Glossa a journal of general linguistics

Nyman, Alexandra and Bruce Tesar. 2019. Determining underlying presence in the learning of grammars that allow insertion and deletion. *Glossa: a journal of general linguistics* 4(1): 37. 1–41. DOI: https://doi.org/10.5334/gjgl.603

RESEARCH

Determining underlying presence in the learning of grammars that allow insertion and deletion

Alexandra Nyman[1] and Bruce Tesar[2]

[1] University of Massachusetts, Amherst, N408 Integrative Learning Center, 650 North Pleasant Street, Amherst, MA, US

[2] Rutgers University, 18 Seminary Place, New Brunswick, NJ, US

Corresponding author: Alexandra Nyman (anyman@umass.edu)

The simultaneous learning of a phonological map from inputs to outputs and a lexicon of phonological underlying forms has been a focus of several research efforts (Jarosz 2006; Apoussidou 2007; Merchant 2008; Merchant & Tesar 2008; Tesar 2014). One of the numerous challenges is that of computational efficiency, which led to the investigation of learning with output-driven maps (Tesar 2014). Prior work on learning with output-driven maps has focused on systems in which the only disparities between inputs and outputs were segmental identity disparities (differences in the value of a feature). Inclusion of segmental insertion and deletion disparities exacerbates computational concerns, as it increases the number of possible correspondence relations between an input and an output, and makes the space of possible inputs for a word infinite due to the possible presence of an unbounded number of deleted segments. We propose an extension of that earlier work to handle phonologies that permit insertion and deletion, and evaluate the proposal by applying it to cases in Basic CV Syllable Theory (Jakobson 1962; Clements & Keyser 1983; Prince & Smolensky 2004). First, we propose that a learner represent information about the possible presence/absence of a segment in an underlying form via a presence feature. The presence feature can be set using the same inconsistency detection method that has previously been used to set other segmental features. This allows the learner to combine evidence from paradigmatically related words in a single compact representation. Second, we propose that the learner only consider for underlying forms segments that surface in at least one surface realization of the morpheme. This approach is justified by the structure of output-driven maps, and avoids the potential for an unbounded number of possibly deleted segments in an underlying form. A proof is given for the validity of the method for avoiding unbounded deletion. The resulting learner is able to learn some grammatical regularities about segmental insertion and deletion; this is shown via two manual step-by-step applications of the algorithm. Verificatory simulations for learning the entire typology of Basic CV Syllable Theory are left to work in the near future.

Keywords: phonology; learning; Optimality Theory

1 Introduction

The simultaneous learning of a phonological map from inputs to outputs and a lexicon of phonological underlying forms has been a focus of several research efforts (Jarosz 2006; Apoussidou 2007; Merchant 2008; Merchant & Tesar 2008; Tesar 2014).[1] One of the numerous challenges is that of computational efficiency: how can a phonology be learned using only a plausible amount of computational effort? Contributing to this challenge is the vast quantity of possible lexica for even modest collections of underlying forms. This fact led to the investigation of learning with output-driven maps, resulting in the

[1] References containing, at the end, the text "ROA-" are available for download on the Rutgers Optimality Archive, http://roa.rutgers.edu/. The number after "ROA-" indicates the ROA number of the paper.

Output-Driven Learner (Tesar 2014). A key property of output-driven maps is that they impose a kind of formal structure on the space of underlying forms that makes it possible to search the space without needing to explicitly construct and evaluate all, or even most, of the possibilities in the space.

Prior work on the Output-Driven Learner has focused on systems in which the only disparities between inputs and outputs introduced by the phonology were segmental identity disparities (differences in the value of a feature). In particular, no segmental deletion or insertion was considered. The present work proposes an algorithm for learning phonologies that permit insertion and deletion, using an approach that leverages the structure of output-driven maps to contend with the additional challenges raised by insertion and deletion.

Two specific problems raised by insertion/deletion are tackled in this paper. The first is that the surface form for an individual word does not overtly indicate which segments are present underlyingly, and which are inserted, let alone indicate the underlying presence of segments that have been deleted. How does the learner *learn* about inserted and deleted segments? We argue that the integration of a *presence feature* into the underlying representations of segments provides an answer. A learner uses the presence feature to represent their knowledge about the status of hypothesized segments in an underlying form. For each segment of the surface realizations of a morpheme, the learner constructs a corresponding hypothesized underlying segment in the underlying form for the morpheme, with a presence feature that is initially unset. If the learner determines that a hypothesized segment must be present in the underlying form, then that segment's presence feature is set to +presence. If the learner is convinced that a hypothesized segment cannot be present in the underlying form, then that segment's presence feature is set to –presence. An unset presence feature indicates uncertainty on the part of the learner as to whether the bearing segment is in fact part of the underlying form. Ultimately, the addition of presence features to underlying segments allows the learner to combine evidence from paradigmatically related words in a single compact representation.

The second problem tackled in this paper is the potential for an unbounded number of possibly deleted segments in an underlying form. If the number of such segments is unbounded, then it is infeasible to pursue a simplistic strategy like explicitly representing every possibly present segment in an underlying form, and evaluating the presence feature for every possible segment. The structure of output-driven maps makes the solution proposed here possible, by ensuring that a segment can only be necessarily present in an underlying form if it surfaces in at least one surface realization of the morpheme. The learner can thus limit the hypothesized segments for an underlying form to the modestly sized set of segments that have been thus far observed in at least one surface realization (and can add hypothesized underlying segments during the course of learning if and when observation of new surface realizations suggests it).

The presence feature allows the learner to approach insertion and deletion in the same way as other phonological phenomena. The same kinds of evidence and reasoning previously used in the learning of segmental features in underlying forms can be applied to the learning of presence "features", in turn allowing the learner to learn when and where segments are deleted or inserted. It is important to note that the presence feature is not being proposed as a phonological segmental feature in any traditional sense. The presence feature, as proposed here, is strictly a representation constructed and utilized by the learner.

2 Basic CV Syllable Theory

The illustrations given in this paper involve Basic CV Syllable Theory, abbreviated BST (Jakobson 1962; Clements & Keyser 1983; Prince & Smolensky 2004). The version used here derives from the Optimality Theoretic system developed by Prince and Smolensky, while using correspondence-based faithfulness (McCarthy & Prince 1995) rather than containment-based faithfulness, and with the constraints against insertion conditioned simply on the segmental type (C or V) of the inserted segment (rather than being conditioned on what type of syllabic position they are in). This system has the virtues of being familiar and simple. The phonological activity of BST consists largely of insertion and deletion, lending itself to evaluation of the proposals of this paper.

In BST, an input is any string of the symbols C and V (C for consonant, V for vowel). Possible outputs consist of syllabified C's and V's, where a syllable mandatorily has a nucleus containing exactly one V, and may have an onset containing exactly one C, and may have a coda containing exactly one C. No metathesis or multiple correspondence is permitted. Input-Output corresponding segments must be of the same type: a C in the input can only have a C as an output correspondent, etc.

The theory has five constraints, shown in Table 1: two markedness constraints and three faithfulness constraints. Of the faithfulness constraints, MAX is violated by instances of deletion, and DEPC and DEPV are each violated by certain instances of insertion.

BST was the first Optimality Theoretic system to be studied with respect to learning (Tesar & Smolensky 1994; Tesar 1995), but that was in a context in which underlying forms were provided as part of the input, and the learner was only attempting to learn the constraint ranking. Recent work on the simultaneous learning of constraint ranking information and a lexicon of underlying forms has focused on systems that only have identity disparities, where underlying segments may surface non-identically, but there is no insertion or deletion. BST provides a convenient system for studying the complexities of learning with insertion and deletion, while being simple in that it has no issues of identity disparities: an underlying C may only surface identically as a C, and an underlying V may only surface identically as a V. Setting aside identity disparities in this way makes the analysis simpler, but is not essential to the success of the presence feature proposal given in this paper.

3 Output-driven maps

Recent work has demonstrated that the learning of phonologies, in particular the simultaneous learning of constraint rankings and underlying forms, can be greatly facilitated if the learner knows that the phonological input-output map has a property known as output-drivenness. The structure of output-driven maps is here shown to be just as valuable when learning is extended to include insertion and deletion.

Table 1: The constraints of Basic CV Syllable Theory.

Name	Description
ONSET	A syllable should not lack an onset.
NOCODA	A syllable should not have a coda.
MAX	An input segment should not lack an output correspondent.
DEPC	An output C should not lack an input correspondent.
DEPV	An output V should not lack an input correspondent.

3.1 Output-drivenness

Output-drivenness (Tesar 2014) concerns entailment relations between different input-output mappings in a map. Roughly stated, an input-output map is *output-driven* if whenever an input maps to an output, all other inputs with greater "similarity" to that output also map to the same output.

A *phonological map* contains a set of representations, each representation relating an input to an output. A particular input-output representation will be commonly referred to as a *candidate*. It is often convenient to think of a map as a function, with one input-output mapping for each possible input: any given map contains, for each possible input, one of the many candidate input-output representations containing that input. Borrowing a system for illustration from (Tesar 2014) involving stress and vowel length, a possible input in that system is /paká:/, where the first vowel is unstressed (–stress) and short (–long), while the second vowel is stressed (+stress) and long (+long). There are a total of eight candidates containing that input, as shown in (1). Any phonological map for this system would include one of the eight candidates for that input, along with one candidate for each other input.

(1) The candidates for the input /pa:ká/

/paká:/ → [paká]	/paká:/ → [páka]
/paká:/ → [paká:]	/paká:/ → [páka:]
/paká:/ → [pa:ká]	/paká:/ → [pá:ka]
/paká:/ → [pa:ká:]	/paká:/ → [pá:ka:]

While candidates compete when they share the same input, the concept of similarity behind output-driven maps actually concerns the comparison between different candidates, which share the same *output*. Similarity is based on disparities. A *disparity* is a specific difference between the input and the output of a candidate. The overall similarity between an input and an output is expressed as the set of disparities between them. In the stress/length system, the relevant disparities are identity disparities. An *identity disparity* is a difference in the values of a feature for corresponding input and output segments. The corresponding input and output segments are not identical, and the identity disparities identify the ways in which they are not identical (the features on which they differ). The two candidates in (2) illustrate both identity disparities and relative similarity.

(2) Candidate b. has greater similarity than candidate a.

a. /páká/ → [paká:]

b. /paká/ → [paká:]

Candidate (2)b has one disparity: the second vowel of the input is –long [a], while the corresponding second vowel of the output is +long [a:]. Candidate (2)a has two disparities: the same disparity in length between the second vowels, and a disparity in stress between the first vowels (input first vowel is +stress, output first vowel is –stress).

The candidates in (2) have the same output, [paká:]. They differ only in their inputs. Given that, we can relate the two candidates, based on the shared output form, and say that the disparity in length in the second vowels of each of the candidates are identical corresponding disparities. The disparities are identical because each has an input –long vowel corresponding to an output +long vowel. The disparities are corresponding because they involve the same segment of the same output, the second vowel of [paká:]. The other disparity of (2)a (stress on the first vowels) has no corresponding disparity in (2)b. Because every disparity in (2)a has a corresponding disparity in (2)b, (2)a is said to have *greater similarity* than (2)b. Importantly, this *similarity relation* is not simply a matter of

having fewer disparities: the disparities of one candidate must effectively be a subset of the disparities of the other. Every disparity in the greater similarity candidate must have an identical corresponding disparity in the lesser similarity candidate.

Given that candidates must have the same output in order to possibly have a similarity relation between them, the similarity relation can (to a large extent) be thought of as an expression of the relative similarity that two different inputs have to the same output. In (2), the input of (2)b, the greater similarity candidate, is "more similar", or "closer", to the shared output than is the input of (2)a, the lesser similarity candidate. Relative to the input of (2)a, the input of (2)b is a step closer to the output because the first vowel's stress has been changed to match the output, thus eliminating a disparity. Candidates with greater similarity have inputs that are "closer" to the shared output in this relational, subset-like sense of closeness.

The definition of an output-driven map invokes the similarity relation just described. A map is output-driven if, for every candidate contained in the map, every candidate of greater similarity is also part of the map. For the candidates in (2), if candidate (2)a is part of a map, then candidate (2)b must also be part of that map. Put another way, if a given input maps to an output, then all other inputs which are more similar to that output must also map to that same output. If an output-driven map introduces several disparities in mapping an input to its output, then that knowledge entails the output for numerous other inputs (all those with greater similarity to the same output).

The best that a candidate can do with respect to input-output similarity is zero disparities. Thus, if a candidate like /páká/ → [paká:] is included in a map, and the map is output-driven, then it follows that the candidate /paká:/ → [paká:] is also part of the map. Generally speaking, output-driven maps are idempotent: roughly speaking, if something maps to an output, then that output maps to itself.

The complete relative similarity relation for [paká:] is given in Figure 1. Each node of the graph represents a candidate with output [paká:], and the input of the candidate is the text shown inside the node's oval. The figure is oriented so that higher in the graph means

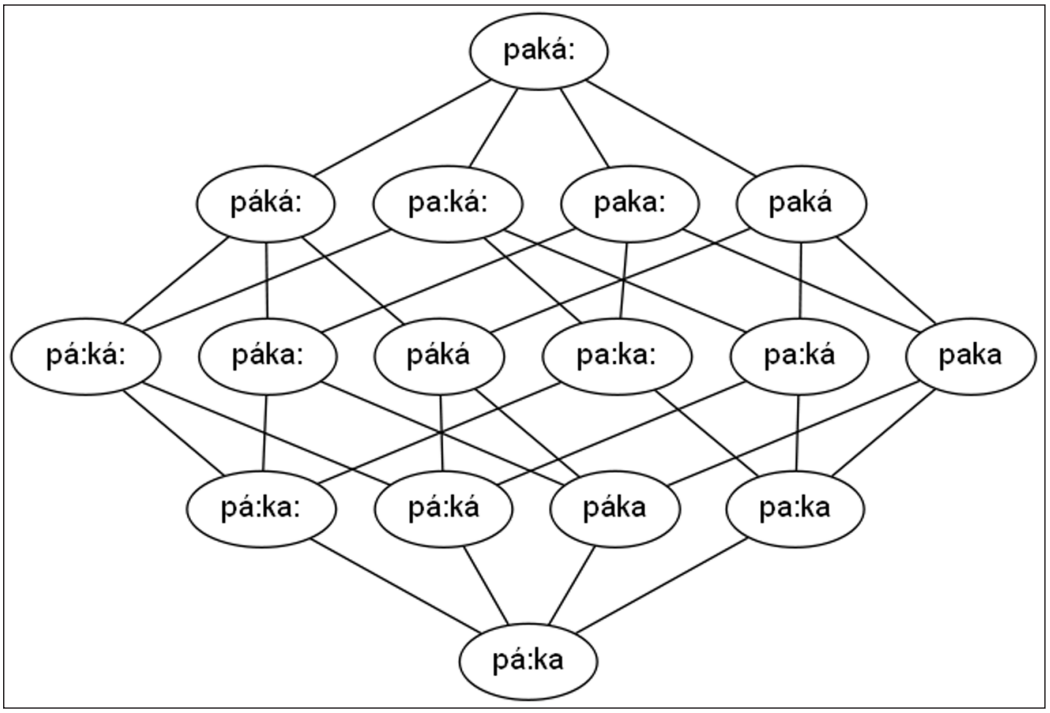

Figure 1: The relative similarity relation for output [paká:].

greater relative similarity. The top node has the greatest similarity: it contains the input that is identical to the output, with no disparities. The nodes in the next row down each represent a candidate containing a single disparity. The bottom node includes the input with the least similarity: every single vowel feature value (stress and length) differs from its correspondent in the output. Each candidate has greater similarity than the candidates below it in the figure, either directly below or via transitivity.

3.2 Insertion and deletion disparities

The problem addressed in this paper involves disparities that are not identity disparities, but insertion disparities and deletion disparities. An *insertion disparity* is an output segment with no corresponding input segment. The candidate in (3) has an insertion disparity: the second vowel of the output has no corresponding segment in the input. The subscripts in this and subsequent examples indicate input-output correspondents.

(3) $/C_1V_2C_3/ \rightarrow [C_1V_2C_3V]$

A *deletion disparity* is an input segment with no corresponding output segment. The candidate in (4) has a deletion disparity; the final consonant of the input has no corresponding segment in the output.

(4) $/C_1V_2C/ \rightarrow [C_1V_2]$

The map in (5) illustrates output-drivenness with insertion and deletion disparities. Note that all four of the candidates in (5) have the same output, [CV].

(5) An output-driven map with insertion and deletion disparities

a. $/V_2C/ \rightarrow [CV_2]$

b. $/C_1V_2C/ \rightarrow [C_1V_2]$

c. $/V_2/ \rightarrow [CV_2]$

d. $/C_1V_2/ \rightarrow [C_1V_2]$

In an output-driven map, the inclusion of (5)a in the map automatically entails the inclusion of (5)b, (5)c, and (5)d in the map, as each of those has greater similarity than (5)a. Candidate (5)a has two disparities: the final consonant of the input is a deletion disparity, and the first consonant of the output is an insertion disparity. Candidate (5)b has an identical deletion disparity, but lacks the insertion disparity. Candidate (5)c has the insertion disparity but not the deletion disparity, and candidate (5)d has no disparities at all.

In the map in (5), candidate (5)d has greater similarity than each of the other candidates. Neither of candidates (5)b and (5)c has greater similarity than the other; each one has a disparity that the other lacks.

3.3 Learning with output-driven maps

The structure of output-driven maps provides a great deal of power for phonological learning, especially with respect to the learning of underlying forms for morphemes. This power can be leveraged in learning approaches that make use of inconsistency detection to determine when the value of an underlying feature can be set (Tesar 2006; Merchant 2008). A learner determines that a given feature of the underlying form for a morpheme must be set to a given value if all possible underlying forms with a conflicting value for that feature are inconsistent with what the learner already knows about the grammar (a kind of process of elimination). The structure of output-driven maps greatly accelerates the determination of inconsistency.

Returning to the stress/length illustration from section 3.1, if the output [paká:] is generated by the grammar, then some input must map to it. The structure of output-driven maps says that if any input maps to it, then the input with zero disparities must map to it: /paká:/ → [paká:]. However, if a learner observes a particular word [paká:], they are not guaranteed that the lexicon's input for that particular word is /paká:/. One or more of the underlying features might have different values for that word, and those values are changed by the phonology to reach the output. The learner can determine which underlying feature values are necessary to reach the output by testing each one separately. To determine if the length feature of the second vowel *must* be +long, the learner could test all of the possible inputs that have a second vowel with the feature value –long, and see if any of those could map to the output [paká:] via some constraint ranking consistent with the learner's ranking information. If none of those inputs can lead to [paká:], then the correct input *must* have the second vowel +long. The feature value of –long has been determined to be inconsistent, allowing the learner to set, in their lexicon, the value of that second vowel's length feature to +long.[2] This approach requires determining that all possible inputs with the second vowel +long be inconsistent. If that were to require separately constructing and evaluating every possible such input, the computational cost of the approach could become quite high. This is where the structure of output-driven maps makes its contribution.

The structure of output-driven maps is, at its heart, entailment relations between candidates: if A→X is in the map, then every candidate B→X of greater similarity must also be in the map. Learning underlying feature values takes advantage of the logically equivalent contrapositive form: if candidate B→X is *not* in the map, then any other candidate A→X of lesser similarity also cannot be in the map.

In the illustration from section 3.1, because the target map is output-driven, the learner can test the length feature of the second vowel by constructing a candidate with an input which differs from the output, [paká:], solely on the value of the length feature of the second vowel, /paká/. This input has only one disparity. The resulting candidate is (2)b, /paká/ → [paká:]. If (2)b proves to be inconsistent with the learner's current grammatical knowledge, then it cannot be part of the map being learned. Any other candidate with an underlying value of –long for the second vowel will have that same disparity, plus others. Candidate (2)a, /páká/ → [paká:], has lesser similarity than (2)b. In an output-driven map, if (2)a is grammatical, then (2)b is also grammatical. The contrapositive direction states that if (2)b is *not* grammatical (B→X is *not* in the map), then (2)a is also *not* grammatical (A→X is *not* in the map).

The inconsistency of the single disparity candidate /paká/ → [paká:] obviates the need for constructing or computing any of the possibly many other candidates with a value of –long for the input second vowel such as (2)a, due to the greater internal similarity of the single disparity candidate, combined with the structure of output-driven maps. Thus, the single disparity candidate acts as a proxy for the many possible candidates with the underlying value –long for the second vowel. If the single disparity candidate is inconsistent, then all such candidates are inconsistent, and the learner has determined that the underlying value of length for that vowel must be +long.

3.4 Contrast, alternation, and learning

When an underlying feature is determined to necessarily have a specific value, the linguistic interpretation is that the particular instance of the feature is *contrastive* for that environment. It is contrastive in that, if the underlying feature were instead given

[2] See (Tesar 2014) for further discussion of how inconsistency is detected computationally.

a different value, then the resulting input would have a different surface realization. In order for a learner to set a feature's value with complete confidence, they must observe that feature in an environment where it is contrastive, where the feature's value makes a difference in the surface realization of a word.

Setting the underlying value for a feature can lead to further information for the learner when it is part of a morpheme that appears in multiple words. If the surface word [paká:] discussed in section 3.3 consists of two morphemes, with the first syllable realizing morpheme r1 ("root 1") and the second syllable realizing morpheme s4 ("suffix 4"), then setting the underlying value of the length feature for the second vowel means setting it in the underlying form for s4 (the phonological input for the word is constructed by concatenating the underlying forms of the morphemes of the word). This value of +long for s4, set in the environment of the word r1s4, will then be carried into the learner's analysis of any other word containing the morpheme s4.

Observing a morpheme in a different environment is particularly informative if the set feature is neutralized to the feature's other value in the different environment. For example, when suffix s4 is combined with a different root, r3, to form the word r3s4, with output [páka]. The key point here is that, whereas s4 surfaced as [ká:] in word r1s4, it surfaces as [ka] in r3s4. s4 exhibits a *morphemic alternation*, and in particular alternates with respect to its length feature. Because the learner knows that s4 is underlyingly +long, it can conclude that the length feature is being neutralized in word r3s4: the phonology is shortening the vowel of s4 in this context. That allows the learner to obtain non-phonotactic ranking information: ranking information that requires evidence of disparities forced by the grammar.

In this approach to learning, contrast indicates aspects of underlying forms that must be faithfully preserved in a particular context, and alternation indicates aspects of underlying forms that must be neutralized in a particular context. Both the faithful preservation and the neutralization are the responsibility of the constraint ranking. Extending this approach to linguistic systems involving insertion and deletion requires identifying instances of contrast with insertion and deletion, representing them appropriately in underlying forms, and using those representations to uncover and interpret instances of insertional and deletional neutralization.

4 The idealization of the learning situation

The idealized learning situation used in this paper posits that the learner receives grammatical word outputs, each in the form of a syllabified sequence of C's and V's. For each observed output, the learner is provided with an indication of the morphemes making up the corresponding word. Each segment of each output is labeled as affiliated with a morpheme. Thus, for each output, the learner is provided with the information of what morphemes make up the word, and which segments of the output are the surface realization of each morpheme of the word.

The learner is also provided with some correspondence information between paradigmatically related outputs. In previous work involving the Output-Driven Learner, this information was entirely implicit. Because the only disparities considered were identity disparities, every surface realization of a morpheme had the same number of segments, and the learner "assumed" that the first segment of one surface realization corresponded to the first segment of another surface realization of the same morpheme, the second with the second, and so forth. If a morpheme surfaced as [ka] in one word, and as [ká:], the learner infers that the same underlying segment is the input correspondent for [a] in the first word and for [á:] in the second.

Once insertion and deletion are introduced, morphemes can have surface realizations of different lengths in different words. Under the present idealization, the learner is provided with an indication of which segments correspond to each other in different surface realizations of the same morpheme. If a morpheme surfaces as [V] in one word and as [VC] in another word, the learner is given the knowledge that the V in the first corresponds to the V in the second; they are both the "same" segment, in this sense. Note that this does not guarantee that the V has an underlying correspondent segment (it could be an inserted V in both cases), but if the V in the first word has an underlying correspondent (in the underlying form of the morpheme), then the V in the second word has the same underlying correspondent. Similarly, there is no guarantee that the C in the second surface realization does not have an underlying correspondent; it might well have one, but it is deleted in the first word. See section 9.5 for further discussion.

The paradigmatic correspondence information just described is assumed to ultimately result from the same paradigmatic morphological analysis, performed by the learner, that is ultimately necessary for the learner to determine what morphemes are present in which words in the first place. The present idealization side-steps the very complex topic of how the learner performs such analysis to gain the information, and simply posits that the information is provided. The research presented here aims to better understand how such paradigmatic information could be effectively used by a learner to learn the underlying forms and the constraint ranking. How the learner arrives at that paradigmatic information is left to future research.

5 The presence feature

The learning algorithm for identity disparities was able to representationally distinguish between underlying features for which the learner had committed to a specific value and underlying features for which the learner had not (yet) made any such commitment. If a similar learning strategy is to be applied to learning in the face of insertion and deletion disparities, the learning algorithm must also be able to representationally distinguish between underlying segments for which the learner has committed to being present/absent, and underlying segments for which the learner has no commitment (the learner has not yet determined if the hypothesized segment is actually there in the underlying representation or not).

The proposal here is for the learner to use a feature-like representational structure, called the *presence feature*. By making it a feature-like representational structure, the same feature-setting strategy used for learning underlying feature values can be applied to the learning of the presence feature, and thus the determination of which segments are actually present in the underlying forms. The current proposal is that the presence feature is a representational device constructed and used by the learner for the purpose of learning, and is not part of the linguistic theory itself. The presence feature is added to segments that are contained in underlying forms and linguistic inputs.

5.1 Representing presence and absence

Each hypothesized segment explicitly represented in the input has a presence feature. If a segment is set to +presence (present) in the input, this indicates that the segment must be present underlyingly. If none of the other features (the regular phonological features) have been set on that segment, that the learner has determined that a segment of some sort must be there, but nothing about the segment's phonological featural specifications. If some of the phonological features have been set on the segment, then the value +presence indicates that a segment with those set feature values must be present at that location in the input.

If a segment is set to –presence (absent) in the input, this indicates that the segment cannot be present underlyingly. Having a segment marked as –presence in an underlying form serves to block the learner from hypothesizing that type of segment in that position in response to other data.

A feature is unset if it has not been set to a specific value underlyingly. An unset feature is denoted with a question mark (?) in place of the value of the feature. At a given point in learning, a feature might be unset because the learner has not yet acquired the information necessary to set the feature, or it might be unset because it is not contrastive in any environment for the target language, so no information is forthcoming which would justify setting it one way or the other. A chart summary of the notations for a presence feature is given in Table 2.

The representation in (6) has two potential input segments: the first potential input segment is a potential correspondent to the first output segment [C], and the second potential input segment is a potential correspondent to the second output segment [V]. Both input segments have their presence feature unset; the learner is uncertain if either segment is actually present in the correct input for the word.

(6) /(?,C)(?,V)/ → [CV]

In (7), the underlying consonant segment has been set to +presence. The learner has committed to the presence of that segment in the input. The underlying vowel's presence feature has not been set.

(7) /(+,C)(?,V)/ → [CV]

In (8), the underlying vowel segment has been set to –presence. The learner has committed to the absence of that segment in the input. The effective candidate represented is /C_1/ → [C_1V], meaning that the output consonant has an input correspondent, while the output vowel is inserted.

(8) /(+,C)(–,V)/ → [CV]

5.2 Setting presence features

When a learner first starts learning underlying forms, it adjusts the underlying form for a morpheme on the basis of the output forms of words containing that morpheme, and in particular on the basis of the identified surface realization of that morpheme in those words. Each segmental representation in the underlying form for a morpheme is a potential correspondent to an output segment in at least one of the surface realizations of that morpheme.

All constructed input segments initially have their presence feature unset, just as all other segmental features are initially unset. As described in section 3.3, the learner separately tests each feature using inconsistency detection. For a given word, with a given

Table 2: Notations for the possible states of a presence feature.

Symbol	Meaning
+	Segment must be in the input
–	Segment cannot be in the input
?	Unset

output, the learner has evidence that a feature must be set to one value if setting that feature to its other value would force the word's output to be something different from its actual (observed) output. When a presence feature cannot have one value, the learner can commit to the feature being set to the other value.

To foreshadow the illustration in section 8.2.2, suppose the learner observes a word with a single morpheme, r1, surfacing as [V], and the underlying form for r1 is /(?,V)/. The solitary potential segment in the input is the potential correspondent of the solitary V in the output. An assigned value of +presence to the presence feature in r1 will clearly be consistent: the resulting candidate is /V_1/ → [V_1], an identity candidate for an attested output, which is guaranteed to be grammatical by output-drivenness. The underlying V could be +presence; the learner will test to see if it must be +presence by testing the value –presence. The test candidate, /(–,V)/ → [V], with no segments in the input, cannot be optimal, as it will always lose to a candidate with no segments in the output. No ranking of the constraints of BST will ever compel an entire inserted syllable. This detected inconsistency tells the learner that, because the value of the presence feature cannot be –presence, it must therefore be +presence, and the learner sets the corresponding feature in the lexicon to +presence.

As discussed in section 3.4, a feature can be set when it is contrastive. Here, the presence feature is contrastive: if this particular presence feature were set to –presence instead, a different output would result. Positing a presence feature on input segments allows this general principle of feature setting to extend to the learning of presence vs. absence of segments.

5.3 Learning deletion and insertion

As discussed in section 3.4, an underlying feature that has been set can reveal evidence of neutralization if it surfaces unfaithfully in a different morphemic context. If an underlying segment has been set to +presence, then a context in which it is unfaithfully realized (does not surface) provides evidence of deletion. To neutralize a +presence segment is to delete it.

If an underlying segment has been set to –presence, then a context in which it is unfaithfully realized (has a possible output correspondent) provides evidence of insertion. While a bit counterintuitive, an output segment with a potential input correspondent set to –presence is the way that an inserted segment is represented in this system. In essence, it states that the output segment cannot have an input correspondent segment that is actually present in the underlying form, because if it were present, it would cause a different word containing that morpheme to surface with the wrong output.

Features set to +presence create the opportunity for observing deletion; features set to–presence create the opportunity for observing insertion.

5.3.1 Deletion

A segment can only be set to +presence in a context in which it has a realized output correspondent (and is contrastive in that context). If the same morpheme appears in a different context and that (+presence) segment does not have an output correspondent, then the learner has direct evidence of neutralization in the form of segment deletion. The learner can conclude that the segment in question has been deleted in the second context, because it knows that the segment actually is present underlyingly as a result of the first context. Identifying a concrete instance of deletion allows the learner to obtain ranking information that forces the learner's ranking to delete the segment in the latter context, with the effect of also forcing deletion of analogous other segments in analogous other contexts.

To foreshadow the illustration in section 8.2.4, suppose the learner has already set two presence features for morpheme r1, so that its underlying form is /(+,V)(+,C)/. The presence feature of the second segment, the C, was set on the basis of a word containing the morpheme r1 in which that segment was contrastively present in the output. However, the output of the word containing only the morpheme r1 is [V], and the corresponding candidate is $/V_1C/ \rightarrow [V_1]$. This candidate is necessarily grammatical, and the underlying C is deleted. In this context, the C is neutralized; the same output would result whether or not this C was present in the input.

5.3.2 Insertion

A segment can only be set to –presence in a context in which it does not have an output correspondent (and is contrastive in that context). If the same morpheme appears in a different context and that (–presence) segment does have a potential output correspondent, then the learner has direct evidence of neutralization in the form of segment insertion. The learner can conclude that the segment in question has been inserted in the second context, because it knows that there is no input correspondent for the segment as a result of the first context. Identifying a concrete instance of insertion allows the learner to obtain ranking information that forces the learner's ranking to insert the output segment in the latter context, with the effect of also forcing insertion of analogous other segments in analogous other contexts.

To foreshadow the illustration in section 8.3.6, suppose the learner has already set the underlying form for a morpheme r2 to /(+,V)(+,C)(–,V)/. The second V is set to –presence: the learner is certain there is not a second V there in the underlying form. That presence feature was set on the basis of a word containing the morpheme r2 in which that segment was contrastively absent from the output. However, the output of the word containing only the morpheme r2 is [V.CV], and the corresponding candidate is $/V_1C_2/ \rightarrow [V_1.C_2V]$. This candidate is necessarily grammatical, and the second V of the output is inserted. In this context, the second V is neutralized; the same output would result whether or not the output V had an actual correspondent in the input.

6 The unbounded deletion problem

6.1 The problem

Allowing insertion and deletion expands the range of possible inputs for a learner to consider for a given output. Different candidates for an output can vary in the number of insertions and deletions they include. Since the output form has a fixed, finite number of segments, the number of segments that could be inserted is at most the number of output segments. This allows for exponential growth in the number of possible inputs as a function of the size of the output.

Deletion adds a greater degree of complexity. There is no fixed bound on the number of input segments that can be deleted within a candidate, regardless of the size of the output form. On its own, this results in a space of possible inputs that is infinite in size. Matters are made more complex by the fact that the infinite space cannot be bounded simply through inconsistency. In fact, in some circumstances the correct ranking will map each of an infinite subset of the possible inputs to the same output: with respect to the ranking itself, there are an infinite number of inputs that are equally good. Grammatical consistency alone will not bound the size or number of inputs to be considered.

Consider a grammar in which codas are banned, and this restriction is enforced via consonant deletion; the upcoming example in section 8.2 is such a case. Such a grammar would have a ranking in which NoCoda and DepV both dominate Max. If such a grammar admits a candidate like (9)a as grammatical, then it will also admit candidate (9)b, where the input C is deleted (rather than putting the C in the coda of the preceding

syllable, or inserting a vowel after the C to create another syllable with C in the onset). That grammar will necessarily apply the same deletion to multiple consonants that are stacked at the end of the input (for identical reasons), so candidates like (9)c and (9)d will also be admitted. There is no bound on the number of C's that can be lined up at the end of the input (following a solitary V), and all such inputs will map to the same output, [V].

(9) Unbounded deletion allows an infinite number of inputs to have the same output.
   a. /V_1/ → [V_1]
   b. /V_1C/ → [V_1]
   c. /V_1CC/ → [V_1]
   d. /V_1CCC/ → [V_1]

Figure 2 illustrates how the relative similarity relation for /V/ becomes unboundedly large as deletion disparities increase. The top node, (9)a, has no disparities. In the second row, each node contains a single deletion disparity; (9)b is illustrated as the third node from the left. The third row of nodes illustrates how rapidly the amount of candidates expands as deletion disparities increase; (9)c is illustrated as the fourth node from the right. Because there is no limit to the amount of deletion disparities an output can have, the final row illustrates unboundedness with ellipses.

6.2 Deletion and output-drivenness

As discussed in section 3, the structure of output-driven maps ensures that, if one candidate is in the map, then any other candidate formed solely by removing disparities must also be in the map. Such a relationship holds among the candidates in (9): removing a deletion disparity from candidate (9)c results in (9)b, and removing a deletion disparity from (9)b results in (9)a, the candidate with zero deletion disparities. For any single output, for any grammatical candidate with deletion disparities, there will be a candidate for the same output (but different input) having no deletion disparities that is also grammatical.

More generally, output-drivenness gives us the result in Lemma 1.

Lemma 1: Suppose that a grammar defines an output-driven map, where input-output correspondence is purely segment to segment, and individual input segments lacking output correspondents constitute deletion disparities. Then for any grammatical candidate, if the input contains a segment *seg* lacking an output correspondent, then the candidate formed by removing *seg* from the input is also grammatical.
Proof: Follows directly from output-drivenness. The input segment lacking an output correspondent constitutes a deletion disparity. Removing that segment from the input removes one disparity, introduces no other ones, and has the same output. By output-drivenness, that candidate is optimal.
End of Proof

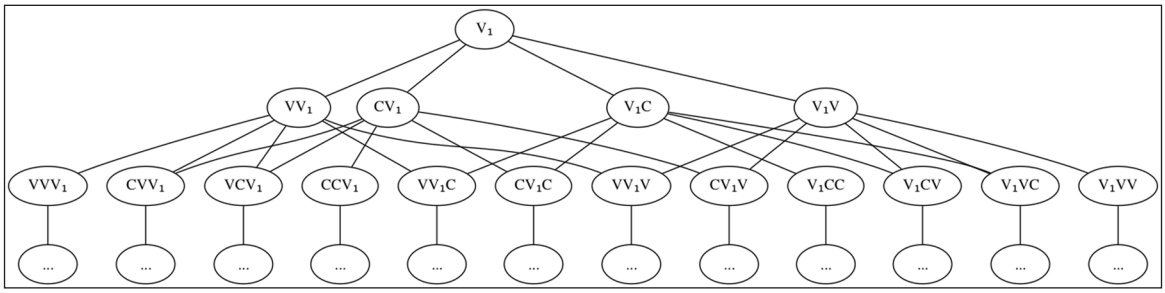

Figure 2: The relative similarity order for unbounded deletion case.

Under the conditions of Lemma 1, if a single output is considered in isolation, there will be a grammatical candidate for that output with no deletion disparities, that is, no input segments lacking output correspondents. That candidate could be described as "deletionally minimal", in that it minimizes the number of deletions in the candidate.

Lemma 1 can be generalized in a non-trivial way to underlying forms for morphemes. To do so, we need to be sufficiently precise about what "success" is. The learner is engaged in identifying a ranking of the constraints and a lexicon of underlying forms that "works" for the language. A grammar works if, for any word of the language, the input constructed by composing the underlying forms for the morphemes of the word is mapped (by the constraint ranking) to the correct output for that word. Focusing on a single morpheme *target_morph*, while leaving all other morphemes and the constraint ranking correct, an underlying form *uform_1* for *target_morph* "works" if, when included in the input for a *target_morph*-containing word, results in the correct output for that word.

For each word containing *target_morph*, the output segments affiliated with *target_morph* constitute the surface realization of *target_morph* in that word. A candidate with the correct (observed) output and *uform_1* as the underlying form for *target_morph* has an IO correspondence between the segments of *uform_1* and the surface realization of *target_morph*. For a given word containing *target_morph*, any given segment of *uform_1* either does or does not have an output correspondent in the word, as determined by the IO correspondence.

For any given segment of *uform_1*, it either has an output correspondent in at least one surface realization of *target_morph*, or it does not. Thus, it is coherent to identify a potential property of a segment of *uform_1* as not having an output correspondent in any surface realization. We will label that as the property of having *no correspondent anywhere*, and commonly use *seg_nca* to denote a segment with that property.

Theorem 1: Suppose that a grammar defines an output-driven map, where input-output correspondence is purely segment to segment, and individual input segments lacking output correspondents constitute deletion disparities. Suppose that a morpheme *target_morph* has underlying form *uform_1*, and that a segment *seg_nca* of *uform_1* has no correspondent anywhere. Let the underlying form *uform_2* be constructed by removing *seg_nca* from *uform_1*. Changing the underlying form for *target_morph* to *uform_2* will result in each word containing *target_morph* having the same output as before.
Proof: The theorem follows from applying Lemma 1 to each word containing *target_morph*. For any word *word_tm* containing *target_morph*, if *uform_2* is substituted for *uform_1* as the underlying form for *target_morph*, then the fact that *seg_nca* has no correspondent anywhere entails that it has no correspondent in *word_tm*. Lemma 1 ensures that the input with *uform_2* substituted for *uform_1* will have the same output.
End of Proof

Corollary 1: Under the conditions of Theorem 1, for any morpheme *target_morph*, there exists an underlying form *uform_corr* that works, and has the property that each segment present in *uform_corr* has an output correspondent in at least one of the surface realizations of *target_morph*.
Proof: By hypothesis, there exists at least one working underlying form *uform* for *target_morph*. If every segment of *uform* has an output correspondent, then adopt *uform* as the value of *uform_corr*, and we are done. Otherwise, construct *uform_corr* by removing from *uform* all segments that have no correspondent anywhere, as justified by Theorem 1.
End of Proof

A couple of clarifications are in order. Corollary 1 ensures that each segment of *uform_corr* has an output correspondent in at least one surface realization of *target_morph*. It does *not* ensure that there is a single surface realization that contains output correspondents for all of the segments of *uform_corr*.

Corollary 1 also does not ensure that there is a unique *uform_corr*, or that any given possible *uform_corr* necessarily has the fewest segments of any working underlying form. For instance, consider a language where $/V_1/ \rightarrow [CV_1]$, that is, consonants are inserted when necessary to ensure that syllables have onsets. For a monomorphemic word with output [CV], both /CV/ and /V/ are working underlying forms, where $/C_1V_2/ \rightarrow [C_1V_2]$. The underlying form /CV/ is clearly longer than /V/, but each input segment of /CV/ does have an output correspondent in the word.

6.3 Limiting the range of underlying forms in learning

Corollary 1 guarantees that the learner can limit the range of possible underlying forms to be considered. A crude argument to this effect is as follows. Given that the learner observes only a finite amount of data, they can only observe a finite number of surface realizations of any given morpheme. As each surface realization has a finite number of segments, there are only a finite number of distinct segments across all of the observed surface realizations of the morpheme. Because the learner only need consider segments in the underlying form that might possibly correspond to a segment in one of the surface realizations, the learner need only consider a finite number of possible segments in the underlying form.[3]

The learner proposed in this paper exploits Corollary 1, and the structure of output-drivenness behind it, to solve the unbounded deletion problem. It does so by adding a potential segment to an underlying form only if that segment could potentially have an output correspondent for one of the surface realizations of its morpheme. The first time the learner processes a word containing a given morpheme, it creates a lexical entry for that morpheme. For each segment of the surface realization, the learner adds a corresponding potential segment to the underlying form in the lexical entry. The learner does not commit to the presence of any of the potential segments in the underlying form at this point (that was discussed in section 5), but it does represent the possibility of having each such segment.

Subsequently, whenever the learner processes a different word containing the same morpheme, it constructs the IO correspondence between the underlying form for the morpheme and the surface realization of the morpheme in that word. If there is a new segment in the surface realization that does not have an existing potential input correspondent in the underlying form, then a new input segment is added to the underlying form as the potential input correspondent for the new output segment.

Here is an illustration, taken from the learning example that will be presented in detail in section 8.2. Specifically, this comes from the part of the example shown in section 8.2.3. A morpheme, r1, had been observed as a monomorphemic word, surfacing as [V]. The learner constructed an underlying form for r1, /(?,V)/, with a single segment in the input that potentially corresponded to the V in the output. At that point, it was immediately able to determine that the underlying segment must be present, and set the presence feature accordingly, yielding the underlying form /(+,V)/ for r1. This state of affairs is shown in (10).

[3] This simple reasoning relies on the idealization that there is no coalescence. If multiple input segments can correspond to the same output segment, then the number of distinct output segments does not necessarily bound the number of input segments that need to be considered in such a simple way. Issues involving coalescence and splitting are left for future research.

(10) Morpheme r1: /(+,V)/
Word r1: /(+,V_1)/ → [V_1]

The learner then observes a different word, r1s1, which contains morpheme r1. The output of r1s1 is [V.CV]. The morphemic affiliation information for the word indicates that the first two segments of the output are affiliated with morpheme r1, and that the first segment, V, is the same segment as the V in the output of the word r1. Thus, for the surface realization [V_1C_2] of r1 in the word r1s1, V_1 has an input correspondent (both potential and actual) in the underlying form, but C_2 does not currently have a potential input correspondent. The learner at this point adds one to the underlying form, yielding the underlying form /(+,V)(?,C)/. The result is depicted in (11).

(11) Morpheme r1: /(+,V)(?,C)/
Word r1s1: /(+,V_1)(?,C_2); (?,V_3)/ → [V_1.C_2V_3]

The learner has not yet committed to whether the C is present in the underlying form, but if it is present, then it will correspond to the output C in the surface realization of r1 in the word r1s1. That is what the subscript "2" indicates in (11): the C in the output has a potential input correspondent, but it is not yet known if it is present, and thus an actual input correspondent or not.

Once the additional potential segment is added to the underlying form, it is relevant everywhere, including the previously processed word r1. The correspondence situation for the word r1 no longer looks as it did in (10). Instead, it is now as shown in (12).

(12) Morpheme r1: /(+,V)(?,C)/
Word r1: /(+,V_1)(?,C)/ → [V_1]

In (12), the learner still isn't certain if the C is present in the underlying form of r1 or not. However, *if* the C is present underlyingly, *then* it is deleted in the word r1, because that segment has no potential output correspondent in word r1.

The fact that the correspondence situation has changed for word r1 does *not* invalidate the information obtained by the learner when r1 was previously examined by the learner. This is because of output-drivenness. If the potential C in the underlying form of r1 ultimately proves to be absent (set to –presence), then the learner is effectively in exactly the same situation as it was earlier, with (10). If, on the other hand, the C ultimately proves to be present (set to +presence), then it constitutes a deletion disparity in the word r1. By output-drivenness, if /VC/ → [V], it automatically follows that /V/ → [V], as a deletion disparity has been removed, and no other disparity has been added. Thus, the mapping /V/ → [V] entertained earlier is still valid. This will always be true of a potential underlying segment added in response to a later word; because the added segment has no potential output correspondent in earlier words (otherwise it would have been included in the underlying form at the time), it will register as a potential deletion disparity for the earlier words, and by output-drivenness any grammatical information previously obtained for those words is still valid.

By attending to the structure of output-driven maps, the learner is able to solve the unbounded deletion problem: it only need consider underlying form segments that potentially have a correspondent in at least one observed surface realization of the morpheme.

6.4 Explicit vs. implicit non-presence

The learner uses a hybrid scheme for representing hypothetical segments which are not present in an underlying form. Some such segments are represented explicitly, each as a potential input segment with a presence feature set to the value –presence. Others are

only implicitly represented, through lack of any symbolic representational structure at all. The split is not arbitrary. The non-present hypothetical segments that end up explicitly represented have important differences from the non-present hypothetical segments that end up implicitly represented.

In order for a potential input segment to be set to the value –presence, the segment must have been posited in the input in the first place. The learner only posits a potential segment in the input on the basis of an output where there is an output segment that does not yet have a potential input correspondent. A potential input segment set to –presence entails the existence of an inserted output segment in at least one surface realization of the morpheme: the output segment that was the basis for constructing the potential input segment in the first place. Direct evidence of disparities, such as inserted segments, have the potential to provide non-phonotactic ranking information: the constraint ranking must be such that it forces the evidenced disparity.

A segment that is not explicitly represented in an underlying form has no potential output correspondent in any of the words observed by the learner. Such a segment's lack of presence cannot provide evidence for non-phonotactic ranking information: it cannot constitute a deletion disparity (it is not present in the input), and it cannot indicate an insertion disparity (it has no potential output correspondent that is inserted).

In cases where unbounded deletion is possible, the overwhelming majority of possibly deleted segments are completely uninformative, and the learner does not waste any resources contemplating them. The possibly deleted segments that might be informative are represented by the learner with potential input segments, and the set of such segments is clearly bounded, both for an individual word and for a morpheme that appears in a variety of words.

7 Overview of the learner

7.1 Overview of the learner

The learner presented here is based on the Output-Driven Learner (Tesar 2014), with some modifications made to incorporate the proposals for contending with insertion and deletion.

The learner's first phase is phonotactic learning (Hayes 2004; Prince & Tesar 2004 and references therein). The learner observes entire output forms without knowledge of the morphological constituency of the words. For each word, the learner constructs a candidate with no disparities (the input is segmentally the same as the output) and, in keeping with the structure of output-driven maps, concludes that the candidate must be grammatical. The learner then determines what (if any) new ranking information is needed to ensure that grammaticality of that candidate, and adds that information to the learner's support (its store of ranking information).

When processing a word, the learner tests each unset feature of the underlying forms of the word, to see if any can be set. If a feature can be set, the learner (a) sets the feature in its lexicon; (b) checks for non-phonotactic ranking information. The pursuit of non-phonotactic ranking information consists of searching its memory of observed words, looking for any words in which the morpheme containing the just-set feature alternates with respect to that feature, that is, in which the just-set feature is neutralized to the value opposite its underlying value. If such a word is found, then the learner checks it for further ranking information, using essentially the same procedure as phonotactic learning, but with the set features in the lexicon providing at least one disparity in the candidate (any features unset in the lexicon are temporarily assigned values matching the surface realization of the word being tested). The pursuit of settable underlying features and non-phonotactic ranking information continues until the learner determines that it has a grammar that successfully produces all of the observed words.

7.2 Top-level outline of the Output-Driven Learner

A top-level outline of the Output-Driven Learner, upon which the present proposal is based, is given in Figure 3. The learning examples presented in this paper only require single form learning, so the use of contrast pairs (steps 4 and 5) will not be elaborated on further in this paper. This outline also does not include reference to fewest set features, which is proposed and discussed in Tesar (2014) but not referenced in the examples here; see Tesar (2014) for extensive additional discussion of these topics.

The following subsections provide more detailed outlines of key parts of the Output-Driven Learner, modified to explicitly reflect the handling of the presence feature.

7.3 Single form learning

Once phonotactic learning has completed, the learner gets knowledge of the morphemic constituency of each word, including the morphemic affiliation of each segment of each output form. The learner then starts processing morphologically analyzed words one at a time, using a procedure labeled *single form learning*. An outline of *single form learning* is given in Figure 4.

Whenever the learner encounters a new morpheme, it creates an entry in its lexicon for that morpheme, with one underlying segment for each segment of the (just-observed) surface realization of that morpheme. All presence features of all segments of the underlying form are initially unset.

For present purposes, it is simply assumed that the learner is able to construct the correct IO correspondence in step 3, based largely on the supplied information about morphemic affiliation. Whenever the learner observes a morpheme in a new context, it constructs a new IO correspondence and checks to see if there is an affiliated surface segment that did not appear in any previously observed surface realization. If such a segment is identified, then the learner adds a potentially corresponding input segment to the underlying form of the morpheme. This implements the approach described in section 6.3. An example of this

Output-Driven Learner

1. *phonotactic learning*
2. repeat for each stored word until no word yields new information
   single form learning (sec. 7.3) on the word
   end
3. if no stored words fail *initial word evaluation* (sec. 7.4)
   wait until a new word is observed
   store the new word
   go to step 2
   else
   go to step 4
   end
4. *contrast pair learning*
5. if an informative contrast pair was found
   go to step 2
   else
   wait until a new word is observed
   store the new word
   go to step 2
   end

Figure 3: Outline of the Output-Driven Learner.

single form learning

1. given a morphologically segmented *output*
2. construct *input* from the learner's underlying forms for the morphemes of *output*
3. construct an IO correspondence *corr* between *input* and *output*
4. construct *word* from *input*, *output*, and *corr*
5. *initial word evaluation* with *word* (sec. 7.4)
6. if *word* cannot provide new information
   return
   end
7. *check for ranking information*
8. for each input segment *seg* with an unset presence feature:
   if *seg* has a potential output correspondent
   test the value –presence for *seg*
   else
   test the value +presence for *seg*
   end
   if the tested value is inconsistent
   set the presence feature of *seg* to the value opposite the tested value
   let *morph* be the morpheme affiliated with *seg*
   non-phonotactic ranking information with *morph* and *seg* (sec. 7.5)
   end
   end
9. return

Figure 4: Outline of single form learning.

kind of segment addition to an existing underlying form occurs in the upcoming illustration of learning deletion in section 8.2.3.

7.4 Initial word evaluation

An outline of the procedure of initial word evaluation is given in Figure 5. This procedure determines if the learner's current grammar accounts for the word. To make this determination, the procedure must contend with two kinds of uncertainty in the learner's grammatical information: uncertainty about the input, and uncertainty about the ranking.

The learner deals with uncertainty about the ranking by generating its "best guess" ranking given the learner's support of ranking information, which is computed by running Biased Constraint Demotion with a faithfulness-low bias (Prince & Tesar 2004). This occurs in step 4 of Figure 5.

The learner contends with uncertainty about the input by taking advantage of output-drivenness. In earlier work on the Output-Driven Learner, the space of possible inputs for a word formed a finite lattice, such as the one shown in Figure 1. The viable inputs for a word (those that did not contradict set features in the learner's lexicon) always formed a finite sublattice. The learner would construct the input corresponding to the bottom of the finite sublattice. This input is sometimes referred to as the *maximal mismatch input*, because every unset feature is temporarily assigned the value opposite its surface realization in the word.

The learner then checks to see if the observed word is the sole optimum for the maximal mismatch input with respect to the constructed ranking. If it is, then the grammar is currently accounting for the word, and is said to pass initial word evaluation; otherwise, it fails initial word evaluation. This occurs in steps 5 and 6 of Figure 5. This approach works because, by the logic of output-drivenness, if the ranking maps the maximal mismatch

initial word evaluation

1. *given a word*
2. let *mismatch_input* be the input for word matching the learner's lexicon
3. for each segment *seg* of *mismatch_input* with an unset presence feature:
   let *unset_f* be the presence feature of *seg*
   if *seg* has an output correspondent in *word*:
   assign *unset_f* the value –presence
   else
   assign *unset_f* the value +presence
   end
   end
4. construct the *ranking* from the learner's support
5. find the optimal candidate(s) for *mismatch_input* using *ranking*
6. if a candidate other than *word* is optimal
   return fail
   else
   return pass
   end

Figure 5: Outline of initial word evaluation.

input to the correct word, then it must map every other viable input to the correct word (because all of the other viable inputs are higher in the lattice, and thus have greater relative similarity). Because the learner's current best guess ranking maps all of the viable inputs to the correct word output, there is nothing further to be learned from that word at that point.

That procedure, as described, requires modification when deletion is an option. The reason is that the similarity relation is not bounded from below (as illustrated in Figure 2), so there is no bottom element. However, if the learner (justifiably) restricts their space of viable inputs to just those involving the potential input segments that they have constructed in the underlying forms making up the input for the word, then a finite space of viable inputs is quickly recovered. In the cases of BST considered here, the finite space of viable inputs forms a lattice with a bottom element, the maximal mismatch input. The construction of that is depicted in steps 2 and 3 of Figure 5.

7.5 Ranking information

The procedure outlined in Figure 6 is the standard one for pursuing ranking information in the Output-Driven Learner, applied to the presence feature. Given a similarity relation of viable inputs for the word, the learner selects the one at the top of the relation, the input with greatest similarity to the output, what could be called the *maximal match input*. For an unset presence feature of a segment, this involves assigning the value reflecting the presence/absence of the segment's potential output correspondent, as is depicted in steps 2 and 3 of Figure 6.

The procedure *check for ranking information* is given a single word, and attempts to get ranking information from it. The procedure *non-phonotactic ranking information*, shown in Figure 7, searches the learner's stored words for words which meet certain criteria, and then calls *check for ranking information* on each of those words.

check for ranking information

1. given a *word*
2. let *match_input* be a copy of the input for *word*
3. for each segment *seg* of *match_input* with an unset presence feature:
   let *unset_f* be the presence feature of *seg*
   if *seg* has an output correspondent in *word*:
   assign *unset_f* the value +presence
   else
   assign *unset_f* the value –presence
   end
   end
4. construct *winner* with *match_input* and the output of *word*
5. while an informative *loser* can be found:
   form a winner-loser *pair* from *winner* and *loser*
   add *pair* to the learner's support of ranking information
   end
6. return

Figure 6: Outline of check for ranking information.

non-phonotactic ranking information

1. given a morpheme *morph* and an underlying segment *seg*
2. let *feature* be the presence feature of *seg*
3. construct a *list* of all stored words containing *morpheme* where either:
   (*feature* has value +presence and *seg* has no output correspondent)
   or
   (*feature* has value –presence, and
   seg's potential output correspondent exists in the output)
   end
4. for each *word* in *list*:
   check for ranking information with *word*
   end
5. return

Figure 7: Outline of non-phonotactic ranking information.

Step 5 of Figure 6 searches for an informative loser. The learner takes the input of the winner and parses it with their constraint ranking. If the optimal candidate is not the winner, then it is an informative loser; otherwise, the learner decides that it cannot find one.[4]

When a feature has been newly set, *non-phonotactic ranking information* is called specifically on the newly set feature. It looks specifically for words in which the newly set feature is not realized, that is, words where a disparity is necessarily introduced with respect to the newly set feature. For the presence feature specifically, if a potential input segment is set to +presence, this procedure looks for words in which that segment does not have an output correspondent (a deletion disparity). If a potential input segment is set to –presence, then the learner looks for words in which the hypothetical segment's

[4] See (Tesar 2014 and references therein) for details about the selection of informative losers and the computational properties of winner-loser pairs.

potential (but not actual) output correspondent does occur in the output (an insertion disparity). In general, obtaining non-phonotactic ranking information requires grammatical mappings with disparities.

8 Learning deletion and insertion

We will now see how the learner solves the problem of insertion and deletion with the data they are given during learning. Section 8.1 outlines phonotactic learning and what the learner can determine prior to consideration of morphemic identity. Section 8.2 is an illustration of learning about deletion for a particular segment, while 8.3 is an illustration of learning insertion for a particular segment. These examples will also demonstrate how learning the value of the presence feature for these particular segments makes it possible for the learner to obtain additional, non-phonotactic ranking information. In particular, the non-phonotactic ranking information results in a grammar that more generally applies deletion or insertion in the relevant contexts.

8.1 Phonotactic learning

The learning examples, in upcoming sections 8.2 and 8.3, involve different grammars within BST. While the two grammars require different constraint rankings, one thing they have in common is basic phonotactics: the languages for both grammars include syllables with onsets and syllables without onsets, and syllable codas are not permitted. The raw phonotactic outputs (that is, lacking any indication of morphemic affiliation) for both languages include the basic word forms in (13).

(13) [V], [CV], [V.CV], [V.V]

The learner constructs a set of phonotactic ranking information from these outputs. Because this learning is strictly phonotactic from the start, the learner assumes that the input for [V] is identical to [V]. The learner then accumulates a list of winner ~ loser pairs. A winner ~ loser pair consists of the targeted, grammatical output (the winner) and an output that competes with that winner based on current grammatical knowledge (the loser). The winner-loser pairs constructed from the observed form [V] are given in (14).

(14) Phonotactic ranking information

Word	Input	W~L	Onset	NoCoda	Max	DepC	DepV
r1	V	V ~ CV	L	e	e	W	e
r1	V	V ~ ∅	L	e	W	e	e

The symbol W marks constraints preferring the winner of the pair, the symbol L marks constraints preferring the loser of the pair, and the symbol e indicates that the constraint assigns an equal number of violations to the winner and loser. The W~L column indicates the output forms for the winner (left of the tilde) and the loser (right of the tilde). The symbol ∅ denotes a form (either input or output) with no segments. In (14), the first winner-loser pair indicates that the candidate which surfaces faithfully as [.V.] must beat the candidate in which a consonant is inserted into a syllable onset [.CV.], while the second winner-loser pair indicates that the candidate which surfaces faithfully as [.V.] must beat the candidate in which the input vowel is simply deleted, leaving nothing [∅]. Notice that this grammar prefers the violation of Onset rather than the violation of both DepC and Max. From this ranking information, the learner can use Biased Constraint Demotion (BCD) to determine the most restrictive ranking consistent with the information (Prince & Tesar 2004). The resulting constraint ranking is shown in (15).

(15) NoCoda ≫ {Max, DepC} ≫ Onset ≫ DepV

In the examples given in sections 8.2 and 8.3, phonotactic learning has resulted in the ranking information shown in (14), and the examples continue from that point.

8.2 Learning example 1: Deletion

8.2.1 Learning data for example 1

The target language in example 1 allows syllables with and without onsets, but forbids syllable codas. The ban on codas is enforced via consonant deletion. The grammar will delete a consonant rather than allow a coda or insert a vowel to create a new syllable.

This example focuses on two morphologically-related words: the root r1 in isolation, and root r1 combined with suffix s1. The words, with their outputs and morphemic affiliations, are shown in (16).

(16) Learning data for example 1

Word	Output	Morphemic Affiliation	
r1	[V]	r1: V	
r1s1	[V.CV]	r1: VC	s1: V

The morphemic affiliation given for word r1s1 indicates that the first two segments of the output, VC, are affiliated with r1; they constitute the surface realization of r1 in this context. The final segment of the output, V, is affiliated with the suffix s1.

The data include one key morphemic alternation: r1 surfaces as V in one context, and VC in another. Learning the grammar requires addressing two closely related matters: determining if the underlying form for r1 includes the C that appears in r1s1, and determining if the constraint ranking results in consonant deletion in the bare root context or results in consonant insertion in the suffixed context.

8.2.2 Processing word r1

Morpheme r1 is processed for the first time in word r1, surfacing as [V], so a new lexical entry is created, with an underlying form containing a segmental representation for the solitary segment [V] in the surface realization. The constructed underlying segmental representation, /(?,V)/, has its presence feature unset. The other content of the segment, /V/, matches the surface realization, [V], because in BST an underlying C cannot correspond to a surface V.

(17) Example 1 lexicon after creation of an entry for morpheme r1

Morpheme	Underlying Form
r1	/(?,V)/

The input for word r1 is the same as the underlying form for morpheme r1, and it has exactly one unset feature. The learner proceeds to test the unset feature, to see if it can be set. Because the segment is present on the surface in word r1, the structure of output-driven maps ensures that adopting an underlying value of +presence for the feature will be consistent: that would result in the input being identical to the output. The learner tests the value of the feature that is opposite its surface realization: –presence. Because the segment in question is the only segmental representation in the input, assigning a value of –presence to the feature is equivalent to using an input with no segments. The candidate to be tested for consistency is shown in (18). The underlying segmental representation that is the target of testing is emphasized in bold.

(18) /(–,V)/ = ∅ → [V]

Figure 8 shows part of the relative similarity order for the word r1. In section 6.3, we discussed how the learner considers candidates that have a potentially corresponding output segment. Because the only surfacing segment for r1 observed so far is a single V, the relevant similarity sublattice for [V] consists of the fully faithful input /V/ and the input /∅/ with a single insertion disparity, as is illustrated by the solid line between /∅/ and /V/ in Figure 8. The other nodes in the second row illustrate the candidates with single deletion disparities for V, which are not currently relevant candidates for r1, and which the learner does not consider. Therefore, the learner only tests /∅/ → [V] for consistency.

To test the candidate in (18), the learner selects, as an informative loser, ∅ → ∅, and forms the winner-loser pair shown in the bottom row of (19).

(19) The candidate ∅ → [V] is inconsistent.

Input	W ~ L	Onset	NoCoda	Max	DepC	DepV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
∅	V ~ ∅	L	e	e	e	L

The result is inconsistency. While (19) shows the tested winner-loser pair (below the thick black line) along with the learner's support, in this instance the tested winner-loser pair is inconsistent all on its own:[5] while there are two constraints that prefer the loser, there are no constraints that prefer the winner, so the highest-ranked constraint with a preference will select the loser over the winner no matter how the constraints are ranked. The hypothesized winner cannot possibly win.

The inconsistent candidate resulted from assigning the value –presence to the underlying presence feature, and the inconsistency justifies setting the feature to +presence in the lexicon, resulting in the lexicon shown in (20).

(20) Example 1 lexicon after feature setting for r1

Morpheme	Underlying Form
r1	/(+,V)/

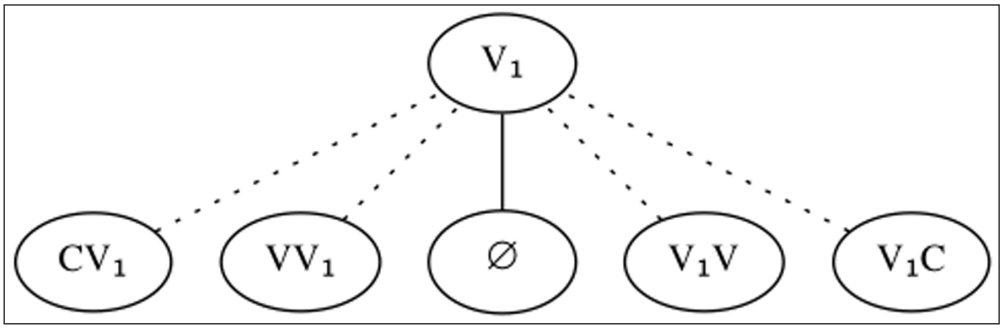

Figure 8: A partial similarity order for r1, surfacing as [V]. The dashed lines are to candidates with single unmotivated deletion disparities. The learner need not bother with these (nor with candidates having multiple unmotivated deletion disparities), and only tests the insertion disparity candidate, with input /∅/, connected by the solid line.

[5] It is equivalent to a trivially false elementary ranking condition (Prince 2002).

Morpheme r1 does not alternate with respect to the presence feature (it surfaces with a solitary V in every observed environment), so no non-phonotactic ranking information can be obtained via this feature.

The data considered thus far (the word r1) do not warrant positing any other segments, so feature setting for morpheme r1 is complete *for now*.

8.2.3 Processing word r1s1

Word r1s1 has output [V.CV], with morphemic affiliations of r1 = [VC] and s1 = [V].

The learner matches the lexical entry for r1 to the surface realization of r1 in r1s1. The V of the underlying form for r1 is matched to the V of the surface realization, and then a new segment entry is added to the underlying form of r1 for the C following the V in the surface realization of r1. The added segmental representation is (?,C), with its presence feature initially unset.

Morpheme s1 is observed for the first time in word r1s1, surfacing as [V], so a new lexical entry is created, with /(?,V)/. The result is the lexicon shown in (21).

(21) Example 1 lexicon after creation of an entry for morpheme s1

Morpheme	Underlying Form
r1	/(+,V)(?,C)/
s1	/(?,V)/

The input for word r1s1 is the combination of the underlying forms for morphemes r1 and s1, /(+,V)(?,C); (?,V)/, where the semi-colon indicates the boundary between the morphemes. The learner proceeds to test each of the unset features in turn.

First, the learner tests the presence feature for the second underlying segment of root r1. Because the segment is present on the surface in word r1s1, the test value of the feature is –presence. The other unset feature (the vowel of s1) is temporarily assigned the value matching its surface realization, +presence. The candidate to be tested for consistency is shown in (22), and the viable similarity relation is in Figure 9.

(22) r1s1: /(+,V)(**–,C**); (+,V)/ = /VV/ → [V.CV]

Because the presence feature of /V/ in r1s1 has been set to +presence, it remains a constant throughout the sublattice. In the top node, /V/ is accompanied by the other

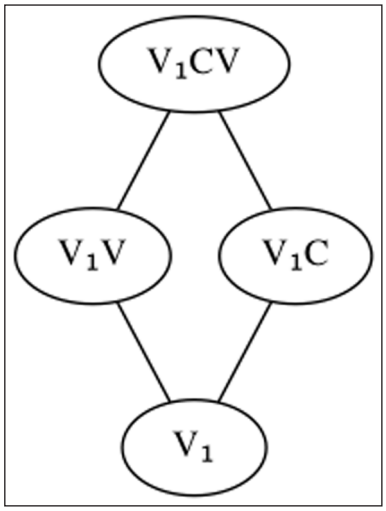

Figure 9: Viable similarity relation for r1s1.

segments that surface in r1s1. In the second row, each node contains a single insertion disparity. The bottom node contains two deletion disparities, leaving the single set segment /V/.

The learner selects, as an informative loser, /VV/ → [V.V], and forms the winner-loser pair shown in the bottom row of (23).

(23) The candidate /VV/ → [V.CV] is inconsistent with the learner's support

Input	W ~ L	ONSET	NOCODA	MAX	DEPC	DEPV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
VV	V.CV ~ V.V	W	e	e	L	e

The result is inconsistency. The ranking requirements of the tested winner-loser pair contradict the learner's existing ranking information, as highlighted by the shaded cells in (23). As a consequence, the learner sets the underlying value of the target feature to the surface-matching value +presence in the lexicon, resulting in the lexicon in (24).

(24) Example 1 lexicon after first feature setting for r1s1

Morpheme	**Underlying Form**
r1	/(+,V)(+,C)/
s1	/(?,V)/

Morpheme r1 *does* alternate with respect to the presence feature of its second segment. This creates an opportunity for the learner to pursue non-phonotactic ranking information.

8.2.4 Non-phonotactic ranking information

Word r1s1 was the basis for creating a segmental representation of C in r1, and for setting that segment's presence feature to +presence. The learner now returns to word r1, where the morpheme r1 surfaces without a corresponding C. The lexicon now requires an input of /VC/, and the output for word r1 is [V]. Therefore, the candidate /VC/ → [V] must be grammatical. The learner determines that /VC/ → [VC] is an informative loser, forms a winner-loser pair, and adds it to the learner's permanent support (ranking information). This is not information generated for purposes of inconsistency detection. The learner knows that this ranking information *must* be true, and adds it to the permanent support.

The learner then determines that there is another informative loser, /VC/ → [V.CV], and it forms another winner-loser pair, resulting in the support depicted in (25). The non-phonotactic ranking information is given in the bottom two rows (the winner is not identical to the loser in these pairs).

(25) Example 1 the learner's support after obtaining non-phonotactic ranking information

Word	Input	W ~ L	ONSET	NOCODA	MAX	DEPC	DEPV
r1	V	V ~ CV	L	e	e	W	e
r1	V	V ~ ∅	L	e	W	e	e
r1	VC	V ~ VC	e	W	L	e	e
r1	VC	V ~ V.CV	e	e	L	e	W

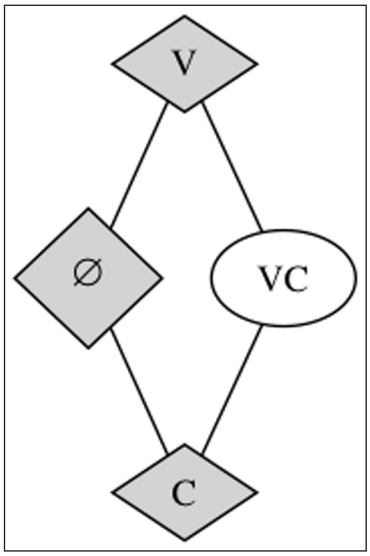

Figure 10: Similarity relation for the ranking information of r1.

The top node of Figure 10 is no longer a viable candidate for r1 because /C/ has been set to +presence underlying. Because both /V/ and /C/ have now been set to +presence underlyingly, the other nodes for r1 are non-viable candidates. So, node /VC/ becomes the top node of the viable sublattice by default, while the others are now diamond shaped and shaded.

(26) NoCoda ≫ {DepC, DepV} ≫ Max ≫ Onset

The first of the new winner-loser pairs, /VC/ V ~ VC, indicates that the grammar will delete a consonant rather than allow it in a syllable coda, providing the explanation for the lack of syllable codas in the language. The second of the new winner-loser pairs, /VC/ V ~ V.CV, indicates that the grammar will delete a consonant rather than insert a vowel to allow the consonant into the onset of a following syllable. These two pairs together determine a general pattern of deletion. While learned on the basis of words r1 and r1s1, the resulting grammar will apply this pattern of deletion generally in the language. The most restrictive constraint ranking consistent with the learner's support is given in (26).

At this point, there are no more informative losers available for the candidate /VC/ → [V]. The learner then returns to the processing of unset features in word r1s1.

8.2.5 Resume processing of r1s1

At this point, the input for word r1s1 is /(+,V)(+,C); (?,V)/. The learner now tests the unset presence feature for suffix s1. Because the segment is present on the surface in word r1s1, the test value of the feature is –presence. The candidate to be tested for consistency is shown in (27).

Figure 11 shows the same similarity relation for r1s1 as Figure 9, updated to reflect the setting of the presence feature for the underlying C in r1, as was justified in (23). The nodes that are now diamond shaped and shaded are such because they are now non-viable. This is because they conflict with the learner's current lexicon, which includes /C/ as +presence for r1.

(27) /(+,V)(+,C); (**–**,**V**)/ = /VC/ → [V.CV]

The learner selects, as an informative loser, /VC/ → [V], and forms the winner-loser pair shown in the bottom row of (28).

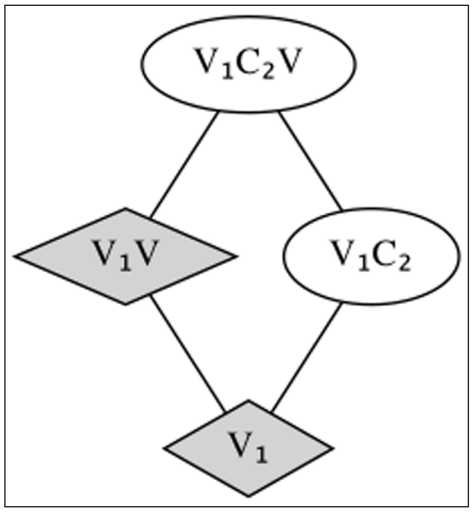

Figure 11: Similarity relation for r1s1.

(28) The candidate /VC/ → [V.CV] is inconsistent with the learner's support.

Input	W~L	ONSET	NOCODA	MAX	DEPC	DEPV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
VC	V ~ VC	e	W	L	e	e
VC	V ~ V.CV	e	e	L	e	W
VC	V.CV ~ V	e	e	W	e	L

The result is inconsistency. In particular, the tested winner-loser pair directly contradicts the second piece of the non-phonotactic ranking information, as highlighted by the shaded cells. The non-phonotactic ranking information has made it possible to set the underlying presence feature for suffix s1. As a consequence, the learner sets the underlying value of the presence feature to the surface-matching value +presence in the lexicon, resulting in the lexicon in (29).

(29) Example 1 final lexicon

Morpheme	**Underlying Form**
r1	/(+,V)(+,C)/
s1	/(+,V)/

Morpheme s1 does not alternate, so no new non-phonotactic ranking information can be obtained.

At this point, the learner has succeeded. The lexicon in (29) is sufficient, and in fact every feature has been correctly set. The support in (25), and its corresponding constraint ranking in (26), correctly map the inputs for both words to the correct outputs.

In particular, the learner has successfully determined that r1 has an underlying consonant, even though it is deleted in word r1. Learning this required combining information from the related words r1 and r1s1, and the persistent linking representation between the two was the presence feature on the consonant for r1. Setting the C in the underlying form of r1 to +presence enables to learner, upon reexamination of word r1, to see the that C is deleted in word r1, and to obtain non-phonotactic ranking information indicating that C's must delete to avoid codas generally in the language.

8.3 Learning example 2: Insertion

8.3.1 Learning data for example 2

The target language in example 2 allows syllables with and without onsets, but forbids syllable codas. The ban on codas is enforced via vowel insertion. The grammar will insert a vowel after a consonant to create a new syllable rather than allow a coda or delete a consonant.

This example focuses on four morphologically-related words: the root r1 in isolation, the root r2 in isolation, and each of the roots combined with suffix s1. The words, with their outputs and morphemic affiliations, are shown in (30).

(30) Learning Data for Example 2

Word	**Output**	**Morphemic Affiliation**	
r1	[V]	r1: V	
r1s1	[V.V]	r1: V	s1: V
r2	[V.CV]	r2: VCV	
r2s1	[V.CV]	r2: VC	s1: V

The data include one key morphemic alternation: r2 surfaces as VCV in one context, and VC in another. Learning the grammar requires addressing two closely related matters: determining if the underlying form for r2 includes the final V that appears in r2, and determining if the constraint ranking results in vowel deletion in the suffixed context or results in vowel insertion in the bare root context.

8.3.2 Processing word r1

Morpheme r1 is processed for the first time in word r1, surfacing as [V], so a new lexical entry is created, with an underlying form containing a segmental representation for the solitary segment [V] in the surface realization.

(31) Example 2 lexicon after creation of an entry for morpheme r1

Morpheme	**Underlying Form**
r1	/(?,V)/

The learner proceeds to test the unset feature of r1, to see if it can be set. The candidate to be tested for consistency is shown in (32).

(32) /(**–,V**)/ = ∅ → [V]

The learner selects, as an informative loser, ∅ → ∅, and forms the winner-loser pair shown in the bottom row of (33).

(33) The candidate ∅ → [V] is inconsistent.

Input	W~L	Onset	NoCoda	Max	DepC	DepV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
∅	V ~ ∅	L	e	e	e	L

The result is inconsistency. This is a repeat of the evaluation at the beginning of Example 1. The same winner-loser pair is inconsistent, for the same reason, with the same result: the underlying presence feature for r1 is set to +presence, resulting in the lexicon shown in (34).

(34) Example 2 lexicon after feature setting for r1

Morpheme	Underlying Form
r1	/(+,V)/

Morpheme r1 does not alternate with respect to the presence feature (it surfaces with a solitary V in every observed environment), so no non-phonotactic ranking information can be obtained via this feature.

8.3.3 Processing word r1s1

Word r1s1 has output [V.V], with morphemic affiliations of r1 = [V] and s1 = [V].

The learner matches the lexical entry for r1 to the surface realization of r1 in r1s1. Every segment of the surface realization has a matching possible segment in the underlying form, so the learner adds no new segmental representations to the underlying form.

Morpheme s1 is observed for the first time in word r1s1, surfacing as [V], so a new lexical entry is created, with /(?,V)/. The result is the lexicon shown in (35).

(35) Example 2 lexicon after creation of an entry for morpheme s1

Morpheme	Underlying Form
r1	/(+,V)/
s1	/(?,V)/

The input for word r1s1 is /(+,V); (?,V)/. The learner tests the presence feature for s1. Because the segment is present on the surface in word r1s1, the test value of the feature is –presence. The candidate to be tested for consistency is shown in (36).

(36) /(+,V); (**–,V**)/ = /V/ → [V.V]

The learner selects, as an informative loser, /V/ → [V], and forms the winner-loser pair shown in the bottom row of (37).

(37) The candidate /V/ → [V.V] is inconsistent.

Input	W ~ L	Onset	NoCoda	Max	DepC	DepV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
V	V.V ~ V	L	e	e	e	L

The result is inconsistency. In BST, there is no grammatical mechanism to motivate inserting an entire syllable, with no segments of the syllable having input correspondents. As a consequence, the learner sets the underlying value of the target feature to the surface-matching value +presence in the lexicon, resulting in the lexicon in (38).

(38) Example 2 lexicon after feature setting for s1

Morpheme	Underlying Form
r1	/(+,V)/
s1	/(+,V)/

Morpheme s1 does not alternate with respect to its presence feature, so no non-phonotactic ranking information can be obtained via this feature.

8.3.4 Processing word r2

Morpheme r2 is observed for the first time in word r2, surfacing as [V.CV], so a new lexical entry is created, with an underlying form containing a segmental representation for each segment of the surface realization.

(39) Example 2 lexicon after creation of an entry for morpheme r2

Morpheme	Underlying Form
r1	/(+,V)/
r2	/(?,V)(?,C)(?,V)/
s1	/(+,V)/

The learner proceeds to test the first unset feature of r2, to see if it can be set. The candidate to be tested for consistency is shown in (40).

(40) /(**–,V**)(+,C)(+,V)/ = /CV/ → [V.CV]

The learner selects, as an informative loser, /CV/ → [CV], and forms the winner-loser pair shown in the bottom row of (41).

(41) The candidate /CV/ → [V.CV] is inconsistent.

Input	W~L	Onset	NoCoda	Max	DepC	DepV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
CV	V.CV ~ CV	L	e	e	e	L

The result is inconsistency. There is no grammatical motivation for inserting the initial vowel. As a consequence, the learner sets the underlying value of the target feature to the surface-matching value +presence in the lexicon, resulting in the lexicon in (42).

(42) Example 2 lexicon after setting the first feature of r2

Morpheme	Underlying Form
r1	/(+,V)/
r2	/(+,V)(?,C)(?,V)/
s1	/(+,V)/

The first segment of morpheme r2 does not alternate with respect to its presence feature, so no non-phonotactic ranking information is obtained at this time.

Next, the learner tests the second feature of r2, to see if it can be set. The candidate to be tested for consistency is shown in (43).

(43) /(+,V)(**–,C**)(+,V)/ = /VV/ → [V.CV]

The learner selects, as an informative loser, /VV/ → [V.V], and forms the winner-loser pair shown in the bottom row of (44).

(44) The candidate /VV/ → [V.CV] is inconsistent.

Input	W~L	ONSET	NOCODA	MAX	DEPC	DEPV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
VV	V.CV ~ V.V	W	e	e	L	e

The result is inconsistency. In particular, the tested winner-loser pair directly contradicts the first piece of the non-phonotactic ranking information, as highlighted by the shaded cells. As a consequence, the learner sets the underlying value of the target feature to the surface-matching value +presence in the lexicon, resulting in the lexicon in (45).

(45) Example 2 lexicon after setting the second feature of r2

Morpheme	Underlying Form
r1	/(+,V)/
r2	/(+,V)(+,C)(?,V)/
s1	/(+,V)/

The second segment of morpheme r2 does not alternate with respect to its presence feature, so no non-phonotactic ranking information is obtained at this time.

Next, the learner tests the second feature of r2, to see if it can be set. The candidate to be tested for consistency is shown in (46).

(46) /(+,V)(+,C)(**–,V**)/ = /VC/ → [V.CV]

The learner selects, as an informative loser, /VC/ → [V], and forms the winner-loser pair shown in the bottom row of (47).

(47) The candidate /VC/ → [V.CV] is consistent.

Input	W~L	ONSET	NOCODA	MAX	DEPC	DEPV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
VC	V.CV ~ V	e	e	W	e	L

The result is consistent. All three winner-loser pairs in (47) are consistent with the constraint ranking shown in (48). Furthermore, under that ranking, the candidate in (46), /VC/ → [V.CV], is optimal: it is more harmonic than all of its competitors, not just the loser /VC/ → [V]. The learner is not guaranteed that the final winner-loser pair in (47) is correct, or that the ranking in (48) is correct. The learner only knows at this point that they are possibilities at this stage, because they are consistent with the learner's current knowledge of the target grammar.

(48) {NOCODA, MAX, DEPC} ≫ {ONSET, DEPV}

Because setting the presence feature of the final V to –presence, i.e., omitting the final V from the input, is consistent, the learner cannot be sure if that final V is part of the input or not, so the presence feature is not set. However, the segment, with its unset presence feature, remains part of the underlying form for r2; it is the potential input correspondent for the final vowel of the surface realization of r2 as a word, and other evidence may

help determine the value of its presence feature. Thus, the learner's lexicon remains as depicted in (45).

8.3.5 Processing word r2s1

Word r2s1 has output [V.CV], with morphemic affiliations of r2 = [VC] and s1 = [V].

The learner matches the lexical entries for r2 and s1 to the surface realization. One aspect of the matching isn't fully obvious: does the final V of the output get matched to the necessarily underlying V of s1, or the potential V at the end of r2? A full discussion of this would involve how morphemic segmentation is computed and learned, which is outside the scope of this paper. Here, we will simply take it that the learner has determined that the suffix is in fact realized on the surface here. None of the output segments lacks a potential input correspondent, so none of the lexical entries are immediately modified, and the lexicon is still as was depicted in (45).

The input for word r2s1 is currently /(+,V)(+,C)(?,V); (+,V)/. The relative similarity relation for r2s1 is shown in Figure 12. The learner tests the sole unset presence feature in the input: the final segment of the underlying form of r2. Because the segment does not have a surface correspondent in this word, the test value of the feature is +presence. The candidate to be tested for consistency is shown in (49).

(49) /(+,V)(+,C)(+,**V**); (+,V)/ = /VCVV/ → [V.CV]

All underlying forms that have output correspondents for r2s1 have segments that have been set to +presence. Because /?V/ in morpheme r2 has surfaced in another context, the learner posits it as a deletion disparity here by testing it as /+V/.

The learner selects, as an informative loser, /VCVV/ → [V.CV.V], and forms the winner-loser pair shown in the bottom row of (50).

(50) The candidate /VCVV/ → [V.CV] is inconsistent.

Input	W~L	ONSET	NOCODA	MAX	DEPC	DEPV
V	V ~ CV	L	e	e	W	e
V	V ~ ∅	L	e	W	e	e
VCVV	V.CV ~ V.CV.V	W	e	L	e	e

The result is inconsistency. In particular, the tested winner-loser pair directly contradicts the second piece of the non-phonotactic ranking information, as highlighted by the shaded cells. As a consequence, the learner sets the underlying value of the target feature to the surface-matching value –presence in the lexicon, resulting in the lexicon in (51).

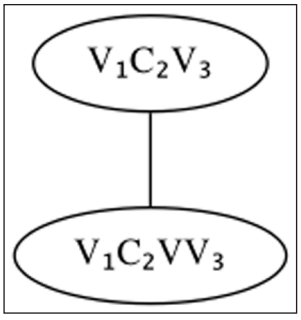

Figure 12: Relative similarity relation for r2s1.

(51) Example 2 lexicon after setting the third feature of r2

Morpheme	Underlying Form
r1	/(+,V)/
r2	/(+,V)(+,C)(–,V)/
s1	/(+,V)/

8.3.6 Non-phonotactic ranking information

Word r2s1 set the presence feature of the second V (the final underlying potential segment) of r2 to –presence. Morpheme r2 alternates with respect to the presence of a second V, as the morpheme surfaces with a second V in the word r2. For that reason, the learner now returns to word r2. The lexicon now requires an input of /VC/, and the output for word r2 is [V.CV]. This is illustrated in Figure 13, which shows the similarity relation for word r2. Now that the final input /V/ has been set to –presence, the top node of the similarity relation for r2 is no longer viable. The viable relation solely consists of the bottom node, with input /VC/. That candidate is the top (and only) candidate of the viable similarity relation. Therefore, the candidate $/V_1C_2/ \rightarrow [V_1.C_2V]$ must be grammatical. This reveals an instance where an insertion disparity is forced.

The learner then tests the candidate $/V_1C_2/ \rightarrow [V_1.C_2V]$ to see if any additional ranking information can be obtained. The learner determines that $/V_1C_2/ \rightarrow [V_1C_2]$ is an informative loser, forms a winner-loser pair, and adds it to the learner's permanent support (ranking information), shown in (52).

(52) Example 2 the learner's support after obtaining non-phonotactic ranking information

Word	Input	W~L	Onset	NoCoda	Max	DepC	DepV
r1	V	V ~ CV	L	e	e	W	e
r1	V	V ~ ∅	L	e	W	e	e
r2	VC	V.CV ~ VC	e	W	e	e	L

The new winner-loser pair indicates that the grammar will insert a vowel after an consonant rather than allow that consonant into a syllable coda. The most restrictive constraint ranking consistent with the learner's support is given in (53).

(53) NoCoda ≫ {Max, DepC} ≫ Onset ≫ DepV

At this point, the learner has succeeded. The lexicon in (51) is sufficient, and in fact every feature has been correctly set. The support in (52), and its corresponding constraint rank-

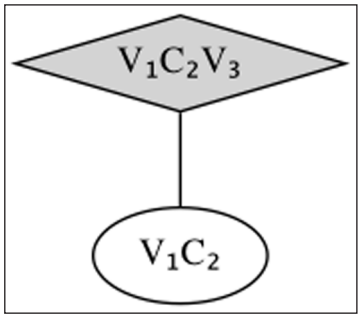

Figure 13: The similarity relation for r2; the only viable candidate has input /VC/, and therefore an insertion disparity of the final vowel.

ing in (53), correctly map the inputs for both words to the correct outputs. In particular, the learner has successfully determined that the second V in word r2 [V.CV] is inserted, not present underlyingly. Learning this required combining information from the related words r2 and r2s1, and the persistent linking representation between the two was the presence feature on the second vowel for r2. Setting the second V in the underlying form of r2 to –presence enables to learner, upon reexamination of word r2, to see that the second V in the output of word r2 is inserted, and to obtain non-phonotactic ranking information indicating that V's must be inserted to avoid codas generally in the language.

9 Discussion

9.1 The role of output-drivenness

Prior work on learning with output-driven maps focused on systems with identity disparities, where the space of possible inputs for a given output is strictly finite but grows exponentially in the size of the output. The structure of output-driven maps was used to compute efficiently over the space of inputs, needing to only construct and evaluate a small number out of the many possible candidates. The present work focuses on deletion and insertion disparities. Allowing for unbounded insertion and deletion results in a space of inputs for an output that is countably infinite. As shown in section 6, the very same structure of output-driven maps, with a straightforward interpretation of insertion and deletion in relative similarity, permits this space of inputs to be reasoned over, by guaranteeing that the instances of deletion that need be considered can be limited to segments corresponding to at least one observed surface realization of a morpheme.

Reifying the learner's knowledge about segmental presence/absence in the form of the presence feature allows the existing learning algorithm for setting underlying features to be extended straightforwardly to determining the presence/absence of particular segments via inconsistency detection. As with other segmental features, the presence feature allows the learner to combine information across different surface realizations of the same morpheme (morphemic alternations), combining the information into a single location (the lexical entry for the morpheme). When a segment is present in one surface realization but not in another, the learner can determine if the segment is necessarily underlyingly present in one of the environments where it surfaces, or if it is necessarily underlyingly absent in one of the environments where it does not surface. Either way, determined necessity allows the presence feature to be set, which then generalizes to the other surface realizations, providing direct evidence of deletion if the segment is underlyingly present, and direct evidence of insertion if the segment is underlyingly absent. Such evidence gives rise straight away to further ranking information, which then generalizes to other words with other morphemes.

9.2 Limitations and uncertainties

The learner proposed here builds on the existing Output-Driven Learner, and inherits many of its demonstrated results, known limitations, and uncertainties. The original Output-Driven Learner was shown, via simulations, to learn all 24 languages of a particular typology for stress and vowel length. The formal limits on the range of linguistic systems the algorithm can learn are not yet well understood, however. That uncertainty remains when the present proposal for insertion and deletion is added. Simulations specifically with Basic CV Syllable Theory will likely be informative. This will require further elaboration on the representation of correspondence across surface forms, to be addressed in future work.

One known difficulty for inconsistency-based approaches is perfect correlation of two features when either is capable of being contrastive. If feature A and feature B are always both +, or both –, in all environments of the language, the grammar is ordinarily expected

to enforce that phonotactic pattern. If the enforcement mechanism is symmetric (such as with a traditional AGREE constraint), then a normal solution would be for one of the features to be "contrastive", and the other to be modified as necessary to agree. But if either feature could serve as the contrastive one, then the nature of inconsistency detection prevents the learner from committing to either. Inconsistency detection only sets a feature to a value when the opposite value cannot work. It won't set feature A to match the surface form, because it could be that feature A doesn't match and is harmonized to feature B, and vice-versa. There is no mechanism for "simply picking one" of the features to set. The linguistic plausibility of such a situation has yet to be investigated in depth.

9.3 Simultaneously learning the presence feature and other segmental features

While the system used in the present study, Basic CV Syllable Theory, abstracts away from other segmental features, reintroducing such features will put the presence feature alongside the other features of an underlying segment, and the learner will be able to work on setting all of them via the same method based on output-drivenness and inconsistency detection. There are, however, some issues regarding the combination that aren't immediately obvious.

9.3.1 Unset features and candidate evaluation

The original Output-Driven Learner only evaluated candidates that were fully specified, in the sense that any feature values unset underlyingly are assigned temporary values for the purpose of evaluation. That stance is maintained here in the handling of presence features. When a candidate is evaluated for consistency, for example, each presence feature is assigned a value. The treatment of the presence feature perfectly parallels the earlier treatment of other features, but it has some subtle consequences that are different. Because deletion was not considered in the earlier work, any segment that existed in an underlying form for a morpheme had an output correspondent in any word containing that morpheme. Every segment, and thus every feature of every segment, was necessarily present, and each feature could be reasoned about independently. But if a presence feature is given the value –presence, either permanently set or temporarily assigned, then the entire segment is absent from the input (and from the entire candidate). This impacts the other features of the segment: they are not "present" in the candidate, and are not considered in the evaluation of the candidate. An underlying segment assigned the value –presence vacuously satisfies MAX; it does not constitute a deletion disparity. An underlying segment assigned the value –presence also has no output correspondent, and therefore vacuously satisfies IDENT constraints, regardless of what feature values have been assigned to the other features of the segment. Other features of an underlying segment, as represented in the lexicon, are only part of a candidate if the presence feature for that segment has the value +presence. In this sense, the other features are dependent on the presence feature.

Evaluating candidates with features that have no value can sometimes introduce complications, as has been shown by Magri (2018). In particular, Magri defines a partial feature to be one which can be evaluated by constraints without having an assigned value; these are distinguished from total features, which always have an assigned value when evaluated by constraints. Magri shows that an IDENT constraint can lead to maps which are not output-driven if it does not assign a violation to some candidates where a feature with no assigned value has an IO corresponding feature with an assigned value. Those complications do not arise here; the circumstance of evaluating a candidate in which an underlying feature has no value does not arise. In the terms of Magri, the proposal described in this paper restricts itself to total features.

9.3.2 Possibly deleted segments with unset features

Learning via inconsistency detection can set the value of an underlying feature by determining that none of the alternative values for that feature are consistent. Thus, the learner sets a presence feature to –presence if it can show that the value +presence is inconsistent. An example of this was shown in (49) of section 8.3.5. The third underlying segment of morpheme r2, a V with an unset presence feature, was posited because it corresponded to a V in the surface form of r2 in isolation. The surface form of r2s1, however, has no vowel corresponding to this underlying segment. The learner tests the possibility of this V being not present (–presence) by evaluating the candidate where the V is present underlyingly (+presence). The inconsistency of the +presence case allows the learner to set the feature for that vowel to –presence.

In the earlier learning work examining only identity disparities, if a (binary) feature could be set at all, it could be set via evaluation of a single alternative candidate, where the tested feature was assigned the value opposite its surface value, while any other unset features were assigned values matching their surface values. The logic supporting that approach changes when a segment that has no surface correspondent is considered. This is because there is in general no unique assignment of values to unset features that has greatest similarity: the learner cannot assign values that match the output realization, because the segment has no output realization (output correspondent) in the word. A deletion disparity is individuated on an entire segment, not feature by feature as with identity disparities. The deletion of an [n], for instance, is not the same type of disparity as the deletion of an [t], and relative similarity requires that corresponding disparities be identical; for extended discussion of this point, see (Tesar 2014: 55–57).

For the learner to rule out the possibility that a segment is present underlyingly but deleted on the surface, it needs to separately check all of the possible segments that could be the underlying segment, and show that none of them could be deleted, consistent with the rest of the grammar. In the example of (49) of section 8.3.5, there was only one option to check: the potential third underlying segment of r2 is a V, because the "V vs. C" contrast is not subject to identity violation, so the learner evaluates with the underlying segment (+,V). Suppose, instead, that the linguistic theory in use admitted the feature +/–tense for vowels (–tense being equivalent to lax). Each vowel would have two features, presence and tense. If the tense feature for the tested vowel in (49) had not yet been set, then the learner would have to separately evaluate (+presence, +tense, V) and (+presence, –tense, V), and could only set the vowel to –presence if both of those cases proved inconsistent. Given multiple such unset features, the learner would have to separately evaluate all combinations of values for the unset features. The soundness of the learner is unchanged, but the computational effort required could escalate.

There are counterpoints to this potential computational escalation, however. The vowel with the unset presence feature is only represented in the underlying form of r2 because in a different word, bare r2, it potentially corresponds to an output vowel. If that output vowel is –tense, and phonotactic learning has already determined that there is a contrast between +tense and –tense vowels in the language, then at the time the underlying vowel is posited, it will set the underlying feature to –tense, even if it cannot set the presence feature at that point. The learner is, in effect, representing that if the surface –tense vowel has an input correspondent, then that input must be set to –tense (if it were set to +tense, then the output vowel would necessarily have been +tense). Thus, it is plausible that some features of a segment may be set when it is initially posited even if the presence feature cannot be set. More set features for a segment means fewer unset features which means fewer combinations to test to determine if the segment is possibly not present underlyingly at all.

Another counterpoint would be a case where a vowel surfaces as +tense in one environment, and does not surface in another environment. If the learner's linguistic information to that point were such that it guaranteed that epenthetic vowels must be –tense, then any attempt to set the vowel to underlyingly –presence would fail: the grammar might insert a vowel, but not a +tense vowel. This would allow the learner to set the presence feature to +presence *as a consequence of* other segmental features, avoiding the need to test the presence feature in an environment in which the input segment has no output correspondent.

An analogous computational escalation is possible with the learning of non-phonotactic ranking information. In the example described in section 8.2.4, the learner has set the presence feature of the consonant in morpheme r1 to +presence, based on how the morpheme surfaces for word r1s1. However, the consonant does not surface when r1 has no suffix. The learner is able to obtain non-phonotactic ranking information because it knows that the C is underlyingly present, and can gain the ranking information necessary to ensure that it deletes in that context. The ranking information is based on the knowledge that the viable input with greatest similarity to the output must map to that output. Once again, when a deletion disparity appears, there is not guaranteed to be a unique input with greatest similarity if the deleted segment has unset features. Each possible assignment of values to the unset features of the deleted segment constitutes a separate input for the same output, and none of them will be related to another with respect to similarity. It is still possible for the learner to obtain ranking information at this point, by computing the additional ranking information resulting from each of the different possible inputs (assignments of values to the unset features of the deleted segment), and taking the join of those sets of ranking information, as defined by Merchant (2008). Again, however, this is more computational effort that having a single candidate from which to obtain ranking information. The common situation between this case and the feature-setting case described earlier is the effect of having an underlying segment with the presence feature assigned the value +presence and at least one feature (other than presence) unset, appearing in an environment where that segment is deleted.

Future research will need to examine what interesting interactions can occur between setting the presence feature and setting other features.

9.4 Global ambiguity of underlying forms

A general issue in the learning of underlying forms is the prevalence of global ambiguity, in which multiple underlying forms are equally adequate for a given morpheme. This is a natural consequence of neutralization and richness of the base: when a grammar enforces patterns by neutralizing multiple inputs to the same output, any of the neutralized inputs is empirically adequate. Inconsistency detection will not distinguish between multiple empirically adequate underlying forms. The consideration of insertion and deletion can amplify this issue, as it is possible to have an infinite set of neutralizing, empirically adequate underlying forms.

The current proposal combines two strategies to deal with different aspects of this global ambiguity. It inherits from prior work the strategy of using unset features to represent a space of possible inputs in a compact form, along with an approach that does not set an underlying feature to one value unless the other value(s) is shown to be inconsistent. Tolerance of unset features allows the learner to avoid forcing an arbitrary (and perhaps premature) setting of values to features. The other strategy, newly proposed here, is to only represent possible underlying segments that could possibly correspond to an actual output segment. This strategy deliberately prefers underlying forms that do not contain any unnecessary segments, relying on output-drivenness to justify their lack of necessity.

In the case of an infinite number of empirically adequate underlying forms for the same morpheme, the learner succeeds in eliminating from consideration all possible underlying

forms that would extend beyond the (necessarily finite) range of segments that actually surface for the morpheme. As demonstrated in the illustrations in section 8, the learner can adopt this approach without needing to wait until after they have observed all possible surface realizations: the learner can safely add hypothesized underlying segments during the course of learning in response to new surface realizations of a morpheme.

Successful learning can produce a potential underlying segment with an unset presence feature. This can happen when the feature is completely non-contrastive: the same output will result no matter which value is assigned to the feature, such as with the underlying C when /CV/→[CV] and /V/→[CV]. For each word containing the related morpheme, the learner will stop trying to set the feature once the word begins passing initial word evaluation (section 7.4). See (Tesar 2014: 385–390) for further discussion.

9.5 Insertion and morphemic identity

The learning algorithm used here operates under the idealization that the learner is provided with information about the morphemic structure of the observed words. In particular, each segment of a word is marked as being affiliated with a particular morpheme. This makes it possible for the learner to identify when the same morpheme is surfacing in different environments.

In earlier work that didn't involve insertion and deletion, the morphemic affiliation of each segment was unambiguous. The present work creates a slight complication: what is the morphemic affiliation of an inserted segment? Intuitively, it would seem that the correct answer would be that an inserted segment has no morphological affiliation: to be affiliated with a morpheme is to be in correspondence with some element of the underlying form of the morpheme. However, in the example in section 8.3, the inserted segment is marked as affiliated with a particular morpheme (specifically, the root r2). This was quite deliberate. Not marking a segment as being affiliated with any morpheme would be an overt indication that the segment was inserted, obviating the need for the learner to reason toward that conclusion based on the distributional evidence. As it stands, the learner is faced with data in which a segment marked as affiliated with a morpheme surfaces in one environment and not in another, and has to determine whether the segment is present underlyingly and deleted in one environment, or is not present underlyingly and is inserted in the other environment.

While marking epenthetic segments as affiliated with a particular morpheme seems reasonable for studies like the current one, it does indicate an issue that will be more fully addressed only by learning algorithms that engage in the learning of morphemic affiliation. Direct confrontation with learning morphemic affiliation in this linguistic framework is left for future research. It seems likely that such work will involve simultaneous learning of constraint ranking, morphemic affiliation, and morphemic underlying forms. In such an approach, the presence feature proposed in this paper could play a valuable role in the learning of morphemic affiliation when insertion and deletion are engaged in the language. Past computational research, completely apart from work in Optimality Theory, has suggested that knowing the phonotactic patterns of a language could be useful in computing morpheme segmentation in words (Brent & Cartwright 1997). The present work hints at an approach in which a learner, in possession of the results of phonotactic learning, learns non-phonotactic grammatical information in tandem with morphemic affiliation in words.

9.6 What has been accomplished

We have proposed that a learner represent information about the possible presence/absence of a segment in an underlying form via a presence feature. Along with the presence feature, we have proposed an extension of the Output-Driven Learner that uses the combination of

output-driven map structure and inconsistency detection to learn underlying forms when the presence/absence of underlying segments is not known in advance. Output-driven map structure allows the learner to only hypothesize underlying segments that would possibly correspond to an output segment in at least one surface realization of the morpheme, greatly limiting the number of possible underlying forms that are actively considered by the learner. The presence feature can be set using the same inconsistency detection method that has previously been used to set other segmental features, an approach that makes it possible to combine information across paradigmatically related words. Representing knowledge about the presence/absence of an underlying segment enables the learner to obtain non-phonotactic ranking information that enforces the inferred insertion and deletion disparities. The resulting learner is able to learn grammatical regularities about segmental insertion and deletion, based on paradigmatic evidence.

Abbreviations

BST = Basic CV Syllable Theory, C = consonant, V = vowel, BCD = biased constraint demotion

Acknowledgements

The authors wish to thank Alan Prince, the audience at the 2015 NECPhon conference, the audience at the 2015 GLEEFUL conference, and three Glossa reviewers for useful comments. All errors are exclusively the responsibility of the authors.

Funding Information

Thank you to the Aresty Research Center at Rutgers University, New Brunswick for conference funding and support.

Competing Interests

The authors have no competing interests to declare.

References

Apoussidou, Diana. 2007. *The learnability of metrical phonology*. Amsterdam: University of Amsterdam dissertation.

Brent, Michael R. & Timothy A. Cartwright. 1997. Distributional regularity and phonotactic constraints are useful for segmentation. In Michael R. Brent (ed.), *Computational approaches to language acquisition*, 93–125. Cambridge, MA: MIT Press.

Clements, George N. & Samuel Jay Keyser. 1983. *CV phonology*. Cambridge, MA: MIT Press.

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

Jakobson, Roman. 1962. *Selected writings 1: Phonological studies*. The Hague: Mouton.

Jarosz, Gaja. 2006. *Rich lexicons and restrictive grammars – maximum likelihood learning in Optimality Theory*. Baltimore, MD: The Johns Hopkins University dissertation. ROA-884.

Magri, Giorgio. 2018. Output-drivenness and partial phonological features. *Linguistic Inquiry* 49(3). 577–598. DOI: https://doi.org/10.1162/ling_a_00283

McCarthy, John J. & Alan Prince. 1995. Faithfulness and reduplicative identity. In Jill Beckman, Laura Walsh Dickey & Suzanne Urbancyzk (eds.), *University of Massachusetts Occasional Papers 18: Papers in Optimality Theory*, 249–384. Amherst, MA: GLSA, University of Massachusetts.

Merchant, Nazarré. 2008. *Discovering underlying forms: Contrast pairs and ranking*. New Brunswick, NJ: Rutgers University dissertation. ROA-964.

Merchant, Nazarré & Bruce Tesar. 2008. Learning underlying forms by searching restricted lexical subspaces. *Proceedings of the Forty-First Conference of the Chicago Linguistics Society (2005), vol. II: The Panels*, 33–47. ROA-811.

Prince, Alan. 2002. Entailed ranking arguments. Ms. Rutgers University. ROA-500.

Prince, Alan & Bruce Tesar. 2004. Learning phonotactic distributions. In René Kager, Joe Pater & Wim Zonneveld (eds.), *Constraints in phonological acquisition*, 245–291. Cambridge: Cambridge University Press.

Prince, Alan & Paul Smolensky. 2004. *Optimality Theory: Constraint interaction in generative grammar*. Malden, MA: Blackwell. DOI: https://doi.org/10.1002/9780470759400

Tesar, Bruce. 1995. *Computational Optimality Theory*. Boulder, CO: University of Colorado dissertation. ROA-90.

Tesar, Bruce. 2006. Learning from paradigmatic information. In Christopher Davis, Amy Rose Deal & Youri Zabbal (eds.), *Proceedings of the 36th Meeting of the North East Linguistics Society*, 619–638. Amherst, MA: GLSA, University of Massachusetts. ROA-795.

Tesar, Bruce. 2014. *Output-driven phonology* [Cambridge Studies in Linguistics]. Cambridge: Cambridge University Press.

Tesar, Bruce & Paul Smolensky. 1994. The learnability of Optimality Theory. In Raul Aranovich, William Byrne, Susanne Preuss & Martha Senturia (eds.), *Proceedings of the Thirteenth West Coast Conference on Formal Linguistics*, 122–137. CSLI.

How to cite this article: Nyman, Alexandra and Bruce Tesar. 2019. Determining underlying presence in the learning of grammars that allow insertion and deletion. *Glossa: a journal of general linguistics* 4(1): 37.1–41. DOI: https://doi.org/10.5334/gjgl.603

Submitted: 02 January 2018 **Accepted:** 27 November 2018 **Published:** 15 March 2019

]u[*Glossa: a journal of general linguistics* is a peer-reviewed open access journal published by Ubiquity Press. **OPEN ACCESS**