are four different languages $(Lex_1^{\text{tar}}, \gg^{\text{tar}})$, $(Lex_2^{\text{tar}}, \gg^{\text{tar}})$, $(Lex_3^{\text{tar}}, \gg^{\text{tar}})$ and $(Lex_4^{\text{tar}}, \gg^{\text{tar}})$ in the Paka typology which are all consistent with the paradigm and thus count as target languages. They share the same ranking and only differ for how their four lexicons set the value of the feature [LENGTH] for the two suffixes nom and acc (two values times two suffixes gives four possibilities). Theorem 1 explains why ODL leaves those two feature values unset. Indeed, suppose there existed a run of ODL where the algorithm ended up setting those two feature values as well. For concreteness, suppose that ODL happened to set them as Lex_1^{tar}. The final lexicon learned by ODL would then be inconsistent with the other three target lexicons Lex_2^{tar}, Lex_3^{tar}, and Lex_4^{tar}. And this is forbidden by theorem 1: the final lexicon learned by ODL must be consistent with all four of these target lexicons, and thus must leave unset any feature value where they disagree. In other words, when there are different target languages, ODL refuses to choose one over the other, and prudently abstains from making any move that would lead to an inconsistency with any of the target languages.

In the case of the Paka language considered here, the failure of ODL to set the value of the feature [LENGTH] for the suffixes nom and acc causes no trouble: the partial lexicon returned by ODL can be completed by setting the two missing values at random. Yet, now that we have understood why ODL fails at setting those two values, it is easy to imagine variants where ODL's failure to set some values does lead into trouble. Indeed, suppose that the paradigm that ODL is trained on is consistent with two target languages $(Lex_1^{\text{tar}}, \gg^{\text{tar}})$ and $(Lex_2^{\text{tar}}, \gg^{\text{tar}})$ which have different lexicons Lex_1^{tar} and Lex_2^{tar} but (to keep things simple) share the ranking $\gg^{\text{tar}}$. Suppose that the two target lexicons Lex_1^{tar} and Lex_2^{tar} differ in particular for the way they set a certain feature φ for a certain morpheme, say the root meaning M. This is the same case as, say, the feature [LENGTH] and the morpheme nom above. Yet, suppose that there is some other morpheme, say another root $\widehat{\mathsf{M}}$ (possibly equal to M), and some other feature ψ (possibly equal to φ) such that consistency with the input paradigm requires the value assigned to feature ψ for $\widehat{\mathsf{M}}$ to crucially depend on the value assigned to feature φ for M. The two lexicons Lex_1^{tar} and Lex_2^{tar} will thus also necessarily differ for the value they assign to the feature ψ for $\widehat{\mathsf{M}}$. Again, when ODL is trained on this paradigm, it will fail to set the value of the feature φ for M and the value of the feature ψ for $\widehat{\mathsf{M}}$, because it would otherwise be inconsistent with one of the two target lexicons, in contradiction of theorem 1. Yet, in this case ODL has stopped short of succeeding. In fact, because of the correlation between the two feature values for the two morphemes, these values which ODL has failed to learn cannot be set at random, they are a crucial component of the learning task.

At this point, two crucial questions arise, which unfortunately I have to leave open. The *first question* is the following: is it possible to guarantee that ODL succeeds at least in the benign cases where the "pathologies" just described do not hold? More explicitly, consider the case of a paradigm Π which is consistent with a *single* language, so that ODL's excessive prudence cannot lead into trouble. Is it possible to guarantee that ODL succeeds in this benign case? The *second question* is the following: is it possible to construct phonologically plausible typologies which yield paradigms consistent with multiple languages where ODL's excessive prudence leads into trouble in the way just formally characterized? Or is it instead the case that such cases are hard to come by in natural language phonology? Whatever the answer to this question, it changes nothing for the importance of Tesar's work. If it turns out that no such cases are attested, this match between the typology of what is attested and the restrictions on what is learnable will provide exciting evidence in favor of ODL. If it turns out instead that such cases are well instantiated, then we will have made nonetheless substantial progress by understanding the shortcomings of ODL's reliance on inconsistency detection. And this will only have been made possible by the kind of careful and sophisticated computational work reported in Tesar's book.

References

Alderete, John, Adrian Brasoveanu, Nazarré Merchant, Alan Prince, and Bruce Tesar. 2005. Contrast analysis aids in the learning of phonological underlying forms. In *Proceedings of the 24th West Coast Conference on Formal Linguistics*, ed. John Alderete, Chung-hye Han, and Alexei Kochetov, 32–42. Somerville, MA, USA: Cascadilla Proceedings Project.

Hayes, Bruce. 2004. Phonological acquisition in Optimality Theory: The early stages. In *Constraints in phonological acquisition*, ed. René Kager, Joe Pater, and Wim Zonneveld, 158–203. Cambridge: Cambridge University Press.

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

Merchant, Nazarré Nathaniel. 2008. Discovering underlying forms: contrast pairs and ranking. Doctoral Dissertation, Rutgers University.

Merchant, Nazarré, and Bruce Tesar. 2008. Learning underlying forms by searching restricted lexical subspaces. In *Proceedings of the 41st annual meeting of the Chicago Linguistic Society*, ed. L Edwards Rodney. Chicago: Chicago Linguistic Society.

Moreton, Elliott. 2004. Non-computable functions in Optimality Theory. In *Optimality theory in phonology: A reader*, ed. John J. McCarthy, 141–163. Malden: MA: Wiley-Blackwell.

Prince, Alan. 2002. Entailed ranking arguments. Ms., Rutgers University, New Brunswick, NJ. Rutgers Optimality Archive, ROA 500. Available at http://www.roa.rutgers.edu.

Prince, Alan, and Bruce Tesar. 2004. Learning phonotactic distributions. In *Constraints in phonological acquisition*, ed. R. Kager, J. Pater, and W. Zonneveld, 245–291. Cambridge University Press.

Tesar, Bruce. 2006. Faithful contrastive features in learning. *Cognitive Science* 30:863–903.

Tesar, Bruce, John Alderete, Graham Horwood, Nazarré Merchant, Koichi Nishitani, and Alan Prince. 2003. Surgery in language learning. In *Proceedings of the 22nd West Coast Conference on Formal Linguistics*, ed. Gina Garding and Mimu Tsujimura, 477–490. Somerville, MA, USA: Cascadilla Proceedings Project.

Appendix A. Merchant's (2008) CPR

This section sketches Merchant's (2008) CPR algorithm. In subsection A.1, I explain how CPR adheres to the architecture (28), only with a more abstract representation of the current partial ranking information in terms of ERCs. I then review CPR's subroutines for extracting ranking and lexical information. I start from the easier case where they are applied to a single paradigmatic entry (subsections A.3 and A.5) and then turn to the extension to contrast pairs (subsections A.4 and A.6).

A.1. **ERCs and CPR's architecture**

An *elementary ranking condition* (ERC) is an assignment of one of the three symbols W, L, and *e* to each of the constraints in the constraint set (Prince 2002). An ERC is usually represented as a row of these symbols, with the constraints annotated on top. To illustrate, an ERC for the constraint set (5) of the Paka typology is provided in (37). An ERC *matrix* is a finite collection of ERCs, stacked one on top of the other (the order does not matter). An arbitrary ERC matrix is denoted by $\mathfrak{R}$. Given two ERC matrixes $\mathfrak{R}_1$ and $\mathfrak{R}_2$, $\mathfrak{R}_1 + \mathfrak{R}_2$ denotes the ERC matrix obtained by stacking one on top of the other (the order does not matter).

(37)
$$\begin{array}{cc|cccc}
\text{Ident[Stress]} & \text{Ident[Lenght]} & \text{MainLeft} & \text{MainRight} & \text{NoLong} & \text{WSP} \\
e & \text{W} & \text{W} & \text{L} & \text{L} & e
\end{array}$$

A constraint ranking is *consistent* with an ERC provided it satisfies condition (38); it is consistent with an ERC matrix provided it is consistent with each of its rows. An ERC matrix *entails* another ERC matrix provided every ranking consistent with the former is also consistent with the latter. Thus, $\mathfrak{R}_1 + \mathfrak{R}_2$ entails both $\mathfrak{R}_1$ and $\mathfrak{R}_2$.

(38) At least one constraint which has a W in the ERC considered is ranked above every constraint which instead has an L (constraints which have an *e* play no role in the consistency condition).

Here is the phonological underpinning of these notions. Consider the mapping $(\mathbf{Aa}, \mathbf{Xx})$ of a certain underlying form $\mathbf{Aa}$ to a certain surface form $\mathbf{Xx}$, which is thus construed as the winner. For any other loser candidate $\mathbf{Yy}$, consider the ERC defined as in (39). Stack all the ERCs thus constructed for all loser candidates one on top of the other into an ERC matrix, denoted by $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$. Then, (the OT grammar corresponding to) a ranking maps the underlying form $\mathbf{Aa}$ to the surface form $\mathbf{Xx}$ if and only if that ranking is consistent with the ERC matrix $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ corresponding to that mapping.

(39)
a. A constraint has a W provided it prefers the winner mapping $(\mathbf{Aa}, \mathbf{Xx})$ over the loser mapping $(\mathbf{Aa}, \mathbf{Yy})$, namely it assigns less violations to the former mapping than to the latter.
b. A constraint has an L provided it instead prefers the loser mapping $(\mathbf{Aa}, \mathbf{Yy})$ over the winner mapping $(\mathbf{Aa}, \mathbf{Xx})$, namely it assigns less violations to the former mapping than to the latter.
c. A constraint has an *e* provided it has no preference between the two mappings $(\mathbf{Aa}, \mathbf{Xx})$ and $(\mathbf{Aa}, \mathbf{Yy})$, namely it assigns them the same number of violations.

To illustrate, consider the Paka typology (1)-(5) and focus on the mapping (/páːká/, [páːka]) of the underlying form /páːká/ to the surface form [páːka]. The ERC matrix $\mathfrak{R}$(/páːká/, [páːka]) corresponding to this mapping is provided in (40). It consists of seven ERCs, because there are seven loser candidates. For convenience, each ERC is annotated on the left with the corresponding underlying, winner, and loser form (loser forms are stricken out as a mnemonic). The top ERC in (40) says that the winner mapping (/páːká/, [páːka]) wins over the loser mapping (/páːká/, ~~[paká]~~) provided at least one of the two winner-preferring constraints Ident[length] or MainLeft is ranked above both loser-preferring constraints MainRight and NoLong.

(40)

$$\begin{array}{r|cc|cccc|}
 & \text{Ident[Stress]} & \text{Ident[Lenght]} & \text{MainLeft} & \text{MainRight} & \text{NoLong} & \text{WSP} \\
\text{/páːká/, [páː-ka], }\cancel{\text{[pa-ká]}} & & W & W & L & L & \\
\text{/páːká/, [páː-ka], }\cancel{\text{[pa-káː]}} & & W & W & L & & \\
\text{/páːká/, [páː-ka], }\cancel{\text{[paː-ká]}} & & & W & L & & W \\
\text{/páːká/, [páː-ka], }\cancel{\text{[paː-káː]}} & & W & W & L & W & W \\
\text{/páːká/, [páː-ka], }\cancel{\text{[pá-ka]}} & & W & & & L & \\
\text{/páːká/, [páː-ka], }\cancel{\text{[pá-kaː]}} & & W & & & & W \\
\text{/páːká/, [páː-ka], }\cancel{\text{[páː-kaː]}} & & W & & & W & W
\end{array}$$

Grammatical information was represented in 4.1.2 as a (possibly partial) collection G of mappings $(\mathbf{Aa}, \mathbf{Xx})$ of an underlying form $\mathbf{Aa}$ to a surface form $\mathbf{Xx}$. Each of these mappings can be equivalently represented with the corresponding ERC block $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$. And the grammatical information G can thus be represented with the ERC matrix obtained by stacking all these blocks one of top of the other. Merchant's CPR adopts the architecture (28), with the only difference that the current partial grammatical information is represented not as a partial grammar G but as an ERC matrix $\mathfrak{R}$. This makes the algorithm more flexible in the type of partial grammatical information it maintains: any (partial) grammar G can be represented through the corresponding ERC matrix; while not just any ERC matrix $\mathfrak{R}$ corresponds to some partial collection of mappings. This additional flexibility in the data structure maintained by the algorithm is crucial in order to define the subroutine for extracting ranking information, as explained in the next subsection.

A.2. **The join**

The *join* of some ERC matrices $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$ is any ERC matrix which satisfies the two conditions (41). Any such ERC matrix will be denoted by $J(\mathfrak{R}_1, \ldots, \mathfrak{R}_n)$. Condition (41a) says that the join is weaker than (namely entailed by) each matrix $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$. Condition (41b) says that, among all ERC matrices which are weaker than (namely entailed by) each matrix $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$, the join is a strongest one (namely one which entails all the others). In other words, condition (41a) says that the join captures some of the ranking information shared by the matrices $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$; and condition (41b) says that it does not miss any shared ranking information.

(41) a. Each of the ERC matrices $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$ entails their join;
b. If each of the ERC matrices $\mathfrak{R}_1, \ldots, \mathfrak{R}_n$ entails some ERC matrix $\mathfrak{R}$, their join entails $\mathfrak{R}$ as well.

The following lemma collects some properties of the join which will be used in the rest of this appendix.

Lemma 3. **(I)** *Suppose that one of the ERC matrices $\mathfrak{R}_1, \mathfrak{R}_2, \ldots, \mathfrak{R}_n$ is entailed by each of the others. For concreteness, assume that it is $\mathfrak{R}_1$ which is entailed by each of the other ERC matrices $\mathfrak{R}_2, \ldots, \mathfrak{R}_n$. Then, $\mathfrak{R}_1$ is the join of $\mathfrak{R}_1, \mathfrak{R}_2, \ldots, \mathfrak{R}_n$.* **(II)** *Given the ERC matrices $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3, \ldots, \mathfrak{R}_n$, replace two of them with the ERC matrix obtained by stacking the two matrices one on top of the other. For concreteness, let's replace $\mathfrak{R}_1$ and $\mathfrak{R}_2$ with $\mathfrak{R}_1 + \mathfrak{R}_2$. Then, the join of the $n-1$ ERC matrices $\mathfrak{R}_1 + \mathfrak{R}_2, \mathfrak{R}_3, \ldots, \mathfrak{R}_n$ entails the join of the original n ERC matrices $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3, \ldots, \mathfrak{R}_n$, while the reverse does not hold in general.* **(III)** *Given four ERC matrices $\mathfrak{R}_1, \mathfrak{R}_2, \widehat{\mathfrak{R}}_1, \widehat{\mathfrak{R}}_2$, consider the ERC matrix $J(\mathfrak{R}_1, \mathfrak{R}_2) + J(\widehat{\mathfrak{R}}_1, \widehat{\mathfrak{R}}_2)$ obtained by stacking one on top of the other the join of $\mathfrak{R}_1$ and $\mathfrak{R}_2$ and the join of $\widehat{\mathfrak{R}}_1$ and $\widehat{\mathfrak{R}}_2$. Then, $J(\mathfrak{R}_1, \mathfrak{R}_2) + J(\widehat{\mathfrak{R}}_1, \widehat{\mathfrak{R}}_2)$ coincides with the join of the four ERC matrices $\mathfrak{R}_1 + \widehat{\mathfrak{R}}_1$, $\mathfrak{R}_1 + \widehat{\mathfrak{R}}_2$, $\mathfrak{R}_2 + \widehat{\mathfrak{R}}_1$, $\mathfrak{R}_2 + \widehat{\mathfrak{R}}_2$.*

Proof. An ERC matrix $\mathfrak{J}$ is the join of two ERC matrixes $\mathfrak{R}_1$ and $\mathfrak{R}_2$ if and only if the set of rankings consistent with $\mathfrak{J}$ is the *union* of the two sets of rankings consistent with $\mathfrak{R}_1$ and $\mathfrak{R}_2$. Furthermore, the set of rankings consistent with $\mathfrak{R}_1 + \mathfrak{R}_2$ is the *intersection* of the two sets of rankings consistent with $\mathfrak{R}_1$ and $\mathfrak{R}_2$. Claims (I)-(III) then follow from elementary set-theoretic manipulations on the sets of consistent rankings. □

A.3. **A subroutine for extracting ranking information through the *join***

A.3.1. *Description of the subroutine*

The subroutine (42) takes as input some ranking information in the form of an ERC matrix $\mathfrak{R}$, some lexical information in the form of a (possibly partial) lexicon *Lex*, and a paradigm Π (line 1). And it extends the ranking information encoded in the input ERC matrix $\mathfrak{R}$ by adding to that input ERC matrix some additional ERCs as follows. The subroutine considers a certain meaning combination Mm (line 2). And it reads off the paradigm Π

the surface realization $\mathbf{Xx}$ of that meaning combination Mm (line 3). The subroutine then constructs the set Γ of all underlying concatenations $\mathbf{Aa}$ which are consistent with the lexicon *Lex* as underlying forms for that meaning combination Mm (line 4). This means that, if *Lex* sets a feature for the root M (or the suffix m) to a certain value, then the root morpheme $\mathbf{A}$ (the suffix morpheme $\mathbf{a}$) of each underlying form $\mathbf{Aa}$ in Γ has that feature set to that value. Any of these underlying forms in Γ could be the actual underlying form according to the partial lexical information encoded by *Lex*. In other words, the target grammar could enforce any of the mappings $(\mathbf{Aa}, \mathbf{Xx})$ of any underlying form $\mathbf{Aa}$ in Γ to the surface form $\mathbf{Xx}$. The best the subroutine can do is thus to extract any ranking information shared by all these mappings. That is done by computing the *join* of all ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ corresponding to all mappings $(\mathbf{Aa}, \mathbf{Xx})$ of all underlying forms $\mathbf{Aa}$ in Γ to the surface form $\mathbf{Xx}$ (line 5). The input ERC matrix $\mathfrak{R}$ is updated by adding this join to it (line 6). The ERC matrix thus updated is returned (line 7).

(42)
```
1: Require: an ERC matrix $\mathfrak{R}$, a (possibly partial) lexicon $Lex$, a paradigm $\Pi$
2: Require: a meaning combination $\mathsf{Mm}$ in the paradigm $\Pi$
3: read the surface realization $\mathbf{Xx}$ of the meaning combination $\mathsf{Mm}$ off the paradigm $\Pi$
4: construct the set $\Gamma$ of all concatenations $\mathbf{Aa}$ consistent with $Lex$ as underlying forms for $\mathsf{Mm}$
5: compute the join of the ERC matrices $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ for all underlying forms $\mathbf{Aa}$ in $\Gamma$
6: add this join to the ERC matrix $\mathfrak{R}$
7: Return: the updated ERC matrix $\mathfrak{R}$
```

The join of the ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ computed at line 5 is an ERC-theoretic construct which might very well not correspond — in the sense of (39) — to any mapping of an underlying form to a surface form. For this reason, CPR needs the additional flexibility of representing the current (partial) grammatical/ranking information as an ERC matrix rather than as a (partial) grammar. Subsection B.1 will explain that output-drivenness in effect guarantees that the join computed in line 5 always corresponds to a mapping of a certain underlying form to the surface form $\mathbf{Xx}$ read at line 3.

A.3.2. *Analysis of the subroutine*

The following lemma provides a straightforward correctness result for Merchant's subroutine (42). Note that the proof only uses property (41a) of the join, not property (41b). In other words, the lemma only hinges on the fact that the join extracts ranking information which is shared by all the ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$. Property (41b) is nonetheless crucial, because it ensures that the join does not miss any of this shared ranking information, making Merchant's subroutine (42) optimal. The crux of Merchant's subroutine (42) is its time complexity. Merchant (2008, chapter 4) develops an elegant algorithm for computing the join which can be used to implement line 5. Unfortunately, his algorithm needs to compute the *fusional closure* of each of the input ERC matrices it has to join. Although there are no results (I am aware of) concerning the time complexity of that computation, it is most plausibly intractable in the general case.

Lemma 4. *Merchant's subroutine (42)* preserves consistency*, in the following sense. Assume that the input paradigm* Π *is consistent with some target lexicon and some target ranking, that the input ERC matrix* $\mathfrak{R}$ *is consistent with that target ranking, and that the input (possibly partial) lexicon Lex is consistent with that target lexicon. The updated ERC matrix returned by Merchant's subroutine (42) is also consistent with that target ranking.*

Proof. Let $\mathbf{Aa}$ be the underlying form assigned to the meaning combination Mm by the target lexicon. This underlying form $\mathbf{Aa}$ belongs to the set Γ constructed by the subroutine at line 4, because of the hypothesis that the input lexicon *Lex* is consistent with the target lexicon. Thus, the corresponding ERC block $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ is one of the blocks that the subroutine takes the join of at line 5. Since the target ranking is consistent with this ERC block $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$, then it is also consistent with the join, by property (41a) of the join. Since the target ranking is also consistent with the input ERC matrix $\mathfrak{R}$, it is then consistent with the updated ERC matrix returned by the subroutine. □

A.4. **Extension to contrast pairs**

A.4.1. *Description of the subroutine*

The extension of Merchant's subroutine (42) to contrast pairs is provided in (43). This subroutine takes as input some ranking information in the form of an ERC matrix $\mathfrak{R}$, some lexical information in the form of a (possibly partial) lexicon *Lex*, and a paradigm Π (line 1). The subroutine considers a certain contrast pair (line 2). For concreteness, I am assuming here that the two meaning combinations of the contrast pair share the suffix meaning, namely have the shape Mm and $\widehat{\mathsf{M}}\mathsf{m}$ (the case where they share the root meaning is handled analogously). The subroutine reads off the paradigm Π the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of those two meaning combinations (line 3). Although $\mathbf{x}$ and $\widehat{\mathbf{x}}$ are the surface realizations of the same underlying suffix segment corresponding to the suffix meaning m, they can be different, because of morphophonemic alternations in the surface realization of that underlying suffix triggered by the two different roots M and $\widehat{\mathsf{M}}$. The subroutine then constructs the set Γ of all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of underlying concatenations consistent with the lexicon *Lex* as underlying forms for the meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ and furthermore share the underlying suffix segment $\mathbf{a}$ (line 4). The latter requirement encodes the fact that we are focusing on a pair of meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ which share the suffix meaning m. For each pair $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of underlying concatenations in Γ, the subroutine considers the corresponding ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ and $\mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$, stacks them together into the ERC matrix $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ and constructs the *join* of all ERC matrices thus obtained (line 5). The input ERC matrix $\mathfrak{R}$ is updated by adding this join to it (line 6). The ERC matrix thus updated is then returned (line 7).

(43)
```
1: Require: an ERC matrix $\mathfrak{R}$, a (possibly partial) lexicon $Lex$, a paradigm $\Pi$
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix meaning $\mathsf{m}$
3: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
4: construct the set $\Gamma$ of pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of concatenations sharing the suffix morpheme such that
       • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
       • $\widehat{\mathbf{A}}\mathbf{a}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
5: compute the join of the ERC matrices $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ for all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ in $\Gamma$
6: add this join to the ERC matrix $\mathfrak{R}$
7: Return: the updated ERC matrix $\mathfrak{R}$
```

A subtle point is worth stressing. At line 5, the subroutine considers the join in (44a), not the one in (44b). The difference is that in (44a), we sum together the two ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ and $\mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ into $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ before taking the join. This is indeed the right thing to do, because the join in (44a) is stronger than the one in (44b), by lemma 3-II.

(44) a. $J\big\{\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}}) \,\big|\, (\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a}) \in \Gamma\big\}$

b. $J\big\{\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}),\ \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}}) \,\big|\, (\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a}) \in \Gamma\big\}$

A.4.2. *Analysis of the subroutine*

The correctness lemma 4 trivially extends from the original subroutine (42) for single meaning combinations to the variant (43) for contrast pairs. The inefficiency of the original subroutine (42) for single meaning combinations is of course aggravated in the variant (43) for contrast pairs, as the latter requires taking the join of a larger number of larger ERC matrices. Yet, the subroutine (43) applied to the contrast pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ is able to extract more ranking information than the double application of the original subroutine (42) to the two separate meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$. To see that, suppose that the root meaning M admits a unique underlying form $\mathbf{A}$ consistent with Lex, the root meaning $\widehat{M}$ admits a unique underlying form $\widehat{\mathbf{A}}$, and the suffix meaning m admits only two underlying forms $\mathbf{a}_1$ and $\mathbf{a}_2$. Suppose we first apply the original subroutine (42) to the meaning combination Mm. At line 4, it constructs the set Γ consisting of the two underlying forms $\mathbf{Aa}_1$ and $\mathbf{Aa}_2$. At line 5, it computes the join $J(\mathfrak{R}_1, \mathfrak{R}_2)$ of the two ERC blocks $\mathfrak{R}_1 = \mathfrak{R}(\mathbf{Aa}_1, \mathbf{Xx})$ and $\mathfrak{R}_2 = \mathfrak{R}(\mathbf{Aa}_2, \mathbf{Xx})$. Suppose we next apply in turn the original subroutine (42) to the meaning combination $\widehat{\mathsf{M}}\mathsf{m}$. At line 4, it constructs the set Γ consisting of the two underlying forms $\widehat{\mathbf{A}}\mathbf{a}_1$ and $\widehat{\mathbf{A}}\mathbf{a}_2$. At line 5, it computes the join $J(\widehat{\mathfrak{R}}_1, \widehat{\mathfrak{R}}_2)$ of the two ERC blocks $\widehat{\mathfrak{R}}_1 = \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}_1, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ and $\widehat{\mathfrak{R}}_2 = \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}_2, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$. In the end, the two consecutive applications of the original subroutine (42) to the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ extract the ranking information captured by the sum of the two joins in (45).

(45) $J(\mathfrak{R}_1, \mathfrak{R}_2) + J(\widehat{\mathfrak{R}}_1, \widehat{\mathfrak{R}}_2)$

Lemma 3-III ensures that (45) is equivalent to the join in (46). The four ERC matrices being joined here correspond to all four combinations of underlying forms for the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$. In particular, the learner is also joining together the two ERC blocks $\mathfrak{R}_1 + \widehat{\mathfrak{R}}_2$ and $\mathfrak{R}_2 + \widehat{\mathfrak{R}}_1$, whereby the shared suffix meaning m is assigned a different underlying form in the case of Mm than in the case of $\widehat{\mathsf{M}}\mathsf{m}$. In other words, since the learner is processing the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ separately one at the time, it is blind to the fact that these meaning combinations share the suffix meaning m, which must therefore have the same underlying form in the two cases.

(46) $J(\mathfrak{R}_1 + \widehat{\mathfrak{R}}_1,\ \mathfrak{R}_1 + \widehat{\mathfrak{R}}_2,\ \mathfrak{R}_2 + \widehat{\mathfrak{R}}_1,\ \mathfrak{R}_2 + \widehat{\mathfrak{R}}_2)$

Suppose we instead apply the subroutine (43) to the contrast pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$. At line 4, it constructs the set Γ which consists of the two pairs $(\mathbf{Aa}_1, \widehat{\mathbf{A}}\mathbf{a}_1)$ and $(\mathbf{Aa}_2, \widehat{\mathbf{A}}\mathbf{a}_2)$. At line 5, it computes the join (47) of the two ERC blocks $\mathfrak{R}(\mathbf{Aa}_1, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}_1, \widehat{\mathbf{X}}\widehat{\mathbf{x}}) = \mathfrak{R}_1 + \widehat{\mathfrak{R}}_1$ and $\mathfrak{R}(\mathbf{Aa}_2, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\mathbf{a}_2, \widehat{\mathbf{X}}\widehat{\mathbf{x}}) = \mathfrak{R}_2 + \widehat{\mathfrak{R}}_2$.

(47) $J(\mathfrak{R}_1 + \widehat{\mathfrak{R}}_1,\ \mathfrak{R}_2 + \widehat{\mathfrak{R}}_2)$

The join (47) is stronger than the one in (46), because it joins less matrices (the smaller the number of matrices, the larger the amount of shared information for the join to extract). And that is because the two blocks $\mathfrak{R}_1 + \widehat{\mathfrak{R}}_2$ and $\mathfrak{R}_2 + \widehat{\mathfrak{R}}_1$ have disappeared from the join. That is what we want, as they correspond to the unreasonable assumption that the suffix m has different underlying forms in the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$. In other words, since the learner is processing the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ together as a contrast pair, it is able to take into account the fact that these meaning combinations share the suffix meaning m, which must therefore have the same underlying form in the two cases.

A.5. **A subroutine for extracting lexical information through *inconsistency detection***

A.5.1. *Description of the subroutine*

The subroutine (48) takes as input some lexical information in the form of a partial lexicon *Lex*, some ranking information in the form of a (possibly partial) grammar G (or an ERC matrix $\mathfrak{R}$), and a paradigm Π (line 1). The subroutine then considers a certain meaning combination Mm with some feature unset according to the input partial lexicon *Lex* for either the root M or the suffix m (line 2). The goal of the subroutine is to try to extend the input partial lexicon by setting some of these unset features. To this end, the subroutine reads off the paradigm Π the surface realization $\mathbf{Xx}$ of that meaning combination Mm (line 3). It constructs the set Γ of all underlying concatenations $\mathbf{Aa}$ which are consistent with both the input lexical information *Lex* and the input ranking information captured by the grammar G (or the ERC matrix $\mathfrak{R}$) (line 4). Consistency with the input lexical information means that, if *Lex* sets a feature for the root M (or the suffix m) to a certain value, then the root morpheme $\mathbf{A}$ (the suffix morpheme $\mathbf{a}$) of each underlying form $\mathbf{Aa}$ in Γ has that feature set to that value. Consistency with the input ranking information means that the input grammar G (or the input ERC matrix $\mathfrak{R}$) is consistent with the mapping $(\mathbf{Aa}, \mathbf{Xx})$ of each such underlying form $\mathbf{Aa}$ in Γ to the surface form $\mathbf{Xx}$. The subroutine then looks for a feature which is unset by *Lex* for the root meaning M (for the suffix meaning m) and such that each root morpheme $\mathbf{A}$ (each suffix morpheme $\mathbf{a}$) has the same value for that feature in all underlying concatenations $\mathbf{Aa}$ in Γ (line 5). For any such feature, the opposite value can be concluded to be *inconsistent* with the available lexical and ranking information. The subroutine thus updates the partial lexicon *Lex* by setting that feature equal to its constant value for the root meaning M (for the suffix meaning m) (line 6). The lexicon thus updated is then returned (line 8).

(48)
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$ (or an ERC matrix $\mathfrak{R}$), a paradigm $\Pi$
2: Require: a meaning combination $\mathsf{Mm}$ in the paradigm $\Pi$
3: read the surface realization $\mathbf{Xx}$ of the meaning combination $\mathsf{Mm}$ off the paradigm $\Pi$
4: construct the set $\Gamma$ of all the underlying concatenations $\mathbf{Aa}$ such that
       • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
       • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$ (or the ERC matrix $\mathfrak{R}$)
5: if some feature unset for the root $\mathsf{M}$ (for the suffix $\mathsf{m}$) has a constant value over $\Gamma$ for $\mathsf{M}$ (for $\mathsf{m}$) then
6:     update the lexicon $Lex$ by setting that feature to that value for the root $\mathsf{M}$ (for the suffix $\mathsf{m}$)
7: end if
8: Return: the updated lexicon $Lex$
```

This subroutine is called *inconsistency detection* (Kager 1999; Tesar et al. 2003; Tesar 2006), because it sets a certain feature to a certain value based on the detection at line 6 of the fact that any other value for that feature would be inconsistent with the available lexical and ranking information.

A.5.2. *Analysis of the subroutine*

The following lemma provides a straightforward correctness result for inconsistency detection (48). Its time complexity is controlled by the size of the set Γ of consistent underlying forms constructed at line 4. In the worst case (when both the input lexicon and the input ranking information are empty), Γ consists of all underlying forms, namely all concatenations of root and suffix morphemes. In this case, the size of Γ grows polynomially in the number of morphemes and thus in turn exponentially in the number of features. In conclusion, the subroutine (48) is efficient relative to the generous notion of efficiency defined in (14), which measures time complexity in terms of the number of morphemes; but is not efficient relative to the more demanding notion of efficiency defined in (15), which measures time complexity in terms of the number of features.

Lemma 5. *Inconsistency detection (48)* preserves consistency*, in the following sense. Assume that the input paradigm Π is consistent with some target lexicon and some target grammar, that the input (possibly partial) grammar G (or ERC matrix $\mathfrak{R}$) is consistent with that target grammar, and that the input partial lexicon Lex is consistent with that target lexicon. The updated lexicon returned by Merchant's subroutine (48) is also consistent with that target lexicon.*

Proof. Assume by contradiction that the lemma is false. This means that the subroutine has set some feature φ for, say, the root meaning M to a value which is different from the value assigned by the target lexicon. For concreteness, assume that the value assigned by the subroutine is $+$ and that the value assigned by the target lexicon is $-$. Thus in particular, the target lexicon assigns to the meaning combination Mm an underlying form $\mathbf{Aa}$ which has feature φ set to the value $-$ in the root $\mathbf{A}$. This underlying form $\mathbf{Aa}$ is consistent with the partial lexicon *Lex* (because it is consistent with the target lexicon which is in turn consistent with *Lex*). Furthermore, the mapping $(\mathbf{Aa}, \mathbf{Xx})$ of this underlying form $\mathbf{Aa}$ to the surface form $\mathbf{Xx}$ is consistent with the input grammar G (because the target ranking is consistent with both). In other words, this underlying form $\mathbf{Aa}$ satisfies the two conditions stated by the two bullets in line 4 and thus belongs to the set Γ constructed by the subroutine. Since this underlying form $\mathbf{Aa}$ has feature φ set equal to $-$ for the root $\mathbf{A}$, then the subroutine cannot have possibly set that feature to the opposite value $+$ at line 6. □

A.6. **Extension to contrast pairs**

A.6.1. *Description of the subroutine*

The extension of Inconsistency detection (48) to contrast pairs is provided in (49). This subroutine takes as input some lexical information in the form of a partial lexicon *Lex*, some ranking information in the form of a (possibly partial) grammar G (or an ERC matrix $\mathfrak{R}$), and a paradigm Π (line 1). The subroutine considers a certain contrast pair (line 2). For concreteness, I am assuming here that the two meaning combinations of the contrast pair share the suffix meaning, namely have the shape Mm and $\widehat{\mathsf{M}}\mathsf{m}$ (the case where they share the root meaning is handled analogously). The subroutine reads off the paradigm Π the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of those two meaning combinations (line 3). It constructs the set Γ of all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of underlying concatenations which are consistent with both the input lexical information *Lex* and the input ranking information G (or $\mathfrak{R}$) and furthermore share the underlying suffix segment $\mathbf{a}$ (line 4). The subroutine then looks for a feature which is unset by *Lex* for the root meaning M (or the root meaning $\widehat{\mathsf{M}}$ or the suffix meaning m) such that each underlying root morpheme $\mathbf{A}$ (the underlying root morpheme $\widehat{\mathbf{A}}$ or the underlying suffix morpheme $\mathbf{a}$) has the same value for that feature in all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ in Γ (line 5). For any such feature, the the subroutine updates the partial lexicon *Lex* by setting that feature equal to its constant value for the root meaning M (for the root meaning $\widehat{\mathsf{M}}$ or for the suffix meaning m) (line 6). The lexicon thus updated is then returned (line 8).

(49)
```
1: **Require:** a partial lexicon $Lex$, a (possibly partial) grammar $G$ (or an ERC matrix $\mathfrak{R}$), a paradigm $\Pi$
2: **Require:** a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix meaning $\mathsf{m}$
3: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
4: construct the set $\Gamma$ of pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of concatenations sharing the suffix morpheme such that:
       • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
       • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$
       • $\widehat{\mathbf{A}}\mathbf{a}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
       • the mapping $(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ is consistent with the grammar $G$
5: **if** some unset feature has a constant value over $\Gamma$ for any of the three meanings $\mathsf{M}$, $\widehat{\mathsf{M}}$ or $\mathsf{m}$ **then**
6:     update the lexicon $Lex$ by setting that feature to that constant value for that meaning
7: **end if**
8: **Return:** the updated lexicon $Lex$
```

A.6.2. *Analysis of the subroutine*

The correctness lemma 5 trivially extends from the original subroutine (48) for single meaning combinations to the variant (49) for contrast pairs. Furthermore, the variant (49) for contrast pairs remains efficient as the original version (48) for single meaning combinations, when efficiency is generously measured in terms of the number of morphemes (again, not when it is measured in terms of the number of features). Yet, a number of authors (including Alderete et al. 2005 and Merchant and Tesar 2008) have pointed out that the subroutine (49) applied to the contrast pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ can extract more lexical information than the double application of the original subroutine (48) to the two separate meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$. To see that, focus on one of the two meaning combinations of the contrast pair, say Mm. The original subroutine (48) considers all underlying forms $\mathbf{Aa}$ for Mm consistent with the input lexical and ranking information and only sets the features which are constant across all these underlying forms. If the number of consistent underlying forms is large, the number of features which are constant across them is small, and the subroutine thus extracts only little lexical information. The subroutine (49) might instead be able to extract more lexical information, because it looks for features which are constant over a smaller set of underlying forms for Mm, namely those underlying forms $\mathbf{Aa}$ which are not only consistent with the input lexical and ranking information but also use a suffix morpheme $\mathbf{a}$ which can be combined with a root morpheme $\widehat{\mathbf{A}}$ into an underlying form $\widehat{\mathbf{A}}\mathbf{a}$ for $\widehat{\mathsf{M}}\mathsf{m}$ which is also consistent with the input lexical and ranking information.

Appendix B. Equivalence between Merchant's and Tesar's subroutines

B.1. **Subroutines for extracting ranking information from a single meaning combination**

Let me denote Tesar's subroutine (31) for the extraction of ranking information from a single meaning combination by $\mathrm{ERI}_{\mathrm{Tesar}}$; let me denote Merchant's subroutine (42) by $\mathrm{ERI}_{\mathrm{Merchant}}$. The following lemma says that, if the underlying typology is output-driven, then $\mathrm{ERI}_{\mathrm{Tesar}}$ and $\mathrm{ERI}_{\mathrm{Merchant}}$ are equivalent because they update the current ranking/grammatical information with the "same" information. This equivalence is striking, given that the time complexity of $\mathrm{ERI}_{\mathrm{Tesar}}$ is only linear in the number of features while the time complexity of $\mathrm{ERI}_{\mathrm{Merchant}}$ is unknown but probably exponential in both the number of features and the number of morphemes. Output-drivenness has thus afforded a significant reduction of time complexity. The correctness lemma 1 for $\mathrm{ERI}_{\mathrm{Tesar}}$ (which was proven directly in subsection 4.4) can also be made to follow from the correctness lemma 4 for $\mathrm{ERI}_{\mathrm{Merchant}}$ through the equivalence lemma 6.

Lemma 6. *Suppose that the underlying typology is output-driven. Suppose that* $\mathrm{ERI}_{\mathrm{Tesar}}$ *is fed with a partial grammar* G *and a (partial) lexicon Lex. Suppose that* $\mathrm{ERI}_{\mathrm{Merchant}}$ *is fed with the ERC matrix* $\mathfrak{R}$ *corresponding to the partial grammar* G *and the same lexicon Lex. Then, the ERC block corresponding to the mapping* $(\mathbf{Bb}, \mathbf{Xx})$

constructed by ERI$_{\text{Tesar}}$ *at line 4 is the join of the ERC blocks* $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ *corresponding to the underlying forms* $\mathbf{Aa}$ *in* Γ *constructed by ERI*$_{\text{Merchant}}$ *at line 5.*

Proof. The underlying form $\mathbf{Bb}$ constructed by ERI$_{\text{Tesar}}$ at line 4 is consistent with Lex and therefore belongs to the set Γ constructed by ERI$_{\text{Merchant}}$ at line 4. Consider one of the underlying forms $\mathbf{Aa}$ in this set Γ. The two underlying forms $\mathbf{Aa}$ and $\mathbf{Bb}$ agree on every feature which is set by the input partial lexicon Lex. Furthermore $\mathbf{Bb}$ and the surface form $\mathbf{Xx}$ agree on every remaining feature. Thus, the underlying form $\mathbf{Bb}$ is more similar to the surface form $\mathbf{Xx}$ than the underlying form $\mathbf{Aa}$ is. Since the underlying typology is output-driven, then every ranking which maps the less similar underlying form $\mathbf{Aa}$ to $\mathbf{Xx}$, also maps the more similar underlying form $\mathbf{Bb}$ to $\mathbf{Xx}$. In other words, any ranking consistent with the ERC block $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ is also consistent with the ERC block $\mathfrak{R}(\mathbf{Bb}, \mathbf{Xx})$. Lemma 3-I thus ensures that the ERC block $\mathfrak{R}(\mathbf{Bb}, \mathbf{Xx})$ is the join of the ERC blocks $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx})$ across all underlying concatenations $\mathbf{Aa}$ in Γ. □

B.2. **Subroutines for extracting lexical information from a single meaning combination**

Consider the slight variant (50) of the subroutine (48) for the extraction of lexical information through inconsistency detection. The only difference between the two implementations is the following: Merchant's original subroutine (48) sets any feature which is constant over Γ; the variant (50) instead focuses right from the beginning on a specify feature φ which is unset for either the root M or the suffix m of the meaning combination Mm considered. In line 3, I am assuming for concreteness that the feature φ is unset for the root M (the case where it is unset for the suffix m is handled analogously). The modification is only cosmetic: the original subroutine (48) is equivalent to running this variant (50) for each unset feature.

(50)
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a meaning combination $\mathsf{Mm}$ in the paradigm $\Pi$
3: Require: a feature $\varphi$ unset by $Lex$ for either $\mathsf{M}$ or $\mathsf{m}$ — for concreteness, say it is unset for $\mathsf{M}$
4: read the surface realization $\mathbf{Xx}$ of the meaning combination $\mathsf{Mm}$ off the paradigm $\Pi$
5: construct the set $\Gamma$ of all the underlying concatenations $\mathbf{Aa}$ such that
     • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
     • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$
6: if all underlying forms $\mathbf{Aa}$ in $\Gamma$ have the feature $\varphi$ set to the same value in the root morpheme $\mathbf{A}$ then
7:     update the lexicon $Lex$ by setting the feature $\varphi$ for the root meaning $\mathsf{M}$ equal to that value
8: end if
9: Return: the updated lexicon $Lex$
```

Let me denote Tesar's subroutine (35) for the extraction of lexical information from a single meaning combination by ELI$_{\text{Tesar}}$; let me denote (the cosmetic variant of) the original inconsistency detection subroutine (50) by ELI$_{\text{inc-det}}$. The following lemma says that, if the underlying typology is output-driven, then ELI$_{\text{Tesar}}$ and ELI$_{\text{inc-det}}$ are equivalent because their if-clauses at line 6 are equivalent. This equivalence is striking, given that the if-clause of ELI$_{\text{Tesar}}$ can be checked in time linear in the number of features while the if-clause of ELI$_{\text{inc-det}}$ requires time exponential in the number of features in the worst case. Output-drivenness has thus afforded a significant reduction of time complexity. The correctness lemma 2 for ELI$_{\text{Tesar}}$ (which was proven directly in subsection 4.5) can also be made to follow from the correctness lemma 4 for ELI$_{\text{inc-det}}$ through the equivalence lemma 7.

Lemma 7. *Suppose that the underlying typology is output-driven. Then, the two if-clauses in line 6 ELI*$_{\text{Tesar}}$ *and ELI*$_{\text{inc-det}}$ *are equivalent.*

Proof. To start, let me prove that the if-clause in line 6 of ELI$_{\text{Tesar}}$ entails the if-clause in line 6 of ELI$_{\text{inc-det}}$. Assume by contradiction that the former if-clause is true while the latter if-clause is false. The hypothesis that the if-clause of ELI$_{\text{inc-det}}$ is false entails in particular that the set Γ of consistent underlying forms constructed by ELI$_{\text{inc-det}}$ at line 5 contains an underlying form $\mathbf{Aa}$ whose root $\mathbf{A}$ disagrees with $\mathbf{X}$ relative to the feature φ (indeed, if all the underlying forms $\mathbf{Aa}$ in Γ had a root $\mathbf{A}$ which agreed with $\mathbf{X}$ relative to φ, then the if-clause of ELI$_{\text{inc-det}}$ would hold true). This underlying form $\mathbf{Aa}$ and the underlying form $\mathbf{Bb}$ constructed by ELI$_{\text{Tesar}}$ at line 5 thus agree on the feature φ for the root meaning M, as they both have the opposite value than $\mathbf{X}$. Furthermore, they agree on every feature set by the partial lexicon Lex, because they are both consistent with Lex. Finally, $\mathbf{Bb}$ agrees with $\mathbf{Xx}$ for every remaining feature. The underlying form $\mathbf{Bb}$ is thus more similar to the surface form $\mathbf{Xx}$ than the underlying form $\mathbf{Aa}$ is. Since the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with G (because $\mathbf{Aa}$ belongs to Γ), then there exists a ranking which is consistent with both. Since the grammar corresponding to that ranking is output driven and it maps the less similar underlying form $\mathbf{Aa}$ to $\mathbf{Xx}$, then it also maps the more similar underlying form $\mathbf{Bb}$ to $\mathbf{Xx}$. In other words, that ranking is also consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$. In conclusion, the mapping $(\mathbf{Bb}, \mathbf{Xx})$ and the grammar G are consistent (because both consistent with that ranking), contradicting the hypothesis that the if-clause of ELI$_{\text{Tesar}}$ is true.

Vice versa, let me prove that the if-clause in line 6 of ELI$_{\text{inc-det}}$ entails the if-clause in line 6 of ELI$_{\text{Tesar}}$. Assume by contradiction that the former if-clause is true while the latter if-clause is false. The hypothesis that the if-clause of ELI$_{\text{Tesar}}$ is false means that the input (partial) grammar G is consistent with the mapping $(\mathbf{Bb}, \mathbf{Xx})$ of the underlying form $\mathbf{Bb}$ constructed by ELI$_{\text{Tesar}}$ at line 5 to the surface form $\mathbf{Xx}$. Since this underlying form $\mathbf{Bb}$ is consistent with Lex, then $\mathbf{Bb}$ belongs to the set Γ of consistent underlying forms constructed at line 5 by

$\mathrm{ELI}_{\mathrm{inc-det}}$. Since $\mathbf{B}$ and $\mathbf{X}$ disagree relative to the feature φ, consider the underlying form $\overline{\mathbf{B}}\mathbf{b}$ identical to $\mathbf{Bb}$ but for the fact that $\overline{\mathbf{B}}$ and $\mathbf{X}$ instead agree relative to the feature φ. Obviously, the underlying form $\overline{\mathbf{B}}\mathbf{b}$ is more similar to the surface form $\mathbf{Xx}$ than the underlying form $\mathbf{Bb}$ is. Since $(\mathbf{Bb}, \mathbf{Xx})$ is consistent with G, then there exists a ranking which is consistent with both. Since the grammar corresponding to that ranking is output driven and it maps the less similar underlying form $\mathbf{Bb}$ to $\mathbf{Xx}$, then it also maps the more similar underlying form $\overline{\mathbf{B}}\mathbf{b}$ to $\mathbf{Xx}$. In other words, that ranking is also consistent with the mapping $(\overline{\mathbf{B}}\mathbf{b}, \mathbf{Xx})$. This underlying form $\overline{\mathbf{B}}\mathbf{b}$ thus belongs to the set Γ constructed by $\mathrm{ELI}_{\mathrm{inc-det}}$ at line 5.This conclusion contradicts the hypothesis that the if-clause of $\mathrm{ELI}_{\mathrm{inc-det}}$ holds, because it shows that Γ contains two underlying forms $\mathbf{Bb}$ and $\overline{\mathbf{B}}\mathbf{b}$ such that $\mathbf{B}$ and $\overline{\mathbf{B}}$ disagree relative to the feature φ. □

B.3. Subroutines for extracting lexical information from a contrast pair

Tesar's subroutine for extracting lexical information from contrast pairs is more involved than his other two subroutines for extracting lexical and ranking information from a single meaning combination. For this reason, the latter to subroutines have been reviewed in subsections 4.4 and 4.5, while the former subroutine is relegated to this appendix. Here, I start from Merchant's subroutine (49) for extracting lexical information from contrast pairs. And I slowly build my way towards Tesar's reformulation, provided below in (54). Indeed, this seems to me the best way of making sense of the somewhat complex formulation of Tesar's subroutine. To start, I perform the same cosmetic change done above in appendix B.2: I replace the original subroutine (49) with (51), which only differs because line 3 focuses on a specific feature φ unset by the input lexicon *Lex* for one of the three meanings M, $\widehat{\mathsf{M}}$, or m of the contrast pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$. For concreteness, I assume it is unset for the root meaning M.

(51)
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix $\mathsf{m}$
3: Require: a feature $\varphi$ unset by $Lex$ for either $\mathsf{M}$, $\widehat{\mathsf{M}}$, or $\mathsf{m}$ — for concreteness, say it is unset for $\mathsf{M}$
4: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
5: construct the set $\Gamma$ of pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ of concatenations sharing the suffix morpheme such that:
      • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
      • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$
      • $\widehat{\mathbf{A}}\mathbf{a}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
      • the mapping $(\widehat{\mathbf{A}}\mathbf{a}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ is consistent with the grammar $G$
6: if all the pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\mathbf{a})$ in $\Gamma$ have the underlying root $\mathbf{A}$ set to the same value for feature $\varphi$ then
7:     update the lexicon $Lex$ by setting feature $\varphi$ to that constant value for the meaning $\mathsf{M}$
8: end if
9: Return: the updated lexicon $Lex$
```

The modification is only cosmetic: the original subroutine (49) is equivalent to running this variant (51) for each feature which is unset for one of the three morphemes M, $\widehat{\mathsf{M}}$, or m.

The set Γ constructed by the subroutine (51) at line 5 can equivalently be described as the collection of all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ which satisfy the conditions expressed by the four bullets *plus* the additional condition that the two underlying suffix morphemes $\mathbf{a}$ and $\widehat{\mathbf{a}}$ coincide, because they are the underlying form corresponding to the same suffix meaning m. The condition that $\mathbf{a}$ and $\widehat{\mathbf{a}}$ coincide is in turn equivalent to the condition that they agree for each feature ψ. We could of course loosen up this condition and require the two underlying suffix morphemes $\mathbf{a}$ and $\widehat{\mathbf{a}}$ to agree only for certain features $\psi_1, \ldots, \psi_n$. This intuition leads to the formulation (52). It differs from (51) only because of the new line 4 which introduces the features $\psi_1, \ldots, \psi_n$ and because of the additional fifth bullet in the definition of the set Γ. Note that line 4 assumes the features $\psi_1, \ldots, \psi_n$ to be all unset by the input partial lexicon *Lex* for the shared suffix m because the condition that both $\mathbf{a}$ and $\widehat{\mathbf{a}}$ are consistent with *Lex* as underlying forms for m already entails that they agree on every feature set by *Lex* for m.

(52) Inconsistency detection from a contrast pairs ($\mathrm{ID}_{\mathrm{cp}}$):
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix $\mathsf{m}$
3: Require: a feature $\varphi$ unset by $Lex$ for either $\mathsf{M}$, $\widehat{\mathsf{M}}$, or $\mathsf{m}$ — for concreteness, say it is unset for $\mathsf{M}$
4: Require: some features $\psi_1, \ldots, \psi_n$ all unset for the shared suffix meaning $\mathsf{m}$ according to $Lex$
5: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
6: construct the set $\Gamma$ of pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ of concatenations such that:
      • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
      • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$
      • $\widehat{\mathbf{A}}\widehat{\mathbf{a}}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
      • the mapping $(\widehat{\mathbf{A}}\widehat{\mathbf{a}}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ is consistent with the grammar $G$
      • the two underlying suffix morphemes $\mathbf{a}$ and $\widehat{\mathbf{a}}$ agree for every feature $\psi_1, \ldots, \psi_n$
7: if all the pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ in $\Gamma$ have the underlying root $\mathbf{A}$ set to the same value for feature $\varphi$ then
8:     update the lexicon $Lex$ by setting feature $\varphi$ to that constant value for the meaning $\mathsf{M}$
9: end if
```

```
10: Return: the updated lexicon $Lex$
```

If we consider *no* features ψ's at all in line 4, then (52) effectively considers the two meaning combinations of the contrast pair independently of each other and thus ends up equivalent to the subroutine (48) for extracting lexical information from single meaning combinations. In the sense that the application of the subroutine (52) to the contrast pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ is equivalent to two consecutive applications of the subroutine (48) to the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ one at the time. If instead we consider all unset features ψ's at line 4, then (52) effectively only considers pairs of underlying forms for the two meaning combinations which share the underlying suffix and is therefore equivalent to the original subroutine (51) for extracting lexical information from contrast pairs. In the end, depending on the number of features ψ's considered at line 4, (52) provides a whole range of subroutines for extracting lexical information in between the weaker (48) and the stronger (51).

The subroutine (52) can be equivalently restated as in (53). Instead of requiring the two suffix morphemes **a** and $\widehat{\mathbf{a}}$ to coincide for every feature $\psi_1, \ldots, \psi_n$, we consider all possible combinations $v_1, \ldots, v_n$ of values of those features at line 6. And we require both **a** and $\widehat{\mathbf{a}}$ to have those features set equal to those values in the fifth bullet.

(53)
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix $\mathsf{m}$
3: Require: a feature $\varphi$ unset by $Lex$ for either $\mathsf{M}$, $\widehat{\mathsf{M}}$, or $\mathsf{m}$ — for concreteness, say it is unset for $\mathsf{M}$
4: Require: some features $\psi_1, \ldots, \psi_n$ all unset for the shared suffix meaning $\mathsf{m}$ according to $Lex$
5: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
6: For any combination $v_1, \ldots, v_n$ of values of the features $\psi_1, \ldots, \psi_n$, construct the set $\Gamma(v_1, \ldots, v_n)$ of
   pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ of underlying concatenations such that:
     • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
     • the mapping $(\mathbf{Aa}, \mathbf{Xx})$ is consistent with the grammar $G$
     • $\widehat{\mathbf{A}}\widehat{\mathbf{a}}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
     • the mapping $(\widehat{\mathbf{A}}\widehat{\mathbf{a}}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ is consistent with the grammar $G$
     • the two underlying suffix morphemes $\mathbf{a}$ and $\widehat{\mathbf{a}}$ both have the values $v_1, \ldots, v_n$ for features $\psi_1, \ldots, \psi_n$
7: if all the pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ in all the sets $\Gamma(v_1, \ldots, v_n)$ have the underlying root $\mathbf{A}$ set to the same value
   for feature $\varphi$ then
8:     update the lexicon $Lex$ by setting feature $\varphi$ to that constant value for the meaning $\mathsf{M}$
9: end if
10: Return: the updated lexicon $Lex$
```

By reasoning as above in appendix B.2, the subroutine (53) is easily shown to be equivalent to (54) under the assumption of output-drivenness. Indeed, the two underlying concatenations $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ constructed by (54) at line 6 provide a "summary" of the set $\Gamma(v_1, \ldots, v_n)$ of pairs of underlying concatenations constructed by (53).

(54)
```
1: Require: a partial lexicon $Lex$, a (possibly partial) grammar $G$, a paradigm $\Pi$
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix $\mathsf{m}$
3: Require: a feature $\varphi$ unset by $Lex$ for either $\mathsf{M}$, $\widehat{\mathsf{M}}$, or $\mathsf{m}$ — for concreteness, say it is unset for $\mathsf{M}$
4: Require: some features $\psi_1, \ldots, \psi_n$ all unset for the shared suffix meaning $\mathsf{m}$ according to $Lex$
5: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
6: For any combination $v_1, \ldots, v_n$ of values of the features $\psi_1, \ldots, \psi_n$, construct the two underlying
   concatenations $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ as follows:
     • $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ are consistent with $Lex$ as underlying forms for $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$
     • $\mathbf{B}$ has the opposite value for feature $\varphi$ than $\mathbf{X}$
     • $\mathbf{b}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ have features $\psi_1, \ldots, \psi_n$ set to the values $v_1, \ldots, v_n$
     • $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ are otherwise identical to $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$
7: if the the grammar $G$ is inconsistent with the two mappings $(\mathbf{Bb}_{v_1,\ldots,v_n}, \mathbf{Xx})$ and $(\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ for
   any combination of values $v_1, \ldots, v_n$ then
8:     update the lexicon $Lex$ by setting feature $\varphi$ for the meaning $\mathsf{M}$ equal to the value it has in $\mathbf{X}$
9: end if
10: Return: the updated lexicon $Lex$
```

If the features $\psi_1, \ldots, \psi_n$ considered in line 4 of (54) exhaust the set of features, then the two subroutines (51) and (54) are equivalent. Yet, the running time of (54) is exponential in the number of features in the latter case, because it needs to consider all combinations $v_1, \ldots, v_n$ of values.

B.4. What about the subroutines for extracting ranking information from a contrast pair?

By reasoning as in appendix B.3, I can go from Merchant's original subroutine (43) for extracting ranking information from contrast pairs to the variant in (55).

(55)
```
1: Require: an ERC matrix $\mathfrak{R}$, a (possibly partial) lexicon $Lex$, a paradigm $\Pi$
```

```
2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix meaning $\mathsf{m}$
3: Require: some features $\psi_1, \ldots, \psi_n$ all unset for the shared suffix meaning $\mathsf{m}$ according to $Lex$
4: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
5: For any combination $v_1, \ldots, v_n$ of values of the features $\psi_1, \ldots, \psi_n$, construct the set $\Gamma(v_1, \ldots, v_n)$ of
   pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ of underlying concatenations such that:
     • $\mathbf{Aa}$ is consistent with the lexicon $Lex$ as an underlying form for $\mathsf{Mm}$
     • $\widehat{\mathbf{A}}\widehat{\mathbf{a}}$ is consistent with the lexicon $Lex$ as an underlying form for $\widehat{\mathsf{M}}\mathsf{m}$
     • the two underlying suffix morphemes $\mathbf{a}$ and $\widehat{\mathbf{a}}$ both have the values $v_1, \ldots, v_n$ for features $\psi_1, \ldots, \psi_n$
6: compute the join of the ERC matrices $\mathfrak{R}(\mathbf{Aa}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{A}}\widehat{\mathbf{a}}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ for all pairs $(\mathbf{Aa}, \widehat{\mathbf{A}}\widehat{\mathbf{a}})$ in $\Gamma(v_1, \ldots, v_n)$
   for all combinations of values $v_1, \ldots, v_n$
7: add this join to the ERC matrix $\mathfrak{R}$
8: Return: the updated ERC matrix $\mathfrak{R}$
```

Again, if we consider no features ψ's at line 3, then (55) effectively considers the two meaning combinations of the contrast pair independently of each other and thus ends up equivalent to the subroutine (42) for extracting lexical information from single meaning combinations. If instead we consider all unset features ψ's at line 3, then (55) effectively only considers pairs of underlying forms for the two meaning combinations which share the underlying suffix and is therefore equivalent to Merchant's original subroutine (43) for extracting lexical information from contrast pairs. In the end, depending on the number of features ψ's considered at line 3, (55) provides a whole range of subroutines for extracting ranking information in between the weaker (42) and the stronger (43).

By reasoning as above in appendix B.2, the subroutine (55) is easily shown to be equivalent to (56) under the assumption of output-drivenness. Again, the two underlying concatenations $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ constructed by (56) at line 5 provide a "summary" of the set $\Gamma(v_1, \ldots, v_n)$ of pairs of underlying concatenations constructed by (55).

```
(56) 1: Require: an ERC matrix $\mathfrak{R}$, a (possibly partial) lexicon $Lex$, a paradigm $\Pi$
     2: Require: a pair $(\mathsf{Mm}, \widehat{\mathsf{M}}\mathsf{m})$ of meaning concatenations which share, say, the suffix meaning $\mathsf{m}$
     3: Require: some features $\psi_1, \ldots, \psi_n$ all unset for the shared suffix meaning $\mathsf{m}$ according to $Lex$
     4: read the surface realizations $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$ of $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$ off the paradigm $\Pi$
     5: For any combination $v_1, \ldots, v_n$ of values of the features $\psi_1, \ldots, \psi_n$, construct the two underlying
        concatenations $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ as follows:
          • $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ are consistent with $Lex$ as underlying forms for $\mathsf{Mm}$ and $\widehat{\mathsf{M}}\mathsf{m}$
          • $\mathbf{b}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ have features $\psi_1, \ldots, \psi_n$ set to the values $v_1, \ldots, v_n$
          • $\mathbf{Bb}_{v_1,\ldots,v_n}$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}$ are otherwise identical to $\mathbf{Xx}$ and $\widehat{\mathbf{X}}\widehat{\mathbf{x}}$
     6: compute the join of the ERC matrices $\mathfrak{R}(\mathbf{Bb}_{v_1,\ldots,v_n}, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{B}}\widehat{\mathbf{b}}_{v_1,\ldots,v_n}, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ for all combinations of
        values $v_1, \ldots, v_n$
     7: add this join to the ERC matrix $\mathfrak{R}$
     8: Return: the updated ERC matrix $\mathfrak{R}$
```

Let's focus on the simplest case, where we consider a single feature ψ at line 3. Assume furthermore that ψ is binary, namely ties the two values $v = +$ and $v = -$. At line 5, the subroutine constructs the two underlying forms $\mathbf{Bb}_+$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_+$ corresponding to the value $v = +$ of feature ψ (namely, both $\mathbf{b}_+$ and $\widehat{\mathbf{b}}_+$ have value $+$ for the feature ψ) and the two underlying forms $\mathbf{Bb}_-$ and $\widehat{\mathbf{B}}\widehat{\mathbf{b}}_-$ corresponding to the value $v = -$ (namely, both $\mathbf{b}_-$ and $\widehat{\mathbf{b}}_-$ have value $-$ for the feature ψ). At line 6, we finally compute the join of the two ERC blocks $\mathfrak{R}(\mathbf{Bb}_+, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{B}}\widehat{\mathbf{b}}_+, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$ and $\mathfrak{R}(\mathbf{Bb}_-, \mathbf{Xx}) + \mathfrak{R}(\widehat{\mathbf{B}}\widehat{\mathbf{b}}_-, \widehat{\mathbf{X}}\widehat{\mathbf{x}})$. Even in the simplest case of a single feature ψ, there is no way around computing the join.

Let me take stock. Merchant's develops a new subroutine for extracting ranking information through the notion of join. This subroutine can be applied to either a single meaning combination at the time, as in (42); or to a contrast pair, as in (43). The latter is more powerful than the former: the subroutine extracts more ranking information by processing the two meaning combinations Mm and $\widehat{\mathsf{M}}\mathsf{m}$ as a contrast pair than by processing them separately one at the time. The crux of Merchant's subroutine for extracting ranking information is that it relies on the computation of a join, which is time consuming. The biggest impact of output-drivenness is that it allows Merchant's subroutine for extracting ranking information from a single meaning combination to be restated without having to compute any join. The restated subroutine thus runs in time linear in the number of features. In other words, it is efficient even relative to the more demanding notion of efficiency introduced in subsection 2.6. The situation is unfortunately very different when Merchant's subroutine for extracting ranking information is boosted by being applied to contrast pairs. In this case, output-drivenness is not able to circumvent the computation of the join and thus it does not suffice to make the subroutine efficient.