

Trees probe deeper than strings: an argument from allomorphy

Hossep Dolatian

Department of Linguistics
Stony Brook University
Stony Brook, NY, USA
hossep.dolatian@
alumni.stonybrook.edu

Shiori Ikawa

Department of English
Language and Culture,
Fuji Women's University, Japan
shiori.ikawa@fujijoshi.ac.jp

Thomas Graf

Department of Linguistics
Stony Brook University
Stony Brook, NY, USA
mail@thomasgraf.net

Abstract

Linguists disagree on whether morphological representations should be strings or trees. We argue that tree-based views of morphology can provide new insights into morphological complexity even in cases where the posited tree structure closely matches the surface string. Our argument is based on a subregular case study of morphologically conditioned allomorphy, where the phonological form of some morpheme (the target) is conditioned by the presence of some other morpheme (the trigger) somewhere within the morphosyntactic context. The trigger and target can either be linearly adjacent or non-adjacent, and either the trigger precedes the target (inwardly sensitive) or the target precedes the trigger (outwardly sensitive). When formalized as string transductions, the only complexity difference is between local and non-local allomorphy. Over trees, on the other hand, we also see a complexity difference between inwardly sensitive and outwardly sensitive allomorphy. Just as unboundedness assumptions can sometimes tease apart patterns that are equally complex in the finitely bounded case, tree-based representations can reveal differences that disappear over strings.

1 Introduction

Morphology can be taken to operate over either strings or trees. Consider the simple case of English *undoable*, which is ambiguous between *not doable* with *un-* scoping over *doable*, and *can be undone* with *-able* scoping over *undo*. If one's primary concern is morphotactics, i.e. how morphemes can be arranged to obtain a well-formed word, then it is sufficient to represent *undoable* as a string *un+do+able*, consisting of three morphemes in a particular order. But this representation does not encode the scopal relations between the affixes *un-* and *-able*. Linguists instead use trees to encode the scopal relations between the affixes *un-* and

-able, giving us *[un[do able]]* and *[[un do]able]* for each respective interpretation of *undoable*. But strings and trees are vastly different data structures that greatly affect computational complexity. For instance, every dependency that is context-free over strings is only regular over trees. This paper explores the typology of allomorphy to probe how the choice between strings and trees can affect morphological complexity. Our key insight is that even in cases where trees seem to add little over strings, trees can reveal complexity differences between empirical phenomena that are opaque at the string level.

Tree-based models are still rare in computational morphology, where morphological phenomena are usually modeled with finite-state machinery (Koskenniemi, 1983; Beesley and Karttunen, 2003; Roark and Sproat, 2007). From this perspective, morphological dependencies form regular string languages, and morphological processes can be computed by 1-way finite-state transducers.¹ In fact, many aspects of morphology are *subregular* over strings and fall within remarkably simple subclasses of regular string languages and finite-state transductions (Chandlee, 2014, 2017; Aksënova et al., 2016; Dolatian et al., 2021).

There is little formal work on evaluating the expressivity of morphological dependencies and processes over tree-based representations. In particular, the fine-grained notions of subregular complexity have not been applied to tree-based views of morphology even though many subregular classes can easily be generalized from strings to trees. Previous analyses of morphology that implicitly posit tree structure (Selkirk, 1982, Trost, 1991, a.o.), do not explore the implications of tree structure for complexity, either. This paper seeks to demonstrate

¹The only major exception is total reduplication (Culy, 1985), which we set aside throughout this paper; see Dolatian and Heinz (2020) for detailed discussion.

that this focus on string representations to the exclusion of tree structure means that subtle complexity differences between phenomena may be missed. It is not just cases like *undoable* where trees are useful, but even phenomena where the tree structure provides seemingly no additional information over the string representation.

To this end, we contrast string-based and tree-based views of morphologically conditioned allomorphy in terms of their subregular complexity. Morphologically conditioned allomorphy covers phenomena where some morpheme (the target) has multiple possible realizations, the choice of which is conditioned by the presence of another morpheme (the trigger) within the word. Cross-linguistically, morphologically-conditioned allomorphy can be parameterized in terms of directionality and the degree of locality between the target and trigger morpheme (Carstairs, 1987; Bobaljik, 2000, 2012; Bonet and Harbour, 2012; Embick, 2015).²

Table 1: Parameters for morphologically-conditioned allomorphy between trigger x and target y

Adjacency \ Direction	Inward	Outward
	Local	$x < y$
Non-adjacent	$x < \dots < y$	$y < \dots < x$

If the trigger x is structurally lower than the target y , then allomorphy is *inwardly-sensitive*. If the trigger x is structurally higher than the target y , then allomorphy is *outwardly-sensitive*. If the target and trigger are structurally adjacent, then allomorphy is *locally computed*. If the target and trigger are non-adjacent, and if there can be one or more intervening morphemes, then the process is long-distance or *non-local*. Typologically, local allomorphy is the most common in both directions. Non-adjacent allomorphy is significantly less common, but attested (Božič, 2019).

We find that these four types do not pattern the same way depending on whether one models them over strings or trees (see Table 6). When modeled over strings, there is no complexity difference

²Bobaljik (2000) suggest that the directionality difference correlates with the distinction between phonologically conditioned and morphologically conditioned allomorphy. Following Paster (2006), we take these two splits to form two separate axes of variation and consider only directionality. The phonological nature of the trigger should be examined independently from the formal characteristics of the computation involved.

between inwardly and outwardly sensitive allomorphy. The only relevant split is whether the trigger and target are in a local configuration, which corresponds to a difference between input strictly local (ISL) transductions and sequential transductions. Over trees, we find the same split. But in addition we also see a difference between non-local inwardly sensitive allomorphy and non-local outwardly sensitive allomorphy, with the former but not the latter constituting a sequential tree transduction.

The paper is organized as follows. Section 2 defines relevant families of string and tree transducers, including the (to the best of our knowledge) novel classes of bottom-up and top-down sequential tree transductions. In §3 and §4, we illustrate the typological parameters of allomorphy with attested examples from natural languages. In each section we formalize the respective type of allomorphy over strings as well as trees and contrast their complexity. We then synthesize the main insights in §5. We conclude in §6.

2 Mathematical preliminaries

We cover several classes of subregular string and tree transductions in this paper. Due to space constraints, we cannot give full definitions of each class, but the discussion in the subsequent sections is sufficiently straightforward on a formal level that the reduced rigor should not impact clarity.

2.1 Subregular string transductions

Subregular string transductions are computed by finite-state transducers (FSTs) that obey additional restrictions. One well-known class is the class of *subsequential* transductions, but for our purposes the even more restrictive class of *sequential* transductions will do.³

Definition 1 (Sequential) An FST \mathcal{T} is left-to-right sequential iff \mathcal{T} is deterministic and all states are final. We use $\tau(\mathcal{T})$ to denote the transduction computed by \mathcal{T} . An FST \mathcal{T} is right-to-left sequential iff there is a left-to-right sequential FST \mathcal{T}' such that $\tau(\mathcal{T}) = \{\langle i, o \rangle \mid \langle i^{-1}, o^{-1} \rangle \in \tau(\mathcal{T}')\}$, where s^{-1} is the mirror image of string s . We say that \mathcal{T} (or $\tau(\mathcal{T})$) is sequential iff it is left-to-right sequential or right-to-left sequential.

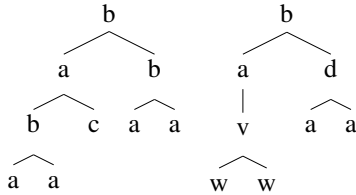
³Our definition of sequential is derived from the non-standard definition of subsequential transducers in Chandlee (2014), which requires all states to be final.

If one further limits the state space of a sequential transducer so that it consists of all and only those states that record the previous k symbols in the input string, one obtains an *input strictly k -local (ISL- k)* transducer. As pointed out in [Chandlee et al. \(2018\)](#), a transduction is guaranteed to be ISL- k if it can be described by a finite set of rewrite rules of the form $a \rightarrow b \mid u.v$ such that $a, b \in \Sigma$, $u, v \in \Sigma^*$, and the combined length of u and v is at most $k - 1$. Crucially, the output of one rewrite rule cannot serve as the input for another rewrite rule. All the rules apply in parallel. We say that a transduction is ISL iff it is ISL- k for some $k \geq 1$.

2.2 Subregular tree transductions

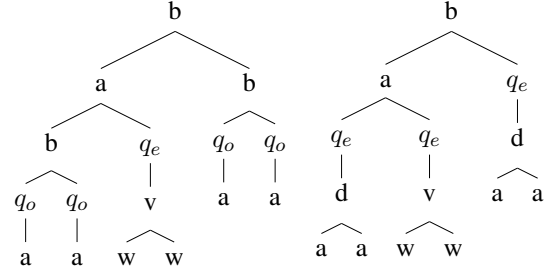
Since the tree transductions encountered in this paper are exceedingly simple, we introduce bottom-up and top-down tree transductions via examples. Our generalizations of sequential transductions from strings to trees then are easily defined as special cases of these two well-known types of tree transducers, full definitions of which can be found in [Comon et al. \(2008\)](#) and [Gécseg and Steinby \(1984\)](#), among others.

Suppose our input trees are strictly binary branching and all nodes are either labeled a , b , or c . Now consider a transduction that leaves almost all nodes the same, except that something special happens to the root of each subtree that contains an even number of as (not counting the root itself). If the label of the subtree's root is b , then it should be relabeled d . If the label is a , then the left subtree will be deleted. In addition, every leaf node c in the input tree is rewritten as $v(w, w)$. Hence the input tree on the left would become the output tree on the right.

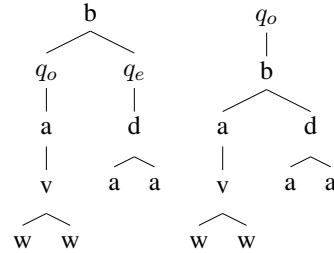


This transduction can be computed by a bottom-up tree automaton. We use two states, q_o and q_e , which keep track of whether a subtree contains an odd or an even number of as . Next, we define transition rules for the leaves: $a() \rightarrow q_o(a)$, $b() \rightarrow q_e(b)$, and $c() \rightarrow q_e(v(w, w))$. Let us also add a rewrite rule for interior node b : $b(q_o(x), q_o(y)) \rightarrow q_e(d(x, y))$ expresses that when we encounter a node labeled b such that the left subtree and the

right subtree both contain an odd number of as (and thus the whole subtree contains an even number of as), b should be replaced with a d while keeping the left subtree x and the right subtree y in the same position. With these rules, we can already begin to rewrite the input tree.

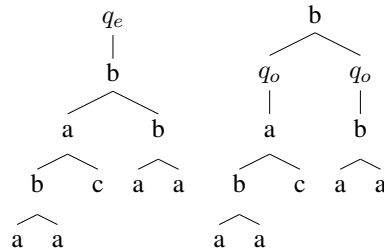


We also need rewrite rules for a as an interior node. For the concrete case at hand, the relevant transition rule is $a(q_e(x), q_e(y)) \rightarrow q_o(a(y))$, which removes the left subtree x . We then add a few more rules to handle the remaining cases. For example, $b(q_o(x), q_e(y)) \rightarrow q_o(b(x, y))$ ensures that nothing is changed when a subtree rooted in b does not contain an even number of as .

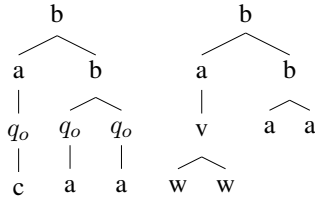


If q_o is a final state, then the subtree beneath it is chosen as the output of the transformation, otherwise it is rejected.

Now consider instead the case of a top-down transducer, which rewrites the input tree from the root towards the leafs. Assume the same input tree as before, but this time something special happens when the root of a subtree is dominated by an odd number of bs (not counting the root itself). In this case, b is rewritten as d , and c is replaced with $v(w, w)$. In addition, a with two daughters has the left one deleted. This will produce the very same output tree as before, but it does so in a different manner. First, we always start with an initial state q_e , and we set $q_e(b(x, y)) \rightarrow b(q_o(x), q_o(y))$.



Next we add one rule for a and one for b : $q_o(a(x, y)) \rightarrow a(q_o(y))$ and $q_o(b(x, y)) \rightarrow d(q_e(x), q_e(y))$. This leaves us will only leaf nodes to rewrite, which is handled by the rules $q_o(c()) \rightarrow v(w, w)$ and $q_o(a()) \rightarrow a$.

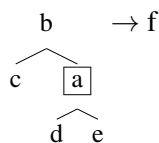


The tree is a valid output for the input because we were able to process the whole tree from the root to all its leaves.

Of course we would have to add more rules to both transducers to also cover the configurations that do not arise in our toy examples. But even then the transducers would still be *deterministic*: given two transitions rules of the form $a \rightarrow u$ and $b \rightarrow v$, $u \neq v$ implies $a \neq b$ (and in addition, the top-down tree automaton has exactly one initial state). In fact, our two example transductions satisfy additional properties that make them natural analogs of sequential string transductions.

Definition 2 (Tree sequential) A *deterministic bottom-up tree transducer* is bottom-up sequential iff all its states are final. A *deterministic top-down tree transducer* is top-down sequential iff it holds for every state q and every leaf symbol σ that the transducer has a transition rule $q(\sigma()) \rightarrow t$, where t is some tree not containing any states.

Finally, we also need a tree analogue of ISL string transductions. We adopt the definition in Graf (2020), but since it spans multiple pages, we only convey the intuition here. An ISL tree transduction is state-free in the sense that what a given node should be rewritten as is fully determined by its label and the local context. For the purposes of this paper, we can limit this even further to just the class of ISL tree transductions that only relabel nodes but do not change the structure of the input tree. As a concrete example, consider this rewrite rule:



This says that a node that is labeled a is relabeled as f if the node has b as its mother, c as its left sister, d as its left daughter, e as its right daughter, and the node has no other sisters or daughters.

3 Inwardly-sensitive allomorphy

We now turn to local (§3.1) and non-local (§3.2) inwardly-sensitive allomorphy, followed by outwardly-sensitive allomorphy in §4. Local inwardly-sensitive allomorphy can be modeled with ISL FSTs and by ISL tree-transducers, suggesting that the choice between strings and trees is innocuous here. Non-local inwardly-sensitive allomorphy only falls within those classes if one assumes a finite upper bound. Otherwise, if no finite bound is assumed, then ISL is insufficient, but the allomorphy phenomena can still be captured by left-to-right sequential string transducers or bottom-up sequential tree transducers.

3.1 Local and inwardly-sensitive

As previously indicated in Table 1, an allomorphic pattern is local and inwardly-sensitive iff the conditioning morpheme (the trigger x) is structurally below the alternating morpheme (the target y) and x is structurally adjacent to y . Table 2 illustrates this with the past suffix alternation in Latin.

Table 2: Local inwardly-sensitive allomorphy from Latin (Embick, 2015, ch4.6)

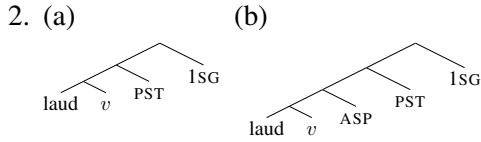
imperfect laud-ā- ba -m	pluperfect laud-ā- ve-ra -m
praise- v - PST _{y} -1SG	praise- v - ASP _{x} - PST _{y} -1SG
T[+past]→- ba	T[+past]→- ra / ASP[perf] ₋

Following Embick (2015), the Latin past suffix is by default $-ba$. After the aspect suffix $-ve$, it is instead realized as $-ra$. In terms of rewrite rules, we have the following:

1. (a) $PST \Rightarrow -ra \mid ASP _$
- (b) $PST \Rightarrow -ba \mid W _$

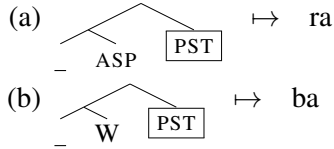
Here, and throughout the rest of the paper, we use W to denote any morpheme that is irrelevant to the alternation, e.g. any morphemes that are not ASP or PST in this example. Since the alternation can be described by finitely many rewrite rules with a context of size 1, it is an ISL-2 string transduction.

The allomorphy is also ISL over trees, but the size of the window increases slightly to ISL-3. Suppose that the two forms in Table 2 have the underlying tree structures below.



In order to derive the pattern in Table 2 given this tree structure, an ISL tree transducer has to include the rewrite rules below. They are ISL-3 rewrite rules because the depth of the tree on the left-hand side is 3.

3. ISL-3 rewrite rules for Latin Past



3.2 Non-adjacent and inwardly-sensitive

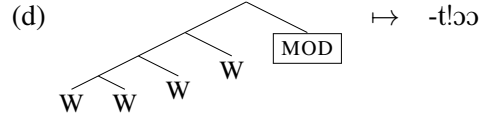
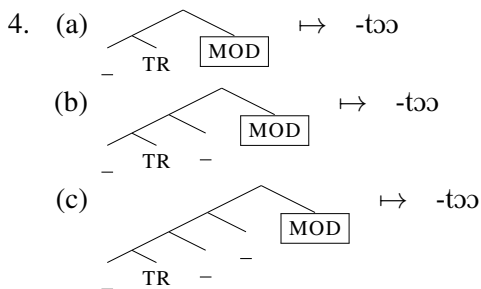
We now turn to non-adjacent inwardly-sensitive allomorphy. Recall that in these cases, the trigger x and target y are not adjacent, and x is structurally above y . We illustrate this with an example from Kiowa.

Table 3: Non-adjacent inwardly-sensitive allomorphy from Kiowa (Bonet and Harbour, 2012, 231)

héib-e-gyū-məw-təw	héib-é-gyū-məw-t!əw
enter-TR _x -DISTR-NEG-MOD _y	enter-INTR _x -DISTR-NEG-MOD _y
MOD → -təw / TR	MOD → -t!əw / INTR

The modality suffix (target y) surfaces as $-təw$ ($-t!əw$) if the verb is transitive (intransitive). Transitivity is marked on the post-root suffix (trigger x). The trigger and target are not adjacent.

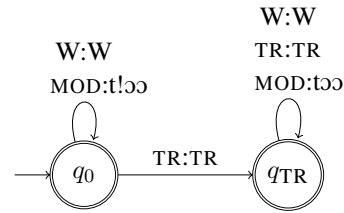
Over strings, the alternation for the attested Kiowa examples would be ISL-4. The context must span at least four 4 morphemes because the target and trigger are separated by at most 2 interveners. Similarly, over a tree, this function can be captured by an ISL-5 transduction. The crucial rewrite rules are shown below. These rules indicate that MOD is rewritten as $təw$ if there is a TR with (i) no intervener, or (ii) one intervener or (iii) two interveners. In all other cases, MOD is rewritten as $t!əw$.



This treatment works for the observed cases as there is necessarily an upper bound on how far the trigger and the target can be apart. But it fails to capture the fact that the interveners do not affect the allomorphy at all. Instead, we may assume that there is no upper bound on the number of intervening morphemes. In that case, Kiowa allomorphy is no longer ISL, neither over strings nor over trees.

That said, over strings the Kiowa allomorphy pattern still falls within the class of left-to-right sequential transductions. The corresponding transducer is shown in Figure 1.

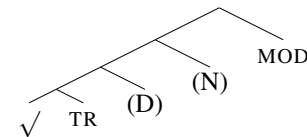
Figure 1: Sequential FST (left-to-right) for Kiowa



Over trees, it is also a fairly simple transduction and is, in fact, bottom-up sequential. In order to capture the allomorphy conditioned by TR, the transducer has to distinguish between a state q_{TR} where it has already processed TR and a state q_W where it has not. If the transducer sees MOD in state q_{TR} , MOD is rewritten as $təw$. Whereas if it sees MOD in state q_W , MOD is transformed into $t!əw$.

Note, however, that the crucial morphemes, MOD and TR, do not stand in a dominance relation if one assumes a phrase structure tree as depicted in Figure 2.

Figure 2: The phrase structure tree for the Kiowa modality suffixes



Thus, even if a bottom-up sequential transducer processes TR and moves to the state q_{TR} , the transducer reads MOD separately from that state transition. The transducer thus needs to delay its output when it reads MOD: instead of immediately choosing an output for MOD, it switches to a state q_{MOD}

without outputting anything. Then, at the next node, if the state from the left branch is q_{TR} , the transducer outputs a tree such that its right branch is $t\omega$. If the state from the left branch is not q_{TR} , on the other hand, then the transducer outputs a tree with the right branch $t!\omega$. The relevant transition rules are shown below, with \cdot as the label of interior nodes.

5. (a) $\text{MOD}() \rightarrow q_{\text{MOD}}()$
- (b) $\text{TR}() \rightarrow q_{\text{TR}}(\text{TR})$
- (c) $W() \rightarrow q_W(W)$
- (d) $\cdot(q_{\text{TR}}(x), q_{\text{MOD}}(y)) \rightarrow q_{\text{TR}}(\cdot(x, -t\omega))$
- (e) $\cdot(q_W(x), q_{\text{MOD}}(y)) \rightarrow q_W(\cdot(x, -t!\omega))$

In sum, the move from a local to a non-local phenomenon continues the parallelism already observed in the local case. Local inwardly-sensitive allomorphy is ISL over strings as well as trees, and its non-local counterpart is left-to-right sequential over strings or bottom-up sequential over trees. The only noteworthy difference is that the bottom-up sequential transducer has to make use of a *delayed output* strategy. While this may seem innocuous, this will be the decisive reason in §4.2 why non-local outwardly-sensitive allomorphy over trees is more complex.

4 Outwardly-sensitive allomorphy

Mirroring inwardly-sensitive allomorphy, outwardly-sensitive allomorphy is either local (§4.1) or non-local (§4.2). As we will see, local patterns are once again ISL over strings as well as trees. If we model non-local patterns as involving potentially unbounded distances between the target and trigger, then these patterns are sequential over strings, but not necessarily over trees.

4.1 Local and outwardly-sensitive

An allomorphic pattern is local and outwardly-sensitive iff the conditioning morpheme (the trigger x) is structurally above the alternating morpheme (the target y), and x is structurally adjacent to y . Table 4 gives an example from Hungarian: the plural suffix surfaces as $-k$ by default but must be $-ai$ before the 1SG possessive suffix $-m$.

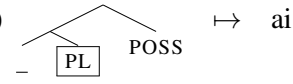
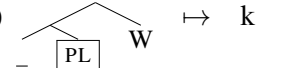
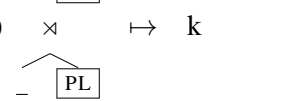
Table 4: Local outwardly-sensitive allomorphy from Hungarian (Carstairs 1987, 165; Embick 2010, 62)

ruhá- m dress-POSS _x	ruhá- k dress-PL _y	ruha- ái-m dress-PL _y -POSS _x
-------------------------------------------	-----------------------------------------	---------------------------------------------------------------

Over strings, the above phenomenon is ISL-2 as it can be described by the following rewrite rules:

6. (a) $\text{PL} \rightarrow ai \mid _ \text{POSS1SG}$
- (b) $\text{PL} \rightarrow k \mid _ W$ (where W may also denote the end of the string).

Over trees, the Hungarian plural suffix alternation is ISL-3. Assume once again a right-linear structure where each affix is the right sibling of a subtree containing all the material to its left. The possessive affix is the right sibling of the interior node that immediately dominates the plural suffix. Hence an ISL tree transduction for the pattern in Table 4 must include the plural alternation rules shown in 7 (\times indicates that the node is the root). The depth of the context specified in the left-hand side of these rules is at most 3, and hence the plural alternation is ISL-3.

7. (a)  $\mapsto ai$
- (b)  $\mapsto k$
- (c)  $\mapsto k$

4.2 Non-adjacent and outwardly-sensitive

We now consider the case of outwardly-sensitive allomorphy where the trigger x is still above the target y , but no longer string-adjacent to it. We illustrate with Slovenian.

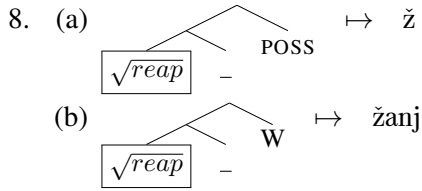
Table 5: Non-adjacent outwardly-sensitive allomorphy from Slovenian (Božič, 2016, 2019, 501)

žanj-e- \emptyset -m reap _y -ASP-PRES-2P.SG	ž-e-l-a reap _y -ASP-PTC _x -F.SG
$\sqrt{\text{reap}} \rightarrow \text{žanj}$	$\sqrt{\text{reap}} \rightarrow \text{ž} / _ \dots \text{PTC}$

The root ‘reap’ surfaces as *žanj* by default. It surfaces as *ž* when the participle suffix $-l$ is present. The root and suffix are not adjacent but are separated by an aspect suffix.

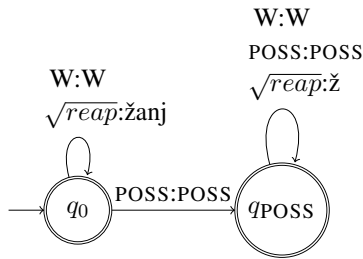
As with Kiowa’s inwardly-conditioned non-adjacent allomorphy in §3.2, the above case can be analyzed as ISL- k with a larger value for k . Over

strings, it would be ISL-3, with the central rewrite rule being $\sqrt{reap} \rightarrow _W\text{PTC}$. Over trees, the alternation is also ISL-3, as is evidenced by the relevant rewrite rules below.



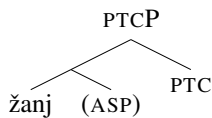
If, for the sake of argument, we treat this allomorphy as truly long-distance, then ISL is no longer sufficient. But as in the case of inwardly-sensitive non-local allomorphy, the parallel between strings and trees remains as we are dealing with a sequential transduction in both cases. The sequential string transducer is shown in Figure 3. Note that this transducer operates right-to-left, whereas inwardly-sensitive non-local allomorphy is left-to-right sequential.

Figure 3: Sequential transducer (right-to-left) for Slovenian root allomorphy



When operating with trees, we observe a curious split: sequentiality hinges on whether interior nodes are labeled with projections of affixes. Suppose that trees are labeled in the manner shown in Figure 4, where the tree’s root has the label PTCP and the suffix has PTC.

Figure 4: A phrase structure tree for Slovenian root allomorphy



In this case, it is easy to provide a top-down sequential tree transducer for the Slovenian root allomorphy. By default, the transducer is in state q_W . When encountering the node PTCP, the transducer changes to a new state q_{PTC} , which then gets passed down into the subtrees along both

branches. We then have two distinct rewrite rules such that $q_W(\sqrt{reap})$ is rewritten as $\check{z}anj$, whereas $q_{\text{PTC}}(\sqrt{reap})$ results in \check{z} . If PTCP is present in the tree, then the transducer, by virtue of moving from the tree root towards the leaves, must have encountered it before reaching the morphological root \sqrt{reap} . The transducer will thus correctly rewrite it as \check{z} in these cases, and only these cases. The key transition rules are explicitly listed in 9a–9c.

9. (a) $q_W(\text{PTCP}(x, y))$
 $\rightarrow \text{PTCP}[q_{\text{PTC}}(x), q_{\text{PTC}}(y)]$
- (b) $q_W(\sqrt{reap}) \rightarrow \check{z}anj$
- (c) $q_{\text{PTC}}(\sqrt{reap}) \rightarrow \check{z}$

But on the other hand, without labels like PTCP, the alternation is not top-down sequential. In fact, it is not even top-down deterministic. The problem is that top-down transition rules are of the form $q(\sigma(x_1, \dots, x_n)) \rightarrow \omega(q_1(x_1), \dots, q_2(x_2))$. This means that the state assigned to a daughter x_i depends only on the label of its mother, and the state assigned to the mother. Neither the label of x_i itself, nor the labels of any of its siblings are taken into consideration. But this is exactly what is needed in the case of Slovenian. Without interior labels like PTCP, the rewrite rules would have to be $q_W(\cdot(x, W)) \rightarrow \cdot(q_W(x), W)$ and $q_W(\cdot(x, \text{PTC})) \rightarrow \cdot(q_{\text{PTC}}(x), \text{PTC})$. The state that controls the processing of the subtree x must be contingent on the label of the right daughter (like PTC), and this is not possible with deterministic top-down transducers, which top-down sequential transducers are a proper subclass of.

However, one could equip the transducer with a finite look-ahead of depth 1, which would allow it to inspect the labels of daughters, too, before assigning states. This would be a *sensing tree transducer* as defined in Graf and De Santo (2019). Note that the need for look-ahead does not arise with sequential string transductions because they can emulate finite look-ahead by delaying their output; and to a more limited extent, this is also an option for the sequential bottom-up transducer. Inwardly-sensitive and outwardly-sensitive allomorphy thus seem to exhibit exactly the same complexity in the string case, but diverge at least for non-local phenomena if one switches from strings to trees. The additional complexity of trees brings to light an additional challenge that is not readily apparent in the string case.

Table 6: Summary of formal results for directionality and locality of allomorphy types; patterns marked with * are ISL if one does not assume unboundedness

Pattern	String-based computation	Tree-based computation
Inward & local	ISL	ISL
Inward & non-local	Left-to-right sequential*	bottom-up sequential*
Outward & local	ISL	ISL
Outward & non-local	Right-to-left sequential*	top-down sequential or STFTT* (sensing)

5 Discussion

This paper surveyed the attested categories of local and non-local allomorphy, with the key findings summarized in Table 6. Our central insight is that even though the choice between strings and trees seems innocuous given how closely our right-linear tree structures mirror the strings, it does reveal a difference in complexity between inwardly-sensitive and outwardly-sensitive allomorphy. Hence the use of trees can be motivated on the same grounds as unboundedness assumptions, namely that it reveals complexity differences that would be missed otherwise.

The relevant complexity difference may seem minor compared to, say, the difference between regular and context-free dependencies, which greatly matters for practical purposes such as parsing. However, this is true for most subregular complexity differences. Since all subregular dependencies and transductions can be handled with finite-state machinery, subregular distinctions do not impact efficiency. That does not mean, though, that the distinctions are irrelevant. They affect learnability, and they make different typological predictions about what kind of patterns we should expect to find across languages. One way to construe our finding, then, is that it urges us to look for empirical differences between inward non-local and outward non-local allomorphy that can be traced back to the gap in computational complexity.

Of course this argument hinges on the assumption that these phenomena are indeed unbounded and operate over trees. Unboundedness is far from a given because inflectional morphology in natural language morphology usually exhibits limits on the linear distance between the targets and triggers of allomorphy. What is unclear is whether this is an intrinsic limitation of allomorphy itself or an accidental confluence of multiple independent factors.

Our findings provide a *modus tollens* argument to address this: if we do find differences between

inwardly-sensitive and outwardly-sensitive allomorphy that can be explained in terms of the subregular complexity split, then that argues in favor of underlying unboundedness and morphological tree structures because that is the only case where we find a difference in subregular complexity. If there are no discernible differences, then either unboundedness or tree structure should be jettisoned for inflectional morphology.

If there is evidence for both unboundedness and tree structure, that would be an interesting parallel to syntax. In fact, the split between inwardly-sensitive and outwardly-sensitive allomorphy already has a connection to syntax. The sensing tree transducers of Graf and De Santo (2019) were motivated by the desire to address shortcomings of deterministic top-down transducers for syntax, so it is interesting that they are also needed for tree-based morphology with the unboundedness assumption.

While the argument we present can be made with the coarse split between ISL and sequential transductions, it would be interesting to explore a possible middle-ground between the two. Tier-based strict locality has been explored in various areas — including phonology, morphology, and syntax — as an extension of the strict locality underpinning ISL (Heinz et al., 2011; Aksënova et al., 2016; Graf, 2018; Burness et al., 2021; Dolatian and Guekguezian, 2021). The non-local case of allomorphy discussed in this paper may be describable along those lines. So far, no counterpart has been defined for tree transductions, but once this happens, the issues explored in this paper should be revisited from the perspective of tier-based strict locality.

6 Conclusion

We have investigated four types of allomorphy from the perspective of strings and trees: local and non-local inwardly-sensitive allomorphy, and local and non-local outwardly-sensitive allomorphy. Even

though our tree structures closely track the surface strings, our findings are not the same over the two types of representations. While the split between local and non-local allomorphy always leads to a complexity difference if one assumes unboundedness, inwardly- and outwardly-sensitive allomorphy are equally complex over strings but not over trees. If there are empirical differences between these allomorphy types that can be derived from the split in complexity, that would provide evidence for both unboundedness and tree structure in inflectional morphology. Our study thus highlights the importance of representations even in cases where the difference of representations seems innocuous at a first glance. An approach firmly rooted in trees and unboundeness may reveal subtle computational differences that would be missed otherwise.

Acknowledgments

Thomas Graf's work on this project was supported by the National Science Foundation under Grant No. BCS-1845344.

References

- Alëna Aksënova, Thomas Graf, and Sedigheh Moradi. 2016. **Morphotactics as tier-based strictly local dependencies**. In *Proceedings of the 14th sigmorphon workshop on computational research in phonetics, phonology, and morphology*, pages 121–130.
- Kenneth Beesley and Lauri Karttunen. 2003. *Finite-state morphology: Xerox tools and techniques*. CSLI Publications, Stanford, CA.
- Jonathan David Bobaljik. 2000. The ins and outs of contextual allomorphy. In Kleanthes K. Grohmann and Caro Struijke, editors, *University of Maryland working papers in linguistics*, volume 10, pages 35–71. University of Maryland, College Park.
- Jonathan David Bobaljik. 2012. *Universals in comparative morphology: Suppletion, superlatives, and the structure of words*. Number 50 in Current Studies in Linguistics. MIT Press, Cambridge, MA.
- Eulàlia Bonet and Daniel Harbour. 2012. **Contextual allomorphy**. In *The morphology and phonology of exponence*, number 41 in Oxford Studies in Theoretical Linguistics, pages 195–235. Oxford University Press, Oxford.
- Jurij Božič. 2016. Locality of exponence in distributed morphology: Root suppletion in Slovenian. In *North East Linguistic Society (NELS)*, volume 46, pages 137–146, Amherst. GLSA.
- Jurij Božič. 2019. **Constraining long-distance allomorphy**. *The Linguistic Review*, 36(3):485–505.
- Phillip Alexander Burness, Kevin James McMullin, and Jane Chandlee. 2021. **Long-distance phonological processes as tier-based strictly local functions**. *Glossa: a journal of general linguistics*, 6(1).
- Andrew Carstairs. 1987. *Allomorphy in inflexion*. Croom Helm, London.
- Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware, Newark, DE.
- Jane Chandlee. 2017. **Computational locality in morphological maps**. *Morphology*, 27(4):1–43.
- Jane Chandlee, Jeffrey Heinz, and Adam Jardine. 2018. **Input strictly local opaque maps**. *Phonology*, 35(2):171–205.
- H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2008. **Tree automata: Techniques and applications**. Published online: <http://www.grappa.univ-lille3.fr/tata>. Release from November 18, 2008.
- Christopher Culy. 1985. **The complexity of the vocabulary of Bambara**. *Linguistics and Philosophy*, 8:345–351.
- Hossep Dolatian and Peter Ara Guekguezian. 2021. **Relativized locality: Phases and tiers in long-distance allomorphy in Armenian**. *Linguistic Inquiry*.
- Hossep Dolatian and Jeffrey Heinz. 2020. **Computing and classifying reduplication with 2-way finite-state transducers**. *Journal of Language Modeling*, 8:79–250.
- Hossep Dolatian, Jonathan Rawski, and Jeffrey Heinz. 2021. **Strong generative capacity of morphological processes**. *Proceedings of the Society for Computation in Linguistics*, 4(1):228–243.
- David Embick. 2010. *Localism versus globalism in morphology and phonology*, volume 60 of *Linguistic Inquiry Monographs*. MIT Press, Cambridge, MA.
- David Embick. 2015. *The morpheme: A theoretical introduction*, volume 31. Walter de Gruyter, Boston and Berlin.
- Thomas Graf. 2018. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*, pages 117–136.
- Thomas Graf. 2020. **Curbing feature coding: Strictly local feature assignment**. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 224–233, New York, New York. Association for Computational Linguistics.

- Thomas Graf and Aniello De Santo. 2019. [Sensing tree automata as a model of syntactic dependencies](#). In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 12–26, Toronto, Canada. Association for Computational Linguistics.
- Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akademiai Kiadó, Budapest.
- Jeffrey Heinz, Chetan Rawal, and Herbert G Tanner. 2011. [Tier-based strictly local constraints for phonology](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short papers-Volume 2*, pages 58–64. Association for Computational Linguistics.
- Kimmo Koskenniemi. 1983. *Two-level morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki.
- Mary Paster. 2006. *Phonological conditions on affixation*. Ph.D. thesis, University of California, Berkeley, Berkeley, CA.
- Brian Roark and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford University Press, Oxford.
- Elisabeth Selkirk. 1982. *The syntax of words*. Number 7 in Linguistic Inquiry Monographs. MIT Press, Cambridge, Mass.
- Harald Trost. 1991. [Recognition and generation of word forms in natural language understanding systems: Integrating two-level morphology and feature unification](#). *Applied Artificial Intelligence*, 4:411–457.