Proceedings of the Society for Computation in Linguistics

Volume 4 Article 22

2021

Strong Generative Capacity of Morphological Processes

Hossep Dolatian
Stony Brook University, hossep.dolatian@alumni.stonybrook.edu

Jonathan Rawski
Stony Brook University, jonathan.rawski@stonybrook.edu

Jeffrey Heinz
Stony Brook University, jeffrey.heinz@stonybrook.edu

Strong generative capacity of morphological processes

Hossep Dolatian, Jonathan Rawski, and Jeffrey Heinz
Department of Linguistics & Institute for Advanced Computational Science
Stony Brook University
{hossep.dolatian,jeffrey.heinz,jonathan.rawski}@stonybrook.edu

Abstract

Morphological processes are generally computable with 1-way finite-state transducers. However, we show that 1-way transducers do not capture the strong generative capacity of certain morphological analyses for more complex processes, including mobile affixation, infixation, and partial reduplication. As diagnostics for strong generative capacity, we use origin semantics and order-preservation. These analyze the input-output correspondences generated by finite-state transducers and their corresponding logical transductions. For some linguistic analyses of these complex processes, their strong generative capacity is matched by more expressive grammars, such as non-order-preserving transductions and their corresponding 2-way finite-state transducers.

1 Introduction

A central goal of computational morphology is to define the minimally sufficient and restrictive classes of grammars which can compute attested morphological processes. Virtually all of morphology is sufficiently computable with 1-way finite-state transducers (FSTs) (Roark and Sproat, 2007). Furthermore, most of morphology can be computed with restricted subclasses of these finite-state grammars (Chandlee, 2017). Thus, 1-way FSTs are adequate in weak generative capacity (WGC).

This paper examines the strong generative capacity (SGC) of 1-way FSTs when computing morphological functions. For a given theory, we find a divergence between the WGC and SGC of different morphological processes, including infixation, mobile affixation, and partial reduplication. There is a longstanding controversy around defining adequate diagnostics for the SGC of linguistic structures (Manaster-Ramer, 1987a; Miller, 1991, 1999). For our purposes, we use two diagnostics which are well-defined in theoretical computer science, but have not been previously applied to computational morphology: origin semantics (Bojańczyk, 2014) and order-preservation (Filiot, 2015). They provide a unique lens for examining the input-output correspondences created by different classes of finite-state grammars and their corresponding logical transductions (Engelfriet and Hoogeboom, 2001).

We use these diagnostics to show that simple affixation is definable with 1-way FSTs both in terms of WGC and SGC. However, depending on the *specific* morphological theory, these diagnostics indicate that 1-way FSTs do not match the SGC of more complex processes. Instead, some morphological analyses are more faithfully computed with more expressive non-order-preserving transductions which themselves are computed by 2-way FSTs. These results do not argue against the practicality or efficiency of 1-way FSTs. Instead, they are scientific results about the computational and mathematical properties of morphology.

This paper is organized as follows. We review mathematical results on generative capacity in linguistics in §2. In §3, we define origin semantics and order-preservation as diagnostics for SGC. We use these diagnostics in §4 to show how 1-way FSTs capture the SGC of simple affixation. In §5, we show how 1-way FSTs do not capture the theory-dependent SGC for other morphological processes, while 2-way FSTs do. We conclude in §6. We provide an appendix §A of some illustrative 2-way FSTs which do capture the SGC of these analyses.

2 Weak vs. strong generative capacity

Given a grammar, its WGC defines the set of forms which it can generate, usually stringsets. In contrast, its SGC defines the type of hidden structure that it posits during the derivation. It is generally harder to determine the SGC of a grammar than its WGC. Informally there are two issues:

Proceedings of the Society for Computation in Linguistics (SCiL) 2021, pages 228-243.
Held on-line February 14-19, 2021

1. *Fundamental issues in SGC*
   (a) **Grounding**: basis for interpretations
   (b) **Diagnostic**: formal tools for evaluations

The **grounding** for SGC is the external basis assumed when evaluating grammars. For syntax, the external basis for evaluating SGC is semantic interpretation and constituency, i.e., if a grammar's phrase structure tree is *similar* to the semantic interpretation. The **diagnostic** for SGC is simply the set of formal tools used to determine 'similarity'. The simplest diagnostic is to require, for example, that the tree and semantics are identical. More elaborate diagnostics utilize nuanced interpretations and deductions from tree geometry (Miller, 1999).

In syntax, WGC and SGC often converge. Most context-free (CF) phenomena are CF in both WGC and SGC (Chomsky, 1956; Pullum and Gazdar, 1982; Gazdar and Pullum, 1985), while most non-CF phenomena are non-CF in both WGC and SGC (Culy, 1985; Radzinski, 1991; Stabler, 2004; Kobele, 2006; Clark and Yoshinaka, 2014). But, WGC and SGC can diverge when the overt syntax is CF, but the associated semantics is non-CF (Radzinski, 1990). For example, both Dutch and Swiss German have cross-serial clause constructions where the languages contain a sequence of noun phrases, followed by a sequence of verbs which subcategorize for these nouns: $N_1N_2N_3V_1V_2V_3$. In terms of their semantics, such constructions are non-CF in both languages (Bresnan et al., 1982; Shieber, 1985), and thus non-CF in SGC. But in Dutch, these sequences are CF in terms of WGC because there is no overt morphological marking for subcategorization between verbs and nouns. In contrast, Swiss German nouns show different case marking based on the verbs which subcategorize for them. Thus, these constructions are non-CF in both WGC and SGC in Swiss German, but only in SGC in Dutch.[1]

In morphology and phonology, there are fewer debates on generative capacity. We speculate that this is due to two issues. First, morphology and phonology have comparatively restrictive WGC. Second, it is unclear what external basis (grounding) should be used for SGC, and thus what diagnostics or metrics to use.

In terms of WGC, virtually all attested morphological and phonological processes are sufficiently characterized by the class of Regular languages and functions (Johnson, 1972; Koskenniemi, 1983; Sproat, 1992; Ritchie, 1992; Kaplan and Kay, 1994; Beesley and Karttunen, 2003; Roark and Sproat, 2007). In fact, most of these processes only require less expressive subclasses of subregular languages and rational functions (Rogers and Pullum, 2011; Rogers et al., 2013; Heinz and Idsardi, 2013; Chandlee, 2014, 2017; Aksënova et al., 2016; Chandlee and Heinz, 2018; Chandlee et al., 2018; Heinz, 2018). The exception is total reduplication which is not definable with FSAs (Culy, 1985) or 1-way FSTs (Chandlee, 2017). Furthermore, many theories of phonology are computationally proven to be notationally equivalent and thus equivalent in WGC. This includes theories for phonotactics (Graf, 2010a,b), vowel harmony (Andersson et al., 2020), syllabification (Strother-Garcia, 2019), and tone (Danis and Jardine, 2019; Jardine et al., 2020; Oakden, 2020).[2] For morphology, many theories are likewise finite-state definable and thus equivalent in WGC (Karttunen, 2003; Roark and Sproat, 2007; Ermolaeva and Edmiston, 2018).

There are few debates on the SGC of phonology and morphology. For phonology, the proper grounding for SGC is unclear. For morphology, the grounding of SGC is often treated as the semantic constituency of words. Due to prefix-suffix dependencies, the semantic constituency of words (SGC) is context-free (Langendoen, 1981; Selkirk, 1982; Carden, 1983; Oseki, 2018; Oseki et al., 2019; Oseki and Marantz, 2020); but in practice, the morphotactics of words (WGC) are regular (Hammond, 1993; Bjorkman and Dunbar, 2016; Aksënova and De Santo, 2019).[3] Furthermore, although partial

[1] A more elaborate example is total reduplication (copying), which is multi-CF in its WGC but debatably parallel multi-CF in its SGC (Stabler, 2004; Clark and Yoshinaka, 2014). A controversial example is the *respectively* construction which seems non-CF in syntax (Kac et al., 1987; Kac, 1987; Manaster-Ramer, 1987b), but is potentially due to pragmatic factors (Pullum and Gazdar, 1982). An entertaining example are *buffalo* sentences which are the regular language *buffalo** in WGC, but CF in their semantics (SGC).

[2] There is debate on the WGC of constraint-interaction grammars like Optimality-Theory (Prince and Smolensky, 2004). There are many finite-state approximations (Eisner, 1997, 2000a,b; Karttunen, 1998; Frank and Satta, 1998; Riggle, 2004; Gerdemann and Van Noord, 2000; Gerdemann and Hulden, 2012) of varying computational tractability (Idsardi, 2006; Heinz et al., 2009). But in principle, constraint-interaction can express non-regular functions (Lamont, 2019a,b, prep; Hao, 2019) and is Turing-complete (Smolensky, 1993).

[3] Some work adds refinements to finite-state systems in order to provide easier-to-design grammars for prefix-suffix dependencies, such as flag-diacritics (Beesley, 1998; Saléschus, 2008; Saléschus and Hautli, 2008), registers (Cohen-Sygal and Wintner, 2006), feature unification (Trost, 1990, 1991; Krieger and Pirker, 1993; Zajac, 1998; Amtrup, 2003, 2004),

reduplication is a rational function in its WGC, there is some debate on its SGC (Dolatian and Heinz, 2018b, 2020).

Root-and-pattern morphology is likewise problematic over FSTs with a single input tape (Kay, 1987; Kiraz, 2001; Dolatian and Rawski, 2020); though easier to use once combined with feature-unification (Gasser, 2009). As a language, it is regular in WGC, but more intuitively expressed with context-sensitive grammars (Botha and Blunsom, 2013). An interesting demonstration for SGC comes from prefix-suffix dependencies. They can be modeled by 2-headed FSAs which can express linear CF languages (Creider et al., 1995). These grammars can however be restricted enough to only generate regular languages (Ramer and Savitch, 1997).

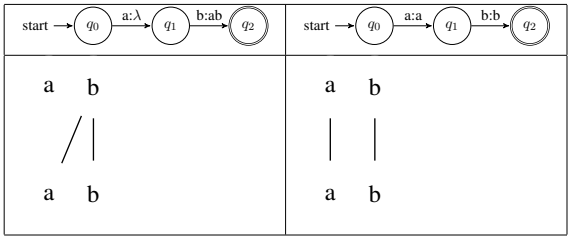
In this paper, we analyze the SGC of morphology when computed over FSTs. We develop an alternative grounding in terms of the segmental correspondence between the input and output representations. Based off of Dolatian (2020), we formulate SGC in terms of two mathematical properties: origin semantics and order-preservation.

3 Preliminaries: Origin semantics and order-preservation

Finite-state transducers (FSTs) are a common computational model for morphology and phonology (Roche and Schabes, 1997). Technically, the FSTs used in computational linguistics and NLP are 1-way FSTs. These are FSTs which process the input in only one direction. 1-way FSTs compute rational functions. In contrast, a 2-way FST is an FST which processes the input in multiple directions by going back and forth on the input (Savitch, 1982; Engelfriet and Hoogeboom, 2001; Shallit, 2008). 2-way FSTs compute the more expressive regular functions (Filiot and Reynier, 2016). 2-way FSTs have recently been applied to model reduplication (Dolatian and Heinz, 2020).

In order to probe the SGC of 1-way FSTs, we use origin semantics and order-preserving logical transductions. Given an FST, its origin semantics (Bojańczyk, 2014) is the origin information of each symbol o_n in the output string.[4] This is the position i_m of the read head on the input tape when the FST had generated o_n. To illustrate, consider a partial function $f_{ab} = \{(ab, ab)\}$ which maps *ab* to itself. This function can be modeled with at least two different 1-way FSTs as in the top row of Figure 1 which differ in when they output the output symbols *a*,*b*. In the bottom row of Figure 1, we use graphs called *origin graphs* (Bojańczyk et al., 2017) to visualize the origin information created by the two FSTs for the mapping (*ab*, *ab*). The FSTs are equivalent in their WGC, but they differ in their origin semantics (SGC).

Figure 1: Pair of 1-way FSTs for the function f_{ab} and their origin information for the mapping $ab \rightarrow ab$.



In terms of formal logic, FSTs correspond to logical transductions which use Monadic Second Order (MSO) logic (Courcelle, 1997; Engelfriet and Hoogeboom, 2001). For logical transductions, the input string is defined in terms of a word signature $< D, R >$. The segments are defined in terms of a set of domain elements D taken from the set of positive integers. The domain elements satisfy a set of relations R which can be unary or binary. Unary relations designate the labels L of these domain elements, e.g., the label $\mathsf{t}(x)$ designates domain elements which are the segment /t/, and the labels $\mathsf{C}(x), \mathsf{V}(x)$ designate consonants and vowels. Domain elements are connected via binary relations such as immediate successor $\mathsf{succ}(x, y)$ or non-immediate precedence $\mathsf{prec}(x, y)$. For example, the word *pat* is defined with the following logical formulas. Input relations use this font.

2. *Word model for* pat
   Domain D: $\{1, 2, 3\}$
   Unary relations or labels $L \subset R$:
   - $\mathsf{C}(x)$ is TRUE for {1,3}
   - $\mathsf{V}(x)$ is TRUE for {2}
   - $\mathsf{p}(x)$ is TRUE for {1}
   - $\mathsf{a}(x)$ is TRUE for {2}
   - $\mathsf{t}(x)$ is TRUE for {3}

     ...
   - $\mathsf{z}(x)$ is TRUE for none

and CF grammars (Ritchie et al., 1992). Except for CF grammars, these refinements can be restricted enough to make them regular in their WGC.

[4] Origin semantics has *no* relation to lexical semantics. It concerns the correspondence information for output symbols.

Binary relations in R:

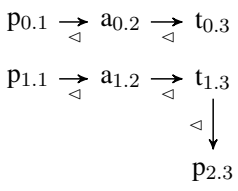
- $\mathsf{succ}(x, y)$ is TRUE for {(1,2), (2,3)}
- $\mathsf{prec}(x, y)$ is TRUE for {(1,2),(2,3),(1,3)}

In order to transform input strings into output strings, MSO logical transductions define a copy set C of some fixed size k. The k members of the copy set act as indexes for copies of the input. Output functions define what output segments can be defined in which copy. These functions use `this font`. For example, to add a *p* at the end of every word, we need a transduction with a copy set of size 2 (3). Copy 1 is used to output the input (3a-b), while Copy 2 is used for adding *p* as another output correspondent of the final segment (3d-e). We use the predicate in (3c) to find the final segment. Predicates are just shorthand and use **this font**. When defining output correspondents, the output functions reference which copy is being used in the form of a superscript, e.g., the output function $\mathtt{p}(x^2)$ generates the label p on the output correspondent of x in Copy 2 based on the properties of the input domain element x.

3. *Logical transduction for adding a final* p
   (a) $\forall \mathsf{label} \in L$: $\mathtt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x)$
   (b) $\mathsf{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y)$
   (c) $\mathbf{final}(x) \stackrel{\text{def}}{=} \neg\exists y[\mathsf{succ}(x, y)]$
   (d) $\mathtt{p}(x^2) \stackrel{\text{def}}{=} \mathbf{final}(x)$
   (e) $\mathtt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \mathbf{final}(y)$

We visualize this transduction for the input-output pair (*pat,patp*) in Figure 2. The index of the domain elements in the input is shown via a subscript $0.i$ where *i* is an element of D. For output nodes, their index is a subscript $c.i$ where *c* is an element of the copy set C. The immediate successor relation is shown as $\lhd$-labeled edges. We do not show the precedence relation.

Figure 2: Applying the transduction (3) on *pat*



1-way FSTs compute rational functions which are computed by order-preserving MSO transduction (Bojańczyk, 2014; Filiot, 2015; Filiot and Reynier, 2016). A transduction is order-preserving if the indexing and linearization of output nodes satisfy the following criteria in (4) (Chandlee and Jardine, 2019). If a transduction does not satisfy these criteria, then it is non-order-preserving. It cannot be modeled by a 1-way FST, but instead requires a 2-way FST.

4. *Criteria for order-preservation*
   (a) For a domain element x and for any pair of its output correspondents y^c, z^d, if $c < d$, then y^c is ordered before y^d.
   (b) For two domain elements x, y, if x precedes y in the input, then the output correspondents of x precede the output correspondent of y.

The transduction in (3) satisfies these criteria as visualized in Figure 2. Visually, the first criterion means that we can't have any upwards-going edges that are vertical, while the second criterion means we can't have any right-to-left edges. In this paper, we use origin semantics and order-preservation as diagnostics for the SGC of morphology, while the grounding of SGC is segmental correspondence.

4 Convergence of weak and strong generative capacity in simple affixation

Simple affixation processes like suffixation and prefixation are rational functions in their WGC. We show they are also rational functions in their SGC.

Suffixes are added at the end of input, while prefixes are added at the beginning, e.g., the English suffix *-ed* and prefix *re-*. In terms of grounding, the desired segmental correspondences are that the suffix (prefix) is in correspondence with the right (left) edge of the input. With this grounding, 1-way FSTs capture these correspondences. In Figure 3, we show the FSTs and their semantics. Throughout this paper, we illustrate these functions with nonce words like *at, tra, pa*.

The suffix is generated if we reached the end of the input string and can no longer read in more symbols.[5] In terms of origin semantics, the FST treats the suffix segments as output correspondents of the final segment. For prefixation, the results are analogous. The prefix *re-* is generated at the beginning of the word with transparent origin semantics.

Similar properties hold for the logical transductions that correspond to these FSTs. For suffixation,

[5]The suffixation FST uses a final outgoing transition arc that outputs *ed* once no more input symbols are read.

Figure 3: 1-way FSTs for suffixing *-ed* (a), prefixing *re-* (b), and their origin information (c,d) for input *at*.

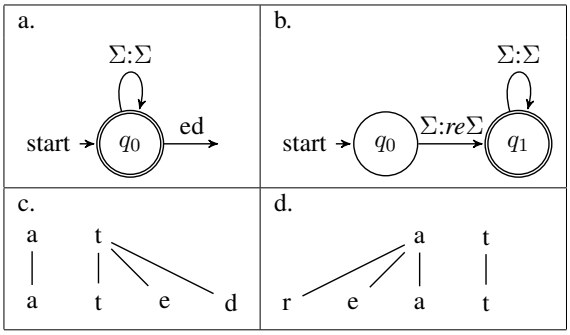
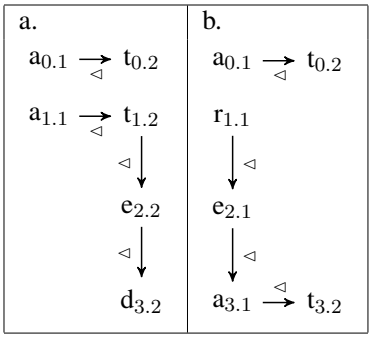


Figure 4: Applying the transductions (5,6) for suffixation (a) and prefixation (b) on the input *at*.



the transduction uses a copy set of size 3 (5). Copy 1 is reserved for outputting the base (5a). The suffix segments are defined in Copies 2-3 as output correspondents of the final segment (5b-c).[6] As for prefixation, the transduction uses a copy set of size 3 (6). In Copies 1-2, the prefix segments are defined as output correspondents of the initial segment (6b-c). The base is defined in Copy 3 (5d).

5. *Order-preserving transduction for suffixation*
   (a) $\forall label \in L$: $\texttt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x)$
   (b) $\texttt{e}(x^2) \stackrel{\text{def}}{=} \mathbf{final}(x)$
   (c) $\texttt{d}(x^3) \stackrel{\text{def}}{=} \mathbf{final}(x)$
   (d) $\texttt{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y)$
   (e) $\texttt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \mathbf{final}(y)$
   (f) $\texttt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \mathbf{final}(y)$
6. *Order-preserving transduction for prefixation*
   (a) $\mathbf{init}(x) \stackrel{\text{def}}{=} \neg\exists y[\mathsf{succ}(y, x)]$
   (b) $\texttt{r}(x^1) \stackrel{\text{def}}{=} \mathbf{init}(x)$
   (c) $\texttt{e}(x^2) \stackrel{\text{def}}{=} \mathbf{init}(x)$
   (d) $\forall label \in L$: $\texttt{label}(x^3) \stackrel{\text{def}}{=} \mathsf{label}(x)$
   (e) $\texttt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$
   (f) $\texttt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$
   (g) $\texttt{succ}(x^3, y^3) \stackrel{\text{def}}{=} \mathsf{succ}(x, y)$

Based on these correspondences, all segments are linearized in a way that satisfies order-preservation (5d-f, 6e-g). The transductions are order-preserving because they satisfy the criteria in (4). Visually, we can see these correspondences based on the indexing of the input and output in Figure 4. For suffixation, the output correspondents of the final segment $t_{0.2}$ form a chain of immediate successor from Copy 1 to Copy 3. Visually, they form a falling line instead of a rising line. Likewise for prefixation, the output correspondents of the initial segment form a similar chain.

Thus in terms of WGC, suffixation and prefixation are computable as order-preserving rational functions in the form of 1-way FSTs. In terms of SGC, these functions capture the desired segmental correspondences based on origin semantics.

5 Divergence between weak and strong generative capacity

Unlike in simple affixation, we argue that WGC and SGC do diverge for more complicated cases of morphology, including infixation (§5.1), mobile affixation (§5.2), and partial reduplication (§5.3).

5.1 Infixation

Unlike simple affixation, the location of an infix is inside the input. For example in Chamorro, the infix <*um*> is before the first vowel, after any consonants (Yu, 2007, 89).[7]

7. epanglo <um>epanglo
   hup g<um>upu
   tristi tr<um>isti

There are two camps of generative theories for infixation: Phonological Subcategorization and Phonological Readjustment (Yu, 2007). These camps differ in terms of what they propose is the underlying location of the infix. In Phonological Subcategorization, surface infixes are treated as 'underlying infixes'. The input is scanned for the infix's surface location (the pivot) and the infix is added directly into this location (McCarthy

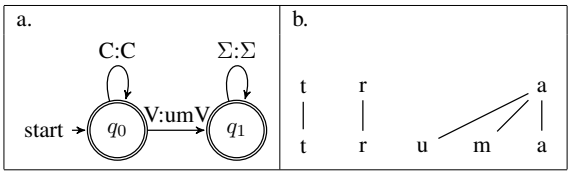
[6] We cannot use a copy set of size 2. Although we only add a single substring *-ed* to the base, the elements of the affix *-ed* must each occupy a unique copy in order to ensure order-preservation.

[7] Glossing is 'hunt crabs', 'to look for crabs'; 'to fly', 'the bird flew'; 'sad', 'becomes sad'.

and Prince, 1990; Blevins, 1999; Nevins and Vaux, 2003; Fitzpatrick, 2004; Yu, 2007; Samuels, 2010). In contrast in Prosodic Readjustment, infixes are treated as underlyingly prefixes or suffixes which get displaced. For Chamorro, the infix would first get added as a prefix, and then it is moved to its surface location. The shift is triggered by the need to optimize syllable structure (McCarthy and Prince, 1993). Though some Readjustment theories assume the shift is triggered by a morphological operation (Embick, 2010; Kalin, 2019, In prep).

Both theories have the same WGC: They generate the same infixed languages. But, they have different SGC when grounded in terms of segmental correspondence. Subcategorization theories are computable with 1-way FSTs where the origin semantics treats the infix as the output correspondent of the infixed location (the pivot).[8]

Figure 5: 1-way FST for infixation with subcategorization (a), and its origin information (b) for an input *tra*.

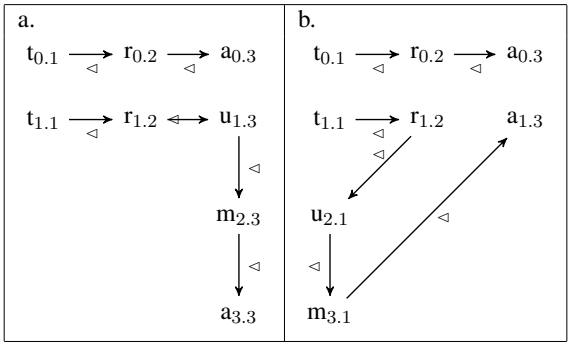


The rational function that is computed by the 1-way FST thus matches the intensional description of subcategorization theories. This is further verified by the order-preserving transduction (8) that's computed by this FST. This transduction uses a predicate $\mathbf{V1}(x)$ (8a) to find the pivot. The predicate finds the first vowel in the input, i.e., a vowel which is not preceded by any other vowels. With this predicate, the base (minus the pivot) is generated in Copy 1 (8b), the infix in Copies 1-2 (8c-d), and the pivot vowel in Copy 3 (8e). We visualize its correspondences in Figure 6a.

8. *Order-preserving transduction for infixation*
   (a) $\mathbf{V1} \stackrel{\text{def}}{=} \mathsf{V}(x) \wedge \neg\exists y[\mathsf{V}(y) \wedge \mathsf{prec}(y, x)]$
   (b) $\forall label \in L - \mathsf{u}(x)$:
   $\texttt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x) \wedge \neg\mathbf{V1}$
   (c) $\mathtt{u}(x^1) \stackrel{\text{def}}{=} \mathbf{V1}(x) \vee \mathsf{u}(x)$
   (d) $\mathtt{m}(x^2) \stackrel{\text{def}}{=} \mathbf{V1}(x)$
   (e) $\forall label \in L$:
   $\texttt{label}(x^3) \stackrel{\text{def}}{=} \mathsf{label}(x) \wedge \mathbf{V1}$
   (f) $\texttt{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y) \wedge \neg\mathbf{V1}(x)$
   (g) $\texttt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{V1}(x) \wedge \mathbf{V1}(y)$
   (h) $\texttt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{V1}(x) \wedge \mathbf{V1}(y)$
   (i) $\texttt{succ}(x^3, y^1) \stackrel{\text{def}}{=} \mathbf{V1}(y) \wedge \mathsf{succ}(x, y)$

[8] In fact, most computational models of infixation exploit the use of pivots in order to make them easier to design (Roark and Sproat, 2007; Wilson, 2018).

Figure 6: For infixation, applying the order-preserving transduction (8) (a) and the non-order-preserving transduction (9) (b) for an input *tra*.



In contrast, the Readjustment theories faithfully match a non-order-preserving transduction (9). Here, the infix is defined as the output correspondent of the initial segment (9b-c), but it is linearized with respect to an internal segment (9e-g). The base is in Copy 1 (9a). This transduction is visualized in Figure 6b. The transduction is not order-preserving because it violates both criteria in (4). It violates the second criterion because there's a right-to-left edge between $r_{1.2}$ and $u_{2.1}$.

9. *Non-order-preserving transduction for infixation*
   (a) $\forall label \in L$: $\texttt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x)$
   (b) $\mathtt{u}(x^2) \stackrel{\text{def}}{=} \mathbf{init}(x)$
   (c) $\mathtt{m}(x^3) \stackrel{\text{def}}{=} \mathbf{init}(x)$
   (d) $\texttt{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y) \wedge \neg\mathbf{V1}(y)$
   (e) $\texttt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{init}(y) \wedge \exists z[\mathbf{V1}(z) \wedge \mathsf{succ}(x, z)]$
   (f) $\texttt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$
   (g) $\texttt{succ}(x^3, y^1) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{V1}$

In terms of finite-state calculus, this non-order-preserving transduction cannot be defined by a 1-way FST. It instead needs a more expressive 2-way FST, as illustrated in the appendix. Thus, the two infixation theories diverge in their SGC.

5.2 Mobile affixation

A similar divergence is found in mobile affixation. Here the affix surfaces in different positions based

on the morphological or phonological properties of the input. This is typologically rare but attested (Fulmer, 1991, 1997; Noyer, 1994; Paster, 2006; Kim, 2010, 2015; Jenks and Rose, 2015). For example in Hamshen Armenian (Vaux, 1998, 2007; Bezrukov and Dolatian, 2020), the indicative is a prefix *g-* for vowel-initial verbs, but a suffix *-gu* for consonant-initial verbs.

10. (a) *g-arne* 'INDC-takes'
    (b) *kale-gu* 'walks-INDC'

Similar to infixation, there are roughly two theoretical strategies for mobile affixation, which we call 'floating' and 'shifting'. The floating analysis posits that the mobile affix consists of morphs which aren't specified as being prefixes or suffixes *g,gu* (cf. Noyer, 1994). The correct morph is chosen based on the phonological properties of the base. In contrast, the shifting analysis posits that the mobile affix consists of a single morph which is underlyingly a prefix *g(u)-* (cf. Kim, 2010). The prefix will shift or move to the right-edge for V-initial bases.[9]

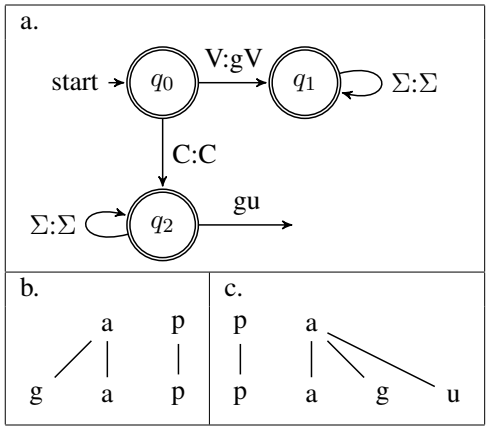
As with infixation, these two theories are grounded in different segmental correspondences. We show that they differ in SGC. First, the floating analysis matches the computation of a 1-way FST. Over a 1-way FST, mobile affixation requires first checking if the input is V-initial or C-initial. If V-initial, then we generate the prefix *g-*. Otherwise, we move to the end of the string and output the suffix *-gu*.[10] This is shown in Figure 7.

The correspondences generated by this 1-way FST are visible in the corresponding order-preserving transduction (11). This transduction is order-preserving and uses a copy set of size 4. We generate the prefix *g-* in Copy 1 as an output correspondent of the initial segment (11a), while we generate the suffix *-gu* in Copies 3-4 as the output correspondents of the final segment (11c-d). The base is generated in Copy 2 (11b). This is visualized in Figure 8a,b. These correspondences match the ones desired for the floating analysis.

[9]For Armenian, the morphs *g-* and *-gu* are segmentally non-identical; but most cases of mobile affixation involve identical morphs. A third approach is suppletive subcategorization whereby the affix has two allomorphs *g-* and *-gu* which are specified as a prefix and suffix respectively (cf. Paster, 2006, 2009; Kim, 2015). This is equivalent in SGC to the floating analysis.

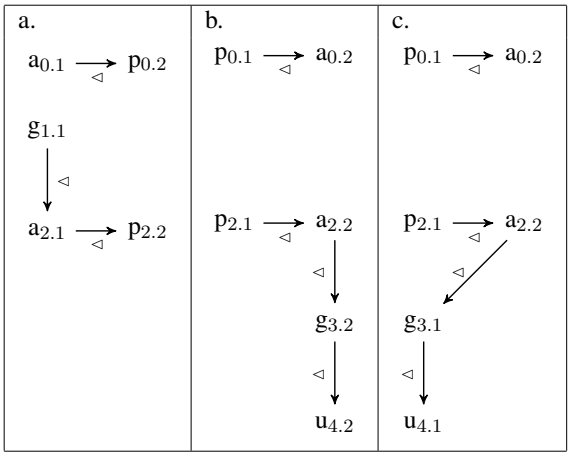
[10]As with suffixation, this FST outputs *gu* when no more input symbols are read at state q_2.

Figure 7: 1-way FST for mobile affixation (a), and its origin information (b,c) for inputs *ap*, *pa*.



11. *Order-preserving transduction for mobile affixation*
    (a) $\mathtt{g}(x^1) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathsf{V}(x)$
    (b) $\forall label \in L\text{: } \mathtt{label}(x^2) \stackrel{\text{def}}{=} \mathsf{label}(x)$
    (c) $\mathtt{g}(x^3) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \exists y[\mathbf{init}(y) \wedge \mathsf{C}(y)]$
    (d) $\mathtt{u}(x^4) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \exists y[\mathbf{init}(y) \wedge \mathsf{C}(y)]$
    (e) $\mathtt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$
    (f) $\mathtt{succ}(x^2, y^2) \stackrel{\text{def}}{=} \mathsf{succ}(x, y)$
    (g) $\mathtt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \mathbf{final}(y)$
    (h) $\mathtt{succ}(x^3, y^4) = \mathbf{final}(x) \wedge \mathbf{final}(y)$

Figure 8: For mobile affixation, applying the order-preserving transduction (11) on input *ap* (a), *pa* (b), and applying the non-order-preserving transduction (12) on input *pa* (c).



In contrast, the shifting analysis is matched by a non-order-preserving function. For example, the 1-way FST in Figure 7 computes an order-preserving transduction (11) which defines the suffix in terms

of the final segment. But if we sacrifice order-preservation (12), we can alternatively define the suffix segments $g_{3.1}$,$u_{4.1}$ as output correspondents to the initial segment $p_{0.1}$ (12c-d). This transduction is not order-preserving as shown in Figure 8c. It violates the second criterion in (4) because there is a right-to-left edge from $a_{2.2}$ to $g_{3.1}$. Such a non-order-preserving function cannot be computed by a 1-way FST, but would need a more expressive 2-way FST, as illustrated in the appendix.

12. *Non-order-preserving transduction for mobile affixation*
    (a) $\mathtt{g}(x^1) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathsf{V}(x)$
    (b) $\forall label \in L\text{: } \mathtt{label}(x^2) \stackrel{\text{def}}{=} \mathsf{label}(x)$
    (c) $\mathtt{g}(x^3) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathsf{C}(x)$
    (d) $\mathtt{u}(x^4) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathsf{C}(x)$
    (e) $\mathtt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$
    (f) $\mathtt{succ}(x^2, y^2) \stackrel{\text{def}}{=} \mathsf{succ}(x, y)$
    (g) $\mathtt{succ}(x^2, y^3) \stackrel{\text{def}}{=} \mathbf{final}(x) \wedge \mathbf{init}(y)$
    (h) $\mathtt{succ}(x^3, y^4) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{init}(y)$

Thus, the floating and shifting analyses are both rational functions in WGC, but the latter is not rational in its SGC.

5.3 Reduplication

Reduplication shows the most drastic differences between WGC and SGC. The typology of reduplication is roughly divided into partial and total reduplication. Partial reduplication occurs when there is a fixed bound on how many segments are copied, while total reduplication occurs when there is no bound (Moravcsik, 1978).

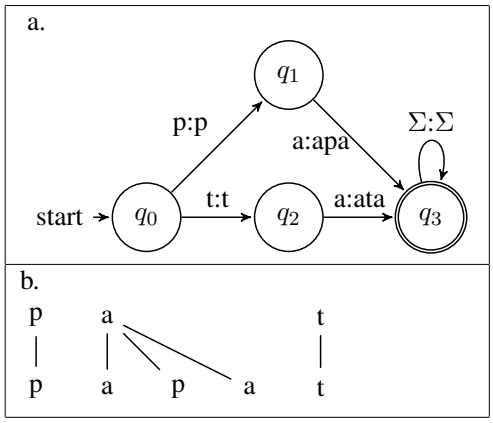
13. (a) *Agta* (Moravcsik, 1978, 311)
    takki→tak∼takki 'leg'→'legs'
    (b) *Indonesian* (Cohn, 1989, 308)
    buku→buku∼buku 'book'→'books'

It is well-known that partial reduplication is definable with 1-way FSTs while total reduplication is not (Culy, 1985; Roark and Sproat, 2007; Chandlee and Heinz, 2012; Chandlee, 2017). Instead, total reduplication requires the more expressive power of 2-way FSTs (Dolatian and Heinz, 2018b, 2020). We show that the divide between 1-way and 2-way FSTs likewise exists for the SGC of partial reduplication.

Across different theories of reduplication, a common assumption is that the reduplicated segments are directly derived from underlying segments, whether via prosodic associations (Marantz, 1982), correspondence (McCarthy and Prince, 1995, 1999), morpheme-repetition (Steriade, 1988; Inkelas and Zoll, 2005), or lax precedence relations (Raimy, 2000). However, 1-way FSTs cannot capture this (SGC) (Dolatian and Heinz, 2020).

Because partial reduplication has a bound on the size of the reduplicant, we can always design a 1-way FST for it. For example, the FST in Figure 9a computes initial CV-copying for a small alphabet {p,a,t}. After we read the vowel, the reduplicant string is generated as *pa* if we are in state q_1; otherwise the string is generated as *ta*. But in terms of origin semantics in Figure 9b, this 1-way FST treats the repeated consonant as an output correspondent of the input vowel.

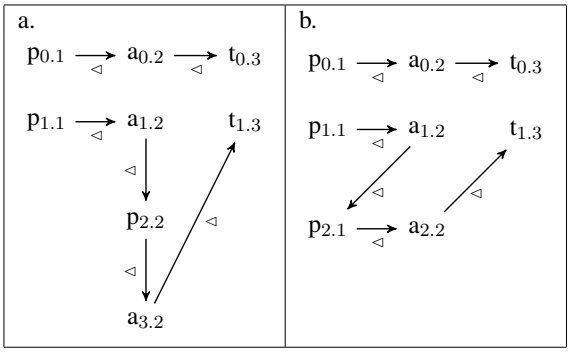
Figure 9: 1-way FST for partial reduplication (a), and its origin information (b) for an input *pat*.



The divergence between theory (SGC) and computation (WGC) is clearer for the corresponding order-preserving transduction (14). This is visualized in Figure 10a. The reduplicants are defined in Copies 2-3 (14c-d) as the output correspondents of the second segment, found via the predicate $\mathbf{2nd}(x)$. The repeated consonant $p_{2.2}$ is defined as the output correspondent of $a_{0.2}$ even though it gets its segmental labels from the first segment $p_{0.1}$.

14. *Order-preserving transduction for partial reduplication*
    (a) $\forall label \in L\text{: } \mathtt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x)$
    (b) $\mathbf{2nd}(x) \stackrel{\text{def}}{=} \exists y[\mathsf{succ}(y, x) \wedge \mathbf{init}(y)]$
    (c) $\mathtt{p}(x^2) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \exists y[\mathbf{init}(y) \wedge \mathsf{p}(y)]$
    (d) $\mathtt{t}(x^2) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \exists y[\mathbf{init}(y) \wedge \mathsf{t}(y)]$
    (e) $\mathtt{a}(x^3) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathsf{a}(x)$
    (f) $\mathtt{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y) \wedge \neg\mathbf{2nd}(x)$
    (g) $\mathtt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathbf{2nd}(y)$

Figure 10: For partial reduplication, applying the order-preserving transduction (14 (a) and the non-order-preserving transduction (15) (b) for an input *pat*.



(h) $\texttt{succ}(x^3, y^3) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathbf{2nd}(y)$
(i) $\texttt{succ}(x^3, y^1) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathsf{succ}(x, y)$

Thus, the 1-way FST and its order-preserving transduction posit a hidden structure which does not match linguistic analyses over the identity between the input consonant *p* and the two output segments *p* (Wilbur, 1973). Based on identity, the desired function is a non-order-preserving transduction (15) where the repeated consonants are defined as output correspondents of the initial consonant (15b-c). This transduction is visualized in Figure 10b. In terms of finite-state calculus, this non-order-preserving transduction cannot be computed with a 1-way FST but needs a more expressive 2-way FST, as illustrated in the appendix.

15. *Non-order-preserving transduction for partial reduplication*
    (a) $\forall label \in L\colon \texttt{label}(x^1) \stackrel{\text{def}}{=} \mathsf{label}(x)$
    (b) $\texttt{p}(x^2) \stackrel{\text{def}}{=} \mathbf{init}(x \wedge \mathsf{p}(x)$
    (c) $\texttt{t}(x^2) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathsf{t}(x)]$
    (d) $\texttt{a}(x^2) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathsf{a}(x)]$
    (e) $\texttt{succ}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ}(x, y) \wedge \neg\mathbf{2nd}(x)$
    (f) $\texttt{succ}(x^1, y^2) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathbf{init}(y)$
    (g) $\texttt{succ}(x^2, y^2) \stackrel{\text{def}}{=} \mathbf{init}(x) \wedge \mathbf{2nd}(x)$
    (h) $\texttt{succ}(x^2, y^1) \stackrel{\text{def}}{=} \mathbf{2nd}(x) \wedge \mathsf{succ}(x, y)$

In contrast to partial reduplication, total reduplication cannot be modeled with a 1-way FST at all. Thus it cannot be computed with an order-preserving MSO transduction. This is because in order to make total reduplication be order-preserving, we would need to have a copy set with a fixed size k such that there would be a single copy-set member for every repeated segment. But because there is no bound on the number of repeated segments in total reduplication, then we cannot use an order-preserving transduction with a fixed copy set. Instead, total reduplication requires a non-order-preserving MSO transduction which is computed by a 2-way FST (Engelfriet and Hoogeboom, 2001). As Dolatian and Heinz (2018b, 2020) show, a 2-way FST provides the right WGC for total reduplication, and the right SGC in terms of origin semantics. In this paper's framework, their result is replicated in terms of order-preservation.

In fact, based on the mismatch between the WGC and SGC for partial reduplication, Dolatian and Heinz (2018b, 2020) argue that the more meaningful implementation is with a 2-way FST. Their argument is based on the difference in origin semantics. By defining partial reduplication with 2-way FSTs, Dolatian and Heinz (2018b, 2020) are able to unify both partial and total reduplication into the same computational framework. Further evidence for this unity between partial and total reduplication comes from learnability (Dolatian and Heinz, 2018a), Deep Learning (Nelson et al., 2020), and the computational typology of reduplication (Dolatian and Heinz, 2019, 2020).

6 Conclusion

Weak and strong generative capacity are separate measures for evaluating the correctness and completeness of formal grammars. 1-way FSTs are adequate to model the weak generative capacity of morphology, and virtually all theories of morphology are finite-state in weak generative capacity. However, in terms of strong generative capacity, different morphological analyses posit different hidden structures in terms of segmental correspondence. Some of these analyses are definable with 1-way finite-state transducers, while some are not. Thus, they differ in their strong generative capacity. This result is surprising given the wide applicability of 1-way FSTs to computational morphology. We based our results on precise mathematically-defined diagnostics that are independently used in theoretical computer science. Our result provides a concrete example of how weak and strong generative capacity may diverge.

References

Alëna Aksënova and Aniello De Santo. 2019. Strict locality in morphological derivations. In *Proceedings*

of the 53rd meeting of Chicago Linguistics Society (CLS 53), pages 1–12.

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.

Jan W Amtrup. 2003. Morphology in machine translation systems: Efficient integration of finite state transducers and feature structure descriptions. *Machine Translation*, 18(3):217–238.

Jan W Amtrup. 2004. Efficient finite state unification morphology. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 453–458.

Samuel Andersson, Hossep Dolatian, and Yiding Hao. 2020. Computing vowel harmony: The generative capacity of search & copy. In *Proceedings of the Annual Meetings on Phonology*, volume 8.

Kenneth Beesley and Lauri Karttunen. 2003. *Finite-state morphology: Xerox tools and techniques*. CSLI Publications, Stanford, CA.

Kenneth R Beesley. 1998. Constraining separated morphotactic dependencies in finite-state grammars. In *Proceedings of International Workshop on Finite State Methods in Natural Language Processing*, pages 118–127, Bilkent. Bilkent University.

Nikita Bezrukov and Hossep Dolatian. 2020. Mobile affixes across Western Armenian: Conflicts across modules. In *University of Pennsylvania Working Papers in Linguistics*, volume 26.

Bronwyn Bjorkman and Ewan Dunbar. 2016. Finite-state phonology predicts a typological gap in cyclic stress assignment. *Linguistic Inquiry*, 47(2):351–363.

Juliette Blevins. 1999. Untangling Leti infixation. *Oceanic linguistics*, pages 383–403.

Mikołaj Bojańczyk. 2014. Transducers with origin information. In *Automata, Languages, and Programming*, pages 26–37, Berlin, Heidelberg. Springer.

Mikołaj Bojańczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle. 2017. Which classes of origin graphs are generated by transducers. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 114:1–114:13, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non-concatenative morphology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP13)*, page 345–56, Stroudsburg, PA. Association for Computational Linguistics.

Joan Bresnan, Ronald M Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635.

Guy Carden. 1983. The non-finite-state-ness of the word formation component. *Linguistic Inquiry*, 14(3):537–541.

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 and Jeffrey Heinz. 2012. Bounded copying is subsequential: Implications for metathesis and reduplication. In *Proceedings of the 12th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, SIGMORPHON '12, pages 42–51, Montreal, Canada. Association for Computational Linguistics.

Jane Chandlee and Jeffrey Heinz. 2018. Strict locality and phonological maps. *Linguistic Inquiry*, 49(1):23–60.

Jane Chandlee, Jeffrey Heinz, and Adam Jardine. 2018. Input strictly local opaque maps. *Phonology*, 35(2):171–205.

Jane Chandlee and Adam Jardine. 2019. Quantifier-free least fixed point functions for phonology. In *Proceedings of the 16th Meeting on the Mathematics of Language (MoL 16)*, Toronto, Canada. Association for Computational Linguistics.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

Alexander Clark and Ryo Yoshinaka. 2014. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96(1-2):5–31.

Yael Cohen-Sygal and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32(1):49–82.

Abigail C Cohn. 1989. Stress in Indonesian and bracketing paradoxes. *Natural language & linguistic theory*, 7(2):167–216.

Bruno Courcelle. 1997. The expression of graph properties and graph transformations in monadic second-order logic. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations*, volume 1, pages 313–400. World Scientific.

Chet Creider, Jorge Hankamer, and Derick Wood. 1995. Preset two-head automata and morphological analysis of natural language. *International journal of computer mathematics*, 58(1-2):1–18.

Christopher Culy. 1985. The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8:345–351.

Nick Danis and Adam Jardine. 2019. Q-theory representations are logically equivalent to autosegmental representations. In *Proceedings of the Society for Computation in Linguistics*, volume 2, pages 29–38.

Hossep Dolatian. 2020. *Computational locality of cyclic phonology in Armenian*. Ph.D. thesis, Stony Brook University.

Hossep Dolatian and Jeffrey Heinz. 2018a. Learning reduplication with 2-way finite-state transducers. In *Proceedings of Machine Learning Research: International Conference on Grammatical Inference*, volume 93 of *Proceedings of Machine Learning Research*, pages 67–80, Wroclaw, Poland.

Hossep Dolatian and Jeffrey Heinz. 2018b. Modeling reduplication with 2-way finite-state transducers. In *Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Brussells, Belgium. Association for Computational Linguistics.

Hossep Dolatian and Jeffrey Heinz. 2019. Redtyp: A database of reduplication with computational models. In *Proceedings of the Society for Computation in Linguistics*, volume 2. Article 3.

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 and Jonathan Rawski. 2020. Multi input strictly local functions for templatic morphology. In *Proceedings of the Society for Computation in Linguistics*, volume 3.

Jason Eisner. 1997. Efficient generation in primitive optimality theory. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 313–320.

Jason Eisner. 2000a. Directional constraint evaluation in optimality theory. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Jason Eisner. 2000b. Easy and hard constraint ranking in ot: Algorithms and complexity. In *Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*, pages 22–33.

David Embick. 2010. *Localism versus globalism in morphology and phonology*, volume 60 of *Linguistic Inquiry Monographs*. MIT Press, Cambridge, MA.

Joost Engelfriet and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. *Transactions of the Association for Computational Linguistics*, 2(2):216–254.

Marina Ermolaeva and Daniel Edmiston. 2018. Distributed morphology as a regular relation. In *Proceedings of the Society for Computation in Linguistics*, volume 1, pages 178–181.

Emmanuel Filiot. 2015. Logic-automata connections for transformations. In *Logic and Its Applications*, pages 30–57, Berlin, Heidelberg. Springer Berlin Heidelberg.

Emmanuel Filiot and Pierre-Alain Reynier. 2016. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19.

Justin Fitzpatrick. 2004. A concatenative theory of possible affix types. In Andrés Salanova, editor, *Papers from EVELIN I. MIT Working Papers in Linguistics*.

Robert Frank and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational linguistics*, 24(2):307–315.

S Lee Fulmer. 1991. Dual-position affixes in Afar: An argument for phonologically driven morphology. In *Proceedings of the Ninth West Coast Conference on Formal Linguistics.*, pages 189–203, Stanford. CSLI.

Sandra Lee Fulmer. 1997. *Parallelism and planes in Optimality Theory: Evidence from Afar*. Ph.D. thesis, University of Arizona, Tucson, AZ.

Michael Gasser. 2009. Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 309–317.

Gerald Gazdar and Geoffrey K Pullum. 1985. Computationally relevant properties of natural languages and their grammars. *New generation computing*, 3:273–306.

Dale Gerdemann and Mans Hulden. 2012. Practical finite state optimality theory. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 10–19.

Dale Gerdemann and Gertjan Van Noord. 2000. Approximation and exactness in finite state optimality theory. In *Procedings of the 5th Meeting of the ACL Special Interest Group in Computational Phonology*, pages 34–45.

Thomas Graf. 2010a. Comparing incomparable frameworks: A model theoretic approach to phonology. In *University of Pennsylvania Working Papers in Linguistics*, volume 16.

Thomas Graf. 2010b. Logics of phonological reasoning. Master's thesis, University of California, Los Angeles.

Michael Hammond. 1993. On the absence of category-changing prefixes in English. *Linguistic Inquiry*, 24(3):562–567.

Yiding Hao. 2019. Finite-state optimality theory: non-rationality of harmonic serialism. *Journal of Language Modelling*, 7(2):49–99.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry Hyman and Frans Plank, editors, *Phonological Typology*, Phonetics and Phonology, chapter 5, pages 126–195. Mouton de Gruyter, Berlin.

Jeffrey Heinz and William Idsardi. 2013. What complexity differences reveal about domains in language. *Topics in Cognitive Science*, 5(1):111–131.

Jeffrey Heinz, Gregory M Kobele, and Jason Riggle. 2009. Evaluating the complexity of optimality theory. *Linguistic Inquiry*, 40(2):277–288.

William J Idsardi. 2006. A simple proof that optimality theory is computationally intractable. *Linguistic Inquiry*, 37(2):271–275.

Sharon Inkelas and Cheryl Zoll. 2005. *Reduplication: Doubling in Morphology*. Cambridge University Press, Cambridge.

Adam Jardine, Nick Danis, and Luca Iacoponi. 2020. A formal investigation of Q-theory in comparison to autosegmental representations. *Linguistic Inquiry*.

Peter Jenks and Sharon Rose. 2015. Mobile object markers in moro: The role of tone. *Language*, 91(2):269–307.

C Douglas Johnson. 1972. *Formal aspects of phonological description*. Mouton, The Hague.

Michael B Kac. 1987. Surface transitivity, respectively coordination and context-freeness. *Natural Language & Linguistic Theory*, 5(3):441–452.

Michael B Kac, Alexis Manaster-Ramer, and William C Rounds. 1987. Simultaneous-distributive coordination and context-freeness. *Computational Linguistics*, 13:25–30.

Laura Kalin. 2019. When allomorphy meets infixation: Cyclicity and separation. Talk presented at the 12th Brussels Conference on Generative Linguistics: Suppletion, allomorphy, and syncretism.

Laura Kalin. In prep. Infixes really are prefixes/suffixes: Evidence from allomorphy on the fine timing of infixation. Ms. Princeton University.

Ronald M Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

Lauri Karttunen. 1998. The proper treatment of optimality in computational phonology. *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 1–12.

Lauri Karttunen. 2003. Computing with realizational morphology. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 203–214. Springer.

Martin Kay. 1987. Nonconcatenative finite-state morphology. In *Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark. Association for Computational Linguistics.

Yuni Kim. 2010. Phonological and morphological conditions on affix order in Huave. *Morphology*, 20(1):133–163.

Yuni Kim. 2015. Mobile affixation within a modular approach to the morphology-phonology interface. In Stela Manova, editor, *Affix ordering across languages and frameworks*, pages 111–123. Oxford University Press, New York.

George Anton Kiraz. 2001. *Computational nonlinear morphology: With emphasis on Semitic languages*. Cambridge University Press, Cambridge.

Gregory Michael Kobele. 2006. *Generating Copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California, Los Angeles.

Kimmo Koskenniemi. 1983. *Two-level morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

Hans-Ulrich Krieger and Hannes Pirker. 1993. Feature-based allomorphy. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 140–147.

Andrew Lamont. 2019a. Majority rule in Harmonic Serialism. In *Supplemental Proceedings of the 2018 Annual Meeting on Phonology*, pages 243–249, Washington, D.C. Linguistic Society of America.

Andrew Lamont. 2019b. Precedence is pathological: The problem of alphabetical sorting. In *Proceedings of the 36th West Coast Conference on Formal Linguistics*, pages 243–249, Somerville, MA. Cascadilla Press.

Andrew Lamont. prep. Optimizing over subsequences generates context-sensitive languages. Unpublished manuscript.

D Terence Langendoen. 1981. The generative capacity of word-formation components. *Linguistic Inquiry*, 12(2):320–322.

Alexis Manaster-Ramer. 1987a. Dutch as a formal language. *Linguistics and Philosophy*, 10(2):221–246.

Alexis Manaster-Ramer. 1987b. Subject-verb agreement in respective coordinations and context freeness. *Computational Linguistics*, 13:64–65.

Alec Marantz. 1982. Re reduplication. *Linguistic Inquiry*, 13(3):435–482.

John McCarthy and Alan Prince. 1990. Prosodic morphology and templatic morphology. In *Perspectives on Arabic linguistics II: Papers from the second annual symposium on Arabic linguistics*, pages 1–54. John Benjamins Amsterdam.

John J McCarthy and Alan Prince. 1993. Prosodic morphology I: Constraint interaction and satisfaction. Unpublished manuscript.

John J McCarthy and Alan Prince. 1995. Faithfulness and reduplicative identity. In Jill N. Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk, editors, *Papers in Optimality Theory*. Graduate Linguistic Student Association, University of Massachusetts, Amherst, MA.

John J McCarthy and Alan Prince. 1999. Faithfulness and identity in prosodic morphology. In René Kager, Harry van der Hulst, and Wim Zonneveld, editors, *The prosody-morphology interface*, pages 218–309. Cambridge University Press, Cambridge.

Philip H Miller. 1991. Scandinavian extraction phenomena revisited: Weak and strong generative capacity. *Linguistics and Philosophy*, 14(1):101–113.

Philip H Miller. 1999. *Strong generative capacity: The semantics of linguistic formalism*. CSLI publications, Stanford.

Edith Moravcsik. 1978. Reduplicative constructions. In Joseph Greenberg, editor, *Universals of Human Language*, volume 1, pages 297–334. Stanford University Press, Stanford, California.

Max Nelson, Hossep Dolatian, Jonathan Rawski, and Brandon Prickett. 2020. Probing rnn encoder-decoder generalization of subregular functions using reduplication. In *Proceedings of the Society for Computation in Linguistics*, volume 3.

Andrew Nevins and Bert Vaux. 2003. Metalinguistic, shmetalinguistic: The phonology of shmreduplication. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society*, volume 39, pages 702–721. Chicago Linguistic Society.

Rolf Noyer. 1994. Mobile affixes in Huave: Optimality and morphological wellformedness. In *Proceedings of the Twelfth West Coast Conference on Formal Linguistics. Stanford: CSLI*, pages 67–82, Stanford. CSLI.

Chris Oakden. 2020. Notational equivalence in tonal geometry. *Phonology*, 37:257–296.

Yohei Oseki. 2018. *Syntactic structures in morphological processing*. Ph.D. thesis, New York University.

Yohei Oseki and Alec Marantz. 2020. Modeling human morphological competence. *Frontiers in Psychology*, 11.

Yohei Oseki, Charles Yang, and Alec Marantz. 2019. Modeling hierarchical syntactic structures in morphological processing. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 43–52, Minneapolis, Minnesota. Association for Computational Linguistics.

Mary Paster. 2006. *Phonological conditions on affixation*. Ph.D. thesis, University of California, Berkeley, Berkeley, CA.

Mary Paster. 2009. Explaining phonological conditions on affixation: Evidence from suppletive allomorphy and affix ordering. *Word structure*, 2(1):18–37.

Alan Prince and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing, Oxford.

Geoffrey K Pullum and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4):471–504.

Daniel Radzinski. 1990. Unbounded syntactic copying in mandarin chinese. *Linguistics and Philosophy*, 13(1):113–127.

Daniel Radzinski. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, 17(3):277–300.

Eric Raimy. 2000. *The Phonology and Morphology of Reduplication*. Mouton de Gruyter, Berlin.

Alexis Manaster Ramer and Walter Savitch. 1997. Generative capacity matters. In *Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)*, page 106.

Jason Alan Riggle. 2004. *Generation, recognition, and learning in finite state Optimality Theory*. Ph.D. thesis, University of California, Los Angeles.

Graeme Ritchie. 1992. Languages generated by two-level morphological rules. *Computational Linguistics*, 18(1):41–59.

Graeme D Ritchie, Graham J Russell, Alan W Black, and Stephen G Pulman. 1992. *Computational morphology: Practical mechanisms for the English lexicon*. MIT press, Cambridge, MA.

Brian Roark and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford University Press, Oxford.

Emmanuel Roche and Yves Schabes, editors. 1997. *Finite-state language processing*. MIT press, Cambridge.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer.

James Rogers and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342.

Dirk Saléschus. 2008. On resolving long distance dependencies in Russian verbs. *Finite State Methods and Natural Language Processing*, pages 184–196.

Dirk Saléschus and Annette Hautli. 2008. On dissolving morphological long distance dependencies in Russian verbs. *Linguistic Issues in Language Technology*, 1(3):1–27.

Bridget Samuels. 2010. The topology of infixation and reduplication. *The Linguistic Review*, 27(2):131–176.

Walter J Savitch. 1982. *Abstract machines and grammars*. Little Brown and Company, Boston.

Elisabeth Selkirk. 1982. *The syntax of words*. Number 7 in Linguistic Inquiry Monographs. MIT Press, Cambridge, Mass.

Jeffrey Shallit. 2008. *A Second Course in Formal Languages and Automata Theory*, 1 edition. Cambridge University Press, New York, NY, USA.

Stuart M Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, pages 333–343.

Paul Smolensky. 1993. Harmonic grammars for formal languages. In *Advances in neural information processing systems 5*, pages 847–854, San Mateo. Morgan Kauffman.

Richard William Sproat. 1992. *Morphology and computation*. MIT press, Cambridge,MA.

Edward P Stabler. 2004. Varieties of crossing dependencies: structure dependence and mild context sensitivity. *Cognitive Science*, 28(5):699–720.

Donca Steriade. 1988. Reduplication and syllable transfer in Sanskrit and elsewhere. *Phonology*, 5(1):73–155.

Kristina Strother-Garcia. 2019. *Using model theory in phonology: a novel characterization of syllable structure and syllabification*. Ph.D. thesis, University of Delaware.

Harald Trost. 1990. The application of two-level morphology to non-concatenative German morphology. In *Proceedings of COLING-90*, pages 371–376. Association of Computational Linguistics, Morristown, NJ.

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.

Bert Vaux. 1998. *The phonology of Armenian*. Clarendon Press, Oxford.

Bert Vaux. 2007. Homshetsma: The language of the Armenians of Hamshen. In Hovhann H. Simonian, editor, *The Hemshin*, pages 257–278. Routledge, New York.

Ronnie Bring Wilbur. 1973. *The phonology of reduplication*. Ph.D. thesis, University of Indiana, Bloomington, Indiana.

Colin Wilson. 2018. Modeling morphological affixation with interpretable recurrent networks: Sequential rebinding controlled by hierarchical attention. In *Proceeding of CogSci40*, pages 2693–2698.

Alan CL Yu. 2007. *A Natural History of Infixation*. Number 15 in Oxford Studies in Theoretical Linguistics. Oxford University Press, Oxford.

Rémi Zajac. 1998. Feature structures, unification and finite-state transducers. In *Finite State Methods in Natural Language Processing*.

A 2-way FSTs for non-order-preserving function

The main paper showed that the SGC for some theories of infixation, mobile affixation, and partial reduplication requires non-order-preserving MSO transductions. These transductions cannot be computed by 1-way FSTs, but require the more expressive class of 2-way FSTs. We go over a definition of 2-way FSTs and then show example 2-way FSTs for these three morphological processes.

A 2-way FST differs from a 1-way FST in that it uses a direction parameter D. For a given transition arc, the read head can either advance (+1), retract (-1), or stay put on the input tape. The formal definition is shown below for a deterministic 2-way FST. This definition is taken from Dolatian and Heinz (2018b, 2020) who adapt it from other definitions (Shallit, 2008; Filiot and Reynier, 2016). We assume input strings are flanked by the start- and end-boundaries $\rtimes$,$\ltimes$.

16) **Definition:** A 2-way, deterministic FST is a six-tuple $(Q, \Sigma_{\ltimes}, \Gamma, q_0, F, \delta)$ such that:

Q is a finite set of states,
$\Sigma_{\ltimes} = \Sigma \cup \{\rtimes, \ltimes\}$ is the input alphabet,
Γ is the output alphabet,
$q_0 \in Q$ is the initial state,
$F \subseteq Q$ is the set of final states,
δ:$Q \times \Sigma \rightarrow Q \times \Gamma^* \times D$ is the transition function where the direction $D = \{-1, 0, +1\}$.

For Readjustment-based infixation, the non-order-preserving transduction from §5.1 (9) is computed by the 2-way in Figure 11a. The 2-way FST uses 2 passes. In the first pass, we output all the segments which will precede the infixed location. After the first pass, we rewind the machine back to the beginning of input ($\rtimes$) and start a second pass. In the second pass, we output the infix as the output correspondent of the initial segment, and then output everything after the word-initial sequence of consonants. The origin information in Figure 11b match that of Readjustment theories. We assume that if the output segment is generated upon reading the start boundary $\rtimes$, then it is treated as an output correspondent of the initial segment.

For the shifting analysis of mobile affixation, the non-order-preserving transduction from §5.2 (12) is computed by the 2-way FST in Figure 12a. The 2-way FST uses two passes over the input. In the first pass, we try to output the prefix and we output the base. We output the prefix only if the initial segment is V. We rewind the machine back to the beginning of input ($\rtimes$) and start a second pass. We read the initial segment again. If it is V, then we output nothing. But if it is a C, then we output the suffix. In terms of origin semantics in Figure 12b, the suffix now acts as an output correspondent to the initial segment.

Finally for partial reduplication, the non-order-preserving transduction from §5.3 (15) is computed by the 2-way FST in Figure 13a. The machines reads the input in 2 passes. In the first pass, we output the first CV substring. In the second pass, we output the entire input. By using two passes, we get the desired origin information. Both output symbols *p* correspond to the input symbol *p*.

Figure 11: 2-way FST for Readjustment-based infixation (a), and its origin information (b) for an input *tra*.

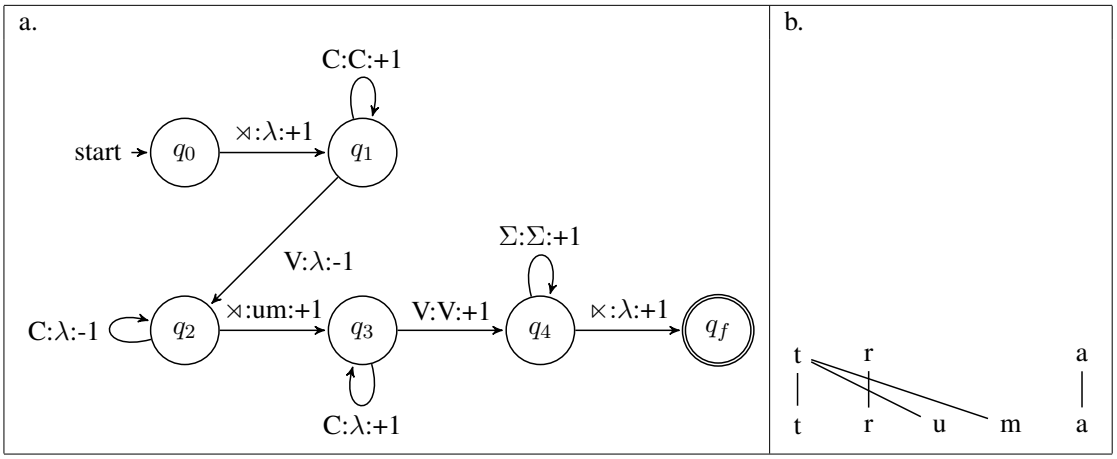


Figure 12: 2-way FST for mobile affixation (a), and its origin information (b,c) for inputs *ap*, *pa*.

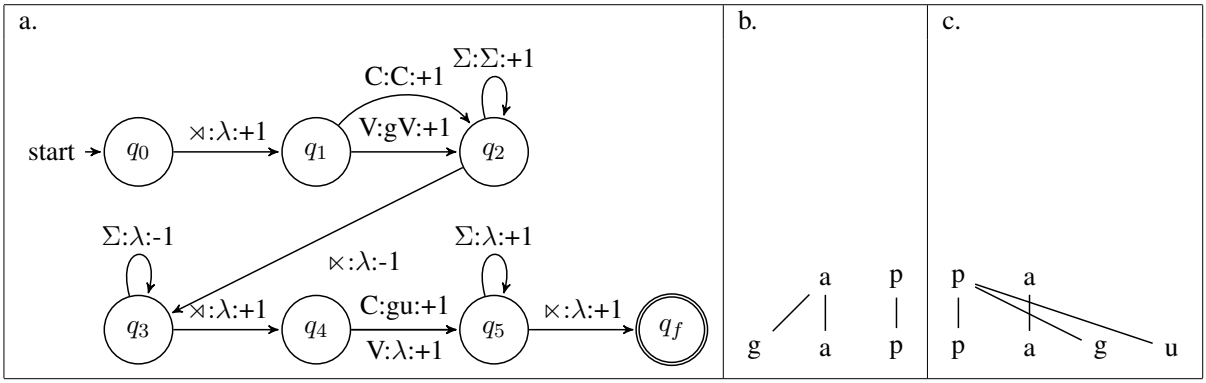


Figure 13: 2-way FST for partial reduplication (a), and its origin information (b) for an input *pat*.