On the proper treatment of weak determinism: Subsequentiality and simultaneous application in phonological maps

Eric Meinhardt,[*] Anna Mai,[*] Eric Baković,[*] Adam G. McCollum[†]
[*]UC San Diego, [†]Rutgers University

Abstract

Recent work has claimed that (segmental) phonological patterns are *subregular* (Heinz 2011a, b, 2018, Heinz and Idsardi 2013), occupying a delimited proper subregion of the regular functions — the *weakly deterministic* regular functions (Heinz and Lai 2013, Jardine 2016, McCollum et al. 2020a). Whether or not it is correct, this claim can only be properly assessed given a complete and accurate definition of weak determinism. We propose such a definition in this article, patching unintended holes in Heinz and Lai's (2013) original definition that we argue have led to the incorrect categorization of some phonological patterns as weakly deterministic. Building on Elgot and Mezei's (1965) demonstration that regular functions can be decomposed into two contradirectional subsequential functions, we show that weakly deterministic functions are those for which the composands do not *interact* in a way that we formally clarify here, making connections with more familiar notions of interaction in the phonological literature.

1 Introduction

Over the past decade, research at the intersection of computational phonology and formal language theory has advanced the SUBREGULAR HYPOTHESIS, which claims that phonological patterns "occupy some area strictly smaller than the regular languages" (Heinz 2011a, p. 147) and that "the study of the typology of the attested transformations [= input-output maps] in the light of the existing categories yields similar conclusions" (Heinz 2018, p. 198). This work has usefully defined various types of subregular functions relevant to the description of phonological transformations, all properly included within the regular functions in terms of their computational expressivity. A recent snapshot of the most significant of these subregular function types, and their hierarchical inclusion relationships with respect to one another, is shown in Figure 1.

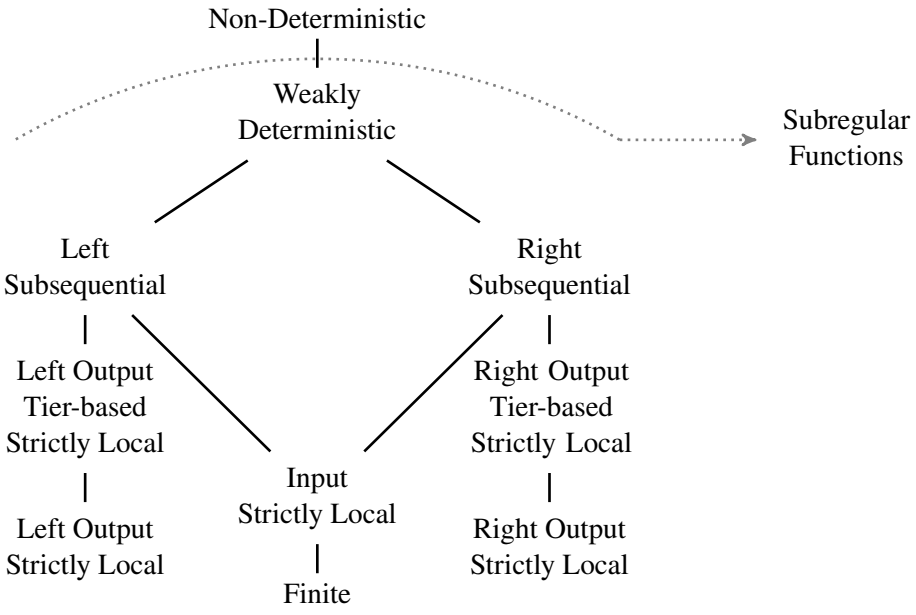

Figure 1 A hierarchy of regular functions, based on Heinz (2018) and Aksënova et al. (2020). All classes define *functional* string transductions, and everything below the non-deterministic regular functions are examples of *subregular* string transductions.

One of the aims of this body of work is to more precisely assess the minimum degree of computational expressivity required to describe some otherwise well-defined phonological phenomena, leading to results such as "local noniterative assimilation is input strictly local" (Chandlee 2014, Chandlee and Heinz 2018), "unidirectional iterative assimilation is output strictly local" (Chandlee et al. 2015), and "bidirectional iterative assimilation is weakly deterministic" (Heinz and Lai 2013). These results lead in turn to readily testable hypotheses about the relative complexity and learnability of different types of phonological patterns. Any results derived from testing these hypotheses are of course only as reliable as the results on which they are based, which are in turn only as reliable as the formal definitions of relevant subregular function types. It is thus imperative that these definitions be as meaningfully accurate as possible. From the perspective of theory development, good mathematical definitions are insightful and often suggest future directions for research or offer an unexpected unification of previously disparate results and intuitions.

Our focus in this article is on the WEAKLY DETERMINISTIC REGULAR FUNCTIONS (WDRFs), at the outer edge of the set of subregular functions depicted in Figure 1. Heinz and Lai (2013) propose the class of WDRFs to demarcate the expressivity boundary between amply attested bidirectional vowel harmony patterns and apparently unattested patterns involving e.g. non-myopic spreading (Wilson 2003, 2006) or majority rules spreading (Baković 2000). Jardine (2016) builds on Heinz and Lai (2013) by identifying attested tonal spreading patterns requiring greater expressivity than that afforded by the WDRFs, leading to a conjecture that the class of WDRFs also demarcates the expressivity boundary between segmental phonology and tonal phonology. McCollum et al. (2020a) respond by identifying attested segmental spreading patterns that require crossing the WDRF expressivity boundary; McCollum et al. (2020b) discuss some examples in greater depth.

The WDRF boundary is thus clearly an important one to define properly, given that it serves as the dividing line between amply attested patterns of bidirectional vowel harmony and other patterns ranging from the apparently rare, to the exclusively tonal, to the unattested. We demonstrate here that Heinz and Lai's (2013) original formal definition of the WDRFs does not accurately delimit the set of functions that it is meaningfully intended to delimit. We offer a revised definition of the WDRFs that accurately delimits the intended set, and show how this revised definition relates to matters of more general interest to theoretical phonologists; namely, priority orders between independently motivated phonological maps (i.e. FEEDING and BLEEDING) and SIMULTANEOUS APPLICATION (Kenstowicz and Kisseberth 1979, Joshi and Kiparsky 1979, 2006, Kiparsky 2015). Our empirical focus is on the bidirectional, dominant-recessive vowel harmony patterns of two Eastern Nilotic languages, Maasai (Tucker and Mpaayei 1955) and Turkana (Dimmendaal 1983), where the former can be described with a WDRF as it is intended to be defined and the latter cannot.

The article is structured as follows. First, in §2, we review both the intent and the limits of Heinz and Lai's (2013) original definition of the WDRFs. Then, in §3, we describe and analyze the vowel harmony patterns of Maasai and Turkana, with special attention to the differences that lead them to be on different sides of the WDRF boundary despite the fact that both technically qualify as weakly deterministic according to Heinz and Lai's (2013) definition. In §4, we introduce a revised definition of weak determinism, based on interaction, that correctly distinguishes the pattern in Maasai (and other WDRFs) from the pattern in Turkana. We end the article with a discussion of the computational and linguistic naturalness of our revised definition of weak determinism in §5 and with a brief conclusion in §6.

2 Formal background

In general, a regular function can be decomposed into two contradirectional subsequential functions (Elgot and Mezei 1965), with one (the 'outer' function) applying to the result of the other (the 'inner' function). We refer to all such decompositions of regular functions as ELGOT-MEZEI (EM) DECOMPOSITIONS.

Heinz and Lai (2013) define the subregular WDRFs in terms of a restriction on EM decompositions. In their initial, informal definition, they state that a WDRF "can be decomposed into [an inner] subsequential and [an outer, contradirectional] subsequential function *without the [inner] function marking up its output in any special way*" (Heinz and Lai 2013, p. 52, emphasis added). To keep these distinct from EM decompositions,

we refer to these as HEINZ-LAI (HL) DECOMPOSITIONS.

There are four main, interrelated things to note about EM decompositions and HL decompositions. The first is that the two functions of the decomposition are SUBSEQUENTIAL. A subsequential function reads its input string incrementally, and is able to remember some (finite) amount of what it has already read in the string unboundedly far in the past[1] but — at least in any linguistically relevant function — only to look boundedly far ahead in the string to aid in deciding what to write to its output at any given point.

The second thing to note is that the two functions of the decomposition are CONTRADIRECTIONAL: one is left-subsequential, reading its input string incrementally starting from the left, and the other is right-subsequential, reading its input string incrementally starting from the right.[2] Each of the two contradirectional subsequential functions thus has different kinds of information access to the input string: the left-subsequential function has access to information unboundedly far to the left and bounded 'look-ahead' access to the right, while the right-subsequential function has access to information unboundedly far to the right and bounded look-ahead access to the left. This difference in string access is schematically represented in Figure 2.

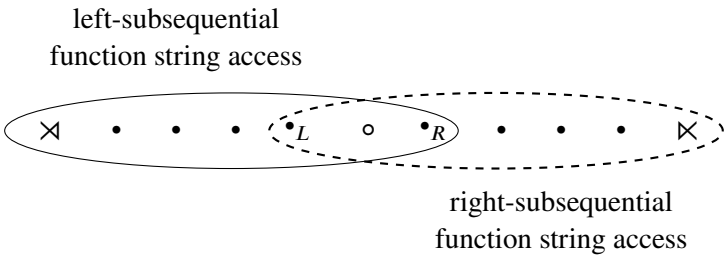

Figure 2 String access at an arbitrary point '∘' by two contradirectional subsequential functions.

In this figure, the end points of the string are represented by a boundary on the left (labeled '⋊') and a boundary on the right (labeled '⋉'), and the amount of bounded look-ahead access that each function has when it is applied to the string is represented by the distance between adjacent points in the string. When reading the point in the string labeled '∘', the left-subsequential function has access from the left string boundary to the adjacent '$\bullet_R$' to the right of '∘', and to none of the points further to the right. When reading the same point '∘', the right-subsequential function has access from the right string boundary to the adjacent '$\bullet_L$' to the left of '∘', and to none of the points further to the left. Each function otherwise has unbounded access from its initial starting point through to the bounded look-ahead access point.

The third thing to note is that the two functions of the decomposition are (assumed to be) ORDERED with respect to one another: one is the 'inner' function I, applying first, and the other is the 'outer' function O, applying to the result of I. In other words, I and O are COMPOSED, conventionally represented as $O \circ I$.

The fourth and most important thing to note is the additional restriction of HL decompositions: that the inner function I does not "[mark] up its output in any special way". The intention behind this restriction is to prevent I from providing information to the outer function O about what has previously been read on I's proximal, unbounded side, which is O's distal, bounded side. If I is able provide such information to O, then O would essentially have unbounded access to information on *both* sides of the string, effectively rendering O more expressive than subsequential — and the composition $O \circ I$ more expressive than weakly deterministic.

[1]More precisely, it must incrementally classify *every* prefix of every input string it applies to as belonging to one of a finite number of equivalence classes, and incrementally update this classification based on each new symbol and the equivalence class for the prefix so far — i.e. what the function 'remembers' about the observed prefix so far.

[2]Indeed, Heinz and Lai (2013, p. 52) use "left" and "right" to characterize the two subsequential functions that comprise an HL decomposition. We use "inner" and "outer, contradirectional" because we find it signals more clearly that the first function to apply may be either left- or right-subsequential so long as the second function to apply is contradirectional.

In the discussion leading up to their formal definition of the WDRFs, Heinz and Lai (2013, pp. 54–55) explicitly identify just two forms of inner function mark-up to be precluded. One is the addition of *novel symbols* to the output of the inner function I, symbols that aren't in the original alphabet of I's input. If I is able to write novel symbols, then those symbols can be strategically deployed in such a way that crucial information about I's unbounded side is communicated to the outer function O. A function that does not add novel symbols in this way is ALPHABET-PRESERVING; we will also refer to a function that *does* add novel symbols as ALPHABET-INCREASING. The other form of explicitly precluded mark-up in Heinz and Lai (2013) is based on an "important observation" by an anonymous reviewer: "*coding* these new symbols as strings formed over the original alphabet" (Heinz and Lai 2013, p. 55, emphasis in the original). In other words, *novel **sequences** of symbols* can also be strategically deployed to communicate crucial information about I's unbounded side to O. A function that effectively encodes new symbols in this way is LENGTH-INCREASING; a function that *does not* encode new symbols in this way is LENGTH-PRESERVING.

It will be useful to examine a couple of examples at this point. The first is a schematic example of bidirectional vowel harmony in §2.1, which can be decomposed into two contradirectional subsequential functions where the inner function is alphabet- and length-preserving — that is, it can be modeled with a WDRF in accordance with Heinz and Lai's (2013) formal definition. The second is a schematic example of unbounded tonal plateauing in §2.2, which cannot be modeled with a WDRF as originally shown by Jardine (2016). Our schematic example involves the introduction of new symbols by the inner function, but the same point could be made by allowing the inner function to use length-increasing codes.

2.1 Bidirectional vowel harmony

In the simplest cases of bidirectional vowel harmony, target vowels in the harmony domain change to agree with the harmonic feature value of a trigger vowel (the nearest root vowel, a vowel with the so-called 'dominant' harmonic feature value, etc.). Assuming the trigger vowel's harmonic feature value is '+' and that targets are '−', then $/ \ldots -^n + -^m \ldots / \mapsto [\ldots +^n + +^m \ldots]$. This transformation can be decomposed into two contradirectional subsequential functions, operating as illustrated in Figure 3.

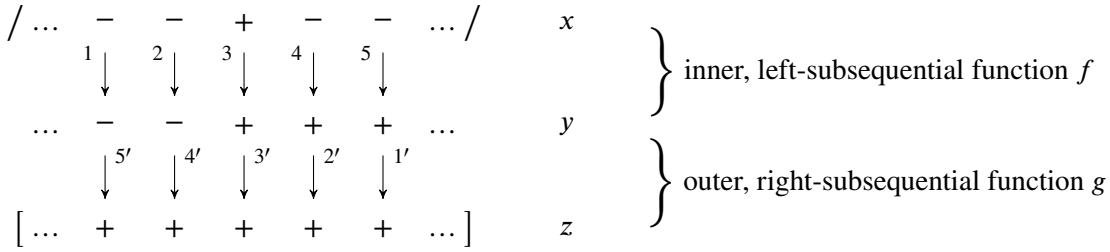

Figure 3 A schematic example of bidirectional vowel harmony.

Figure 3 illustrates the type of function composition necessary to generate this bidirectional vowel harmony mapping. In this example, the inner function f is (arbitrarily) left-subsequential, reading its input string x from left to right and writing the intermediate output string y. Target vowels are initially written faithfully, as shown by the arrows labeled 1 and 2, until a trigger vowel is read (arrow 3). This information results in all subsequent vowels being changed to agree with the trigger vowel (arrows 4–5). The intermediate output string y generated by the inner function f is then fed as input to the outer, right-subsequential function g, which reads this string from right to left and writes the ultimate output string z. Both changed target vowels (arrows $1'$–$2'$) and trigger vowels (arrow $3'$) are written faithfully; the remaining, unchanged target vowels to the left of the trigger (arrows $4'$–$5'$) are then also changed to agree with the trigger.

Notice that every change made by functions f and g is visible in the final output regardless of whether f or g applies first, or indeed whether they apply simultaneously to the input (Kenstowicz and Kisseberth

1979, Joshi and Kiparsky 1979, 2006, Kiparsky 2015).[3] This will prove to be a useful heuristic for diagnosing WDRFs. The left-subsequential function f can effectively be defined as an iterative left-to-right spreading rule, changing − to + *after* +, and the right-subsequential function g can be defined as an iterative right-to-left spreading rule, changing − to + *before* +. Crucially, neither function makes a change that has any effect on the operation of the other; paraphrasing Heinz and Lai (2013) slightly, neither function marks up its output in any special way. This shows that there exists a decomposition of the overall mapping $/ \ldots -^n + \ -^m \ldots / \mapsto [\ldots +^n + \ +^m \ldots]$ into an ordered pair of contradirectional subsequential functions without mark-up. The mapping in Figure 3 is thus an instance of a properly HL-decomposed WDRF.

2.2 Unbounded circumambience

One of Jardine's (2016) key results is that the minimum degree of computational expressivity required to describe what Jardine calls UNBOUNDED CIRCUMAMBIENT processes is *greater* than the expressivity afforded by the WDRFs. Such processes demand the expressivity of NON-DETERMINISTIC REGULAR FUNCTIONS (NDRFs); see now also McCollum et al. (2020a), Hao and Andersson (2019), and Koser and Jardine (2020). Jardine's (2016) definition of unbounded circumambient processes is given in Def. 1 (emphasis added).

Definition 1. An *unbounded circumambient process* is a process: (Jardine 2016, p. 249)

a. whose application is dependent on information (e.g. the presence of a trigger or blocker) *on both sides* of the target; and [= CIRCUMAMBIENT]

b. in which, on both sides, there is *no principled* a priori *bound on how far* this information may be from the target. [= UNBOUNDED]

A regular function describing an unbounded circumambient process can be EM-decomposed into two contradirectional subsequential functions, with each properly identifying critical information at an *a priori* unbounded distance on the side of the target matching the function's origin. Unlike a properly HL-decomposed WDRF, however, the inner function of this EM decomposition must somehow communicate to the outer function whether the critical information is present on the side of the target opposite the outer function's origin in order to correctly describe the unbounded circumambient process. This is why these patterns require greater computational expressivity than that afforded by the WDRFs.

Now we turn to a schematic example. In cases of *unbounded tonal plateauing* (UTP), all low (L) tones between two high (H) tones surface as H: $/ \ldots \text{H L}^n \text{H} \ldots / \mapsto [\ldots \text{H H}^n \text{H} \ldots]$.[4] This transformation can be decomposed into two contradirectional subsequential functions, operating as illustrated in Figure 4.

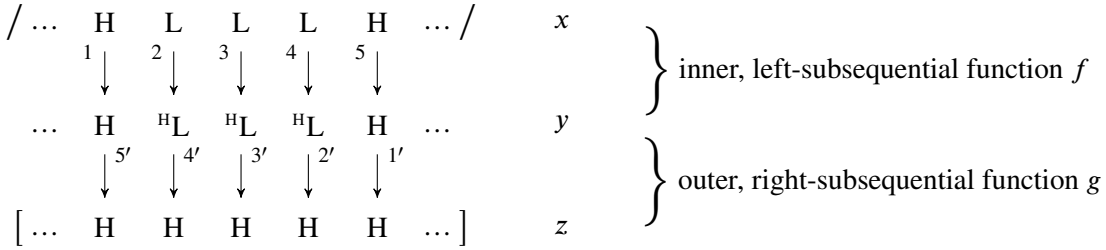

Figure 4 A schematic example of unbounded tonal plateauing.

[3] What we call here 'simultaneous application' is discussed under the banner of the 'direct mapping hypothesis' in Kenstowicz and Kisseberth (1979, Ch. 8); this hypothesis is there pitted against the 'ordered rule hypothesis', on which see §4.3 below. Note also that Chomsky and Halle (1968, p. 344) use the term 'simultaneous application' to refer to a distinct hypothesis about how a *single rule* applies to *multiple locations* within a form; see also Anderson (1974, Ch. 13) and Kenstowicz and Kisseberth (1977, Ch. 5).

[4] Jardine (2016, pp. 251ff) cites Luganda, Digo, Xhosa, Zulu, Yaka, Saramaccan, Papago (Tohono O'odham), and South Kyungsang Korean as examples of languages with some variant of UTP; see that work for discussion and references to sources. In some cases the underlying tonal distinction is analyzed as H vs. toneless rather than H vs. L. This distinction is irrelevant here; all that matters is that there is a two-way opposition, regardless of how it is represented (see also footnote 5).

The first, inner function f reads its input string x in one direction (here, from left to right), writing Ls faithfully to the intermediate output y until it reads and writes an H. When it does, at the arrow labeled 1, there is only enough information to determine that any subsequent sequence of Ls will *potentially* surface as Hs. They will do so only if there is another H further along in the string; otherwise, the sequence of Ls will surface faithfully as Ls. Thus, when f continues reading the subsequent string of Ls at the arrows labeled 2–4, it lacks any information that another H is forthcoming and thus cannot determine how these Ls will ultimately surface. The inner function f must therefore communicate to the outer function g information about the difference between a (potentially unbounded) string of Ls preceded by an H in the input string x, as in this case, as opposed to a string of Ls not preceded by an H in the input string x. The outer function g can then use this information to determine whether a string of Ls is *between* two Hs — one on its proximal, unbounded side, the other on its distal, bounded side — as though g had direct, unbounded access to both sides of the original input string and were thus more expressive than subsequential.

Significantly, this information cannot be encoded simply using symbols from the alphabet used for the input string x (at least, not without increasing the length of the string). The symbol 'H' cannot be used, because the outer function g still needs to distinguish an H in y that was also H in x — which will faithfully surface as H in z regardless — from H in y derived from L in x, which may or may not surface as H in z. That leaves the symbol 'L', and this symbol on its own cannot encode the necessary difference between Ls preceded as opposed to not preceded by H in x. The information must thus be encoded in y with a novel symbol,[5] here written $^{\text{H}}$L. This encoding allows g to read its input string y in the other direction (here, from right to left) and confidently determine whether the unbounded circumambient conditions of UTP are met. If $^{\text{H}}$L were the first symbol encountered by g in its right-to-left sweep, then the conditions of UTP would obviously not be met and only H would surface as H, while L and $^{\text{H}}$L would surface as L. Otherwise, as in the example under discussion in Figure 4, the outer function g reads and writes Ls faithfully until it reads and writes an H, at the arrow labeled $1'$. At this point, g can fully determine the output of subsequent symbols from y to z: H and $^{\text{H}}$L both surface as H, as illustrated by the arrows labeled $2'$–$5'$, and L surfaces as L.

Despite the essential symmetry of the unbounded circumambient conditions of UTP, f and g *cannot* apply simultaneously, nor can they apply in either order and maintain their respective definitions. Whichever function serves as the inner function must be the one to specially encode an L read after an H as $^{\text{H}}$L, and the other, outer function must be the one to read $^{\text{H}}$L and determine whether it should be output as H or as L. In other words — and in contrast to the schematic bidirectional vowel harmony example illustrated in Figure 3 — the inner function must make a change that discernably affects the operation of the outer function. Paraphrasing Heinz and Lai (2013) again, the inner function marks up its output in a special way. The mapping in Figure 4 is thus *not* an HL decomposition of a WDRF; instead, it is an EM decomposition of an NDRF.

2.3 Matching the letter to the spirit

Recall Heinz and Lai's (2013) informal definition of WDRFs quoted at the outset of this section, specifying that the inner function of an HL-decomposed pair of contradirectional subsequential functions cannot mark up its output "in any special way". In the end, though, Heinz and Lai's formal definition, given in Def. 2, precludes only novel symbols and length-increasing codes as mark-up in the relevant sense.[6]

[5] Or equivalently, a length-increasing code for such a new symbol in terms of H and L. Note that one could somehow use tonelessness as a way to encode the relevant information, but this would likewise be a symbol not used in the alphabet of the composition's input string because there is only a two-way tonal opposition in the cases of interest; see Stanley (1967) on the behavior of a 'blank' as a third feature value. The same of course applies to the symbol L if the basic opposition is analyzed as H vs. toneless; see fn. 4.

[6] Note that *LSF* refers to the class of left-subsequential functions, *RSF* refers to the class of right-subsequential functions, and that the left-subsequential function is assumed to be the inner one; however, as already noted and demonstrated, what matters is that the two functions be contradirectional and that the inner one, whichever that is, be both alphabet- and length-preserving. Note also that Def. 2 appears to require that *both* functions be alphabet-preserving, hence $X^* \to X^*$ in both cases; strictly speaking, however, this condition only applies relevantly to the inner function, just like length-preservation. See §4 for an appropriate restatement of Def. 2.

Definition 2. A regular function T is *weakly deterministic* iff there exists $L: X^* \to X^* \in LSF$, and $R: X^* \to X^* \in RSF$ such that L is not length-increasing and $T = R \circ L$. (Heinz and Lai 2013, p. 55)

Heinz and Lai (2013, p. 55) make clear that the second, length-increasing form of mark-up needs to be precluded by this definition "because the only way to unambiguously encode a larger alphabet into a smaller one is with length-increasing functions" — in other words, the alphabet-preserving condition does not preclude all ways for an inner function to make a change that affects the operation of an outer function. Subsequent work (McCollum et al. 2018, O'Hara and Smith 2019, Smith and O'Hara 2019, Lamont et al. 2019) has shown that there are yet more ways for an outer function to receive information from an inner function about the distal end of the string. Some of this work has defended analyses with these other forms of mark-up, safely ensconced in the unintended blindspots of Heinz and Lai's (2013) formal definition in Def. 2, as being WDRF-compliant, thus violating the spirit though not the letter of the definition.

In §3 below, we describe two related bidirectional vowel harmony patterns: one in Maasai (Tucker and Mpaayei 1955) and the other in Turkana (Dimmendaal 1983), with a focus on the differences between them highlighted in Baković (2000, 2002). We argue that the pattern in Maasai can be modeled with a properly characterized WDRF while the pattern in Turkana cannot be, instead requiring the expressivity of an NDRF. The contrast between these two patterns illustrates the need for a revised, strengthened definition of the WDRFs that brings the letter of Heinz and Lai's (2013) formal definition in accord with its informally-stated spirit. The key to this revised definition is that the difference between a WDRF and an NDRF boils down to a precise notion of interaction that we will call MUTATION-INTERACTION or μ-interaction. Informally, a generalization requiring the expressivity of a WDRF can be modeled with two contradirectional subsequential functions that do not crucially μ-interact, while a generalization requiring the expressivity of an NDRF can only be decomposed into two contradirectional subsequential functions that *must* μ-interact. This is the spirit of Heinz and Lai's (2013) definition of the WDRFs: if the two functions don't μ-interact, then there are no mechanisms for the inner function to communicate special information to the outer function.

For two functions f, g to μ-interact in the relevant sense, they must (in any linguistically plausible context) be *non-vacuously ordered* with respect to each other. If there exists an input string x such that applying f and g in one order produces an output string distinct from the output string produced by applying f and g in the opposite order, then f and g are non-vacuously ordered with respect to each other.

Definition 3. Given $f, g: X \to X$, the *order* of f and g is *non-vacuous* iff there exists an input x such that $g \circ f(x) \neq f \circ g(x)$; that is, f and g do not commute on x.

Non-vacuous ordering can be illustrated with 'feeding' and 'bleeding' orders of phonological rules. Consider the case of feeding in (1), based on a Finnish example (Kiparsky 1973, 1993).[7] Raising f in (1a) changes word-final mid front vowels to high, and assibilation g in (1b) changes voiceless coronal stops to fricatives before high front vowels. The feeding order $g \circ f$ is apparent in (1c): only f is applicable to /vete/, resulting in |veti|; g is now newly applicable, yielding [vesi] 'water'. The reverse order $f \circ g$ is shown to the right to yield an incorrect result where g fails to apply; this same incorrect result is obtained if f and g were to apply simultaneously (Kenstowicz and Kisseberth 1979, Joshi and Kiparsky 1979, 2006, Kiparsky 2015).

(1) Raising feeds assibilation in Finnish

a. Raising (f): e ⟶ i / __ # — /kiele/ $\xrightarrow[f]{}$ [kieli] 'tongue'

b. Assibilation (g): t ⟶ s / __ i — /tilati/ $\xrightarrow[g]{}$ [tilasi] 'ordered'

c. $g \circ f$: /vete/ $\xrightarrow[f]{}$ |veti| $\xrightarrow[g]{}$ [vesi] — ($f \circ g$: /vete/ $\xrightarrow[g]{}$ |vete| $\xrightarrow[f]{}$ *[veti])

[7] The discussion of this example and the next is from Baković and Blumenfeld (2019).

Consider now the case of bleeding in (2), based on a Lamba example (Doke 1938, Kenstowicz and Kisseberth 1979). Both lowering f in (2a) and palatalization g in (2b) are potentially applicable to /kosika/ in (2c), but once f applies, yielding |koseka| 'it is strong', g is no longer applicable. The reverse order $f \circ g$ again yields an incorrect result where g applies, also obtained if f and g were to apply simultaneously.

(2) Lowering bleeds palatalization in Lamba

a. Lowering (f): $\mathrm{V} \longrightarrow [-\text{high}] \,/\, \begin{bmatrix}-\text{high}\\-\text{low}\end{bmatrix} \mathrm{C}_0$ — /ponika/ $\xrightarrow[f]{}$ [poneka] 'it falls'

b. Palatalization (g): $\mathrm{s} \longrightarrow \mathrm{ʃ} \,/\, \_\mathrm{i}$ /fisika/ $\xrightarrow[g]{}$ [fiʃika] 'it hides'

c. $g \circ f$: /kosika/ $\xrightarrow[f]{}$ |koseka| $\xrightarrow[g]{}$ [koseka] ($f \circ g$: /kosika/ $\xrightarrow[g]{}$ |koʃika| $\xrightarrow[f]{}$ *[koʃeka])

In both (1) and (2), the application of f *affects* the application of g: in the case of feeding (1) by creating symbols that a subsequent rule can non-vacuously apply to and in the case of bleeding (2) by altering symbols that a subsequent rule would otherwise non-vacuously apply to.[8] Affection can be unrequited, as it is in the cases illustrated in (1) and (2): f affects g, but g does not affect f, which is why the order $f \circ g$ is indistinguishable from simultaneous application. Note that f applies under all six scenarios above: $g \circ f$, $f \circ g$, and f, g applying simultaneously. This is precisely because g does not affect f. But because f affects g, whether g gets to apply depends on its order with respect to f. Given two functions, then, if either affects the other, then their order is non-vacuous. Furthermore, an unaffected function always applies regardless of the order between the functions — or the lack thereof, viz. simultaneous application. Now equipped with these notions, we can delve into the details of the vowel harmony patterns of Maasai and Turkana in §3. Both patterns can be modeled with a pair of contradirectional subsequential functions; in Maasai, these functions do not affect each other and thus do not interact, while in Turkana one function affects the other and thus they do interact.

3 Maasai and Turkana

Maasai (Tucker and Mpaayei 1955, Hall et al. 1973, Baković 2000, 2002, McCrary 2001, Guion et al. 2004, Quinn-Wriedt 2013) and Turkana (Dimmendaal 1983, Noske 1996, 2001, Baković 2000, 2002) are closely related Eastern Nilotic languages. Each possesses an inventory of nine contrastive vowels /i ɪ e ɛ ɑ ɔ o ʊ u/, and has ATR harmony, with [+ATR] spreading bidirectionally from the triggering vowel — often a root vowel, but sometimes a suffix vowel). There exist four harmonic [+ATR]~[−ATR] pairings among the non-low vowels, i~ɪ, e~ɛ, o~ɔ, and u~ʊ. Significantly, the low vowel /ɑ/ does not have a direct [+ATR] counterpart. In certain [+ATR] contexts, underlying /ɑ/ is 're-paired' as [o] (Baković 2000).

This is where the relevant similarities between the Maasai and Turkana vowel harmony patterns end. The key difference between them is the existence and behavior of a set of [−ATR] dominant suffix vowels in the Turkana pattern that are absent from the Maasai pattern. As the remainder of this section shows, the existence and behavior of these [−ATR] dominant suffix vowels makes the function responsible for the Turkana pattern an NDRF, and their absence makes the function responsible for the Maasai pattern a WDRF.

3.1 Maasai

Examples of [+ATR] spreading from roots in Maasai are shown in (3) below; note that we underline roots throughout. (Because tone does not play a role in the harmony pattern, we omit the transcription of tone on vowels for visual clarity of the contrast between /i/ and /ɪ/.) The contrast between (3a,b) shows that the [+ATR] value of the root spreads leftward, controlling the realization of the preceding prefixes. Both prefixes and suffixes are present in (3c,d), and the root spreads [+ATR] in both directions.

[8] See Baković and Blumenfeld (2019, 2020) for more formal detail about these and other ordering relations.

(3) Root-controlled ATR harmony in Maasai

a.	/mɪ-kɪ-rɑɲ/	[mɪ-kɪ-rɑɲ]	‘NEG-1PL-sing’
b.	/mɪ-kɪ-itoki/	[mi-k-intoki]	‘NEG-1PL-do.again’
c.	/kɪ-ɪdɪm-ʊ/	[k-ɪdɪm-ʊ]	‘1PL-be.able-PRES’
d.	/kɪ-ɲorr-ʊ/	[ki-ɲorr-u]	‘1PL-love-PRES’

In addition to roots, some suffixes also trigger [+ATR] spreading. These triggering suffixes are often called ‘dominant’. Examples of suffix-triggered harmony are seen in (4). Observe that both the root meaning ‘wash’ and the intransitive suffix surface as [−ATR] in (4a), but in the presence of the dominant applied suffix, all root and prefix vowels surface as [+ATR] in (4b). In the same manner, the instrumental suffix transforms the [−ATR] root and prefixes in (4c) to [+ATR] in (4d).

(4) Suffix-controlled ATR harmony in Maasai

a.	/ɪsʊj-ɪʃɔ/	[ɪsʊj-ɪʃɔ]	‘wash-INTRANS’
b.	/ɪsʊj-ɪʃɔ-re/	[isuj-iʃo-re]	‘wash-INTRANS-APPLIED’
c.	/ɛ-tɛ-bɛl-ɑ/	[ɛ-tɛ-bɛl-ɑ]	‘3SG-PF-break-PF’
d.	/ɛ-tɛ-bɛl-ie/	[e-te-bel-ie]	‘3SG-PF-break-INST’

The behavior of the low vowel is the final point needing attention. When /ɑ/ occurs to the left of a [+ATR] vowel, it blocks harmony, surfacing faithfully as [ɑ]. This is evident in (5a–d). Observe that the first-person plural prefix /kɪ/ undergoes harmony from the following root /dot/ in (5a), but when the past tense prefix /tɑ/ occurs between the two, it both fails to assimilate to [+ATR] and prevents the more peripheral prefix from harmonizing with the root. In (5c,d), we see the same behavior from root /ɑ/, which blocks the leftward spreading that originates from the dominant applied suffix in (5d).

However, /ɑ/ alternates for harmony when it occurs to the *right* of a [+ATR] trigger. When following a [−ATR] root, /ɑ/ is output faithfully, as in (5e,f). When the low vowel follows a [+ATR] root, though, it undergoes harmony, surfacing re-paired as [o] as in (5g,h). Similarly, when suffixal /ɑ/ is flanked on either side by [+ATR] vowels, as in (5i,j), it is output as [o] due to the influence of the [+ATR] vowel to its left.

(5) The behavior of /ɑ/ in Maasai

a.	/kɪ-dot-ʊn-ie/	[ki-dot-uɲ-ie]	‘1PL-pull-MT-APPLIED’
b.	/kɪ-tɑ-dot-ʊn-ie/	[kɪ-tɑ-dot-uɲ-ie]	‘1PL-PAST-pull-MT-APPLIED’
c.	/ɪ-duŋ-ɪʃɔ-re/	[i-duŋ-iʃo-re]	‘2SG-cut-INTRANS-APPLIED’
d.	/ɪ-ɑs-ɪʃɔ-re/	[ɪ-ɑs-iʃo-re]	‘2SG-do-INTRANS-APPLIED’
e.	/ɪn-lɪpɔŋ-ɑ/	[ɪn-lɪpɔŋ-ɑ]	‘FEM.PL-female-PROD’
f.	/ɔl-mɛn-ɑ/	[ɔl-mɛn-ɑ]	‘MASC.SG-despise-PROD’
g.	/ɪn-mudoŋ-ɑ/	[in-mudoŋ-o]	‘FEM.PL-relative-PROD’
h.	/ɛŋ-komon-ɑ/	[eŋ-komon-o]	‘FEM.SG-pray-PROD’
i.	/ɑ-duŋ-ɑkɪn-ie/	[ɑ-duŋ-okiɲ-ie]	‘3SG-cut-DATIVE-APPLIED’
j.	/ɛ-isud-ɑ-rɪ-ie/	[e-isud-o-ri-jie]	‘3SG-hide-MA-N-APPLIED’

The harmony pattern in Maasai can be decomposed into two very similar, contradirectional rules: a rightward rule spreading [+ATR] to all following vowels and re-pairing /ɑ/ to [o], and a leftward [+ATR] spreading rule that is blocked by /ɑ/. These rules need not be ordered with respect to one another because they affect disjoint sets of elements in the word. Consider an underlying vowel string /ɛ ɑ ʊ i ɛ ɑ ɪ/, with /i/ the [+ATR] trigger. Rightward spreading affects only the final /ɛ ɑ ɪ/ sequence, outputting it as [e o i]; it does not affect the realization of the initial sequence of vowels. Similarly, leftward spreading outputs the initial /ɛ ɑ ʊ/ as [ɛ ɑ u], but has nothing to say regarding the realization of the final sequence of vowels.

When multiple triggers are present, the situation is only superficially different. When a non-low vowel is flanked by two triggers, the medial vowel is output as [+ATR]. This is consistent with the effect of *either* of the two rules. In this context we cannot determine which trigger caused the assimilation of the non-low vowel. When the low vowel /ɑ/ is flanked by two triggers, it is output as [o], but in this context we can uniquely determine the trigger responsible for the alternation: the trigger to the left. Since /ɑ/ blocks leftward spreading, in a sequence like /i ɑ i/, the /i/ to the right will not trigger assimilation (and re-pairing) of the /ɑ/. The /i/ to the left, though, can spread [+ATR] to /ɑ/, and so the string is ultimately output as [i o i].

Stated differently, the surface quality of each vowel in Maasai depends on two sources of information, its underlying ATR specification, and the surface ATR specification of either the preceding or following vowel.[9] Vowels to the left of a trigger depend on the surface ATR value of the following vowel while vowels to the right of a trigger depend on the surface ATR value of the preceding vowel. Crucially, no individual vowel's surface quality depends on ATR specifications in both directions. This is of course what makes the Maasai harmony pattern fit within Heinz and Lai's (2013) class of WDRFs: it can be decomposed into two contradirectional, subsequential functions such that, if ordered, the inner function does not mark up its intermediate output in any special way — and so, the two functions can also apply simultaneously.

This finite-state analysis is demonstrated in two ways. We first present a finite-state transducer (FST) for each directional harmony rule. We then demonstrate the full input-output mapping with tape representations. The FSTs in Figs. 5 and 6 below model the individual transductions computed by each function, and are oriented to correspond to the direction of feature spreading.[10] The ordering of these two non-interacting functions is irrelevant, and we arbitrarily choose the left-to-right function as the first (inner) function to start. Beginning with the initial state q_0 in Fig. 5, the FST reads the left word edge symbol, ⋊, and transitions to state q_1. If the FST then encounters a [−ATR] vowel, it outputs that vowel faithfully and remains in q_1, as shown by the loop above q_1. This looping occurs until either a [+ATR] vowel or the right word edge symbol is read. If the FST encounters an input [+ATR] vowel, it transitions to q_2, and outputs all following vowels as [+ATR], including re-pairing /ɑ/ to |o|, shown by the loop above q_2. If the FST reads the right word edge symbol, ⋉, it transitions directly to q_3, the accepting state, and the first pass is complete.

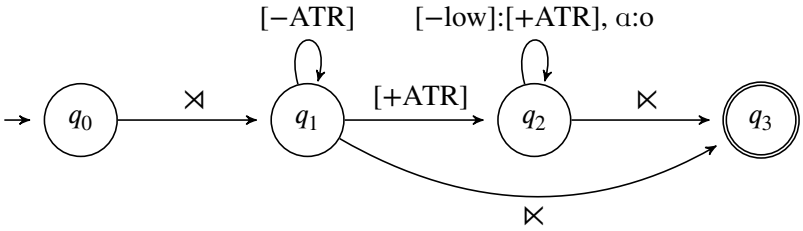

Figure 5 Left-subsequential (left-to-right) FST for Maasai.

The second (outer) function in Fig. 6 reads the output from the first function. Upon reading the right word edge symbol, the FST transitions to q_1, looping there until a [+ATR] vowel or the left word edge is encountered. If no [+ATR] vowel is present, the FST transitions directly from q_1 to q_3 upon reading the left word edge symbol and the transduction is complete. If instead at least one [+ATR] vowel is present in the output of the inner function, then upon reading it the FST transitions to q_2. In this state, all subsequent non-low vowels are output as [+ATR]. The low vowel blocks harmony, meaning that if intermediate |ɑ| is read, the FST outputs it as [ɑ] and moves back to q_1. If no low vowels are encountered, non-low vowels continue to be output as [+ATR] until the left word edge symbol is reached and the transduction is complete.

[9]The surface specification of the adjacent vowels is what matters, not the underlying specification, due to the iterativity of spreading. In other words, each of the two harmony rules is Output Strictly Local (Chandlee et al. 2015).

[10]The use of features in these is simply a shorthand way to refer to the individual segments in each class. The transducers below operate over segmental symbols, and not features, but this is not crucial.

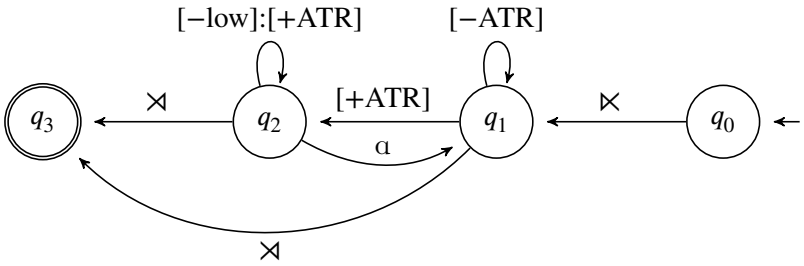

Figure 6 Right-subsequential (right-to-left) FST for Maasai

The tape representations in Figs. 7 and 8 demonstrate the operation of these FSTs with the informative form in (5i). The changes introduced by the inner, left-to-right function are shown in Fig. 7. The intermediate output produced by the inner function serves as the input to the outer, right-to-left function, and the changes introduced by this outer function, resulting in the final output, are shown in Fig. 8.[11]

direction	→							
Fig. 5 state	$q_0 \rightarrow q_1$	q_1	$q_1 \rightarrow q_2$	q_2	q_2	q_2	q_2	$q_2 \rightarrow q_3$
initial input	⋊	ɑ	d u	ŋ ɑ	k ɪ	n i	e	⋉
intermediate	⋊	ɑ	d u	ŋ o	k i	n i	e	⋉

Figure 7 Tape representation of the left-subsequential FST in Fig. 5 operating as the inner function

intermediate	⋊	ɑ	d u	ŋ o	k i	n i	e	⋉
final output	⋊	ɑ	d u	ŋ o	k i	ɲ i	e	⋉
Fig. 6 state	$q_3 \leftarrow q_1$	$q_1 \leftarrow q_2$	q_2	q_2	q_2	q_2	$q_2 \leftarrow q_1$	$q_1 \leftarrow q_0$
direction	←							

Figure 8 Tape representation of the right-subsequential FST in Fig. 6 operating as the outer function

We claimed above that the relative ordering of these two functions is irrelevant. To demonstrate this, consider the opposite ordering, which is exemplified below in Figs. 9 and 10. The transductions exemplified in the tape representations below are identical to those above, but differ only in their ordering. The first function spreads [+ATR] from right to left, as depicted in Fig. 9. After encountering the dominant [+ATR] applied suffix, it transforms /ɪ/ to [i], but halts at the suffixal /ɑ/, since leftward harmony is blocked by the low vowel. Upon encountering the root /u/, the FST is in a position to spread [+ATR] once again, but spreading to the prefixal /ɑ/ fails for the same reason — leftward harmony is blocked by /ɑ/.

direction	←							
Fig. 6 state	$q_3 \leftarrow q_1$	$q_2 \leftarrow q_1$	$q_2 \leftarrow q_1$	$q_1 \leftarrow q_2$	q_2	q_2	$q_2 \leftarrow q_1$	$q_1 \leftarrow q_0$
initial input	⋊	ɑ	d u	ŋ ɑ	k ɪ	n i	e	⋉
intermediate	⋊	ɑ	d u	ŋ ɑ	k i	n i	e	⋉

Figure 9 Tape representation of the right-subsequential FST in Fig. 6 operating as the inner function

[11] Because our transducers ignore consonants, the states and transitions between them shown in these tape representations lump every consonant together with the following vowel. We nevertheless indicate the independent palatalization of /n/ to [ɲ] in the form derived in Figs. 7 and 8 (and again in Figs. 9 and 10).

Under this ordering, the second FST accomplishes a single change, mapping suffixal |ɑ| to [o]. Since rightward harmony is not blocked, assimilation (and re-pairing) occurs, and the correct output is achieved.

intermediate	⋊	ɑ	d	u	ŋ	ɑ	k	i	n	i	e	⋉
final output	⋊	ɑ	d	u	ŋ	o	k	i	ɲ	i	e	⋉
Fig. 5 state	$q_0 \rightarrow q_1$	q_1	$q_1 \rightarrow q_2$		q_2		q_2		q_2		q_2	$q_2 \rightarrow q_3$
direction	→											

Figure 10 Tape representation of the left-subsequential FST in Fig. 5 operating as the outer function

The Maasai functions in Figs. 5 and 6 are thus vacuously ordered, given that neither function affects the other. As a consequence, the functions can be ordered as shown in Figs. 7 and 8 or as shown in Figs. 9 and 10, or indeed they can apply simultaneously. We conclude that bidirectional ATR harmony in Maasai is a WDRF because it can be HL-decomposed as defined by Heinz and Lai (2013). Critically, the Maasai pattern is not unbounded circumambient because the surface realization of each individual vowel is dependent on its input specification and *either* information from (boundedly far to) the left *or* the right, and never from both directions.

3.2 Turkana

As noted at the outset of this section, the harmony pattern in Turkana is in relevant respects identical to the pattern in Maasai except for the existence and behavior of a set of [−ATR] dominant suffix vowels. Let's first establish the commonalities with the Maasai pattern. The contrast in (6a shows that the [+ATR] value of the root spreads leftward, controlling the realization of the preceding prefix. The contrast in (6c shows that the [+ATR] value of the root spreads bidirectionally to both prefixes and suffixes.

(6) Root-controlled ATR harmony in Turkana

a.	/ɛ-kɔrɪ/	[ɛ-kɔrɪ]	'MASC.SG-ratel'
b.	/ɛ-kori/	[e-kori]	'MASC.SG-giraffe'
c.	/ɛ-ɛm-ɪ/	[ɛ-ɛm-ɪ]	'3SG-fear-ASP'
d.	/ɛ-los-ɪ/	[e-los-i]	'3SG-go-ASP'

Dominant suffixes also trigger [+ATR] spreading. The voice suffix /o/ and the gerundial suffix /e/ in (7b cause all other vowels in the word to assimilate to [+ATR].

(7) Suffix-controlled ATR harmony in Turkana

a.	/ɛ-ɑ-kɔkɔ-ʊn-ɪ/	[ɛ-ɑ-kɔkɔ-ʊn-ɪ]	'3-TNS-steal-VEN-ASP'
b.	/ɛ-kɔkɔ-ʊn-ɪ-o/	[ɛ-koko-un-i-o]	'3-steal-VEN-ASP-VOI'
c.	/ɑ-kɪ-dɔk/	[ɑ-kɪ-dɔk]	'INF-K-climb'
d.	/ɛ-dɔk-ʊn-e/	[e-dok-un-e]	'MASC.SG-climb-VEN-GER'

The final parallel between Maasai and Turkana, the behavior of the low vowel in [+ATR] contexts, is illustrated in (8). Observe in (8a that the prefixal /ɑ/ surfaces faithfully regardless of the ATR value of the root. In (8c), the root /rɑm/ is preceded by [−ATR] prefixes. When the low vowel is root-internal and is followed by a dominant suffix, as in (8d), it also fails to undergo harmony. But when a low vowel follows a [+ATR] root (8e-h), or when it is flanked on both sides by [+ATR] vowels (8i), harmony (and re-pairing) obtains, and /ɑ/ is output as [o]. Given that leftward harmony is blocked by the low vowel, we must assume that rightward spreading from the root is what causes /ɑ/ to surface as [o] in (8i).

(8) The behavior of /ɑ/ in [+ATR] contexts in Turkana

a.	/ɑ-dɔk-ʊn/	[ɑ-dɔk-ʊn]	‘INF-climb-VEN’
b.	/ɑ-lim-ʊn/	[ɑ-lim-un]	‘3-steal-VEN-ASP-VOI’
c.	/ɑ-kɪ-ɾɑm/	[ɑ-kɪ-ɾɑm]	‘INF-K-beat’
d.	/ɛ-ɾɑm-e/	[ɛ-ɾɑm-e]	‘MASC.SG-beat-GER’
e.	/ɛ-tɪm-ɑt/	[ɛ-tɪm-ɑt]	‘FEM.SG-shoe-SG’
f.	/ɛ-pur-ɑt/	[e-pur-ot]	‘MASC.SG-beer-SG’
g.	/ɛ-pɛg-ɑɑn-n-ɑ/	[ɛ-pɛg-ɑɑn-n-ɑ]	‘3SG-argue-HAB-SG-VOI’
h.	/ɛ-pup-ɑɑn-n-ɑ/	[ɛ-pup-oon-n-o]	‘3SG-obey-HAB-SG-VOI’
i.	/ɑ-tur-ɑɑn-u/	[ɑ-tur-oon-u]	‘GER-agile-HAB-NOM’

Up to this point, the harmony patterns in Maasai and Turkana appear to be identical. However, there is one crucial difference between the two: there are dominant [−ATR] suffixes in Turkana in addition to dominant [+ATR] suffixes. (These suffixes are circled in the data below.) When a dominant [−ATR] suffix co-occurs with a [+ATR] root, the dominant [−ATR] vowel both resists assimilation and imposes its own ATR value on all preceding vowels (9), which Baković (2000) calls ‘dominance reversal’.

Observe in (9a) that the root /ido/ triggers the assimilation of the ventive and aspect-marking suffixes /ʊn/ and /ɪt/ to [un] and [it], respectively. In the presence of the dominant [−ATR] instrumental-locative suffix (9b), the ventive suffix and the root both surface as [−ATR].[12] Two things are worth noting here. First, these suffixes are idiosyncratically dominant, and there is no obvious phonological information that determines whether a given suffix will alternate or not; compare the final, non-dominant suffix in (8g with the dominant one in (9e). Second, note in (9e) that the dominant [−ATR] suffix may be separated from the affected low vowel by another syllable; we assume that this distance is in principle unbounded.

(9) Dominance reversal in Turkana

a.	/ido-ʊn-ɪt/	[ido-un-it]	‘give.birth-VEN-ASP’
b.	/ɑ-k-ido-ʊn-(ɛt)/	[ɑ-k-ɪdɔ-ʊn-(ɛt)]	‘GER-K-give.birth-VEN-INST.LOC’
c.	/ɑ-k-ɪmɔk-ɑ-kɪn-ɪ/	[ɑ-k-ɪmɔk-ɑ-kɪn-ɪ]	‘GER-K-light.fire-EPI-DAT-V’
d.	/ɛ-ibus-ɑ-kɪn-ɪ/	[e-ibus-o-kin-i]	‘3-drop-EPI-DAT-V’
e.	/ɛ-ibus-ɑ-kɪn-(ɑ)/	[ɛ-ɪbʊs-ɔ-kɪn-(ɑ)]	‘GER-drop-EPI-DAT-VOI’

The key distinction between the Maasai and Turkana patterns is evident in (9c–e). In these examples we see a three-way alternation [ɑ∼o∼ɔ] for the epipatetic suffix vowel. This vowel, which is underlyingly /ɑ/ (9c), is realized as [o] after a [+ATR] root (9d). When preceded by a [+ATR] root and followed by a dominant [−ATR] suffix, though, this vowel is realized as [ɔ] (9e). The surface quality of an underlying suffixal low vowel is thus not able to be determined based solely on whether there is a [+ATR] vowel to its left; it also depends on the presence or absence of a dominant [−ATR] suffix vowel to its right.

Analytically speaking, Maasai and Turkana both exhibit bidirectional [+ATR] spreading, and in addition Turkana exhibits [−ATR] spreading.[13] This additional part of the pattern in Turkana has significant computational repercussions for the analysis. Recall that the realization of every vowel in Maasai is dependent on two sources of information: that vowel’s input specification, and the output specification of a neighboring vowel (to the right for prefixes, potentially to the right or to the left for roots and suffixes). In Turkana, the realization

[12] Noske (2001) transcribes this root once as [edo] (p. 778), and once as [ido] (p. 782). Dimmendaal (1983) always transcribes this root as [ido] or, before a dominant [−ATR] suffix, as [ɪdɔ].

[13] All of the examples of [−ATR] dominant suffixes that we know of appear word-finally, so this additional spreading process is only attested to be leftward. We nevertheless analyze it as being bidirectional in principle. Nothing hinges on this idealization.

of suffixal low vowels depends on additional information: whether there is a preceding [+ATR] vowel (in the root or in a dominant [+ATR] suffix), and whether there is a following [−ATR] dominant suffix vowel.

Both sources of information are in principle at an unbounded distance from the suffixal low vowel in question. This dependency is thus consistent with the more expressive unbounded circumambient class of patterns (Jardine 2016; Heinz 2018, §6.2.4). Turkana should thus *not* be amenable to HL decomposition and thus to an analysis in terms of a WDRF. If only the class of Maasai-like bidirectional harmonies are able to be modeled with Heinz and Lai's (2013) restrictions on alphabet- and length-preserving compositions, then such an analysis should fail for Turkana. We show here that this is not the case, and that a pair of contradirectional, alphabet- and length-preserving subsequential functions is capable of modeling the Turkana pattern.

It is first necessary to describe the FSTs required to analyze the pattern. Consider the transitions through the FST in Fig. 11, which spreads [+ATR] rightward. Upon reading the left word edge symbol, the FST transitions to q_1. If the first vowel is [−ATR], the FST stays in q_1. So long as [−ATR] vowels are encountered, the FST will stay in this state. If a [+ATR] vowel is encountered, though, the FST transitions to q_2. In q_2, non-low vowels are output as their [+ATR] counterparts, and /ɑ/ is re-paired to |o|. This continues until either the right word edge symbol is encountered — which causes the FST to enter q_4, the accepting state — or a dominant [−ATR] suffix is read, at which point the FST transitions to q_3, and all following vowels (if any exist; see fn. 13) will be output as [−ATR]. If no [+ATR] vowels are present in the input, then the FST will either transition directly from q_1 to q_4, or via q_3 if a dominant [−ATR] vowel is present.

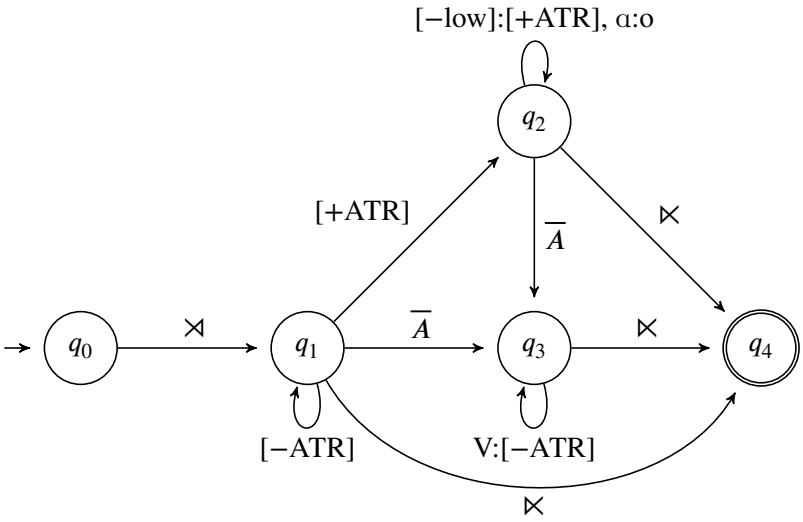

Figure 11 Left-subsequential (left-to-right) FST for Turkana. $\overline{A}$ indicates a dominant [−ATR] suffix.

Once the first pass of the transduction is complete, the output of the inner function is passed to the outer function, which is presented in Fig. 12. Upon reading the right word edge symbol, this FST moves from the initial state to q_1. At q_1, paths through the FST split, based on the ATR value of the final vowel. If the final vowel is [+ATR], this means that a [+ATR] root or dominant [+ATR] suffix was present in the initial input, and it also indicates that no dominant [−ATR] suffix was present in the initial input. As a result, [+ATR] spreading, represented by the loop on q_3, may proceed leftward through to the left edge of the word, q_4, or up to a blocking low vowel, which forces a transition to q_2 and all remaining vowels are output as [−ATR]. If the rightmost vowel in the word is not [+ATR], this entails that either no [+ATR] vowels were present in the initial input, or alternatively that a dominant [−ATR] suffix was present in the initial input. Regardless of which of these facts about the initial input is actually the case, encountering a [−ATR] vowel at the edge of the word causes the FST to move to q_2, where all remaining vowels are output as [−ATR].

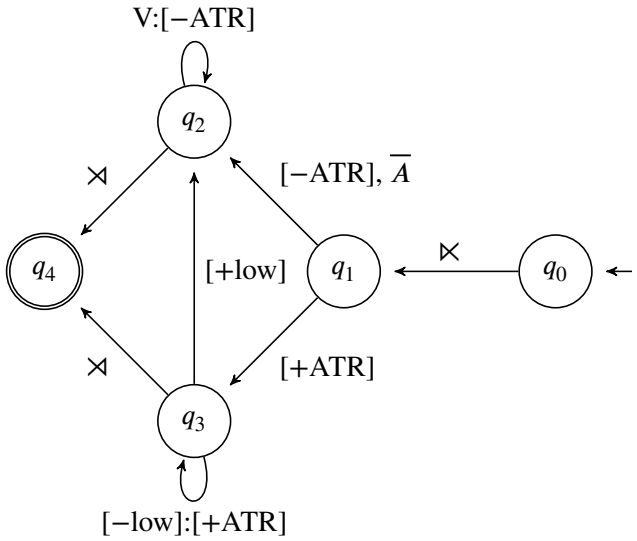

Figure 12 Right-subsequential (right-to-left) FST for Turkana. $\overline{A}$ indicates a dominant [−ATR] suffix.

To make the analysis concrete, consider the mapping modeled in Figs. 13 and 14. The first, rightward pass with the FST in Fig. 11 rewrites suffixal /ɑ/ and /ɪ/ to |o| and |i|, respectively, but fails to assimilate the word-final dominant [−ATR] suffix vowel.

direction	→							
Fig. 11 state	$q_0 \to q_1$	q_1	$q_1 \to q_2$	q_2	q_2	q_2	$q_2 \to q_3$	$q_3 \to q_4$
initial input	⋊	ɛ	i	b u	s ɑ	k ɪ	n ⓐ	⋉
intermediate	⋊	ɛ	i	b u	s o	k i	n ⓐ	⋉

Figure 13 Tape representation of the left-subsequential FST in Fig. 11 operating as the inner function

The second, leftward pass with the FST in Fig. 12 first encounters a word-final [−ATR] vowel, which also happens to be a dominant suffix. As a consequence, the FST rewrites all vowels as their [−ATR] counterparts — including suffixal |o| from underlying /ɑ/ as [ɔ].

intermediate	⋊	ɛ	i	b u	s o	k i	n ⓐ	⋉
final output	⋊	ɛ	ɪ	b ʊ	s ɔ	k ɪ	n ⓐ	⋉
Fig. 12 state	$q_4 \leftarrow q_2$	q_2	q_2	q_2	q_2	q_2	$q_2 \leftarrow q_1$	$q_1 \leftarrow q_0$
direction	←							

Figure 14 Tape representation of the right-subsequential FST in Fig. 12 operating as the outer function

We have just modeled the Turkana harmony pattern with two contradirectional alphabet- and length-preserving functions, despite the qualitatively different, unbounded circumambient dependency present in this pattern compared to the pattern in Maasai. Recall that when we re-ordered the Maasai FSTs, the final output was the same; we now attempt to apply the Turkana FSTs in the opposite order. In Fig. 15, the leftward function applies first, and since there is a dominant [−ATR] vowel at the right edge of the word, all non-low vowels are rewritten as their [−ATR] counterparts — and, crucially, the low vowel /ɑ/ remains |ɑ|.

direction	←								
Fig. 12 state	$q_4 \leftarrow q_2$	q_2	q_2	q_2	q_2	q_2	$q_2 \rightarrow q_1$	$q_1 \leftarrow q_0$	
initial input	⋊	ɛ	i	b u	s ɑ	k ɪ	n ⓐ	⋉	
intermediate	⋊	ɛ	ɪ	b ʊ	s ɑ	k ɪ	n ⓐ	⋉	

Figure 15 Tape representation of the right-subsequential FST in Fig. 12 operating as the inner function

As a consequence, there are now no [+ATR] vowels in the word. The outer, rightward function applies vacuously, and the final output is identical to the intermediate output, as shown in Fig. 21. Note that the final output in Fig. 21 is critically distinct from the attested output, which is shown in Fig. 14.

intermediate	⋊	ɛ	ɪ	b ʊ	s ɑ	k ɪ	n ⓐ	⋉
final output	⋊	ɛ	ɪ	b ʊ	s ɑ	k ɪ	n ⓐ	⋉
Fig. 11 state	$q_0 \rightarrow q_1$	q_1	q_1	q_1	q_1	q_1	$q_1 \rightarrow q_3$	$q_3 \rightarrow q_4$
direction	→							

Figure 16 Tape representation of the right-subsequential FST in Fig. 11 operating as the outer function

So what is it about the Turkana pattern that makes it unbounded circumambient? It may at first appear to be the fact that there is a three-way alternation [ɑ∼o∼ɔ], and/or that a minimum of two binary features are required to distinguish the three segments involved in the alternation (both [ɑ] and [ɔ] are [−ATR] while [o] is [+ATR], and both [o] and [ɔ] are [−low] while [ɑ] is [+low]). As it turns out, however, this apparently key part of the Turkana pattern can be reduced to a two-way alternation requiring only [±ATR]. The result no longer requires one or the other ordering between the functions of the EM decomposition, but it requires *some* ordering between them because they still interact in the relevant sense.

Imagine a hypothetical, minimal variation on Turkana — call it Turkana′ — with an inventory of ten vowels instead of nine, all of them perfectly harmonically paired: i∼ɪ, e∼ɛ, o∼ɔ, u∼ʊ, and — crucially — ɑ∼ɐ. Instead of being re-paired to [o], then, the [+low, −ATR] vowel /ɑ/ becomes the [+low, +ATR] vowel [ɐ] due to a preceding [+ATR] vowel. The three critical Turkana forms in (9c–e) would thus be as in (10c–e) below (glosses removed, since these are hypothetical examples in a hypothetical language).

(10) Dominance reversal in Turkana′

c. /ɑ-k-mɔk-ɑ-kɪn-ɪ/ [ɑ-k-mɔk-ɑ-kɪn-ɪ]
d. /ɛ-ibus-ɑ-kɪn-ɪ/ [e-ibus-ɐ-kin-i]
e. /ɛ-ibus-ɑ-kɪn-ⓐ/ [ɛ-ɪbʊs-ɑ-kɪn-ⓐ]

It is still the case in this hypothetical language that the surface realization of an underlying suffixal low vowel is dependent on unbounded circumambient information. It cannot be determined solely on whether there is a [+ATR] vowel to its left — a 'no' answer to this question is sufficient to determine that it must surface as [ɑ], but a 'yes' answer also requires knowing whether there is a dominant [−ATR] suffix to its right; if the answer to *this* question is 'no', then the surface realization is [ɐ]; if it's 'yes', then it is [ɑ].[14]

The contradirectional subsequential functions required to analyze Turkana′ are the same as those given in for actual Turkana in Figs. 11 and 12, except that low vowels are simply changed to [+ATR] by the rightward function rather than also being re-paired. This in turn means that the two functions can apply in either order to obtain the intended output. If the rightward function applies first to an /i ɑ ⓔ/ sequence, the /ɑ/ will change to |ɐ|, only to change back to [ɑ] by the leftward function. If the leftward function applies first, the /i

[14] See Meinhardt et al. (2020) for an elaboration of this approach to the analysis of Turkana and generally decomposing lookahead phenomena into questions, based on van Rooij (2003).

ɑ/ sequence will change to |ɪ ɑ|, and the rightward function will not have any effect because there are no [+ATR] vowels remaining in the form. Under either order, then, the final output is [ɪ ɑ ɛ̈].

The two functions still interact, however, which can be appreciated by consideration of the fact that they cannot apply simultaneously to this input: the rightward function demands that the /i ɑ/ sequence change to [i ɐ], and the leftward function demands that it change to [ɪ ɑ]. The only way that this direct conflict between the two functions can be adjudicated is by ordering them with respect to one another, and it just so happens that they can be ordered either way to obtain the intended result.

So despite the fact that the surface quality of /ɑ/ in Turkana depends on unbounded circumambient information, we see that the pattern can be modeled without introducing novel symbols or length-increasing codes. This shows that Heinz and Lai's (2013) definition of WDRFs is not able to properly distinguish the Maasai pattern from the unbounded circumambient Turkana pattern, and that the sole formal restrictions on Maasai-like patterns are not alphabet- and length-preservation. Further support for the insufficiency of these restrictions comes from the ability to use methods other than novel symbols or length-increasing codes to mark up an intermediate form, as noted in §2.3, for instance in the form of phonotactically illicit symbols or symbol sequences (McCollum et al. 2018, O'Hara and Smith 2019, Smith and O'Hara 2019, Lamont et al. 2019). We discuss analyses that make use of this form of mark-up in §5.1 and in the Appendix.

The class of WDRFs should thus not be specifically restricted to alphabet- and length-preserving functions, nor by some enumeration of the myriad possible strategies for an inner function to use mark-up in order to communicate information to an outer function. Rather, WDRFs are properly constrained by a more general prohibition on function interaction. In the analysis of the Maasai pattern, the ordering of the functions (or indeed the lack thereof) does not matter because of the way the functions do not interact with one another: no vowel depends on unbounded circumambient information to determine its surface quality, and this fact makes the Maasai pattern weakly deterministic. However, in the analysis of the Turkana pattern, we see that ordering of the functions is crucial, precisely because the realization of suffixal /ɑ/ is dependent on potentially long-distance information both to its right and to its left. This dependence on distal information in both directions makes the Turkana pattern, like other unbounded circumambient patterns, an NDRF.

4 Revising the definition of weak determinism

> I'm fixing a hole where the rain gets in
> And stops my mind from wandering
> Where it will go
>
> *John Lennon / Paul McCartney*

In §3 we demonstrated that the nature of the dependencies in the harmony patterns of Maasai and Turkana are distinct, and that the Turkana pattern is more computationally expressive due to its unbounded circumambience. Despite this difference, Heinz and Lai's (2013) formal definition of the WDRFs treats the two patterns as equivalent since they can both be decomposed into two contradirectional subsequential functions that do not increase the size of the alphabet or augment the length of the string. We formalize an alternative definition, given as Def. 4 below, centered on a notion of interaction between composed functions that we elucidate in this section. Intuitively, two composed functions interact in the relevant sense if the set of changes made by the composition differs from the union of the changes each function would make separately, a notion similar though not identical to familiar notions of feeding and bleeding. The two functions of a properly decomposed NDRF interact in this sense ('μ-interaction'), while the two functions of a properly decomposed WDRF do not ('μ-conservation'). This definition addresses the inadequacies of Heinz and Lai's original definition and correctly differentiates Maasai-like patterns from unbounded circumambient Turkana-like patterns.

Definition 4. A regular function $\tau : X^* \to Z^*$ is WEAKLY DETERMINISTIC iff there exist two subsequential functions $I : X^* \to Y^*$ such that $X \subseteq Y$ and $O : Y^* \to Z^*$ such that O is either left- or right-subsequential, I is the other, $\tau = O \circ I$ and $O \circ I$ is a μ-conserving composition.

Def. 4 captures our interpretation of the intended scope of Heinz and Lai's (2013) original definition, forbidding the inner function in a proper HL decomposition from affecting or overwriting ('interacting with') the behavior of the outer function. We define this notion of interaction for functions on strings — examining related but distinct properties or dimensions of functions from existing notions of 'feeding' and 'bleeding' — and conclude the section with an illustrative discussion of how this particular notion of interaction captures the difference in complexity between weakly deterministic patterns and non-deterministic patterns.

4.1 μ-interaction as a property of function composition

As described in Heinz and Lai (2013), McCollum et al. (2018), Smith and O'Hara (2019), O'Hara and Smith (2019) and Lamont et al. (2019), numerous strategies for caching information in the intermediate form of an EM decomposition are available to the enterprising analyst. Rather than continuing to discover new strategies and enumerating each of them in a stipulative modified definition, we suggest defining the WDRFs by banning the general and extensionally-definable property shared by all these strategies.

An informal statement of what underlies all current strategies for mark-up is that the observable changes made by the outer function O to apply in compositions $\tau = O \circ I$ can depend on or be affected by changes made by the inner function I. That is, in each such strategy, exactly what kind of change I makes to an input w to $\tau = O \circ I$ can in general depend on which changes I makes to w in producing an intermediate representation that is the input to O. Informally, then, we suggest defining the WDRFs as those regular functions which result from the 'non-interacting' composition of subsequential functions — i.e., where changes made by the outer function are not dependent on or affected by changes made by the inner function.

In this section we build up a small inventory of concepts for talking about the set of changes (pairs of non-indentical input-output values) a function f makes and how this set of changes is altered by a particular composition $g \circ f$ with some other function g. We offer formal definitions of 'simultaneous application' as a special case of or variation on priority union (Kaplan 1987) and of analogues to 'feeding' and 'bleeding' through this lens of alterations to the set of changes made by a function. For ease of exposition we only consider here partial functions on a discrete set — e.g. unconditioned SPE-style rewrite rules considered extensionally as sets of pairs of input and output symbols. Our running examples will be context-free-simplifications of two process interactions within the same segment ('on focus' interactions; McCarthy 1999, Baković 2011) before returning to more naturalistic examples of string functions.

The first interaction is a chain shift ('counterfeeding on focus') in (11), based on a Western Basque example (de Rijk 1970, Hualde 1991).[15] Raising-to-high f in (11a) changes prevocalic /e/ to [i], and raising-to-mid g in (11b) changes prevocalic /a/ to [e]. The counterfeeding order $g \circ f$ is apparent in (11c): f is not applicable to /alabaa/, but g is, resulting in [alabea] 'daughter'; f is now newly applicable, but at this point it's too late for f to apply. (Note that the same result is obtained if f and g were to apply simultaneously — see Kenstowicz and Kisseberth 1979, Ch. 8 and Joshi and Kiparsky 1979, 2006.) The reverse order $f \circ g$ is shown to the right to yield an incorrect result where prior application of g enables application of f.

(11) Raising-to-mid counterfeeds raising-to-high in Western Basque

a. Raising-to-high (f): e ⟶ i / __ V /semee/ $\xrightarrow[f]{}$ [semie] 'son'

b. Raising-to-mid (g): a ⟶ e / __ V /alabaa/ $\xrightarrow[g]{}$ [alabea] 'daughter'

c. $g \circ f$: /alabaa/ $\xrightarrow[f]{}$ |alabaa| $\xrightarrow[g]{}$ [alabea] ($f \circ g$: /alabaa/ $\xrightarrow[g]{}$ |alabea| $\xrightarrow[f]{}$ *[alabia])

d. Context-free simplifications: $f(x) = \begin{cases} \text{i} & \text{if } x = \text{e} \\ x & \text{otherwise} \end{cases}$ $g(x) = \begin{cases} \text{e} & \text{if } x = \text{a} \\ x & \text{otherwise} \end{cases}$

[15] The discussion of this example and the next follows Baković (2011). We leave 'feeding', etc. unformalized for the moment, and indulge freely in temporal metaphors.

Our second interaction is a case of 'counterbleeding on focus' in (12), based on a Low German example (Kiparsky 1973, Kenstowicz and Kisseberth 1971). Both spirantization f in (12a) and devoicing g in (12b) are potentially applicable to /taːg/ in (12c), and both indeed apply, yielding [taːx] 'day'. (This result is also obtained if f and g are applied simultaneously.) The reverse order $f \circ g$ is shown to the right to yield an incorrect result where prior application of g disables application of f.

(12) Devoicing counterbleeds spirantization in Low German

a. Spirantization (f): $\begin{bmatrix}-\text{son}\\+\text{voi}\end{bmatrix} \longrightarrow$ [+cont] / V— /taːgə/ $\xrightarrow[f]{}$ [taːɣə] 'days'

b. Devoicing (g): [−son] $\longrightarrow$ [−voi] / —# /haʊz/ $\xrightarrow[g]{}$ [haʊs] 'house'

c. $g \circ f$: /taːg/ $\xrightarrow[f]{}$ |taːɣ| $\xrightarrow[g]{}$ [taːx] ($f \circ g$: /taːg/ $\xrightarrow[g]{}$ |taːk| $\xrightarrow[f]{}$ *[taːk])

d. Context-free simplifications, modeled as sets of non-vacuous input-output pairs:
$f = \{(\text{g}, \text{ɣ}), (\text{b}, \text{β}), (\text{d}, \text{ð}), \dots\}$ $g = \{(\text{ɣ}, \text{x}), (\text{β}, \text{ɸ}), (\text{ð}, \text{θ}), (\text{z}, \text{s}), (\text{g}, \text{k}), (\text{b}, \text{p}), (\text{d}, \text{t}), \dots\}$

Consider now any partial function $f : X \to X$. For such a function, we distinguish those elements of f's domain of definition — $dom(f)$ — where f behaves exactly like the identity function $id_X(x) = x$ from those elements of $dom(f)$ that f changes. The first set of elements — compactly denoted by $fix(f)$ — constitutes f's *fixed points* or *fixset*, defined in Def. 5; unaware of a convenient existing term for the second set, we refer to its elements as f's *mutation points* or *mutation set* and denote it by $\mu(f)$, defined in Def. 6.[16]

Definition 5. The *fixset* for any partial function $f : X \to X$ is given by

$$fix(f) = \{x | x \in dom(f) \wedge f(x) = x\}$$

Definition 6. The *mutation set* for any partial function $f : X \to X$ is given by

$$\mu(f) = \{x | x \in dom(f) \wedge f(x) \neq x\}$$

Note that the mutation set and the fixset of any partial function f are complementary with respect to f's domain: $fix(f) = dom(f) - \mu(f)$. For example, with respect to the context-free simplifications of the Western Basque functions in (11), if Σ represents the inventory of Western Basque and thus the domain of both raising-to-high f and raising-to-high g, then we have the mutation sets and fixsets shown in (13).

(13) Mutation sets and fixsets for Western Basque raising-to-high, raising-to-mid, and their compositions

$\mu(f) = \{\text{e}\}$ $\mu(g) = \{\text{a}\}$ $\mu(g \circ f) = \{\text{e}, \text{a}\}$ $\mu(f \circ g) = \{\text{a}, \text{e}\}$
$fix(f) = \Sigma - \{\text{e}\}$ $fix(g) = \Sigma - \{\text{a}\}$ $fix(g \circ f) = \Sigma - \{\text{e}, \text{a}\}$ $fix(f \circ g) = \Sigma - \{\text{a}, \text{e}\}$

Similarly, for the context-free simplifications of the Low German functions in (12), we have the mutation sets in (14). (The fixsets can be obtained, as noted above, by $fix(f) = \Sigma - \mu(f)$, where $\Sigma = dom(f)$.)

(14) Mutation sets for Low German spirantization, devoicing, and their compositions

$\mu(f) = \{\text{g}, \text{b}, \text{d}, \dots\}$ $\mu(g) = \{\text{ɣ}, \text{β}, \text{ð}, \text{z}, \text{g}, \text{b}, \text{d}, \dots\}$
$\mu(g \circ f) = \{\text{ɣ}, \text{β}, \text{ð}, \text{z}, \text{g}, \text{b}, \text{d}, \dots\}$ $\mu(f \circ g) = \{\text{ɣ}, \text{β}, \text{ð}, \text{z}, \text{g}, \text{b}, \text{d}, \dots\}$

We are not only interested in the set of things that are changed, but also *how* each of those elements are changed: we denote the set of *mutation maps* of a partial function $f : X \to X$ by $\vec{\mu}(f)$, as given in Def. 7.

[16] Strictly speaking, the two definitions here hold for any partial function $f : X \to Y$ such that $X \subseteq Y$, but we ignore that throughout the text for simplicity of presentation.

Definition 7. The set of *mutation maps* of a partial function $f : X \to X$ is given by

$$\vec{\mu}(f) = \{(x, f(x)) | x \in \mu(f)\}$$

Continuing our examples from before, we have the mutation maps in (15) for the simplified, context-free Western Basque example. The mutation maps for the Low German example are similar (if less compactly presentable), but they could plausibly be given in terms of either symbols or feature bundles, with different implications for present purposes, as discussed below.

(15) Mutation maps for Western Basque raising-to-high f, raising-to-mid g, and their compositions

$$\vec{\mu}(f) = \{(\text{e}, \text{i})\} \qquad \vec{\mu}(g) = \{(\text{a}, \text{e})\} \qquad \vec{\mu}(g \circ f) = \{(\text{e}, \text{i}), (\text{a}, \text{e})\} \qquad \vec{\mu}(f \circ g) = \{(\text{a}, \text{i}), (\text{e}, \text{i})\}$$

Using mutation maps, we offer two equivalent ways[17] of distinguishing function compositions that involve interaction (are 'μ-altering' or 'μ-interacting') from those that do not involve interaction (are 'μ-conserving').[18] These two definitions are previewed in (16).

(16) Two informal definitions of μ-conserving partial function composition:

a. $g \circ f$ is μ-conserving iff $g \circ f$ makes all and only the same changes as the union of the changes made by f with the changes made by g.

b. $g \circ f$ is μ-conserving iff $g \circ f$ is equivalent to simultaneous application of g and f.

We offer two characterizations of what qualifies as μ-interaction in acknowledgment of the different interests that bring analysts to consideration of interaction. The first is a more general characterization than the second and useful when there is, in fact, μ-interaction: approaching μ-interaction using the definition given in (16a) aids identification of where two composands interact within their domains and the kind of μ-interaction at play (i.e. pushing, pulling, shifting; see §4.3). The second characterization is most relevant for establishing when it is that no μ-interaction exists.

In more detail, the first definition of μ-conservation presented here treats partial functions as a set of (input, output) pairs and compares the collection of mutation maps of each composand to that of the composition:

Definition 8. For all partial functions $f, g : X \to X$, f and g are *μ-conserving* when composed as $g \circ f$ iff

$$\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f)$$

That is, for any non-interacting composition $g \circ f$, every change made by f is a change made by $g \circ f$, every change made by g is a change made by $g \circ f$, and there are no other changes that $g \circ f$ makes.

As noted, this definition treats functions as collections of mappings, and it treats composition as a particular binary operator that takes two functions and returns a new function, identifying a key property of μ-conserving compositions that makes them such. In principle, however, any operation on two partial functions f, g — not just ordered application — that yields a new partial function could also be said to be μ-conserving iff the result of the operation conserves the mutation maps of the operands.

An example of an alternative operation on mappings is *priority union* — an operation on relations originally introduced for LFG and argument-value matrices in the 1980s (Kaplan 1987), but also applied to strings by Kaplan and Newman (1997) and Karttunen (1998). We present a general (non-string) formalization of *simultaneous application*[19] defined in terms of priority union that exhibits exactly this property of μ − *conservation*. For this reason, priority union lies at the heart of the second definition of μ-conserving compositions, introduced in (16b). As we will show, this definition states that μ-conserving compositions are exactly those which can be equivalently defined in terms of priority union. Roughly, $g \circ f$ is μ-conserving

[17] See §B.3 for a third definition and proofs of the equivalence between the three definitions.

[18] As noted in §4.1, the notion of function 'interaction' in this paper is related but distinct from other notions of function interaction.

[19] We offer a generalization to string functions in §4.2 that accords with linguistic intuitions.

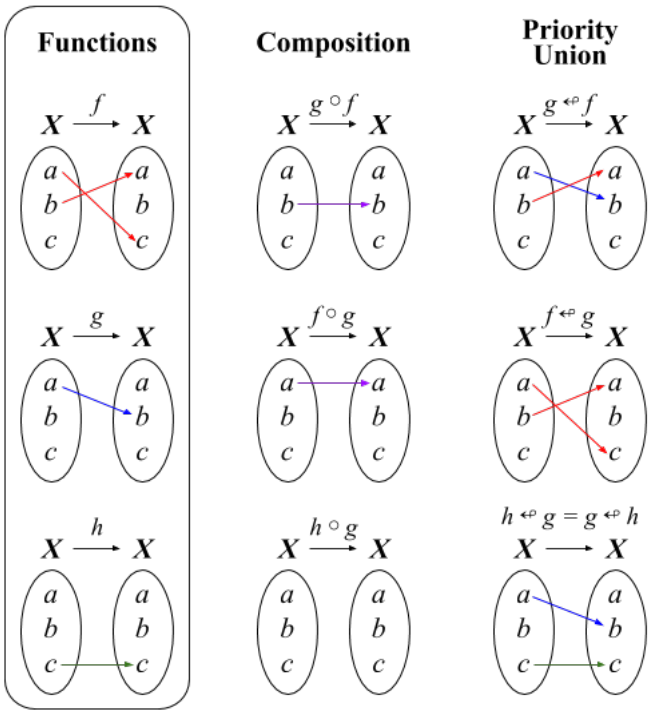

Figure 17 **Left column:** a set of simple functions, f, g, h. **Middle column:** compositions of f, g, and h. In $g \circ f$, g inverts f's change to b and is undefined on $c = f(a)$. In $f \circ g$, f inverts g's change to a. In $h \circ g$, g never maps anything into $dom(h)$, so $h \circ g$ is (one way of defining) the unique function that is defined nowhere, i.e. that maps everything to $\perp$. **Right column:** g's maps are conserved and only those maps of f that don't conflict with g are conserved. f and g conflict everywhere, so $f \leftarrow\!\!\!\rho\, g = f$. g and h do not conflict anywhere, so $g \leftarrow\!\!\!\rho\, h = h \leftarrow\!\!\!\rho\, g$.

iff $\vec{\mu}(g \circ f) = \vec{\mu}(g \leftarrow\!\!\!\rho\, f) = \vec{\mu}(f \leftarrow\!\!\!\rho\, g)$, i.e. when the priority union of the mutation maps of the composands commutes. To arrive at the definition introduced in (16b), we extend this basic idea to define an operation on functions that intuitively corresponds to 'simultaneous application' in the setting of partial functions on discrete sets. Below we introduce priority union and its relation to simultaneous application before fleshing out the (16b) definition of μ-conserving compositions in depth.

Intuitively, given two partial functions or relations g and f, the (left) priority union — $g \leftarrow\!\!\!\rho\, f$ — keeps all mappings defined in g, and then adds to that anything in f that doesn't 'conflict' with any mappings already defined in g: if there is a mapping $(x, g(x))$ defined in g and a mapping $(x, f(x))$ defined in f, then only the mapping of g defined on x survives the union because g gets 'priority' over f. The only mappings of f that will be found in $g \leftarrow\!\!\!\rho\, f$ are those mappings of f already found in g, plus mappings $(x, f(x))$ where $g(x)$ is undefined. See Fig. 17 for visual depiction of examples contrasting priority union with composition.

Definition 9. Given any two (partial) relations $R, S \subseteq X \times Y$, the *priority union of R with S* is given by

$$R \leftarrow\!\!\!\rho\, S = R \cup \{(x, y) | (x, y) \in S \wedge \nexists y'.(x, y') \in R\}$$

Note that we have introduced *left* priority union. Right priority union $\rho\!\!\rightarrow$ is defined analogously. We will use 'priority union' henceforth to refer to left priority union. Note that partial functions are closed under priority union — but not under general set-theoretic union.

The following definition (equivalent for relations that are partial functions) is evocative of 'pattern matching' expressions in modern functional programming languages, where the output for some input value is determined by attempting to match the input value against a sequence of 'cases' or 'patterns', and the first

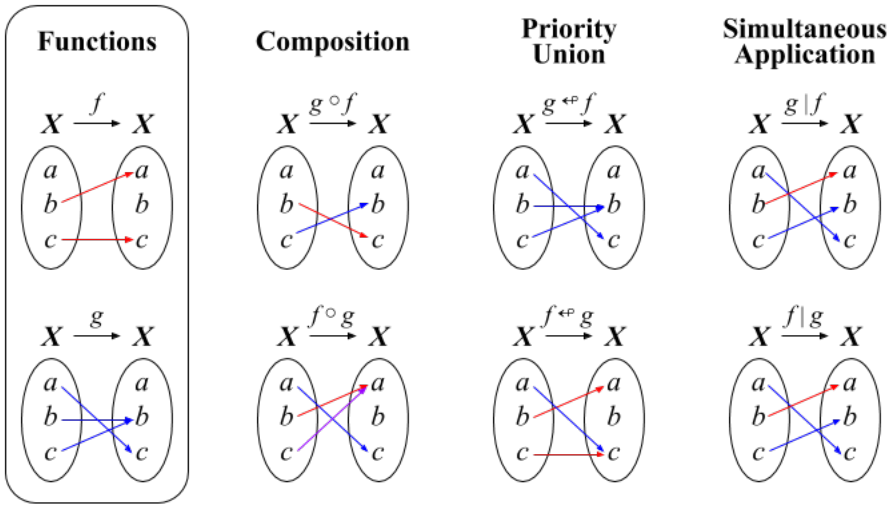

Figure 18 **Left column:** a set of simple functions, f and g. **Left middle column:** compositions of f and g. In $g \circ f$, g maps $a = f(b)$ to c and g maps $c = f(c)$ to b. In $f \circ g$, f maps $b = g(b) = g(c)$ to a and maps $c = g(a)$ to c. **Right middle column:** priority unions of f and g. In $g \twoheadleftarrow f$, g's maps are necessarily conserved (it has 'priority') and everywhere f is defined g is already defined ('f conflicts with g everywhere it is defined'), so no maps of f surive. In $f \twoheadleftarrow g$, all of f's maps are present because it has priority; the only map of g present is the one that doesn't conflict with any defined mapping of f. **Right column:** simultaneous applications of f and g. g and f's mutation maps never disagree about what to do — in fact they are never modify the same things — so they are simultaneously applicable. Simultaneous application — where defined — is always commutative.

one to match (starting from the top and going down) determines what is ultimately output:[20]

Definition 10. Given partial functions $f, g : X \to X$, the *priority union of f with g* is given by

$$f \twoheadleftarrow g(x) = \begin{cases} f(x) & \text{if } x \in dom(f) \\ g(x) & \text{if } x \in dom(g) \wedge x \notin dom(f) \end{cases}$$

These two definitions of priority union are entirely exchangeable for partial functions — we encourage the reader to prove this for themselves.

Two functions can only be 'simultaneously applied' to any element x *precisely* when only one of them applies (non-vacuously), or when both apply in the same way ($x \in equalizer(f, g)$)[21] — i.e. roughly when the two functions don't 'conflict'. Formally, this is exactly when their mutation maps commute under priority union or (equivalently) when the priority union of their mutation maps is the same as normal set-theoretic union:

Definition 11. Given two functions $f, g : X \to X$, simultaneous application of f and g is *possible* iff

$$\vec{\mu}(f) \twoheadleftarrow \vec{\mu}(g) = \vec{\mu}(g) \twoheadleftarrow \vec{\mu}(f) = \vec{\mu}(f) \cup \vec{\mu}(g)$$

This defines a *condition* for simultaneous application but doesn't define simultaneous application *per se*. Given two partial functions f, g that meet the condition for simultaneous application, $\vec{\mu}(f) \twoheadleftarrow \vec{\mu}(g)$ has all the

[20] If P, Q, R are predicates that define membership in, say, $dom(f), dom(g), dom(h)$, then $(f \twoheadleftarrow g \twoheadleftarrow h)(x)$ is equivalent to 'If $P(x)$, then $f(x)$; (else) if $(\neg P(x) \wedge) Q(x)$, then $g(x)$; (else) if $(\neg P(x) \wedge \neg Q(x) \wedge) R(x)$, then $h(x)$'.

The more functional programming and algebra- (than automata- or logic-) flavored formulations of the regular ($\supset$ *rational*) or polyregular transductions we are aware of all contain something comparable (see the *conditional choice* operator in Alur et al. 2014; the *else* operator in Alur et al. 2015; the *disjoint union* combinator in Bojańczyk et al. 2018; the *case* function in Bojańczyk 2018).

[21] See §B.1.

mutation maps of simultaneous application, but is missing the fixmaps of f and g. Since we do not want the fixmaps of f or g to interfere with conservation of mutation maps, the definition for simultaneous application below adds fixmaps after adding together mutation maps; note that the order of adding one set of fixmaps vs. the other is arbitrary.

Definition 12. Given two functions $f, g : X \to X$ that meet the condition for simultaneous application,[22] the *simultaneous application of f and g* — denoted $f \mid g$ — is defined as

$$f \mid g = (\vec{\mu}(f) \nleftarrow \vec{\mu}(g)) \nleftarrow (\overrightarrow{fix}(f) \nleftarrow \overrightarrow{fix}(g))$$

where $\overrightarrow{fix}(f)$ is the fixset analogue of $\vec{\mu}(f)$, i.e. $\overrightarrow{fix}(f) = \{(x, x) | x \in fix(f)\}$.

See Fig. 18 for visual depiction of examples and contrast with both priority union and composition.

With the definition of simultaneous application in place, we return to the second definition of μ-conservation, first introduced in (16b). One can compare Def. 8 with the condition on simultaneous application and see a straightforward connection: when simultaneous application is applicable, $f \mid g$ results in a function that conserves the mutation maps of both operands, just like a μ-conserving composition. We can define μ-conserving compositions as precisely those that are equivalent to simultaneous application:

Definition 13. Given $f, g : X \to X$, $g \circ f$ is μ-conserving iff

$$g \circ f = g \mid f$$

While a more detailed discussion of the properties of μ-interaction is a topic for future work, we conclude this introduction to μ-interaction with some observations about ordering before discussing examples and defining analogues of 'feeding' and 'bleeding' in the context of μ-interaction and μ-conservation. First, recall that composition is *ordered* and that μ-interaction in $g \circ f$ does not in general entail that $f \circ g$ involves μ-interaction. Second, note that non-commutativity (non-vacuous ordering) at any point x (i.e. $g \circ f(x) \neq f \circ g(x)$) entails μ-interaction at x, but commutativity at a point x does not entail μ-conservation at x.[23]

To see how these definitions play out, consider the context-free variants of the Basque raising processes from (11). Recall:

(17) Mutation maps for Western Basque raising-to-high, raising-to-mid, and their compositions

$$\vec{\mu}(f) = \{(\mathrm{e}, \mathrm{i})\} \qquad \vec{\mu}(g) = \{(\mathrm{a}, \mathrm{e})\} \qquad \vec{\mu}(g \circ f) = \{(\mathrm{e}, \mathrm{i}), (\mathrm{a}, \mathrm{e})\} \qquad \vec{\mu}(f \circ g) = \{(\mathrm{a}, \mathrm{i}), (\mathrm{e}, \mathrm{i})\}$$

By Defs. 11–13, $g \circ f$ is μ-conserving because $\vec{\mu}(g \circ f) = \vec{\mu}(g) \nleftarrow \vec{\mu}(f) = \vec{\mu}(f) \nleftarrow \vec{\mu}(g)$. We could also conclude that $g \circ f$ is μ-conserving by alternatively noting that 'simultaneous application' is possible ($\vec{\mu}(f) \nleftarrow \vec{\mu}(g) = \vec{\mu}(g) \nleftarrow \vec{\mu}(f)$) and $g \mid f = g \circ f$. The other application order $f \circ g$, however, is μ-altering by Def. 8 because there is a mutation map of g (viz. (a, e)) missing from $\vec{\mu}(f \circ g)$.

The analysis of the context-free variants of the functions from the Low German example given in (12) are similar, but complicated by whether we treat these mappings as functions on symbols or on feature bundles. As functions on feature vectors, we can conclude that $g \circ f$ (spirantization preceding devoicing) is μ-conserving either by noting that all mutation maps of the composands are conserved (per Def. 8) or by noting that simultaneous application results in the same mutation maps as $g \circ f$ (per Def. 13). As functions on symbols, however, devoicing acts (non-vacuously) on some symbols that are themselves the result of non-vacuous application of spirantization, meaning some of the mutation maps of spirantization are missing from $\vec{\mu}(g \circ f)$. The other order of application $f \circ g$ results in the loss of all of spirantization's mutation maps ($\vec{\mu}(f)$) from $\vec{\mu}(f \circ g)$ — regardless of whether f or g are modeled as functions on symbols or feature vectors.

[22] Note that iff the condition for simultaneous application is not met, then the right-hand side of $f \mid g$ is still perfectly calculable, but the equality $f \mid g = g \mid f$ will not, in general, hold, nor will linguistic intuitions about what 'simultaneous application' ought to be.

[23] See Theorems 3–4 in §B.4.

The astute reader might be tempted to conclude from the examples so far that 'μ-interacting' compositions are all and only those where 'feeding' or 'bleeding' interactions occur, but there are some discrepancies between what properties of 'map interaction' are relevant to the context of this article vs. how the terms *feeding* and *bleeding* are often used in phonology as introduced in Kiparsky (1965, 1968) and discussed in Kenstowicz and Kisseberth (1977, Ch. 4) and Kenstowicz and Kisseberth (1979, Ch. 8). We formalize 'feeding', etc. for the simple case of partial functions on discrete sets in Defs. 14–16.

Definition 14. Given $f, g : X \to X$, f *feeds* g at x in $g \circ f$ iff

$$x \in \mu(f) \wedge f(x) \in \mu(g)$$

...i.e. f changes x and maps x into $\mu(g)$.

Definition 15. Given $f, g : X \to X$,f *bleeds* g at x in $g \circ f$ iff

$$x \in \mu(f) x \in \mu(g)$$

...i.e. f changes x and maps x from $\mu(g)$.

Definition 16. Given $f, g : X \to X$, f *transfuses* g at x in $g \circ f$ iff f both feeds and bleeds g at x.

...i.e. f changes x and maps x from $\mu(g)$ to (a distinct point) y in $\mu(g)$.

We will refer to cases where f feeds (bleeds) g at x without also tranfusing g at x as 'proper feeding' ('proper bleeding'). Note that one species of interaction at x with respect to $g \circ f$ can in general co-occur with a different species of interaction at some other domain element y with respect to $g \circ f$.

The typology of map interaction discussed in Defs. 14–16 — focused on the dynamics of mapping into and out of the outer function's fixset and mutation set — is related to but clearly distinct from whether the inner or outer function's mutation maps are altered vs. conserved when composed with the other. In Defs. 17–19, we provide definitions for species of μ-interaction that are parallel to the vampirically-inspired species of classic interaction.

Definition 17. Given $f, g : X \to X$, f *pushes* g at x iff

$$(x, y) \in \vec{\mu}(f) \wedge (x, y) \notin \vec{\mu}(g \circ f)$$

Here a mutation map of f is missing from the set of mutation maps of the composition $g \circ f$ and it has been replaced by a map $(x, g \circ f(x))$.[24]

Definition 18. Given $f, g : X \to X$, f *pulls* g at x iff

$$(x, y) \in \vec{\mu}(g) \wedge (x, y) \notin \vec{\mu}(g \circ f)$$

More straightforwardly than pushing, in a pulling interaction a mutation map of g is missing from the set of mutation maps of the composition $g \circ f$.

Definition 19. Given $f, g : X \to X$, f *shifts* g at x iff f both pushes and pulls g at x.

As in the context-free simplifications of Western Basque and Low German discussed above, a feeding composition order is *usually* also a pushing composition order (and vice versa), and likewise for bleeding/pulling and transfusion/shifting. However, one salient edge case highlights how these families of patterns diverge:

[24] Note that for the general setting of partial functions on sets of discrete elements, the key defining element of a pushing interaction is a mutation map of f *missing* from $\vec{\mu}(g \circ f)$, not the appearance of a *novel* map $(x, g \circ f(x))$ in $\vec{\mu}(g \circ f)$: in general, $(x, g \circ f(x))$ could already be a mutation map of g (i.e. $(x, g \circ f(x))$ and $(f(x), g \circ f(x))$ are both in $\vec{\mu}(g)$), but in such a case $(x, f(x))$ will still necessarily be missing from $\vec{\mu}(g \circ f)$.

Definition 20. Given $f, g : X \to X$ and a composition $g \circ f$, *point neutralization* occurs at x iff f bleeds g at x *without* pulling g at x.

In other words, two functions make the same change to the same domain element, and when composed only one of them actually makes the change in question, and that change is not overwritten by yet another change. In terms of other vocabulary introduced here, these are bleeding interactions where f bleeds g at x but $f(x) = g(x)$ and $f(x) \in \mathit{fix}(g)$, so $(x, g(x)) \in \vec{\mu}(g \circ f)$.

In the absence of real example at our fingertips, consider the hypothetical case of point neutralization in (18), combining a reversal of the Lamba lowering rule (f from (2)) with the Finnish word-final raising rule (f from (1)). Both rules raise mid vowels, but harmonic raising f in (18a) raises all mid vowels when a high vowel precedes while word-final raising g in (18b) raises only front mid vowels word-finally. When both conditions are met, as in (18c), both f and g are applicable. This is like bleeding — but, unlike bleeding, applying the rules in either order (or simultaneously) achieves the same result because each rule effects the same change as the other.

(18) Point neutralization (hypothetical)

a. Harmonic raising (f): [−low] ⟶ [+high] / [+high] C_0 — /ikota/ $\xrightarrow[f]{}$ [ikuta]

b. Word-final raising (g): e ⟶ i / — # /ake/ $\xrightarrow[g]{}$ [aki]

c. $g \circ f$ or $f \circ g$: /uke/ $\xrightarrow[f/g]{}$ |uki| $\xrightarrow[g/f]{}$ [uki]

The exact correspondence between interaction relations based on mutation set–fix set dynamics (feeding/bleeding/transfusion) and interaction relations based on mutation map dynamics (pushing/pulling/shifting) is outside the main scope of this paper, but Figs. 19–20 below and proofs in §B.5 (see Theorems 5–10) demonstrate for partial functions on discrete sets that they are distinct, and that each μ-altering relation is generally a restriction or refinement of the analogous mutation set – fix set relation.

Mutation-map relation in $g \circ f$ at x	Definition	μ-fix relation(s)
Pushing	A mutation map of f is missing from $\vec{\mu}(g \circ f)$.	f must feed g at x, but not necessarily properly.
Pulling	A mutation map of g is missing from $\vec{\mu}(g \circ f)$.	f must bleed g at x, but not necessarily properly.
Shifting	f pushes and pulls g at x.	f must transfuse g at x.
Proper pushing	f pushes g at x without shifting g at x.	f must feed g at x, *but not necessarily properly.*
Proper pulling	f pulls g at x without shifting g at x.	f must properly bleed g at x.

Figure 19 The relationship between mutation-map dynamics and μ-fix dynamics, focused on mutation-map properties of a composition at a particular input. If the relationship at left holds with respect to $f, g, g \circ f$ and x (e.g. 'f pushes g at x in $g \circ f$'), then the rightmost column indicates the μ-fix relations that need (or need not) hold between g, f, $g \circ f$, and x. Proofs are given in §B.5; see Theorems 5–7.

μ-fix relation in $g \circ f$ at x	Definition	Mutation-map relations
Feeding	f changes x and maps it into $\mu(g)$.	f must push g at x, but not necessarily properly.
Bleeding	f changes x and maps it from $\mu(g)$.	No mutation-map relationship is necessary.
Transfusion	f both feeds and bleeds g at x.	f must push g at x but need not also pull (and therefore shift) g at x.
Proper feeding	f feeds g at x without transfusing g at x.	f must properly push g at x.
Proper bleeding	f bleeds g at x without transfusing g at x.	f cannot push and may properly pull g at x.

Figure 20 The relationship between properties or relations about mutation set dynamics and mutation-map dynamics, focused on mutation set properties of a composition at a particular input. If the relationship at left holds with respect to $f, g, g \circ f$, and x (e.g. 'f bleeds g at x in $g \circ f$'), then the rightmost column indicates the mutation-map relations that need (or need not) hold between $g, f, g \circ f$ and x. Proofs are given in §B.5; see Theorems 8–10.

More importantly for this article, we suggest the reader may safely walk away with two rough heuristics for identifying μ-interaction in a composition $g \circ f$:

(19) Heuristics for μ-interaction.

a. If f feeds, bleeds, or transfuses g at any x in $g \circ f$, then $g \circ f$ is (probably) a μ-interacting order.

b. If the order of f and g is non-vacuous, this is sufficent to conclude that at least one order of composition involves μ-interaction at some domain element.

4.2 μ-interaction and weak determinism

The definition and examples of μ-alteration and μ-conservation given so far have been easy to describe and reason about, but have not been specific to the context of transducers or string functions.[25] The key difference between functions discussed so far and string functions is that, in addition to determining *whether* a string w is changed by a string function f (i.e. whether $w \in \mu(f)$) and what distinct output string an input string is mapped to, we can also describe *where* a string w is changed by f and how an input symbol within w is changed. This is crucial for extending our definition of interaction to strings.

Just as two functions f, g (not necessarily on strings) are μ-interacting when composed iff the composition $g \circ f$ does not conserve the mutation maps of the two functions when considered in isolation, we intuitively want a finer-grained notion of μ-interaction we can apply to the composition $g \circ f$ of two string functions that identifies such a composition as μ-altering in this more specific sense iff the composition does not conserve the changes to particular locations in the string that the two functions would make in isolation. To distinguish the more general notion of interaction from this location-specific notion of interaction, we use the term FACTOR-INTERACTION. Note, however, that while factor-interaction entails μ-interaction, the reverse is not the case: two string functions that only ever operate on non-overlapping regions of a string cannot

[25] We also leave formal models of substance in representation (e.g. feature-vectors and feature geometry) to future work. We expect such work to simplify some of the corner cases and/or mismatches between feeding/bleeding and pushing/pulling.

(by definition) factor-interact but could, in general, μ-interact. Throughout the rest of the paper we almost exclusively consider string functions and are only interested in factor-interaction, so we abbreviate '(non-)factor-interaction' as '(non-)interaction' henceforth. We also restrict our consideration to length-preserving functions for simplicity of presentation, in line with previous work defining WDRFs (Heinz and Lai 2013).

The easiest kind of interaction among string functions to see involves 'lined up' pairs of output functions from composed transducers $\tau_1 \circ \tau_0$. For example, recall the tape representation of the full derivation of /ɛ-ibus-ɑ-kɪn-ⓐ/ ↦ [ɛ-ɪbʊs-ɔ-kɪn-ⓐ] from Figs. 13–14, consolidated in Fig. 21 for convenience.

initial input	⋊	ɛ	i	b	u	s	ɑ	k	ɪ	n	ⓐ	⋉
intermediate	⋊	ɛ	i	b	u	s	o	k	i	n	ⓐ	⋉
final output	⋊	ɛ	ɪ	b	ʊ	s	ɔ	k	ɪ	n	ⓐ	⋉

Figure 21 A consolidation of Figs. 13–14 highlighting the crucial point of interaction.

In the shaded column, the incremental output function of the first FST changes an input symbol from /ɑ/ to |o|, and then the incremental output function of the second FST proceeds to overwrite ('interfere' or 'interact' with) the change made by the first FST, changing |o| to [ɔ]. More generally, however, a change made by the inner function in a composition can cause the behavior of the outer function to change somewhere else — a relationship akin to 'feeding' or 'bleeding' on environment (McCarthy 1999, Baković 2011). To illustrate this and to formalize the richer notion of 'changes' we can associate with string functions, we first introduce a few toy string functions defined by rewrite rules and an example input string in (20). Throughout, ρ refers to string functions given as rewrite rules.

(20) Example string functions, where $s = cad$

a. $\rho_A\colon c \to d / \_\_a$ $\quad \rho_A(s) = dad$

b. $\rho_B\colon a \to b / c\_\_d$ $\quad \rho_B(s) = cbd$

c. $\rho_C\colon d \to c / da\_\_$ $\quad \rho_C(s) = cad$

d. $\rho_D\colon d \to a / a\_\_$ $\quad \rho_D(s) = caa$

Intuitively, the following compositions (among others) involve interaction:

(21) Examples of pulling and pushing interactions of string functions.

a. $\rho_B \circ \rho_A$ involves pulling (bleeding) on environment.
   - $\rho_B(cad) = cbd$, but $\rho_B \circ \rho_A(cad) = \rho_B(dad) = dad$

b. $\rho_C \circ \rho_A$ involves pushing (feeding) on environment.
   - $\rho_C(cad) = cad$, but $\rho_C \circ \rho_A(cad) = \rho_C(dad) = dac$

In contrast, $\rho_D(cad) = caa$, $\rho_A(cad) = dad$, and the composition $\rho_D \circ \rho_A = \rho_D(dad) = daa$ involves no interaction.

To capture these intuitions, we associate each string-to-string function $f : X^* \to Y^*$ (with $X \subseteq Y$) and potential input $w = x_1 x_2 \ldots x_j \ldots x_n$ to τ with the set of uniquely identifiable *factors* of w[26] that τ would change in computing $f(w)$ and information about how f would change them. We call this set of changed factors (and information about changes) 'the μ-factors of f in w' or 'the μ-factors of w with respect to f' and denote it with $\vec{\mu}(f, w)$. By analogy to mutation-maps and fixmaps, we want to be able to uniquely decompose any mapping $(u, v) = (x_1 x_2 \ldots x_n, y_1 y_2 \ldots y_m)$ in a suitably restricted functional string relation into a set of unchanged factors ('fix-factors') and a set of changed factors such that the whole mapping (u, v) can be straightforwardly reconstructed from these two sets. Equipped with this information, we can calculate and compare for any composition of string functions $g \circ f$ and input string w each of

$$\vec{\mu}(f, w) \qquad \vec{\mu}(g, w) \qquad \vec{\mu}(g \circ f, w)$$

[26] Every substring of a string w is a *factor* of w.

Echoing the more general definition, then, μ-conserving compositions of string functions will be defined as all and only those compositions where the μ-factors of the composition are identical (for all input words) to the union of the μ-factors of the two component functions considered in isolation. Equivalently, μ-conserving compositions can also be defined as those whose non-vacuous changes are always equivalent to those produced by simultaneous application, as defined shortly.

For simplicity of presentation, we restrict our operationalization of μ-factors to the context of substitutions and transpositions in length-preserving functions, though we expect that approaches based on bimachines (cf. (26b) in §5.1) or model-theoretic approaches to string functions would yield a characterization of 'what information is changed, where, and how' that is general enough to both cover non-length-preserving functions and a more linguistically relevant set of atomic string edit operations (viz. insertion, deletion, and copying).

To define the μ-factors of a string function f, we require here that the (length-preserving) string function f have a characterization as a transducer τ that is subsequential, synchronous (letter-to-letter), and equipped with origin semantics. The only reason for these restrictions is to ensure a well-behaved correspondence relation between each output symbol and an input symbol. Synchronous transducers are distinguished from asynchronous transducers by their incremental output function. A synchronous transducer maps each input symbol to (at most) a single output symbol, whereas the output function of asynchronous transducers can map each input symbol to a string of multiple output symbols. A transducer τ with origin semantics applied to an input string $w = x_1x_2 \dots x_j \dots x_n$ not only returns an output string $\tau(w) = w' = y_1y_2 \dots y_i \dots y_m$ but is also associated with an *origin* function $\mathfrak{o}$ that maps information about each position i in the output to information about what position $\mathfrak{o}(i) = j$ was rewritten in the input to produce the output symbol y_i at that output position (Bojánczyk 2014). This information tremendously simplifies bookkeeping and exposition, though we do not expect it to be necessary for defining μ-interaction between string-to-string functions, and it is plausible that because origin semantics-based bookkeeping results in very fine-grained information about μ-factors, a composition might define the same extensional string-to-string transductions as simultaneous application, but differ in μ-factors.

For $w \in \Sigma^*$ such that $w = x_1x_2 \dots x_j \dots x_n$, $f : \Sigma^* \to \Delta^*$, and $f(w) = w' = y_1y_2 \dots y_i \dots y_m$, then, we define the μ-factors of w with respect to f as in Def. 21[27]

Definition 21. Given $f : X^* \to X^*$, the *μ-factors* of w with respect to f are given by

$$\overrightarrow{\mu}(f, w) = \{(\mathfrak{o}(i), x_{\mathfrak{o}(i)}, i, y_i) | y_i \neq x_{\mathfrak{o}(i)} \vee i \neq \mathfrak{o}(i)\}$$

The fix-factors $\overrightarrow{fix}(f, w)$ are defined analogously.

The relevant analogue of priority union and simultaneous application of two functions f, g to some input string w operates on the μ-factors and fix-factors of pairs of concrete mappings[28]

$$(x_1x_2 \dots x_n, f(x_1x_2 \dots x_n)) \qquad (x_1x_2 \dots x_n, g(x_1x_2 \dots x_n))$$

i.e. through abuse of notation, when the string-analogue of the condition from Def. 11 holds —

$$\overrightarrow{\mu}(f, w) \leftthreetimes \overrightarrow{\mu}(g, w) = \overrightarrow{\mu}(g, w) \leftthreetimes \overrightarrow{\mu}(f, w)$$

— we will use $g \mid f(w)$ to denote the elementwise string analogue of simultaneous application given in Def. 22

[27] Note that the first disjunct of the condition ('$y_i \neq x_{\mathfrak{o}(i)}$') will match substitutions and the second ('$i \neq \mathfrak{o}(i)$') will match transpositions (metathesis), but non-length preserving operations like fusion, copying, insertion, or deletion are out of scope for this simplistic model of tracking changes to input symbols and origins of output symbols presented here.

[28] In contrast, previous applications of priority union to string relations in linguistics (Kaplan and Newman 1997, Karttunen 1998) only incidentally involved strings: the priority union of two string relations there is exactly the priority union of any other kind of relation, but the domain and codomain elements just happen to be strings.

Definition 22. Given $f, g : X^* \to X^*$ that meet the relevant conditions for simultaneous application, let $g \mid f(w)$ denote the result of taking

$$(\vec{\mu}(g, w) \mathbin{\leftarrow\!\!\rho} \vec{\mu}(f, w)) \mathbin{\leftarrow\!\!\rho} (\overrightarrow{fix}(g, w) \mathbin{\leftarrow\!\!\rho} \overrightarrow{fix}(f, w))$$

reconstructing the string-to-string input-output map, and selecting just the output string.[29]

To see how this plays out, consider the Finnish- and Lamba-based examples (1-2) of pushing (feeding) and pulling (bleeding) on environment:

(22) Raising pushes/feeds assibiliation in Finnish

a. Raising (f): $\mathrm{e} \longrightarrow \mathrm{i} \;/\; \_\_ \#$ $\qquad /\mathrm{kiele}/ \underset{f}{\longrightarrow} [\mathrm{kieli}]$ ‘tongue’

b. Assibilation (g): $\mathrm{t} \longrightarrow \mathrm{s} \;/\; \_\_ \mathrm{i}$ $\qquad /\mathrm{tilati}/ \underset{g}{\longrightarrow} [\mathrm{tilasi}]$ ‘ordered’

c. $g \circ f$: $/\mathrm{vete}/ \underset{f}{\longrightarrow} |\mathrm{veti}| \underset{g}{\longrightarrow} [\mathrm{vesi}]$ $\qquad (f \circ g$: $/\mathrm{vete}/ \underset{g}{\longrightarrow} |\mathrm{vete}| \underset{f}{\longrightarrow} *[\mathrm{veti}])$

d. μ-factors:

$$\vec{\mu}(f, /\mathrm{vete}/) = \{(4, \mathrm{e}, 4, \mathrm{i})\} \qquad \vec{\mu}(g, /\mathrm{vete}/) = \varnothing$$
$$\vec{\mu}(g \circ f, /\mathrm{vete}/) = \{(4, \mathrm{e}, 4, \mathrm{i}), (3, \mathrm{t}, s, \mathrm{s})\} \qquad \vec{\mu}(f \circ g, /\mathrm{vete}/) = \{(4, \mathrm{e}, 4, \mathrm{i})\}$$
$$\vec{\mu}(g \mid f, /\mathrm{vete}/) = \{(4, \mathrm{e}, 4, \mathrm{i})\} \qquad \vec{\mu}(f \mid g, /\mathrm{vete}/) = \{(4, \mathrm{e}, 4, \mathrm{i})\}$$

(23) Lowering pulls/bleeds palatalization in Lamba

a. Lowering (f): $\mathrm{V} \longrightarrow [-\mathrm{high}] \;/\; \begin{bmatrix} -\mathrm{high} \\ -\mathrm{low} \end{bmatrix} \mathrm{C}_0 \_\_$ $\qquad /\mathrm{ponika}/ \underset{f}{\longrightarrow} [\mathrm{poneka}]$ ‘it falls’

b. Palatalization (g): $\mathrm{s} \longrightarrow \mathrm{ʃ} \;/\; \_\_ \mathrm{i}$ $\qquad /\mathrm{fisika}/ \underset{g}{\longrightarrow} [\mathrm{fiʃika}]$ ‘it hides’

c. $g \circ f$: $/\mathrm{kosika}/ \underset{f}{\longrightarrow} |\mathrm{koseka}| \underset{g}{\longrightarrow} [\mathrm{koseka}]$ $\qquad (f \circ g$: $/\mathrm{kosika}/ \underset{g}{\longrightarrow} |\mathrm{koʃika}| \underset{f}{\longrightarrow} *[\mathrm{koʃeka}])$

d. μ-factors:

$$\vec{\mu}(f, /\mathrm{kosika}/) = \{(4, \mathrm{i}, 4, \mathrm{e})\} \qquad \vec{\mu}(g, /\mathrm{kosika}/) = \{(3, \mathrm{s}, 3, \mathrm{ʃ})\}$$
$$\vec{\mu}(g \circ f, /\mathrm{kosika}/) = \{(4, \mathrm{i}, 4, \mathrm{e})\} \qquad \vec{\mu}(f \circ g, /\mathrm{kosika}/) = \{(4, \mathrm{i}, 4, \mathrm{e}), (3, \mathrm{s}, 3, \mathrm{ʃ})\}$$
$$\vec{\mu}(g \mid f, /\mathrm{kosika}/) = \{(4, \mathrm{i}, 4, \mathrm{e}), (3, \mathrm{s}, 3, \mathrm{ʃ})\} \qquad \vec{\mu}(f \mid g, /\mathrm{kosika}/) = \{(4, \mathrm{i}, 4, \mathrm{e}), (3, \mathrm{s}, 3, \mathrm{ʃ})\}$$

To showcase what happens in the setting of string functions, the example of pushing/feeding here is an on-environment interaction, and in such a case instead of seeing a *loss* of a mutation map of f that is replaced by an associated mutation map $(x, g \circ f(x))$ — as in §4.1 — we see only the apperance of a *novel* mutation map in $\vec{\mu}(g \circ f)$ relative to $\vec{\mu}(g) \cup \vec{\mu}(f)$. The reason for this is that in the general context of partial functions on discrete sets, pushing/feeding (like all other interaction) happens ‘on-focus’ — because there is no other possibility. Revised definitions of pushing, pulling, and shifting interactions for string functions follow:

Definition 23. *f pushes g* at x_i in w iff the μ-factors of w with respect to $g \circ f$ are either missing a μ-factor of w with respect to f, or they contain a novel μ-factor not found in the μ-factors of w with respect to either f or g.

Definition 24. *f pulls g* at x_i in w iff the μ-factors of w with respect to $g \circ f$ are missing a μ-factor of w with respect to g.

Definition 25. *f shifts g* at x_i in w iff f both pushes and pulls g at x_i.

[29] We overload $\mid$ in this way and omit explicit description of the reconstruction process because we do not believe there to be anything interesting or unclear about this process in the limited setting treated here, and because we expect future e.g. model-theoretic or bimachine-based treatments of μ-interaction in string functions to be less impressionistic.

Accordingly, we offer a tentative definition of a μ-conserving composition of string functions:

Definition 26. A composition $g \circ f$ of functions $f : X^* \to Y^*$ and $g : Y^* \to Z^*$ such that $X \subseteq Y$ and such that both f and g are length-preserving and definable by transducers that are subsequential, synchronous, and equipped with origin semantics are μ-CONSERVING iff $\forall w \in X^*, \vec{\mu}(g \circ f, w) = \vec{\mu}(g, w) \cup \vec{\mu}(f, w)$ iff $\forall w \in X^*, g \circ f(w) = g \mid f(w)$.

As an alternative to the definition of weakly deterministic regular functions offered by Heinz and Lai (2013), then, we propose the following (repeated from Def. 4):

Definition 27. A regular function $\tau : X^* \to Z^*$ is WEAKLY DETERMINISTIC iff there exist two subsequential functions $I : X^* \to Y^*$ such that $X \subseteq Y$ and $O : Y^* \to Z^*$ such that O is either left- or right-subsequential, I is the other, $\tau = O \circ I$ and $O \circ I$ is a μ-conserving composition.

This definition excludes any composition that uses extra intermediate symbols, length-increasing codes, phonotactic codes (of the sort discussed in §5), or any other means by which the behavior of the inner and outer functions in a composition can guide, interfere with, or interact with each other.

4.3 Ordering, interaction, and subregular classification

We close this section with a brief discussion of two closely related points yet to be made about the relationship of ordering, interaction, and subregular classification.

The derivational nature of the composed functions used throughout this paper does not entail that all phonological patterns analyzed with rule ordering are NDRFs. Most relevantly, Chandlee et al. (2018) demonstrate that some opaque process interactions are analyzable using INPUT STRICTLY LOCAL (ISL) functions, a highly restricted subset of the subsequential functions. Although the better-known examples of opaque process interaction (counterfeeding and counterbleeding) are known to be analyzable in terms of simultaneous application (Kenstowicz and Kisseberth 1979, Joshi and Kiparsky 1979, 2006, Kiparsky 2015), the lesser-known case of self-destructive feeding (Baković 2007) requires ordering (Baković and Blumenfeld 2017, 2020), as do well-known examples of transparent process interaction (i.e. feeding and bleeding).[30] All of the examples of opaque process interaction considered by Chandlee et al. (2018) are shown to be ISL, by 'waiting' an *a priori* known number of states. Crucially, however, all of the examples in question involve non-iterative processes and they are thus bounded in such a way that a waiting strategy is feasible. By contrast, as Jardine (2016) notes, there is no *a priori* known upper bound on the number of 'waiting' states necessary to successfully map inputs to their corresponding outputs in the case of unbounded circumambient processes, rendering a waiting strategy inapplicable to such cases.

Formally speaking, to determine the complexity of the composition of two specific interacting phonological mappings (whether described in terms of ordered rules or otherwise), the relevant algebraic question is what class the composition of those two particular mappings is in. More generally, this question can be framed or explored as: 'which subclasses of the regular relations are *closed under composition*?' A summary of existing closure results about classes mentioned in this article is provided in (24).

(24) Existing closure results

a. Regular functions are closed under composition. (Mohri 1997)

b. The set containing both left- and right-subsequential functions is not closed under composition; rather, the closure of that class is the set of regular functions. (Elgot and Mezei 1965)

c. However, when considered separately, each of the left- and the right-subsequential functions are closed under composition. (Mohri 1997)

[30] Because they are not equivalent to simultaneous application, these ordered interactions *do* necessarily involve μ-interaction— but this is of course not a sufficient condition for NDRF-hood.

d. A restricted subclass of the input strictly local functions are closed under composition. (Chandlee et al. 2018)

Where does the class of WDRFs fall within these results? First, recall that we define WDRFs as μ-conserving compositions of subsequential functions. While a more rigorous exploration of the properties of classes of μ-conserving string functions — subsequential or not — lies beyond the scope of the present work, we expect future work to show that the WDRFs are not in general closed under composition. The intuition behind this expectation is that the composition of a μ-conserving composition of functions with another μ-conserving composition is not, in general, μ-conserving: the larger a function's mutation set, the more likely it is that any other function will have conflicting mappings that prevent the possibility of simultaneous application, and therefore of there being a μ-conserving composition order.

This mention of closure under composition might suggest to the astute reader several questions for future work on subregular function classes and the computational analysis of phonological processes. Discussions of simultaneous application have historically pitted it against ordering (Kenstowicz and Kisseberth 1979; recall fn. 3), a comparison in which ordering 'wins' because of the existence of well-evidenced process interactions that simultaneous application cannot model (e.g. feeding and bleeding). Our results suggest instead that priority union (and by extension simultaneous application) be considered as a less expressive alternative to composition and/or as an operation freely combinable with composition, and that computational function classes be examined for closure under priority union or the algebraic structure defined by the combination of priority union and composition (see Oakden 2020 for some current work exploring this general direction).

5 Discussion

This work presents a novel formal definition of weak determinism, centering interaction as the core property that distinguishes (weakly) deterministic maps from nondeterministic maps within the computational complexity hierarchy of (sub)regular functions illustrated in Fig. 1. Centering interaction as the arbiter of this boundary, rather than mark-up (Heinz and Lai 2013, McCollum et al. 2018, O'Hara and Smith 2019, Smith and O'Hara 2019, Lamont et al. 2019) or lookahead (Jardine 2016, McCollum et al. 2020a), strengthens the status of weakly deterministic maps as a computational class of linguistic interest because it is a computationally and linguistically more natural concept than those centered in previous formal definitions.

5.1 μ-interaction subsumes mark-up

Heinz and Lai's (2013) vision of weak determinism as a restriction on regular functions implemented through constraints on function composition remains at the core of the reformulation presented in §4. WDRFs remain a class of functions that "can be decomposed into [an inner] subsequential and [an outer, contradirectional] subsequential function *without the [inner] function marking up its output in any special way*"; merely the piecewise nature of initial attempts to formally define "any special way" have been exchanged for the unifying notion of μ-interaction between the inner and outer functions.

Heinz and Lai's formulation of weak determinism identified two possible mark-up strategies and accordingly stipulated that WDRFs be alphabet- and length-preserving. In identifying and prohibiting strategies case by case, the focus on mark-up in this definition of weak determinism inspired further work identifying novel variations on these two mark-up strategies. In these works, an even more clever, third mark-up strategy was identified, a form we call PHONOTACTIC CODING.[31]

In this strategy, substrings of the input alphabet which would otherwise be phonotactically illicit in the language in question are used to mark up the intermediate form. More specifically, given the input alphabet

[31] The works cited call this "using phonotactically-illicit intermediate symbol sequences as 'cheat codes' for extra symbols," (McCollum et al. 2018), "(intermediate) substring markup" (O'Hara and Smith 2019), "using predictable substrings as markup" (Lamont 2018), and "information [that] is smuggled into an intermediate representation using predictable substrings of the symbols already in a language's alphabet" (Smith and O'Hara 2019). Graf (2016) makes a related proposal, strategically identifying auxiliary symbols from the input alphabet ("green flags") that can be used as mark-up.

$\mathcal{A}$ and a set $\mathcal{S}$ of non-occurring, phonotactically illicit substrings over $\mathcal{A}$, the inner function may change some substring of the input to a substring of the same length drawn from $\mathcal{S}$, conditioned on some motivating property of the input string. If this change is made at a location in the string that the outer function would encounter before it would encounter the motivating property of the input string, then the output of the outer function at that point may be conditioned on information possibly an unbounded distance from that decision point, rendering the composition as a whole nondeterministic. For this to occur, it is only required that each otherwise-illicit substring introduced by the inner function be unambiguously interpretable by the outer function (i.e. from the opposite direction) as having been introduced because the inner function encountered particular conditions somewhere in the distal end of the string. The result is alphabet-preserving because the set $\mathcal{S}$ is defined over the input alphabet $\mathcal{A}$, and it is length-preserving because substrings from $\mathcal{S}$ only replace substrings of the same length. Fig. 22 illustrates phonotactic coding with the same schematic unbounded tonal plateauing (UTP) example shown in Fig. 4 (§2.2), but this time with the inner function marking up its output with the phonotactic code suggested by Lamont et al. (2019, p. 197).

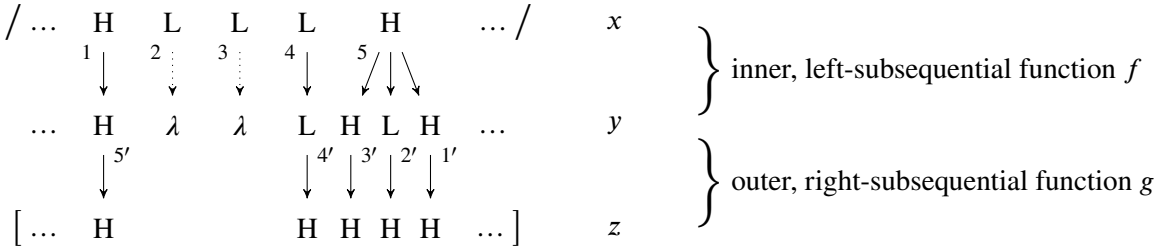

Figure 22 Schematic example of unbounded tonal plateauing, with phonotactic coding

The phonotactic code in this case is the final HLH substring, written in one fell swoop by the inner function f upon encountering an H preceded by two or more Ls which are themselves preceded by an H. More precisely, after reading and faithfully writing the first H in its input string x (arrow 1), the inner function f reads an L. At this point, nothing is written — hence the dotted arrow 2, and the empty symbol λ in the intermediate string y. If the next symbol to be read were an H, the 2-symbol sequence HH would be emitted: the first H taking the place of the previously unwritten L and the second being the faithful realization of the H just read. Lamont et al. (2019, p. 197) note that this HLH $\mapsto$ HHH mapping "model[s] UTP in a local context". This idiosyncratic treatment of plateaux of length 3 is crucial for this mark-up strategy to succeed.

The inner function f instead reads another L and once more writes nothing (arrow 3). All subsequent Ls are written faithfully as L (arrow 4). Finally, upon reading another H, f writes the sequence HLH (the trio of arrows labeled 5). This compensates for the two previous unwritten Ls, and it does so with the substring HLH which, when present as such in an input string, would have been written by the very same inner function as HHH (= "UTP in a local context"). HLH thus functions as a phonotactic code for the outer function g that the conditions for non-local UTP were found in the input string x and that the L of this HLH coded substring in y, and all subsequent Ls until another H is read, should surface as H in the ultimate output string z.

To be clear, the works cited above acknowledge that phonotactic coding is a form of mark-up (see fn. 31), and indeed that "all unbounded circumambient patterns require some sort of mark-up to be used in the application of a pair of subsequential mappings" (O'Hara and Smith 2019, p. 10).[32] Nevertheless, given that it is technically permissible by Heinz and Lai's (2013) formal definition of WDRFs, phonotactic coding has been claimed in these works to critically distinguish unbounded circumambient patterns that can be

[32] Lamont et al. (2019, p. 197) even go so far as to comment negatively on the result of phonotactic coding from the perspective of substantive and methodological assumptions within phonological theory: "Encoding instructions in intermediate representations is strikingly unphonological and is not intended as a plausible interpretation of the process."

described by WDRFs from those requiring more expressive classes of functions because the contradirectional decompositions of these latter functions crucially require alphabet- or length-increasing mark-up.

Perhaps most explicitly, O'Hara and Smith (2019, p. 6) claim that alphabet- or length-increasing mark-up of the kind illustrated in Fig. 4 of §2.2 is "more powerful" than phonotactic coding of the kind illustrated in Fig. 22 above. As evidence of this claim, they argue that the unattested set of "true sour grapes" patterns, defined in (25), can be modeled with novel symbols but not with phonotactic codes.[33] These patterns are intended to be distinguished from "false sour grapes" patterns, a set which includes attested UTP.

(25) An unbounded spreading pattern is an example of *true sour grapes* if, given $\Sigma^* = \{\text{T}, \text{B}, \text{U}\}$ (T = triggers, B = blockers, U = undergoers), and for all $m, n \in \mathbb{N}$,

	if left-to-right…	*if right-to-left…*
a.	$\text{T}^m\text{U}^n\text{B} \mapsto \text{T}^m\text{U}^n\text{B}$, and	$\text{BU}^n\text{T}^m \mapsto \text{BU}^n\text{T}^m$, and
b.	$\text{T}^m\text{U}^n \mapsto \text{T}^{m+n}$.	$\text{U}^n\text{T}^m \mapsto \text{T}^{n+m}$.

O'Hara and Smith (2019) and Smith and O'Hara (2019) show that no length- or alphabet-preserving decomposition can be devised to correctly describe a true sour grapes pattern, whereas such decompositions are possible for false sour grapes patterns (e.g. for UTP in Fig. 22). The key difference between UTP (*qua* false sour grapes) and true sour grapes is that UTP defines unbounded circumambient conditions for making a change to an input string (viz. spreading H), whereas true sour grapes defines unbounded circumambient conditions for *not* making a change to an input string. This key difference guarantees the existence of a phonotactic code for UTP, as follows. Upon encountering an H in a UTP pattern, the inner function will not spread it unless there is another H further down the line, in principle an unbounded distance away. HLH ↦ HHH is a bounded instance of the more general spreading condition ("UTP in the local context"); such maps can be performed by the inner function, freeing up HLH to function as a phonotactic code.

There is no such guarantee of a phonotactic code for true sour grapes. Upon encountering a trigger T, the inner function *will* spread it unless there is a blocker B further down the line, in principle an unbounded distance away. There are thus only bounded instances of the more general *non*-spreading condition, TUB, which map unchanged to TUB. Thus, no substring can be freed up to function as a phonotactic code. But it is important to keep in mind that all possible substrings of Σ^* are assumed to be equally likely in the toy language of (25): T is just as likely as B or U at every point in the string, leaving no substring of any length capable of both encoding the presence of a blocker elsewhere in the string while preserving the original identity of the substring at the location of the mark-up. While this toy example neatly demonstrates that phonotactic coding is a non-viable mark-up strategy for completely unrestricted input languages (free monoids), we argue that this is not a linguistically relevant result at the very least because human languages are not free monoids.[34] In other words, while a bounded instance of the more general non-spreading condition in cases of true sour grapes is not guaranteed, it is nevertheless highly probable if not fully guaranteed that *some* bounded, phonotactically-illicit substring can be recruited to do the necessary mark-up work (Graf 2016).

This brings us back to O'Hara and Smith's (2019) claim that alphabet- and length-increasing mark-up is "more powerful" than phonotactic coding. If "more powerful" is understood to mean "more *expressive*", this is in general false, as expressivity is usually defined and considered. In formal language theory, a function class picks out an extensionally-definable equivalence class of functions.[35] O'Hara and Smith (2019) argue that 'true' sour grapes can only be modeled with alphabet- and length-increasing codes while 'false' sour grapes can be modeled with *either* alphabet-/length-increasing codes *or* phonotactic codes. This is the central logic

[33] On the (un)attestedness of different types of sour grapes ('non-myopic') spreading patterns, see Wilson (2003, 2006), Walker (2010), Kimper (2012), Mascaró (2019), and McCollum et al. (2020b). See also the discussion to come in §5.2 below.

[34] In fact, almost all languages, human or otherwise, are not free monoids.

[35] At the very least, such a function class is expected to have multiple convergent characterizations, including an extensional, 'machine-independent' one, even if we don't currently know what those are.

of their claim that phonotactic-codes are not as expressive as alphabet- and length-increasing codes. However, the difference between the mappings of so-called 'true' and 'false' sour grapes lies in a restriction on the input language of both the inner and outer functions that is unconnected to the unbounded circumambience of the pattern. That the entropy of some languages is too high to support phonotactic coding strategies is really a statement about those languages, not a description of the expressivity of the machinery required to describe a function on those languages. The exchangability of novel intermediate symbols, length-increasing codes for such symbols, and phonotactic codes is apparent from the fact that they all serve the same purpose: creating information about one end of the input string at the other end of the intermediate string. It is the ability of each of these three strategies to accomplish precisely this goal that permits compositions of contradirectional subsequential functions to define NDRFs through any of them.

Beyond the temptation to ascribe computational significance to cosmetic differences between mark-up strategies, viewing weak determinism as a class defined by its resistance to description by a collection of disparate strategies is less parsimonious than the unified account provided by interaction — and it obscures the sense of 'intermediateness' that is characteristic of the class. Heinz and Lai (2013, p. 54) justify the framing of WDRFs as 'weakly' or intermediately deterministic by writing: "they are not necessarily deterministic, but they are 'more' deterministic than regular functions where Elgot and Mezei decomposition *requires* the intermediate mark-up" (emphasis in the original). A more substantive characterization is in (26).

(26) Informal description of deterministic, weakly deterministic, and non-deterministic regular functions

a. **Deterministic (subsequential)**: there exists a side of the string, for every index in all strings, for which the identity of the output element is determined by elements from that side.

b. **Weakly deterministic**: for every index in all strings, there exists a side of the string for which the identity of the output element is determined by elements from that side.[36]

c. **Non-deterministic**: for every index in all strings, both sides of the string may be used to determine the identity of the output element.

Thus, a WDRF is intermediate in expressivity because different output indices may require information from different sides of the form (unlike deterministic functions), but no single output index requires information from *both* sides of the form (unlike non-deterministic functions). In this sense, WDRFs are more deterministic than non-deterministic functions but less deterministic than deterministic functions.

This intuition is preserved in thinking of determinism in terms of μ-interaction. Deterministic functions are regular functions that can be defined as μ-conserving compositions of equi-directional (as opposed to contra-directional) subsequential functions. Weakly deterministic functions are regular functions that can be defined as a μ-conserving compositions of contra-directional subsequential functions. And finally, non-deterministic functions are regular functions that can be defined as a μ-interacting compositions of contra-directional subsequential functions. In this way, interaction captures the essense of what makes weak determinism not quite deterministic; defining WDRFs as the class of EM decompositions for which the inner and outer functions do not interact is computationally more sound than attempts to exhaustively list imaginable forms of mark-up.

5.2 μ-interaction is machine-independent

Previous work has appealed to lookahead as a way to understand the difference between NDRFs and WDRFs (Jardine 2016, McCollum et al. 2020a). Wait states in subsequential transducers have been used to implement bounded lookahead, but wait states are not a machine-independent description of bounded lookahead. Unbounded lookahead, meanwhile, is only indirectly identifiable via subregular classification. In neither case is it clear how to extract a detailed description of lookahead in a particular function or how lookahead affects the behavior of a particular function. A microscopic theory of lookahead might, for instance, describe for

[36] In ongoing work we use this characterization as a starting point for a bimachine-based definition of the WDRFs.

each ambiguity requiring lookahead, the set of 'ambiguous' prefixes (i.e. prefixes that depend on lookahead information to determine output), the set of possible outputs for each ambiguous prefix, and the information that determines which output is correct for each ambiguous prefix. Neither bounded lookahead nor unbounded lookahead have theories with these features, to our knowledge; they do not have formal descriptions that are direct, machine-independent and fine-grained — unlike μ-interaction.

Viewing μ-interaction as the arbiter of the boundary between weak determinism and non-determinism is preferable on computational grounds and is better for formal theory development. First, an analysis of a composition based on μ-factors and μ-interaction offers a much more detailed description of function behavior, including a description of which input strings and what input elements involve interaction (and therefore lookahead). The second reason for preferring a μ-interaction-based differentiation of the WDRFs from the NDRFs over one based on lookahead is that μ-interaction is *necessary* for unbounded lookahead, and — as discussed in more detail in §5.3 — even when unbounded lookahead is technically occurring, it may not be as salient or useful of a description or property to check for as μ-interaction.

Finally, the definitions of μ-interaction given in §4 rely on input-output maps and an origin function. This is in general preferable to transducer-dependent concepts like lookahead, but there are also caveats. In particular, to use μ-interaction as a diagnostic for non-determinism requires a pattern be decomposed into two contradirectional subsequential functions. For bidirectional harmony patterns like those addressed in this paper, the decomposition process seems so intuitive as to be practically predetermined: a left-subsequential function takes care of any rightward feature spreading, and a right-subsequential function attends to any leftward feature spreading. However, the apparent naturalness of these decompositions should not be taken for granted, and in general it depends on the prudence of the linguist-analyst. Therefore, assessing (weak) determinism on the basis of μ-interaction for complex patterns that do not appear to be "naturally" decomposible may prove challenging.

5.3 μ-interaction is linguistically sound

Despite the foregoing caveats, μ-interaction is linguistically as well as computationally sound. To begin with, μ-interaction meshes well with existing concepts in linguistics. Most obviously — and as has already been directly noted in many of the preceding pages — it has a great deal in common with the interactions of rules due to serial ordering in the *SPE* tradition of Chomsky and Halle (1968).[37] Zooming out somewhat, there is also much in common between μ-interaction and interactions of constraints due to ranking and weighting in the traditions of OT (Prince and Smolensky 2004) and HG (Legendre et al. 1990). Any insights gained from the study of any of these forms of interaction are thus highly likely to inform the study of the others — as they indeed already have; see for example Norton (2003), Baković (2007, 2011, 2013), and references therein for relevant formal comparisons of the descriptive and explanatory similarities and differences between these serial rule-based and parallel constraint-based theoretical frameworks.[38]

Bringing our focus back to the more immediate concerns of this article, μ-interaction unites all cases of unbounded circumambience, and is not limited to those cases that satisfy arbitrary criteria resulting from particular decisions about what counts as mark-up or about what it means for a function to 'look ahead' in the string. The Turkana harmony pattern is an instructive case in point. There is clear linguistic evidence for bidirectional spreading from [+ATR] roots and dominant [+ATR] suffixes; for low vowels undergoing rightward but not leftward spreading of [+ATR]; and for leftward spreading from dominant [−ATR] suffixes. Most significantly, rightward spreading of [+ATR] interacts with leftward spreading of [−ATR], the latter

[37] See also Chafe (1968), Kiparsky (1968), Wang (1969), Newton (1971), Koutsoudas et al. (1974), and Kenstowicz and Kisseberth (1977, 1979), among many, many others. For a relatively recent review of relevant concepts and terms, see Baković (2011); for a more thorough exploration of the typology of rule ordering interactions, see Baković and Blumenfeld (2018, 2019, 2020).

[38] The focus in the text on the extremes of this continuum is not meant to ignore the wealth of work aiming for some kind of balance between serialism and parallelism, from Stratal OT (Kiparsky 2015; Bermúdez-Otero, in prep.) to Harmonic Serialism (McCarthy 2010) to Serial HG (Pater 2012).

in some sense has the 'final say' on the realization of vowels in a string, and each direction of spreading corresponds to one of the contradirectional subsequential functions of an EM decomposition. The fact that the surface realization of an underlying low vowel depends on unbounded circumambient information is thus effectively disguised by the independent motivation for each of the interacting spreading processes.

This *feels* unlike unbounded tonal plateauing and other similar, non-myopic patterns (e.g. Tutrugbu ATR harmony; McCollum et al. 2020a), where the linguistically motivated spreading process only corresponds to one of the two functions of the EM decomposition, albeit roughly. The other function must be more indirectly inferred from the distal information that makes the linguistically motivated spreading process non-deterministic. This is what makes some obvious form of mark-up necessary in those cases. The total mapping in Turkana is unbounded circumambient and involves lookahead, but precisely because non-myopia is not analytically salient, μ-interaction remains a linguistically useful tool for identifying that the subregular complexity of Turkana is the same as other more obviously non-myopic unbounded circumambient maps.

6 Conclusion

Phonologists have long concerned themselves with the ways that multiples rules may interact. This article provides a formal definition of function interaction that differentiates properly weakly deterministic regular functions (WDRFs) from the strictly more expressive class of non-deterministic regular functions (NDRFs). Formal comparison of the analyses of the ATR harmony patterns of Maasai and Turkana demonstrates the key difference between weakly deterministic and non-deterministic regular functions: the contradirectional subsequential functions comprising a WDRF do not interact and may thus apply simultaneously, while those comprising an NDRF do interact and thus may require ordering. By framing the distinction between weak determinism and non-determinism in this way, our proposal subsumes the various strategies employed in previous work to delimit the subregular class of WDRFs, thereby offering a single, unified way to characterize the expressivity difference between unbounded circumambient and other bidirectional patterns.

References

Aksënova, A., Rawski, J., Graf, T., and Heinz, J. (2020). The Computational Power of Harmony. To appear in H. van der Hulst (ed.), *Oxford Handbook of Vowel Harmony*, Oxford University Press.

Alur, R., D'Antoni, L., and Raghothaman, M. (2015). DReX: A Declarative Language for Efficiently Evaluating Regular String Transformations. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL '15*, pages 125–137.

Alur, R., Freilich, A., and Raghothaman, M. (2014). Regular Combinators for String Transformations. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic, CSL 2014 and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2014*.

Anderson, S. R. (1974). *The Organization of Phonology*. Academic Press.

Baković, E. (2000). *Harmony, Dominance and Control*. PhD thesis, Rutgers, The State University of New Jersey.

Baković, E. (2002). Vowel harmony and cyclicity in Eastern Nilotic. In Chang, C., Houser, M. J., Kim, Y., Mortensen, D., Park-Doob, M., and Toosarvandani, M., editors, *Proceedings of the 27th Annual Meeting of the Berkeley Linguistics Society*, pages 1–12, Berkeley, CA. Berkeley Linguistics Society.

Baković, E. (2007). A revised typology of opaque generalisations. *Phonology*, 24:217–259.

Baković, E. (2011). Opacity and ordering. In Goldsmith, J. A., Riggle, J., and Yu, A. C. L., editors, *The Handbook of Phonological Theory*, pages 40–67. Wiley Blackwell, Malden, Mass., 2nd edition.

Baković, E. (2013). *Blocking and Complementarity in Phonological Theory*. Equinox, London.

Baković, E. and Blumenfeld, L. (2017). A set-theoretic typology of phonological map interaction. Poster presented at the 2017 Annual Meeting on Phonology, NYU.

Baković, E. and Blumenfeld, L. (2018). Overapplication conversion. In Bennett, R., Angeles, A., Brasoveanu, A., Buckley, D., Kalivoda, N., Kawahara, S., McGuire, G., and Padgett, J., editors, *Hana-bana: A Festschrift for Junko Itô and Armin Mester*. Santa Cruz, CA.

Baković, E. and Blumenfeld, L. (2019). Rule interaction conversion operations. *Loquens*, 6(2):e062.

Baković, E. and Blumenfeld, L. (2020). A typology of map interactions. Manuscript in preparation, UC San Diego and Carleton University.

Bermúdez-Otero, R. (n.d.). *Stratal Optimality Theory*. Oxford University Press, Oxford. In preparation. Pending publication, see the collection of writings available at http://www.bermudez-otero.com/Stratal_Optimality_Theory.htm.

Bojańczyk, M. (2018). Polyregular functions. *arXiv*.

Bojańczyk, M., Daviaud, L., and Krishna, S. (2018). Regular and First-order List Functions.

Bojánczyk, M. l. (2014). Transducers with origin information. In *ICALP 2014: Automata, Languages, and Programming*, pages 26–37.

Chafe, W. L. (1968). The ordering of phonological rules. *International Journal of American Linguistics*, 34:115–136.

Chandlee, J. (2014). *Strictly local phonological processes*. PhD thesis, University of Delaware.

Chandlee, J., Eyraud, R., and Heinz, J. (2015). Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*, pages 112–125.

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

Chandlee, J., Heinz, J., and Jardine, A. (2018). Input Strictly Local opaque maps. *Phonology*, 35(2):171–205.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper & Row.

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

Dimmendaal, G. J. (1983). *The Turkana Language*. Foris, Dordrecht.

Doke, C. M. (1938). *Textbook of Lamba Grammar*. Witswatersrand University Press, Johannesburg.

Elgot, C. C. and Mezei, J. E. (1965). On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68.

Graf, T. (2016). Weak determinism is not too weak: A reply to Jardine (2016). Unpublished ms., Stony Brook University.

Guion, S. G., Post, M. W., and Payne, D. L. (2004). Phonetic correlates of tongue root vowel contrasts in maa. *Journal of Phonetics*, 32(4):517–542.

Hall, B. L., Hall, R. M. R., and Pam, M. D. (1973). African vowel harmony systems from the vantage point of kalenjin. *Afrika und Übersee: Sprachen, Kulturen*, 57(4):241–267.

Hao, Y. and Andersson, S. (2019). Unbounded Stress in Subregular Phonology. In *SIGMORPHON 16*, pages 135–143. ACL.

Heinz, J. (2011a). Computational phonology–Part I: Foundations. *Language and Linguistics Compass*, 5(4):140–152.

Heinz, J. (2011b). Computational phonology–Part II: Grammars, learning, and the future. *Language and Linguistics Compass*, 5(4):153–168.

Heinz, J. (2018). The computational nature of phonological generalizations. *Phonological Typology*, 23:126–195.

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

Heinz, J. and Lai, R. (2013). Vowel harmony and subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 52–63.

Hualde, J. I. (1991). *Basque Phonology*. Routledge, London.

Jardine, A. (2016). Computationally, tone is different. *Phonology*, 33(2):247–283.

Jardine, A. (2017). On the logical complexity of autosegmental representations. In *Proceedings of the 15th Meeting on the Mathematics of Language*, pages 22–35.

Jardine, A. (2019). The expressivity of autosegmental grammars. *Journal of Logic, Language and Information*, 28(1):9–54.

Jardine, A., Danis, N., and Iacoponi, L. (2020). A formal investigation of q-theory in comparison to autosegmental representations. *Linguistic Inquiry*, pages 1–25.

Joshi, S. D. and Kiparsky, P. (1979). Siddha and asiddha in Paninian phonology. In Dinnsen, D. A., editor, *Current Approaches to Phonological Theory*, pages 223–250. Indiana University Press, Bloomington, IN.

Joshi, S. D. and Kiparsky, P. (2006). The extended *siddha*-principle. *Annals of the Bhandarkar Oriental Research Institute 2005*, pages 1–26.

Kaplan, R. (1987). Three seductions of computational psycholinguistics. In Whitelock, P., Wood, M. M., Somers, H. L., Johnson, R., and Bennett, P., editors, *Linguistic Theory and Computer Applications*, pages 149–181. Academic Press.

Kaplan, R. and Newman, P. (1997). Lexical resource reconciliation in the xerox linguistic environment. In *ACL/EACL'98 Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 54–61.

Karttunen, L. (1998). The Proper Treatment of Optimality in Computational Phonology. *arXiv*, page 12.

Kenstowicz, M. and Kisseberth, C. (1971). Unmarked bleeding orders. *Studies in the Linguistic Sciences*, 1:8–28. Reprinted in Charles Kisseberth (ed.), *Studies in Generative Phonology*, 1–12. Champaign, IL: Linguistic Research, 1973.

Kenstowicz, M. and Kisseberth, C. (1977). *Topics in phonological theory*. Academic Press.

Kenstowicz, M. and Kisseberth, C. (1979). *Generative phonology: Description and theory*. Academic Press.

Kimper, W. A. (2012). Harmony is Myopic: Reply to Walker (2010). *Linguistic Inquiry*, 43(2):301–309.

Kiparsky, P. (1965). *Phonological change*. PhD thesis, Massachusetts Institute of Technology.

Kiparsky, P. (1968). Linguistic universals and linguistic theory. In Bach, E. and Harms, R., editors, *Universals in linguistic theory*. Holt, New York.

Kiparsky, P. (1973). How abstract is phonology? In Fujimura, O., editor, *Three Dimensions of Linguistic Theory*, chapter 1 of Part I, *Phonological Representations*, pages 5–56. TEC, Tokyo. Reprinted in *Explanation in Phonology*, pp. 119-163, 1982.

Kiparsky, P. (1993). Blocking in nonderived environments. In Hargus, S. and Kaisse, E. M., editors, *Studies in Lexical Phonology*, volume 4 of *Phonetics and Phonology*, pages 277–313. Academic Press, San Diego, CA.

Kiparsky, P. (2015). Stratal OT: A Synopsis and FAQs. In Hsiao, Y. E. and Wee, L.-H., editors, *Capturing Phonological Shades Within and Across Languages*, pages 2–44. Cambridge Scholars Library, Newcastle upon Tyne.

Koser, N. and Jardine, A. (2020). Stress assignment and subsequentiality. In *Proceedings of the 2019 Annual Meeting on Phonology*.

Koutsoudas, A., Sanders, G., and Noll, C. (1974). On the application of phonological rules. *Language*, 50:1–28.

Lamont, A. (2018). Majority rule in harmonic serialism. *Poster presented at AMP 2018, San Diego*.

Lamont, A., O'Hara, C., and Smith, C. (2019). Weakly deterministic transformations are subregular. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 196–205.

Legendre, G., Miyata, Y., and Smolensky, P. (1990). Harmonic Grammar – A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 388–395. Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

Magri, G. (2018). Idempotency, output-drivenness and the faithfulness triangle inequality: some consequences of mccarthy's (2003) categoricity generalization. *Journal of Logic, Language and Information*, 27(1):1–60.

Mascaró, J. (2019). On the lack of evidence for nonmyopic harmony. *Linguistic Inquiry*, 50:862–872.

McCarthy, J. J. (1999). Sympathy and phonological opacity. *Phonology*, 16(3):331–399.

McCarthy, J. J. (2010). An introduction to harmonic serialism. *Language and Linguistics Compass*, 4(10):1001–1018.

McCollum, A. G., Baković, E., Mai, A., and Meinhardt, E. (2018). The expressivity of segmental phonology and the definition of weak determinism. *Unpublished ms., University of California San Diego*.

McCollum, A. G., Baković, E., Mai, A., and Meinhardt, E. (2020a). Unbounded circumambient patterns in segmental phonology. *Phonology*, 37:215–255.

McCollum, A. G., Baković, E., Mai, A., and Meinhardt, E. (2020b). On the existence of non-myopic harmony. Unpublished manuscript, under review.

McCrary, K. (2001). Low Harmony, ATR Harmony and Verbal Morphology in Kisongo Maasai. *Papers in Phonology*, 5:179–206. UCLA Working Papers in Linguistics, no. 7.

Meinhardt, E., Mai, A., Baković, E., and McCollum, A. G. (2020). Questioning to resolve transduction problems. Poster presented at the Workshop on Formal Language Theory, 3rd Meeting of the Society for Computation in Linguistics (SCiL 2020), New Orleans.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Newton, B. E. (1971). Ordering paradoxes in phonology. *Journal of Linguistics*, 7:31–53.

Norton, R. J. (2003). *Derivational phonology and optimality phonology: formal comparison and synthesis*. PhD thesis, Rutgers, The State University of New Jersey.

Noske, M. (1996). [ATR] harmony in Turkana. *Studies in African Linguistics*, 25(1):61–99.

Noske, M. (2001). [ATR] Harmony in Turkana: A Case of Faith Suffix » Faith Root. *Natural Language & Linguistic Theory*, 18(4):771–812.

Oakden, C. (2020). Southern Min tone circles: Parallel satisfaction using recursive schemes. Poster presented at the Annual Meeting on Phonology.

O'Hara, C. and Smith, C. (2019). Computational complexity and sour-grapes-like patterns. In *Proceedings of the Annual Meeting on Phonology*, volume 7.

Pater, J. (2012). Serial Harmonic Grammar and Berber syllabification. In Borowsky, T., Kawahara, S., Shinya, T., and Sugahara, M., editors, *Prosody Matters: Essays in Honor of Elisabeth O. Selkirk*, pages 43–72. Equinox, London.

Prince, A. and Smolensky, P. (2004). *Optimality Theory: Constraint interaction in generative grammar*. Wiley-Blackwell.

Quinn-Wriedt, L. T. (2013). *Vowel Harmony in Maasai*. PhD thesis, University of Iowa.

Smith, C. and O'Hara, C. (2019). Formal characterizations of true and false sour grapes. volume 2, pages 338–341.

Stanley, R. (1967). Redundancy Rules in Phonology. *Language*, 43(2):393–436.

Tesar, B. (2014). *Output-driven phonology: Theory and learning*. Number 139. Cambridge University Press.

Tucker, A. N. and Mpaayei, J. T. O. (1955). *A Maasai grammar, with vocabulary*. Longmans, Green.

van Rooij, R. (2003). Questioning to resolve decision problems. *Linguistics and Philosophy*, 26(6):727–763.

Walker, R. (2010). Nonmyopic harmony and the nature of derivations. *Linguistic Inquiry*, 41(1):169–179.

Wang, W. S.-Y. (1969). Competing changes as a cause of residue. *Language*, 45:9–25.

Wilson, C. (2003). Analyzing unbounded spreading with constraints: marks, targets, and derivations. *Unpublished manuscript, UCLA*.

Wilson, C. (2006). Unbounded spreading is myopic. *Paper presented at the Phonology Fest Workshop on Current Perspectives on Phonology, Indiana University*.

Wit, G. d. (2015). *Liko phonology and grammar: a Bantu language of the Democratic Republic of the Congo*. LOT, Utrecht.

Appendices

A Phonotactic coding

As discussed in various work, it possible to pass information from one function to another without introducing new symbols into the alphabet or increasing the length of the string, thus abiding by the letter of Heinz and Lai (2013) definition of weak determinism while increasing the expressivity of the transduction (Graf 2016, McCollum et al. 2018, O'Hara and Smith 2019, Smith and O'Hara 2019, Lamont et al. 2019). We provided one example of this in Figure 22 for unbounded tonal plateauing from Lamont et al. (2019). Here we further illustrate phonotactic coding as a markup scheme for ATR harmony in Liko.

Liko (Bantu) is very similar to Turkana (Wit 2015). Liko possesses a nine-vowel inventory, like Maasai and Turkana i ɪ e ɛ a ɔ o ʊ u, exhibits ATR harmony, and maps /a/ to [o] in [+ATR] contexts. Harmony in Liko is typically controlled by roots (27). In (27a), all affixes surface with their underlying, i.e. [−ATR] feature values, while in (27b,c) the [+ATR] root causes input /a/ to surface as [o].

(27) Liko ATR harmony (Wit 2015)

a.	/ta-púk-a/	[ta-púk-a]	'1P-leave-FV'	p. 77, ex. 3.27a
b.	/ta-ɓín-a/	[to-ɓín-o]	'1P-dance-FV'	p. 77, ex. 3.26c
c.	/ká-lut-án-ág-á/	[kó-lut-ón-óg-ó]	'9B-pull-ASS-PLUR-FV'	p. 82, ex. 3.38b

Like Turkana, Liko possesses dominant [+ATR] and dominant [−ATR] morphemes. The negative enclitic, /=gʊ/, is invariantly [−ATR] in (28). In (28a), the enclitic does not harmonize with a [+ATR] root although the preceding [+high] suffix does. When preceded by an underlying /a/, though, the enclitic not only resists [+ATR] harmony, but also prevents preceding low vowels from undergoing harmony (28b,c). In (28b), the enclitic blocks harmony on a single preceding /a/, and in (28c), the negative enclitic blocks harmony on both preceding low vowels.

(28) Unbounded circumambience in Liko ATR harmony (Wit 2015, pp. 94-96)

a.	/ná-ká-ɓín-í=gʊ/	[ná-kó-ɓín-i=gʊ]	'1S.PST-NEG-dance-FV=NEG'
b.	/ná-ká-ɓín-á=gʊ/	[ná-kó-ɓín-a=gʊ]	'1S.PST-NEG-dance-FV.PST=NEG'
c.	/ná-ká-ɓín-ág-á=gʊ/	[ná-kó-ɓín-ág-á=gʊ]	'1S.PST-NEG-dance-PLUR-FV.PST=NEG'

In examples like (28b, c), the root value for [ATR] is not sufficient to determine whether suffixal /a/ undergoes harmony to [o]. The surface realization of suffixal /a/ depends on the [ATR] value of the root, as well as the presence or absence a dominant suffix, which may be either [+ATR] or [−ATR]. There is no obvious bound on the distance between roots and suffixal /a/, nor is there any obvious bound on the distance between suffixal /a/ and a dominant suffix.

The Liko pattern is different from Turkana in that the inner, left-to-right function cannot simply map /a/ and /ɔ/ to |o|. If one does, the outer, right-to-left function cannot distinguish between underlying /a/ and /ɔ/ since both would be mapped onto intermediate |o| by the first transducer. This matters because underlying /a/ surfaces faithfully in between a [+ATR] root and a [−ATR] suffix (as long as no [+high] vowels intervene), whereas underlying /ɑ/ in Turkana surfaces as [ɔ], just like underlying /ɔ/ in the language.

One possible analysis is to introduce a novel, tenth vowel into the alphabet. One possible option is /ʌ/, a [+low, +ATR] vowel. If the inner function maps /a/ to |ʌ|, then it is possible to maintain the distinction between /a/ and /ɔ/. In effect, /a/ maps to itself via intermediate |ʌ|, /a/ ⇒ |ʌ| ⇒ [a], while /ɔ/ maps to itself via intermediate |o|, ɔ ⇒ |o| ⇒ [ɔ]. This sort of function composition clearly violates the Heinz and Lai (2013) definition of weak determinism.

With that as foreground, now consider the possibility of using a symbol already present in the language to indicate an underlying /a/ that is preceded by a [+ATR] vowel. One cannot use another back vowel since they

all occur in the suffixal domain; to use one of u ʊ o ɔ is to neutralize a contrast elsewhere. However, the front mid vowel [e] never occurs in suffixes, removing the possibility of some unintended destruction of a contrast. If suffixal /a/ is mapped to intermediate |e| instead of |ʌ|, the mapping is entirely consistent with Heinz and Lai (2013) definition since the size of the alphabet is not increased, nor is the length of the string manipulated.

This sort of distrubtional restriction is present in many languages, making it possible to use a vowel that would other never occur in a given environment as markup. Even if [e] did occur in suffixes and there was no similar way to exploit the vocalic inventory, one could use consonants. For instance, since the language allows only a limited set of syllable types, then it is possible use a consonant exactly like |ʌ| to indicate an underlying /a/ preceded by a [+ATR] vowel. If a function outputs all suffixal /a/ as |p| in (28c), then the second function can interpret the string |...g-p=gʊ| at the right edge of the word unambiguously as /...g-a=gʊ/ with a [+ATR] vowel occurring somewhere earlier in the word precisely because [p] is illicit immediately after [g].

As noted by Lamont et al. (2019), delimiting weakly deterministic functions in terms of alphabet size and string length suggests that expressivity is intimately tied to alphabet size. If a given alphabet is smaller, then there are, all else being equal, fewer opportunities to use a given symbol as markup. More concretely, the proposed difference in the expressivity of tonal patterns relative to segmental patterns in (Jardine 2016) could be reconstrued as a difference in alphabet size, since tonal inventories are smaller than segmental inventories. Moreover, this predicts a possible distinction in the expressivity of two related patterns thats stems entirely from the size of the alphabet and/or phonotactic restrictions elsewhere in the language. Imagine a language with unbounded tonal plateauing, but the additional stipulation that triggering tones must be initial and final. Lamont et al. (2019) demonstate that this sort of pattern is not analyzable with the same coding trick described above (22). This observation, however, holds only if there are only two tones in the language. Imagine a language with three contrastive tones, high, mid, and low. If mid occurs only on some fixed position(s), perhaps the initial syllable, then mid can serve as markup for the outer function because mid tones would never occur in medial syllables otherwise, rendering tonal plateauing in that language as less complex than in a language with only two tones. In our estimation, such results are spurious, and the fact that alphabet size does not affect interaction is a strength of our proposal.[39]

B Partial function algebra

This appendix contains proofs of statements made in the main text about partial functions. §B.3 overlaps with portions of §4.1 because it is meant to be a coherent presentation of three equivalent formulations of μ-conserving compositions and demonstration that they are equivalent, where the portions in the main text only focus on presenting the two formulations relevant to calculations and simultaneous application without proof and with a focus on linguistically approachable examples.

B.1 Preliminaries

Definition 28. A function $f : X \to Y$ is *total* on X iff $\forall x \in X, f(x) \in Y$, i.e. f is *defined* to map x to a particular value of Y.

The set of all total functions from X to Y is commonly abbreivated as $X \to Y$ or (equivalently) as Y^X, because the number of extensionally distinct functions from X to Y — when $|X| = n$ and $|Y| = m$ — is given by m^n.

Definition 29. A function $f : X \to Y$ is *partial* iff it is not total on X.

The set of all partial functions from X to Y may be abbreviated as $(Y + 1)^X$, because any partial function f can be trivially thought of as a total function on $Y \cup \{\bot\}$, where $f(x) = \bot$ iff f is undefined on x: $\bot$ is a distinguished or reserved element that denotes the unique 'undefined' value.

[39] Although representations are important pieces of defining computational classes (Heinz 2018, Jardine 2017, 2019, Jardine et al. 2020), these differences are primarily about what kind of representations are employed, e.g. string or autosegmental, rather than the size of the inventory of atomic elements in a particular type of representation.

Definition 30. For all partial functions $f, g : X \to Y$,

$$equalizer(f, g) = \{x | f(x) = g(x)\}$$

Definition 31. Let $f : X \to Y$ and let $A \subseteq X$. The *image* of A under f is given by

$$f[A] = \{y | y = f(x) \wedge x \in A\}$$

Definition 32. Let $f : X \to Y$. The *fiber* of y under f is given by

$$f^{-1}(y) = \{x | f(x) = y\}$$

We can also speak of *the fibers* of f — elements of the partition of X indexed by elements of Y.

Definition 33. Let $f : X \to Y$. The *kernel* of f will be given here as

$$\ker f = \{(x, x') | f(x) = f(x')\}$$

We will say that $x \sim_f x'$ iff $(x, x') \in \ker f$. Note that two elements are equivalent under f ('$x \sim_f x'$') iff they are elements of the same fiber of f.

B.2 Conditions on mutation maps and fixmaps

The two lemmas here spell out different ways an element x can come to be a mutation point or fixpoint of a composition $g \circ f$ in terms of whether or not x is a fixpoint or mutation-point of g and/or f and related properties. These lemmas are referred to by later subsections, and are best visited as called for.

Lemma 1. $\forall f, g : X \to X, \forall x, z, (x, z) \in \vec{\mu}(g \circ f)$ iff exactly one of the three following relations holds between x and z:

(F) $inner(x, z) \equiv (x, z) \in \vec{\mu}(f) \wedge z \in fix(g)$

(G) $outer(x, z) \equiv (x, z) \in \vec{\mu}(g) \wedge x \in fix(f)$

(FG) $fed(x, z) \equiv \exists y.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g) \wedge x \neq z$

In the first case, f is the proximal cause of (x, z)'s presence in $\vec{\mu}(g \circ f)$, in the second g is the proximal cause, and in the third, both f and g are proximal causes. In the case of both F and G, the last conjunct identifies a distal cause or background condition involving the other function.

Proof. First note that $\forall (x, z) \in \vec{\mu}(g \circ f)$, exactly one of $A = \{x \in fix(f), x \in \mu(f)\}$ and exactly one of $B = \{f(x) \in fix(g), f(x) \in \mu(g)\}$ must hold. We proceed through each of the four logically possible cases in $A \times B$.

(–) $x \in fix(f) \wedge x \in fix(g)$: $g \circ f(x) = g(f(x)) = g(x) = x$, so x cannot be in $\mu(g \circ f)$: this case is impossible under the assumptions we are investigating.

(F) $x \in \mu(f) \wedge f(x) \in fix(g)$: $g \circ f(x) = g(f(x)) = g(z) = z$.

(G) $x \in fix(f) \wedge f(x) \in \mu(g)$: $g \circ f(x) = g(f(x)) = g(x) = z$.

(FG) $x \in \mu(f) \wedge f(x) \in \mu(g)$: $g \circ f(x) = g(f(x)) = g(y) = z$. $x \in \vec{\mu}(g \circ f)$ if and only if it is further the case that $x \neq z$. □

Lemma 2. $\forall f, g : X \to X, \forall x, x \in fix(g \circ f)$ iff exactly one of these two propositions holds:

- $x \in fix(f) \wedge x \in fix(g)$: x is a mutual fixpoint of both f and g.
- $x \in \mu(f) \wedge f(x) \in \mu(g) \wedge g \circ f(x) = x$: g is the inverse of f at x.

Proof. Analogously to the previous lemma, $\forall(x) \in fix(g \circ f)$, exactly one of $A = \{x \in fix(f), x \in \mu(f)\}$ holds and exactly one of $B = \{f(x) \in fix(g), f(x) \in \mu(g)\}$ holds. By cases:

(*FG*) $x \in fix(f) \wedge x \in fix(g)$: $g \circ f(x) = g(f(x)) = g(x) = x$:

(–) $x \in \mu(f) \wedge f(x) \in fix(g)$: $g \circ f(x) = g(f(x)) = g(z) = z$, so x cannot be a fixpoint of $g \circ f$ and is impossible by hypothesis.

(–) $x \in fix(f) \wedge f(x) \in \mu(g)$: $g \circ f(x) = g(f(x)) = g(x) = z$, so x cannot be a fixpoint of $g \circ f$ and is impossible by hypothesis.

(DoY) $x \in \mu(f) \wedge f(x) \in \mu(g)$: $g \circ f(x) = g(f(x)) = g(y) = z$. $x \in fix(g \circ f)$ if and only if it is further the case that $x = z$. □

B.3 Three equivalent definitions of μ-conservation

In this section, we first present three definitions of μ-conservation of partial functions on a discrete set, and then show their equivalence.

Perhaps the most accessible definition of when two partial functions are μ-conserving is an extension of the ideas of mutation sets and maps and related to the algebraic concepts of *idempotence* — a property of mappings independently noted to be relevant to phonology and to facilitate phonological learning (Tesar 2014, Magri 2018) — and *one-sided identities*.

Definition 34. Given $f : X \to X$, f is *idempotent* iff

$$\forall x \in dom(f), f \circ f(x) = f(x)$$

For example, define $doubleOdd : \mathbb{N} \to \mathbb{N}$ as

$$doubleOdd(x) = \begin{cases} 2x & \text{if } x \bmod 2 \neq 0 \\ x & \text{otherwise} \end{cases}$$

Because doubling an odd number always yields an even number, *doubleOdd* will never be apply again to the result of something that it had once applied to: *doubleOdd* is idempotent.

In terms of identities, a function f is idempotent if it is its own *two-sided identity* element under composition: $f \circ f = f$.

We can think of the definition of μ-conservation below as a generalization of idempotence to two functions — or idempotence as a special instance of μ-conservation.

Definition 35. Given $f, g : X \to X$, f and g are *μ-conserving* when composed as $g \circ f$ iff both

- $\forall x \in \mu(f), g \circ f(x) = f(x)$
- $\forall x \in \mu(g), g \circ f(x) = g(x)$

i.e. g is a *left-identity* of f for all of f's mutation points and f is a *right-identity* of g for all of g's mutation points.

That is, in a μ-conserving composition, no change that f can cause to an input x in $f(x)$ can be rewritten by g and any change g *would* cause to an input (as $g(x)$) either isn't interfered with by f in $g \circ f$, or f makes the same change g would have, given the opportunity to apply first or in isolation.

Equipped with the concepts of fixpoints and mutation points, we can understand idempotence equivalently as in Def. 36.

Definition 36. Given $f : X \to X$, f is idempotent iff

$$\forall x \in \mu(f), f(x) \in \mathit{fix}(f)$$

That is, an idempotent function always maps its own mutation points to its own fixpoints.

The second definition of μ-conservation— the one most useful for calculations and that motivates considering operations on functions besides composition — treats partial functions as a set of (input, output) pairs and compares the mutation maps of both composands to those of the composition: μ-conserving compositions are just those that *conserve* the mutation maps of their composands:

Definition 37. For all partial functions $f, g : X \to X$, f and g are *μ-conserving* when composed as $g \circ f$ iff

$$\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f)$$

That is, for any μ-conserving composition $g \circ f$, every change made by f is a change made by $g \circ f$, every change made by g is a change made by $g \circ f$, and there are no other changes that $g \circ f$ makes.

Where the first formulation focuses on the elements $x \in X$ acted on by f and g and their dynamics in a composition, the second definition is more general: it treats functions as wholistic collections of mappings, treats composition as a particular binary operator that takes two functions and returns a new function, and identifies a key property of μ-conserving compositions that makes them such. In principle, however, any operation on two partial functions f, g — not just ordered application — that yields a new partial function could also be said to be μ-conserving iff the result of the operation conserves the mutation maps of the operands.

An example of an alternative operation on mappings is *priority union* — an operation on relations originally introduced for LFG and argument-value matrices in the 1980s (Kaplan 1987), but also applied to strings by Kaplan and Newman (1997) and Karttunen (1998). We present a general (non-string) formalization of *simultaneous application*[40] defined in terms of priority union that exhibits exactly this property of $\mu - conservation$. For this reason, priority union lies at the heart of the next definition of μ-conserving compositions. As we will show, this definition states that μ-conserving compositions are exactly those which can be equivalently defined in terms of priority union. Roughly, $g \circ f$ is μ-conserving iff $\vec{\mu}(g \circ f) = \vec{\mu}(g \overset{\rho}{\leftarrow} f) = \vec{\mu}(f \overset{\rho}{\leftarrow} g)$, i.e. when the priority union of the mutation maps of the composands commutes. To arrive at this next definition, we extend this basic idea to define an operation on functions that intuitively corresponds to 'simultaneous application' in the setting of partial functions on discrete sets. Below we introduce priority union and its relation to simultaneous application before introducing the third equivalent definition of μ-conserving compositions.

Intuitively, given two partial functions or relations g and f, the (left) priority union —- $g \overset{\rho}{\leftarrow} f$ — keeps all mappings defined in g, and then adds to that anything in f that doesn't 'conflict' with any mappings already defined in g: if there is a mapping $(x, g(x))$ defined in g and a mapping $(x, f(x))$ defined in f, then only the mapping of g defined on x survives the union because g gets 'priority' over f. The only mappings of f that will be found in $g \overset{\rho}{\leftarrow} f$ are those mappings of f already found in g, plus mappings $(x, f(x))$ where $g(x)$ is undefined. See Fig. 17 for visual depiction of examples contrasting priority union with composition.

Definition 38. Given any two (partial) relations $R, S \subseteq X \times Y$, the *priority union of R with S* is given by

$$R \cup \{(x, y) | (x, y) \in S \wedge \nexists y'.(x, y') \in R\}$$

[40] We offer a generalization to string functions in §4.2 that accords with linguistic intuitions.

Note that we have introduced *left* priority union. Right priority union $\twoheadrightarrow$ is defined analogously. We will use 'priority union' henceforth to refer to left priority union. Note that partial functions are closed under priority union — but not under general set-theoretic union.

The following definition (equivalent for relations that are partial functions) is evocative of 'pattern matching' expressions in modern functional programming languages, where the output for some input value is determined by attempting to match the input value against a sequence of 'cases' or 'patterns', and the first one to match (starting from the top and going down) determines what is ultimately output:[41]

Definition 39. Given partial functions $f, g : X \to X$, the *priority union of f with g* is given by

$$f \twoheadleftarrow g(x) = \begin{cases} f(x) & \text{if } x \in dom(f) \\ g(x) & \text{if } x \in dom(g) \wedge x \notin dom(f) \\ \bot & \text{otherwise} \end{cases}$$

These two definitions are entirely exchangeable for partial functions — we encourage the reader to prove this for themselves.

Two functions can only be 'simultaneously applied' to any element x *precisely* when only one of them applies (non-vacuously), or when both apply in the same way ($x \in equalizer(f, g)$) — i.e. roughly when the two functions don't 'conflict'. Formally, this is exactly when their mutation maps commute under priority union or (equivalently) when the priority union of their mutation maps is the same as normal set-theoretic union:

Definition 40. Given two functions $f, g : X \to X$, simultaneous application of f and g is *possible* iff

$$\vec{\mu}(f) \twoheadleftarrow \vec{\mu}(g) = \vec{\mu}(g) \twoheadleftarrow \vec{\mu}(f) = \vec{\mu}(f) \cup \vec{\mu}(g)$$

This defines a *condition* for simultaneous application but doesn't define simultaneous application *per se*. Given two partial functions f, g that meet the condition for simultaneous application, $\vec{\mu}(f) \twoheadleftarrow \vec{\mu}(g)$ has all the mutation maps of simultaneous application, but is missing the fixmaps of f and g. Since we do not want the fixmaps of f or g to interfere with conservation of mutation maps, the definition for simultaneous application below adds fixmaps after adding together mutation maps; note that the order of adding one set of fixmaps vs. the other is arbitrary.

Definition 41. Given two functions $f, g : X \to X$ that meet the condition for simultaneous application,[42] the *simultaneous application of f and g* — denoted $f \mid g$ — is defined as

$$f \mid g = (\vec{\mu}(f) \twoheadleftarrow \vec{\mu}(g)) \twoheadleftarrow (\overrightarrow{fix}(f) \twoheadleftarrow \overrightarrow{fix}(g))$$

where $\overrightarrow{fix}(f)$ is the fixset analogue of $\vec{\mu}(f)$, i.e. $\overrightarrow{fix}(f) = \{(x, x) | x \in fix(f)\}$.

See Fig. 18 for visual depiction of examples and contrast with both priority union and composition.

With the definition of simultaneous application in place, one can compare the second definition of μ-conserving composition (Def. 37) with the condition on simultaneous application and see a straightforward connection: when simultaneous application is applicable, $f \mid g$ results in a function that conserves the mutation maps of both operands, just like a μ-conserving composition. We can define μ-conserving compositions as precisely those that are equivalent to simultaneous application:

[41] If P, Q, R are predicates that define membership in, say, $dom(f), dom(g), dom(h)$, then $(f \twoheadleftarrow g \twoheadleftarrow h)(x)$ is equivalent to 'If $P(x)$, then $f(x)$; (else) if $(\neg P(x) \wedge) Q(x)$, then $g(x)$; (else) if $(\neg P(x) \wedge \neg Q(x) \wedge) R(x)$, then $h(x)$'.

The more functional programming and algebra- (than automata- or logic-) flavored formulations of the regular ($\supset$ *rational*) or polyregular transductions we are aware of all contain something comparable(see the *conditional choice* operator in Alur et al. 2014; the *else* operator in Alur et al. 2015; the *disjoint union* combinator in Bojańczyk et al. 2018; the *case* function in Bojańczyk 2018).

[42] Note that iff the condition for simultaneous application is not met, then the right-hand side of $f \mid g$ is still perfectly calculable, but the equality $f \mid g = g \mid f$ will not, in general, hold, nor will linguistic intuitions about what 'simultaneous application' ought to be.

Definition 42. Given $f, g : X \to X$, $g \circ f$ is μ-conserving iff

$$g \circ f = g \mid f$$

We now demonstrate that every composition that meets one of the three definitions presented so far meets the other two. We begin with a helpful lemma.

Lemma 3. If $g \circ f$ satisfies Definition 35, then condition FG of Lemma 1 cannot hold for any $(x, z) \in \vec{\mu}(g \circ f)$.

Proof. Suppose both $g \circ f$ satisfies Def. 35 and $\exists (x, z) \in \vec{\mu}(g \circ f)$ satisfying FG.

1. By hypothesis and Lemma 1, $\exists y, z.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g)$.
2. Because composition always yields a function and $g \circ f(x) = z$ by assumption, then it must be the case that $(x, y) \notin \vec{\mu}(g \circ f)$.
3. $(x, y) \in \vec{\mu}(f) \wedge (x, y) \notin \vec{\mu}(g \circ f)$ means the first condition of Definition 35 is violated.
4. We have reached a contradiction. □

Theorem 1. $g \circ f$ satisfies Definition 37 iff it also satisfies Definition 35.

Proof. First we show that satisfying Definition 37 implies satisfying Definition 35.

1. $\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f) \Longrightarrow \forall (x, y) \in \vec{\mu}(f), (x, y) \in \vec{\mu}(g \circ f)$, so the first condition of Definition 35 is satisfied.
2. $\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f) \Longrightarrow \forall (x, y) \in \vec{\mu}(g), (x, y) \in \vec{\mu}(g \circ f)$, so the second condition of Definition 35 is satisfied.

Second, we show that satisfying Definition 35 implies satisfying Definition 37.

1. By Lemma 3 and Lemma 1, each mutation $(x, z) \in \vec{\mu}(g \circ f)$ must satisfy either condition F or satisfy condition G.
2. $\forall (x, z)$, (x, z) satisfies condition $F \Longleftrightarrow x \in \mu(f) \wedge g \circ f(x) = f(x) \wedge (x, z) \in \vec{\mu}(f)$.
3. $\forall (x, z)$, (x, z) satisfies condition $G \Longleftrightarrow x \in \mu(g) \wedge g \circ f(x) = g(x) \wedge (x, z) \in \vec{\mu}(g)$.
4. Because every mutation map $(x, z) \in \vec{\mu}(g \circ f)$ must either satisfy F (and fall into $\vec{\mu}(f)$) or it must satisfy G (and fall into $\vec{\mu}(g)$), strict equality holds: $\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f)$.

□

Theorem 2. $g \circ f$ satisfies Definition 37 iff it also satisfies Definition 42.

Proof.

1. $g \circ f$ satisfies Definition 37 iff $\vec{\mu}(g \circ f) = \vec{\mu}(g) \cup \vec{\mu}(f)$.
2. $g \circ f$ satisfies Definition 42 iff $g \circ f = g \mid f$.
3. By Definitions 40-41, $\vec{\mu}(g \mid f) = \vec{\mu}(f) \cup \vec{\mu}(g)$.

□

B.4 Commutativity and μ-interaction

Definition 43. Non-vacuous (non-commutative) composition order. Two functions f and g are said to be non-vacuously (or non-commutatively) ordered with respect to x iff $g \circ f(x) \neq f \circ g(x)$. We can also say that f and g are non-commutative and in particular that they do not commute at x.

Theorem 3. Pointwise commutativity does not in general imply pointwise μ-conservation, and pointwise commutativity with pointwise interaction does not entail that f, g are pointwise inverses.

Proof. $g \circ f(x) \neq f \circ g(x) \Longleftrightarrow$ any one of the following cases holds:

1. (NI) $x \in \mathit{fix}(f) \wedge x \in \mathit{fix}(g)$:
   x is a fixpoint of both f, g.

2. (NI) $(x \in \mathit{fix}(f) \wedge x \in \mu(g)) \wedge g(x) \in \mathit{fix}(f)$:
   x is a mutation point of just g and $g(x)$ is left alone by f.

3. (NI) $(x \in \mathit{fix}(g) \wedge x \in \mu(f)) \wedge f(x) \in \mathit{fix}(g)$:
   x is a mutation point of just f and $f(x)$ is left alone by g.

4. (NI) $(x \in \mu(f) \wedge x \in \mu(g)) \wedge f(x) = g(x) \wedge (f(x) \in \mathit{fix}(g) \wedge g(x) \in \mathit{fix}(f))$:
   x is a mutation point of f, g, and is in their equalizer, and neither f nor g then do anything to $f(x) = g(x)$.

5. (I) $(x \in \mu(f) \wedge x \in \mu(g)) \wedge f(x) = g(x) \wedge (f(x) \in \mu(g) \wedge g(x) \in \mu(f) \wedge g \circ f(x) = f \circ g(x))$:
   x is a mutation point of f, g, and is in their equalizer, and $f(x) = g(x)$ is both a mutation point of f, g and in their equalizer. Note that $g \circ f(x) = x = f \circ g(x)$ is possible but not necessary!

6. (I) $(x \in \mu(f) \wedge x \in \mu(g)) \wedge f(x) \neq g(x) \wedge (g \circ f(x) = f \circ g(x))$:
   x is a mutation point of f, g but is not in their equalizer, but $g \circ f(x) = f \circ g(x)$. Again, note that $g \circ f(x) = x = f \circ g(x)$ is possible but not necessary!

Cases (1-4) involve non-interaction, but cases (5-6) involve interaction. The cases that do involve interaction can, but need not involve two functions acting as partial inverses. □

Theorem 4. Pointwise non-commutativity entails pointwise interaction.

Proof. If $g \circ f(x) \neq g \circ f(x)$, then exactly one of the following three cases holds:

1. x is a mutation point of $g \circ f$ but not of $f \circ g$. Per Lemmas 1-2, there are in principle 3×2 ways this conjunction of events could happen.

   (–) $((x, z) \in \vec{\mu}(f) \wedge z \in \mathit{fix}(g)) \wedge (x \in \mathit{fix}(g) \wedge x \in \mathit{fix}(f))$: these conditions are contradictory.

   (–) $((x, z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(f)) \wedge (x \in \mathit{fix}(g) \wedge x \in \mathit{fix}(f))$: these conditions are contradictory.

   (–) $(\exists y.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g) \wedge x \neq z) \wedge (x \in \mathit{fix}(g) \wedge x \in \mathit{fix}(f))$: these conditions are contradictory.

   (a) $((x, z) \in \vec{\mu}(f) \wedge z \in \mathit{fix}(g)) \wedge (x \in \mu(g) \wedge g(x) \in \mu(f) \wedge f \circ g(x) = x)$:
   In $g \circ f$, x is a mutation point of f and g then leaves $f(x)$ alone.
   In $f \circ g$, f inverts g's change at x.
   $\therefore$ $g \circ f$ will be locally non-interacting iff both $(x, z) \in \vec{\mu}(f)$ is conserved (it must be) *and* $(x, z') \in \vec{\mu}(g)$ is conserved (it can be, but needn't be); this second condition will only hold when $x \in \mathit{equalizer}(f, g)$.
   $f \circ g$ *cannot* be locally non-interacting because $f \circ g$ does not conserve $(x, z') \in \vec{\mu}(g)$.
   $\therefore$ at least one order of composition in this case involves interaction at x.
   x could be in either the equalizer of f, g, or not.

(b) $((x, z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(f)) \wedge (x \in \mu(g) \wedge g(x) \in \mu(f) \wedge f \circ g(x) = x)$:
In $g \circ f$, x is a fixpoint of f and g then alters $f(x) = x$.
In $f \circ g$, f inverts g's change at x.
$\therefore g \circ f$ will be locally non-interacting because $(x, z) \in \vec{\mu}(g)$ is conserved and $x \in \mathit{fix}(f)$.
$f \circ g$ *cannot* be non-interacting because $f \circ g$ does not conserve $(x, z) \in \vec{\mu}(g)$.
$\therefore$ at least one order of composition in this case involves interaction at x.
x *cannot* be in the equalizer of f, g.

(c) $(\exists y.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g) \wedge x \neq z) \wedge (x \in \mu(g) \wedge g(x) \in \mu(f) \wedge f \circ g(x) = x)$:
In $g \circ f$, g's change to $f(x)$ does NOT invert f's change at x.
In $f \circ g$, f inverts g's change at x.
$\therefore g \circ f$ does not conserve $(x, y) \in \vec{\mu}(f)$.
$f \circ g$ does not conserve $(x, z') \in \vec{\mu}(g)$.
$\therefore$ BOTH orders of composition in this case involves interaction at x / lack of conservation of changes to x relative to both f, g.
x could either be in the equalizer of f, g (i.e. $y = z'$), or not.

2. x is a mutation point of $f \circ g$ but not $g \circ f$: this has the same 3×2 cases as before, but with f and g swapped.

3. x is a mutation point of both composition orders: per Lemma 1, there are in principle 3×3 ways this conjunction of events could happen:

(a) $((x, z) \in \vec{\mu}(f) \wedge z \in \mathit{fix}(g)) \wedge ((x, z') \in \vec{\mu}(g) \wedge z' \in \mathit{fix}(f))$:
In $g \circ f$, x is a mutation point of f and g then leaves $f(x)$ alone.
In $f \circ g$, x is a mutation point of g and f then leaves $g(x)$ alone.
Since by hypothesis $g \circ f(x) \neq f \circ g(x)$ and $g \circ f(x) = f(x)$ and $f \circ g(x) = g(x)$, $f(x) \neq g(x)$.
$\therefore$ *both* orders of composition involve lack of conservation of changes to x relative to both f, g.
x *cannot* be in the equalizer of f, g.

(–) $((x, z) \in \vec{\mu}(f) \wedge z \in \mathit{fix}(g)) \wedge ((x, z) \in \vec{\mu}(f) \wedge x \in \mathit{fix}(g))$:
In $g \circ f$, x is a mutation point of f and g then leaves $f(x)$ alone.
In $f \circ g$, x is a fixpoint of g and f then alters $g(x) = x$.
This contradicts the hypothesis that $g \circ f(x) \neq f \circ g(x)$.

(b) $((x, z) \in \vec{\mu}(f) \wedge z \in \mathit{fix}(g)) \wedge (\exists y.(x, y) \in \vec{\mu}(g) \wedge (y, z') \in \vec{\mu}(f) \wedge x \neq z')$:
In $g \circ f$, x is a mutation point of f and g then leaves $f(x)$ alone.
In $f \circ g$, f makes a non-inverting change to g's change at x.
Since by hypothesis $g \circ f(x) \neq f \circ g(x)$ and $g \circ f(x) = f(x)$ and $f \circ g(x) \neq g(x)$.
$\therefore$ at least one order of application (viz. $f \circ g(x)$) involves interaction because it does not preserve $(x, g(x))$.
x could either be in the equalizer of f, g (i.e. $y = z$), or not.

(–) $((x, z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(f)) \wedge ((x, z') \in \vec{\mu}(g) \wedge z' \in \mathit{fix}(f))$:
In $g \circ f$, x is a fixpoint of f and g then alters $f(x) = x$.
In $f \circ g$, x is a mutation point of g and f then leaves $g(x)$ alone.
Here, $g \circ f(x) = g(x) = f \circ g(x)$, which contradicts our hypothesis that $g \circ f(x) \neq f \circ g(x)$.

(–) $((x, z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(f)) \wedge ((x, z') \in \vec{\mu}(f) \wedge x \in \mathit{fix}(g))$:
In $g \circ f$, x is a fixpoint of f and g then alters $f(x) = x$.
In $f \circ g$, x is a fixpoint of g and f then alters $g(x) = x$.
These two propositions are contradictory.

(c) $((x,z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(f)) \wedge (\exists y.(x,y) \in \vec{\mu}(g) \wedge (y,z') \in \vec{\mu}(f) \wedge x \neq z')$:
In $g \circ f$, x is a fixpoint of f and g then alters $f(x) = x$.
In $f \circ g$, f makes a non-inverting change to g's change at x.
Here, $g \circ f(x) = g(x)$ and by hypothesis, $g(x) \neq f \circ g(x)$.
$\therefore$ at least one order of composition involves interaction because $(x, g(x))$ is not conserved.
x *cannot* be in the equalizer of f, g.

(d) $(\exists y.(x,y) \in \vec{\mu}(f) \wedge (y,z) \in \vec{\mu}(g) \wedge x \neq z) \wedge ((x,z') \in \vec{\mu}(g) \wedge z' \in \mathit{fix}(f))$:
In $g \circ f$, g makes a non-inverting change to f's change at x.
In $f \circ g$, x is a mutation point of g and f then leaves $g(x)$ alone.
Here, $f \circ g(x) = g(x)$ and by hypothesis, $g(x) \neq g \circ f(x)$.
$\therefore$ in at least one order of application (viz. $g \circ f(x))$ $(x, f(x))$ is not conserved, so at least one order involves interaction.
x could either be in the equalizer of f, g (i.e. $y = z'$), or not.

(e) $(\exists y.(x,y) \in \vec{\mu}(f) \wedge (y,z) \in \vec{\mu}(g) \wedge x \neq z) \wedge ((x,z') \in \vec{\mu}(f) \wedge x \in \mathit{fix}(g))$:
In $g \circ f$, g makes a non-inverting change to f's change at x.
In $f \circ g$, x is a fixpoint of g and f then alters $g(x) = x$.
Here, $f \circ g(x) = f(x)$ and by hypothesis, $f(x) \neq g \circ f(x)$.
$\therefore$ in at least one order of application (viz. $g \circ f(x))$ $(x, f(x))$ is not conserved, so at least one order involves interaction.
x *cannot* be in the equalizer of f, g.

(f) $(\exists y.(x,y) \in \vec{\mu}(f) \wedge (y,z) \in \vec{\mu}(g) \wedge x \neq z) \wedge (\exists y'.(x,y') \in \vec{\mu}(g) \wedge (y',z') \in \vec{\mu}(f) \wedge x \neq z')$:
In $g \circ f$, g makes a non-inverting change to f's change at x.
In $f \circ g$, f makes a non-inverting change to g's change at x.
By hypothesis, $g \circ f(x) \neq f \circ g(x)$.
$\therefore$ in *both* orders of application at least one mutation mapping of either f, g is not conserved, so both orders involve interaction.
x could either be in the equalizer of f, g (i.e. $y = y'$), or not.

□

B.5 Mutation-map relations and μ-fix dynamics

One common pattern of framing 'interactions' — 'feeding' and 'bleeding' — in phonological map composition is in terms of dynamics into and/or out of the mutation set of one of the composed functions:

Definition 44. Given $f, g : X \to X$, f *feeds* g at x in $g \circ f$ iff

$$x \in \mu(f) \wedge f(x) \in \mu(g)$$

...i.e. f changes x and maps x into $\mu(g)$.

Definition 45. Given $f, g : X \to X$, f *bleeds* g at x in $g \circ f$ iff

$$x \in \mu(f) x \in \mu(g)$$

...i.e. f changes x and maps x from $\mu(g)$.

Definition 46. Given $f, g : X \to X$, f *transfuses* g at x in $g \circ f$ iff f both feeds and bleeds g at x.

...i.e. f changes x and maps x from $\mu(g)$ to (a distinct point) y in $\mu(g)$.

We will refer to cases where f feeds (bleeds) g at x without also tranfusing g at x as 'proper feeding' ('proper bleeding'). Note that we could just as easily have called proper feeding just 'feeding', proper bleeding just 'bleeding', and rendered feeding vs. bleeding vs. transfusion a mutually exclusive three-way distinction.

Here we derive analogues of feeding and bleeding ('pushing' and 'pulling') from the first two definitions of μ-interaction presented in §B.3 and then prove the relationship between pushing and pulling vs. feeding and bleeding.

Both of the first two definitions of a μ-conserving composition break down into two components — two conditions for Def. 35 and two sets for Def. 37. Violations of one element will correspond roughly to a feeding-like interaction and violations of the other element will correspond roughly to a bleeding-like interaction.

Consider pointwise versions of Def. 35-37:

Definition 47. Given $f, g : X \to X$, f and g are μ-conserving at x when composed as $g \circ f$ iff both

(L) $x \in \mu(f) \Longrightarrow g \circ f(x) = f(x)$

(R) $x \in \mu(g) \Longrightarrow g \circ f(x) = g(x)$

i.e. (if $x \in \mu(f)$), then g is a *left-identity* of f at x, and (if $x \in \mu(g)$), then f is a *right-identity* of g at x.

Definition 48. Given $f, g : X \to X$, f and g are μ-conserving at x when composed as $g \circ f$ when $\exists y.(x, y) \in \vec{\mu}(g \circ f) \Longleftrightarrow$

(L) $(x, y) \in \vec{\mu}(f) \ldots$

(R) $\vee (x, y) \in \vec{\mu}(g)$.

We will identify 'pushing' ($\approx$ feeding) at x and 'pulling' ($\approx$ bleeding) at x with non-trivial violations at x of conditions L and condition R (respectively) of these (equivalent) definitions:

Definition 49. Given $f, g : X \to X$, f *pushes* g at x iff

$$(x, y) \in \vec{\mu}(f) \wedge (x, y) \notin \vec{\mu}(g \circ f)$$

(See case FG of Lemma 1.) Here a mutation map of f is missing from the set of mutation maps of the composition $g \circ f$ and it has been replaced by a map $(x, g \circ f(x))$.[43]

Definition 50. Given $f, g : X \to X$, f *pulls* g at x iff

$$(x, y) \in \vec{\mu}(g) \wedge (x, y) \notin \vec{\mu}(g \circ f)$$

More straightforwardly than pushing, in a pulling interaction a mutation map of g is missing from the set of mutation maps of the composition $g \circ f$.

Definition 51. Given $f, g : X \to X$, f *shifts* g at x iff f both pushes and pulls g at x.

By analogy to feeding, bleeding, and transfusion, let proper pushing (pulling) refer to pushing (pulling) at x without shifting at x.

The theorems below demonstrate the relationship between pointwise feeding/bleeding(/transfusion) and pushing/pulling(/shifting).

[43] Note that for the general setting of partial functions on sets of discrete elements, the key defining element of a pushing interaction is a mutation map of f *missing* from $\vec{\mu}(g \circ f)$, not the appearance of a *novel* map $(x, g \circ f(x))$ in $\vec{\mu}(g \circ f)$: in general, $(x, g \circ f(x))$ could already be a mutation map of g (i.e. $(x, g \circ f(x))$ and $(f(x), g \circ f(x))$ are both in $\vec{\mu}(g)$), but in such a case $(x, f(x))$ will still necessarily be missing from $\vec{\mu}(g \circ f)$.

Theorem 5. If f properly pushes g at x, then f either maps a fixpoint of g to a mutation point of g or a mutation point of g to another mutation point of g that is equivalent under g.

Proof. Given: $x \in \mu(f) \wedge g \circ f(x) \neq f(x) \wedge \neg(x \in \mu(g) \wedge g \circ f(x) \neq g(x))$

1. $\equiv x \in \mu(f) \wedge g \circ f(x) \neq f(x) \wedge (x \notin \mu(g) \vee g \circ f(x) = g(x))$
2. $\equiv (x \in \mu(f) \wedge g \circ f(x) \neq f(x) \wedge x \notin \mu(g)) \vee$
   $(x \in \mu(f) \wedge g \circ f(x) \neq f(x) \wedge g \circ f(x) = g(x))$
3. Continuing by modifying the left disjunct of step 2: $x \in \mu(f) \wedge g \circ f(x) \neq f(x) \wedge x \in \mathit{fix}(g)$
4. Continuing with the left disjunct: $x \in \mathit{fix}(g) \wedge x \in \mu(f) \wedge \exists y, z.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g)$
5. Continuing from right disjunct of step 2: $x \in \mu(f) \wedge \exists y, z.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g) \wedge (x, z) \in \vec{\mu}(g)$
6. If the left disjunct of Step 2 is true, then Step 4 shows that f maps a fixpoint x of g to a mutation point y of g.
7. If the right disjunct of Step 2 is true, then Step 5 shows that f maps a mutation point x of g to another mutation point y of g such that $x \sim_g y$. □

Note that this means that proper pushing at x entails feeding (not necessarily proper feeding) at x.

Theorem 6. If f properly pulls g at x, then f maps a mutation point of g to a fixpoint of g.

Proof. Given: $x \in \mu(g) \wedge g \circ f(x) \neq g(x) \wedge \neg(x \in \mu(f) \wedge g \circ f(x) \neq f(x))$

1. $\equiv x \in \mu(g) \wedge g \circ f(x) \neq g(x) \wedge (x \notin \mu(f) \vee g \circ f(x) = f(x))$
2. $\equiv (x \in \mu(g) \wedge g \circ f(x) \neq g(x) \wedge x \notin \mu(f)) \vee (x \in \mu(g) \wedge g \circ f(x) \neq g(x) \wedge g \circ f(x) = f(x))$
3. The left disjunct of the previous step describes an impossible state of affairs, so we are left with the right disjunct.
4. Continuing with the right disjunct: by Lemma 1, $g \circ f(x) = f(x) \Longrightarrow f(x) \in \mathit{fix}(g)$.
5. By Step 2 and Step 4, $x \in \mu(g) \wedge f(x) \in \mathit{fix}(g)$. □

Note that this means that proper pulling at x entails proper bleeding at x.

Theorem 7. If f shifts g at x, then f maps a mutation point of g to a distinct mutation point of g that is not equivalent to the first mutation point.

Proof. Given: $x \in \mu(f) \wedge x \in \mu(g) \wedge g \circ f(x) \neq f(x) \wedge g \circ f(x) \neq g(x)$.

1. $\equiv \exists w, y.(x, w) \in \vec{\mu}(f) \wedge (x, y) \in \vec{\mu}(g) \wedge g \circ f(x) \neq w \wedge g \circ f(x) \neq y$
2. $\equiv \exists w, y, z.(x, w) \in \vec{\mu}(f) \wedge (x, y) \in \vec{\mu}(g) \wedge (w, z) \in \vec{\mu}(g) \wedge z \neq w \wedge z \neq y$ □

Note that this means that shifting at x entails transfusion at x.

Theorem 8. If f properly feeds g at x, then f properly pushes g at x.

Proof. Given: $x, y, z.(x, y) \in \vec{\mu}(f) \wedge (y, z) \in \vec{\mu}(g) \wedge x \in \mathit{fix}(g)$.

1. Pushing at x is necessary because $(x, y) \in \vec{\mu}(f) - \vec{\mu}(g \circ f)$.

2. Pulling at x is impossible because there is no mutation map of g at x.
3. Therefore proper feeding of g at x by f entails *proper* pushing of g at x by f.

□

Theorem 9. If f properly bleeds g at x, then f pulling g at x is possible (not necessary) and f pushing g at x is impossible.

Proof. Given: $x, y, z.(x, y) \in \vec{\mu}(f) \wedge (x, z) \in \vec{\mu}(g) \wedge y \in \mathit{fix}(g)$.

1. Pushing at x is impossible because $(x, y) \in \vec{\mu}(f) \cap \vec{\mu}(g \circ f)$. Therefore, if pulling occurs, it must be proper.
2. If $x \in \mathit{equalizer}(f, g)$, then no pulling at x occurs because $(x, z) \in \vec{\mu}(g) \cap \vec{\mu}(g \circ f)$.
3. If $x \notin \mathit{equalizer}(f, g)$, then pulling at x occurs because $(x, z) \in \vec{\mu}(g) - \vec{\mu}(g \circ f)$.

□

Theorem 10. If f transfuses g from x to y, then f must push g at x but need not shift g from x to y (or to anywhere else).

Proof. Given: $\exists w, x, y, z.(x, y) \in \vec{\mu}(f) \wedge (x, w), (y, z) \in \vec{\mu}(g)$.

1. f pushes g at x because $(x, z) \in \vec{\mu}(f) - \vec{\mu}(g \circ f)$.
2. f pulling g at x (and therefore shifting g at x) is possible but not necessary; it will occur if $w \neq z \equiv x \nsim_g y$.

□

Summary:

(29) f pushes g at $x \Longrightarrow f$ feeds g at x.
   a. f properly pushing g at $x \Longrightarrow$ that f (not necessarily properly) feeds g at x.
   b. f shifting g at $x \Longrightarrow$ that f transfuses g at x and therefore (improper) feeding of g at x by f.

(30) f pulls g at $x \Longrightarrow f$ bleeds g at x.
   a. f properly pulling g at $x \Longrightarrow f$ properly bleeds g at x.
   b. f shifting g at $x \Longrightarrow f$ transfuses g at x and therefore (improper) bleeding of g at x by f.

(31) f shifts g at $x \Longrightarrow f$ transfuses g at x.

(32) f feeds g at $x \Longrightarrow f$ pushes g at x.
   a. f properly feeding g at $x \Longrightarrow f$ properly pushes g at x.
   b. f transfusing g at $x \Longrightarrow f$ (not necessarily properly) pushes g.

(33) f bleeds g at x does not in general imply any mutation-map relation between f and g at x.
   a. f properly bleeding g at $x \Longrightarrow f$ does not push g at x.
   b. f properly bleeding g at $x \wedge x \notin \mathit{equalizer}(f, g) \Longrightarrow f$ properly pulls g at x.
   c. f transfusing g from x to $y \Longrightarrow f$ pushes g at x.
   d. f transfusing g from x to $y \wedge x \nsim_g y \Longrightarrow f$ shifts g from x to y (and therefore pulls g at x).

(34) f transfuses g from x to $y \Longrightarrow f$ pushes g at x, but does not necessarily shift g from x to y.
   a. f transfusing g from x to $y \wedge x \nsim_g y \Longrightarrow f$ shifts g from x to y (and therefore pulls g at x).

(35) f pushes g at x iff f feeds g at x.