Computing Vowel Harmony: The Generative Capacity of *Search and Copy*

Samuel Andersson,[1] Hossep Dolatian,[2] and Yiding Hao[1]
[1]*Yale University and* [2]*Stony Brook University*

1 Introduction

Search & Copy (S&C, Nevins, 2010; Mailhot & Reiss, 2007; Samuels, 2009a,b) is a procedural model of vowel harmony in which underspecified vowels trigger *searches* for *targets* that provide them with features. In this paper, we seek to relate the S&C formalism with models of phonological locality proposed by recent work in the *subregular program* (Heinz, 2018; Chandlee, 2014; Chandlee et al., 2015; Hao & Andersson, 2019; Hao & Bowers, 2019). Our goal is to provide a formal description, within the framework of mathematical linguistics, of the range of possible phonological transformations that admit an analysis within S&C. In particular, we do not propose an analysis of any particular linguistic phenomenon; instead, the present study should be viewed as an analysis of the S&C formalism itself. Our analysis allows S&C to be compared with other formalisms in terms of their expressive power, and identifies conditions under which S&C descriptions of vowel harmony and other phenomena may be incorporated into software systems for phonological processing based on finite-state techniques (see Beesley & Karttunen, 2003).

The contributions of this paper to the study of S&C and subregular phonology are as follows. We show that used in its *unidirectional* mode, all transformations described by an S&C analysis can be modeled by *tier-based input strictly local functions* (TISL, Chandlee, 2014; Hao & Andersson, 2019). This result improves the previous result of Gainor et al. (2012), which showed that vowel harmony processes can be modeled by *subsequential functions*. However, non-TISL transformations can be given S&C descriptions in the following ways. Firstly, since TISL functions are not closed under composition (Sakarovitch, 2009; Chandlee et al., 2018), a non-TISL vowel harmony pattern may be obtained by applying two S&C rules sequentially. Secondly, when S&C is used in its *bidirectional* mode, it has the ability to describe transformations that cannot be modeled by finite-state functions.

The structure of this paper is as follows. In Section 2, we review basic background on the phenomenon of vowel harmony, and in Section 3 we describe the S&C framework as it is formulated by Mailhot & Reiss (2007). Section 4 introduces the finite-state, subsequential, and TISL function classes, and Section 5 shows how these formal constructs can be used to give an efficient implementation of S&C. 6 show how our main result of Section 5 does not hold for bidirectional processes or for the composition of multiple processes. Section 7 concludes.

2 Vowel Harmony

Vowel harmony is a commonly attested process in which certain features of vowels take on values that match those of nearby vowels. A well-known example of vowel harmony, from Turkish, is illustrated in (1). The plural suffix consists of two allomorphs, *-lar* and *-ler*, whose distribution is determined by the features of the last vowel in the nominal root. When the last vowel is [+BACK], the suffix surfaces as *-lar*; otherwise, it surfaces as *-ler*. A similar descriptive generalization governs the distribution of the locative suffix *-a*/*-e*.

(1) *Backness Harmony in Turkish*

	'house'	'horse'	'address'
SG	/ev/ ⇝ [ev]	/at/ ⇝ [at]	/adres/ ⇝ [adres]
PL	/ev-lAr/ ⇝ [ev-ler]	/at-lAr/ ⇝ [atlar]	/adres-lAr/ ⇝ [adres-ler]
PL.LOC	/ev-lAr-A/ ⇝ [ev-ler-e]	/at-lAr-A/ ⇝ [at-lar-a]	/adres-lAr-A/ ⇝ [adres-ler-e]

The alternating vowel in the two suffixes is typically viewed as a feature bundle [∘BACK, −HIGH, −ROUND],

Proceedings of *AMP 2019*

here denoted *A*, which is underspecified for backness in its underlying form. When these two suffixes combine with a stem, they receive values for [±back] from the nearest preceding vowel.

Using terminology from Nevins (2010), a traditional analysis for vowel harmony assumes a *donor-based* view of the process, in which features spread from a fully-specified vowel (the *donor*) to underspecified vowels (*recipients*). In (1), the rightmost root vowel is a donor of backness features, while the *A*s in the two suffixes are recipients of backness features. According to the donor-based view, vowel harmony is characterized by three properties.

(2) *Properties of Vowel Harmony*

a. **Donor–Recipient Asymmetry:** Donors lexically bear the features that are being harmonized, while recipients are lexically underspecified for those features.

b. **Directionality:** Individual vowel harmony processes can specify that features must be spread to the left, to the right, or in both directions.

c. **Long-Distance Dependency:** There is no principled upper bound on the distance between a donor and a recipient.

These three properties are naturally captured by rule-based analyses of vowel harmony. These analyses utilize either *simultaneous* or *iterative* rules, which can operate over sequences of segments or autosegmental structures. Examples of such rules for the backness harmony process in (1) are given in (3).

(3) *Rule-Based Analyses of Turkish Backness Harmony*

a. Iterative Rule
$\text{A} \rightarrow [\alpha\text{BACK}] \ / \ [\alpha\text{BACK}]\text{C}^* \ \_\_\_\_\_$

b. Simultaneous Rule
$\text{A} \rightarrow [\alpha\text{BACK}] \ / \ [\alpha\text{BACK}]\{\text{C}, \text{A}\}^* \ \_\_\_\_\_$

In an iterative analysis, a rule is applied repeatedly, changing one segment of the underlying form at a time. At each iteration, the rule operates over the output of its previous iteration. In our running example, an iterative analysis of the backness harmony process in (1) is given by rule (3a), which states that an *A* must take on the feature value $[\alpha\text{BACK}]$ whenever it is preceded by an $[\alpha\text{BACK}]$ vowel followed by a sequence of consonants (the notation C^* denotes a sequence of consonants of arbitrary length). The derivation of the plural locative forms using (3a) are shown in (4) below. Consider the form [ev-ler-e] 'to the houses.' We assume for simplicity that (3a) is not sensitive to morpheme boundaries. First, since the *A* in *-lAr* is preceded in the underlying form by the $[-\text{BACK}]$ vowel *e* and the consonant sequence *vl*, (3a) assigns $[-\text{BACK}]$ features to *A*, yielding *e*. Next, the *A* in *-A* is now preceded by the $[-\text{BACK}]$ vowel *e* in *-ler* along with the consonant *r*, so (3a) assigns $[-\text{BACK}]$ features to this *A* as well, yielding the surface form *ev-ler-e*.

(4) *Iterative Analysis of (1) using* (3a)

	'to the houses'	'to the horses'	'to the addresses'
Underlying Form	/ev-lAr-A/	/at-lAr-A/	/adres-lAr-A/
Harmony	ev-ler-A	at-lar-A	adres-ler-A
Harmony	[ev-ler-e]	[at-lar-a]	[adres-ler-e]

On the other hand, a simultaneous analysis assumes that the rule is applied simultaneously to all underlying segments that occur in the appropriate environment. The simultaneous rule (3b), for example, states that an *A* adopts $[\alpha\text{BACK}]$ features whenever it is preceded by an $[\alpha\text{BACK}]$ vowel and an arbitrary sequence of consonants and *A*s. Using the simultaneous rule, the mapping /ev-lAr-A/ → [ev-ler-e] is derived as follows. As in the iterative analysis, since the *A* in *-lAr* is preceded by the $[-\text{BACK}]$ vowel *e* and the consonant sequence *vl*, (3b) assigns $[-\text{BACK}]$ features to *A*, yielding *e*. Next, since the *A* in *-A* is preceded by the *e* in the root *ev* along with the sequence *vlAr*, (3b) changes this *A* to *e* as well. This derivation is illustrated in (5).

(5) *Simultaneous Analysis of (1) using (3b)*

	'to the houses'	'to the horses'	'to the addresses'
Underlying Form	/ev-lAr-A/	/at-lAr-A/	/adres-lAr-A/
Harmony	[ev-ler-e]	[at-lar-a]	[adres-ler-e]

3 Search & Copy

In contrast to the donor-based view, S&C takes a *recipient-based* approach to vowel harmony, which is argued by Nevins (2005) and others to be computationally different from the donor-based view. S&C replaces the iterative and simultaneous rules of (3) with a *search* operation. Under the recipient-based view, underspecified segments *search* for a *target* that matches a given feature specification. If a target is found, it serves as a donor, *copying* its features to the recipient that triggered the search. SEARCH and COPY are defined by Mailhot & Reiss (2007) as two operations, described by rules of the following form.

(6) a. SEARCH: From a recipient ς matching feature specification R, search in the direction δ for a target γ matching feature specification F. If such a target exists, the target γ found by the search is the nearest eligible segment to ς in the direction δ.

b. COPY: From a target γ, copy the values for the features G to the recipient ς if γ matches the feature specification C.

To illustrate, the backness harmony process in (1) can be analyzed by the following S&C rules.

(7) *S&C Analysis of Turkish Backness Harmony*

a. From A, SEARCH left for a target matching the feature specification $[\pm\text{BACK}]$.

b. From γ, COPY $[\alpha\text{BACK}]$ to A.

Thus, the mapping /ev-lAr-A/ $\rightarrow$ [ev-ler-e] is derived as follows. First, the *A* in *-lAr* searches for a $[\pm\text{BACK}]$ vowel using (7a), and identifies the *e* in *ev*. The *e*'s $[-\text{BACK}]$ feature is copied to the *A* using (7b), yielding *e*. Next, the *A* in *-A* searches for a $[\pm\text{BACK}]$ vowel using (7a), and identifies the *e* in *-ler*. The *e*'s $[-\text{BACK}]$ feature is copied to the *A* using (7b), yielding *e*. Like the rule-based analyses, S&C can be applied iteratively (e.g., Samuels, 2009b) or simultaneously (e.g., Samuels, 2011). The derivation we have described is iterative. In a simultaneous analysis, the target of the second *A*'s search is the root vowel *e* from *ev* rather than the the *e* in *-ler*.

4 Subregular Computation

We have now reviewed relevant background on vowel harmony and on S&C. In this section, we introduce the framework under which we analyze the S&C formalism. In subregular phonology and in mathematical linguistics more generally, phonological transformations are modeled as functions that take a string as input and produce another string as output. These strings are usually interpreted as sequences of phonemes or other atomic symbols such as prosodic markings or morphological boundary markers. The input to the function is the underlying form of a word, while the output of the function is the surface form or intermediate form produced by the transformation. Under this framework, we formalize vowel harmony processes as functions whose inputs are sequences of phonemes, some of which may be underspecified for certain features, and whose outputs are sequences of phonemes that are all fully specified for all possible features. In the following subsections, we show how this framework can be applied to vowel harmony, and we describe various classes of functions that have been used to describe different kinds of phonological phenomena.

4.1 *Vowel Harmony as a String-to-String Function* To illustrate our framework, let us return to the backness harmony process of (1). This process is modeled as a function that takes an input, which may contain one or more instances of *A*, and changes all *A*s to either *a* or *e*.[1] A simple algorithm implementing this vowel harmony function is given in (8).

(8) *Algorithm for Turkish Backness Harmony*
Input: An underlying form $x = x_1x_2 \dots x_n$.
Output: A surface form $y = y_1y_2 \dots y_n$.

a. Initialize a variable f to the value $[\circ\text{BACK}]$.

b. For each i from 1 to n:

[1] Technically, *A* may still surface as an underspecified vowel if the root as no vowels. We do not address this issue here, since it is not relevant to the present discussion.

i. If x_i is specified for $[\pm\text{BACK}]$ and has a feature value $[\alpha\text{BACK}]$, set f to be $[\alpha\text{BACK}]$ and set y_i to be x_i.

ii. If x_i is not specified for $[\pm\text{BACK}]$, set y_i to be a segment with the features of x_i along with the feature value f.

c. Return $y_1 y_2 \dots y_n$ as output.

The algorithm reads the underlying form from left to right, one segment at a time, and constructs the surface form incrementally as it does so. Throughout the computation, the algorithm maintains a variable f, which records the backness feature of the most recently seen vowel that is specified for $[\pm\text{BACK}]$. Whenever an *A* is seen, it is assigned the feature value recorded in the variable f. For example, the mapping /ev-lAr-A/ → [ev-ler-e] is executed by (8) as follows. First, upon seeing *e*, the algorithm sets the value of the variable f to $[-\text{BACK}]$. When the algorithm reaches the first *A*, the value of f is still $[-\text{BACK}]$, so the *A* is assigned the $[-\text{BACK}]$ feature, yielding *e*. When the algorithm reaches the second *A*, the value of f is still $[-\text{BACK}]$, so this *A* also becomes *e* in the output. This computation is illustrated in table (9), which shows the values of x_i, f, and y_i at each point in the computation.

(9) *Computation of /ev-lAr-A/ → [ev-ler-e] using (8)*

i	x_i	f	y_i
		$[\circ\text{BACK}]$	
1	e	$[-\text{BACK}]$	e
2	v	$[-\text{BACK}]$	v
3	l	$[-\text{BACK}]$	l
4	A	$[-\text{BACK}]$	e
5	r	$[-\text{BACK}]$	r
6	A	$[-\text{BACK}]$	e

4.2 *Finite-State Functions* The algorithm (8) has several interesting computational properties.

(10) *Properties of Algorithm (8)*

a. **Streaming:** The algorithm reads its input exactly once, one segment at a time. After the last segment is read, it is not allowed to read any part of the input again.

b. **Finite-Stateness:** The variables maintained in the algorithm's memory (the *state* of the algorithm) can only take on values from a finite set of possibilities. In this case, those possibilities are $[\circ\text{BACK}]$, $[+\text{BACK}]$, and $[-\text{BACK}]$.

c. **Determinism:** At each point during the computation, the algorithm's behavior is completely determined by its state and the identity of the current input segment being read. In particular, the algorithm cannot maintain a set of possible output forms, to be pruned at a later point in the computation.

Functions implemented by algorithms that satisfy the streaming and finite-stateness conditions are known as *finite-state functions* (Kaplan & Kay, 1994). If the algorithm additionally satisfies the determinism condition, then we say that the function is *subsequential* (Raney, 1958; Schützenberger, 1977). Since the algorithm (8) satisfies all three criteria, the Turkish backness harmony process is subsequential.

Subsequential functions can be depicted visually using *state diagrams*, such as the one shown in Figure 1. A state diagram consists of a collection of labeled circles, connected to one another by labeled arrows. Each circle represents a possible configuration of the algorithm's state. In Figure 1, the three circles represent the three possible values of the variable f. Each arrow represents how the algorithm's state might change. For example, the arrow

$$[\circ\text{BK}] \xrightarrow{\text{V}[+\text{BK}]:\text{V}} [+\text{BK}]$$

means that if $f = [\circ\text{BACK}]$ and x_i is a vowel V carrying the feature $[+\text{BACK}]$, then f changes to $[+\text{BK}]$, and $y_i = \text{V}$ is added to the output. Similarly, the arrow

$$[+\text{BK}] \xrightarrow{\text{V}[+\text{BK}]:\text{V, A:e}} [+\text{BK}]$$

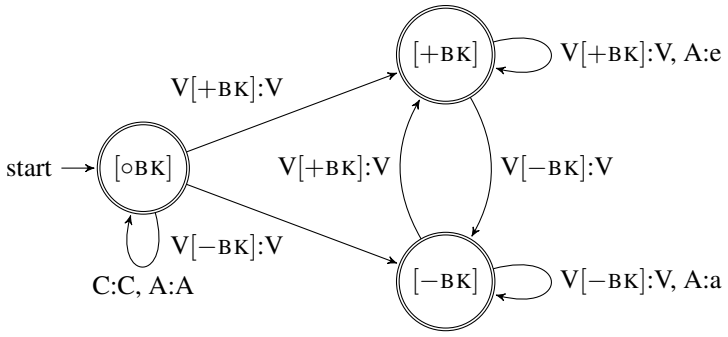

Figure 1: A state diagram for (8).

means that if $f = [+\text{BACK}]$ and x_i is either A or a vowel V[+BACK], then the variable f is unchanged. If $x_i = \text{A}$, then $y_i = \text{e}$ is added to the output, and otherwise, $y_i = x_i$ is added to the output.

In terms of state diagrams, the finite-stateness criterion may be formally defined as the requirement that the state diagram contains finitely many circles. The determinism criterion may be defined as the requirement that the state diagram cannot contain two arrows of the form $q \xrightarrow{\ldots x:y\ldots} r$ and $q \xrightarrow{\ldots x:z\ldots} s$ such that $r \neq s$. Gainor et al. (2012) have shown that, within this framework, all *unidirectional* vowel harmony patterns patterns appearing in the typology of Nevins (2010) can be modeled by subsequential functions. *Bidirectional* patterns, however, cannot be modeled by subsequential functions. Intuitively, this is because the streaming criterion requires that subsequential algorithms read the input string in a single direction, and not in both directions at the same time. Heinz & Lai (2013) have proposed that such processes should be viewed as the composition of a subsequential function with a *right-subsequential* function, which is a function computed by a subsequential algorithm that reads the input from right to left. Functions that can be decomposed in this way are known as *weakly deterministic* functions. Patterns requiring the extra expressivity of weakly deterministic functions are limited to stem-controlled harmony.

4.3 *Locality and Tiers* Finite-state and subsequential functions have enjoyed substantial interest in computational phonology for two reasons. Firstly, finite-state algorithms are computationally efficient, and finite-state functions can be composed with one another, allowing for modular grammar engineering. Secondly, finite-state functions have wide empirical coverage of phonological phenomena. Kaplan & Kay (1994) have shown that the application of SPE-style rules can be implemented using finite-state algorithms, meaning that all phonological transformations admitting a rule-based analysis can be modeled by finite-state functions.[2]

Despite the computational simplicity of finite-state and subsequential functions, Heinz (2010) and Chandlee (2014) have proposed additional restrictions on the form of phonological algorithms in the interest of facilitating learnability in the paradigm of Gold (1967) as well as providing detailed descriptions of phonological typology in terms of computational properties. The first of these is based on the *local languages* of McNaughton & Papert (1971) and the *local functions* of Vaysse (1986).

(11) **Strict Locality:** At each point during the computation, the state of the algorithm can only record the identities of the previous input segments $x_{i-1}, x_{i-2}, \ldots, x_{i-k}$, for some $k \geq 0$.

When a function is computed by a subsequential algorithm that additionally obeys the strict locality criterion for a particular value of k, we say that the function is $(k+1)$-*input strictly local* ($(k+1)$-ISL, Chandlee, 2014). The strict locality criterion formalizes the intuitive notion of locality: phonological alternations can only depend on nearby segments of the input. In order to allow for longer-distance dependencies, Heinz et al.

[2] A notable exceptions to finite-stateness is reduplication (Culy, 1985; Roark & Sproat, 2007; Dolatian & Heinz, 2018, 2019).

(2011) have proposed a relaxation of the strict locality criterion that incorporates the notion of phonological tiers.

(12) **Tier-Based Strict Locality:** The state of the algorithm can only record the identities of the k previous input segments that belong to a special set of segments T, for some $k \geq 0$ and for some set T.

Functions computed by subsequential algorithms obeying the tier-based strict locality criterion are known as $(k+1)$*-input tier-based strictly local* functions ($(k+1)$-TISL, Hao & Andersson, 2019; Hao & Bowers, 2019). Intuitively, the set T represents a phonological tier, and transformations modeled by TISL functions enforce dependencies defined over a phonological tier.[3]

4.4 *Tier-Based Strict Locality in Vowel Harmony* Observe that algorithm (8) can easily be reformulated to meet the tier-based strict locality criterion. In (8), the variable f is recording the backness feature of the most recent fully-specified vowel in the underlying form. The same information can be obtained by recording the identity of the most recent fully-specified vowel instead of its backness feature. In the computation of (9), the values of f would then be $\varnothing$ and e instead of $[\circ\text{BACK}]$ and $[-\text{BACK}]$, respectively. Since only the most recent fully-specified vowel needs to be remembered, the parameter k in the tier-based strict locality criterion has a value of 1, and the tier parameter T is the set of fully-specified vowels. Thus, the Turkish backness harmony process is 2-TISL.

5 Search & Copy as a Tier-Based Input Strictly Local Function

In the previous section, we saw how the backness harmony process of Turkish can be viewed as a 2-TISL function. In this section we extend the algorithm (8) to vowel harmony processes in general by proposing a schema to implement the SEARCH and COPY operations using algorithms of this form. We focus here on implementing S&C analyses consisting of a single SEARCH rule followed by a single COPY rule.

The description (6) of S&C suggests the following naïve algorithm for S&C.[4]

(13) *Naïve Algorithm for S&C*
Input: An underlying form $x = x_1 x_2 \dots x_n$.
Output: A surface form $y = y_1 y_2 \dots y_n$.
Parameters: The directionality parameter $\delta = \pm 1$; feature specifications R, F, C, and G.

a. For each i from 1 to n:
   i. If x_i matches the feature specification R:
      - For each j from 1 to $i-1$:
        - If $x_{i+\delta j}$ does not match the feature specification F, continue to the next iteration of this loop.
        - If $x_{i+\delta j}$ matches the feature specification C, set y_i to be a segment with the feature values of G matching those of $x_{i+\delta j}$, along with the features of x_i.
        - Break out of this loop.
   ii. Otherwise, set y_i to be x_i.
b. Return $y_1 y_2 \dots y_n$ as output.

[3] Since Heinz et al. (2011), there has been a substantial amount of work in showing how the tier-based strictly local criterion can model linguistic phenomena. Tier-based formalisms have mostly been used to model long-distance processes, such as consonant harmony (McMullin, 2016; McMullin & Hansson, 2016), vowel harmony (Aksënova & Deshmukh, 2018; Mayer & Major, 2018), morphotactics (Aksënova et al., 2016), and stress (Hao & Andersson, 2019). Additionally, learnability studies have produced a variety of grammatical inference algorithms for tier-based formalisms (Jardine, 2016; Jardine & Heinz, 2016; Jardine & McMullin, 2017), as well as artificial learning experiments supporting the relevance of tier-based formalisms for cognition (McMullin & Hansson, 2019). Various extensions or refinements to tier-based formalisms have been proposed to handle diverse types of locality domains and blockers over phonological (Graf, 2017; Graf & Mayer, 2018; De Santo, 2018; De Santo & Graf, 2019; Karakaş, 2020), prosodic (Baek, 2018; Hao, 2020), and morphological structures (Aksënova & De Santo, 2018; Moradi et al., 2019). These extensions have some learnability results (McMullin et al., 2019; Burness & McMullin, 2019).

[4] The algorithms stated in this section implement S&C in the simultaneous mode. The algorithms for the iterative mode can be obtained by replacing x_i with y_i in the appropriate places.

The conditional block (13a-i) implements the SEARCH operation, which is triggered when x_i matches the triggering feature specification R. When $\delta = 1$, $x_{i+\delta j}$ occurs to the right of x_i, and when $\delta = -1$, $x_{i+\delta j}$ occurs to the left of x_i. The search continues until $x_{i+\delta j}$ matches the termination criterion F. If $x_{i+\delta j}$ matches the copy criterion C, then the feature values for G of $x_{i+\delta j}$ are copied to the output y_i. While this algorithm captures the SEARCH and COPY procedures defined by Mailhot & Reiss (2007), notice that it violates the streaming criterion of finite-state algorithms. Whereas streaming requires that the algorithm read the input only once, the search operations triggered by underspecified vowels cause previous input segments to be re-read when $\delta = -1$. When $\delta = 1$, the segments after a recipient are read for the first time when the SEARCH is triggered, and they are re-read after the SEARCH has terminated, when the algorithm continues the outer loop that looks for triggers. Thus, the naïve algorithm for S&C is not a finite-state algorithm.[5]

In order to implement S&C in a streaming manner, we need to ensure that triggering a search does not result in re-reading symbols. To do this, we exploit the fact that the target of a SEARCH is always the nearest eligible segment to the trigger. Let us first assume that $\delta = -1$, so that searches are conducted from right to left. While reading an input sequence from left to right, when the algorithm encounters a segment x_i matching the termination criterion F, we know that any future search will terminate at this segment as long as it is triggered before the next input symbol matching F is encountered. Thus, upon encountering an eligible target x_i, we can simply remember the identity of x_i so that when a trigger x_j is encountered at a future time step, the appropriate features can be assigned to the output y_j via COPY based on x_i. This algorithm is fully described in (14).

(14) *TISL Algorithm for S&C*
Input: An underlying form $x = x_1 x_2 \dots x_n$.
Output: A surface form $y = y_1 y_2 \dots y_n$.
Parameters: Feature specifications R, F, C, and G.
   a. Initialize a variable t to the value $\varnothing$.
   b. For each i from 1 to n:
      i. If x_i matches the feature specification F, set t and y_i to be x_i.
      ii. If x_i matches the feature specification R and t matches the feature specification C, then set y_i to be a segment with the features of x_i, along with the values of G carried by t.
   c. Return $y_1 y_2 \dots y_n$ as output.

This algorithm, unlike the naïve algorithm, satisfies the streaming criterion. The variable t records the identity of the most recent vowel matching the feature specification F. Thus, (14) also satisfies the tier-based strictly local criterion, with $k = 1$ and T being the set of vowels matching the feature specification F. This means that S&C can be represented as a 2-TISL function when the SEARCH is conducted from right to left.

In the case where SEARCH is conducted from left to right (i.e., $\delta = 1$), we simply change the for-loop in (14b) to read *For each i from n to 1*. This causes the algorithm to read the input sequence from right to left instead of from left to right. Functions computed by such algorithms are known as *right-to-left tier-based input strictly local functions* (right-TISL).

The TISL algorithm for S&C has the advantage that it is less computationally complex than the naïve algorithm. Because the naïve algorithm causes the input string to be re-read every time a SEARCH is triggered, its asymptotic running time is a quadratic function of the length of its input $O(n^2)$. On the other hand, because the TISL algorithm only reads its input once, its running time is linear in the length of its input $O(n)$. Thus, the TISL algorithm is a much more efficient implementation of S&C than the naïve algorithm.

6 Variants of Search & Copy

Let us summarize our results so far.

(15) *Interim Summary*
   a. When SEARCH is conducted from right to left, S&C describes processes modeled by 2-TISL functions.

[5] In fact, the naive implementation of S&C resembles the behavior of finite-state transducers which are augmented with pebbles (Engelfriet & Maneth, 2002; Engelfriet, 2015).

b. When SEARCH is conducted from left to right, S&C describes processes modeled by right-2-TISL functions.

Our results from the previous section only apply to individual processes consisting of a single unidirectional SEARCH rule combined with a single COPY rule. In this section we show that our results do not hold when these assumptions are violated.

6.1 *Bidirectional Search & Copy* *Bidirectional vowel harmony* is a type of vowel harmony in which donors of a feature are found to the left and to the right of a recipient. An example of bidirectional vowel harmony occurs in Woleaian (Howard, 1972). In this language, the underspecified vowel *A* surfaces as [+HIGH] if there is a [+HIGH] donor to the left *and* to the right of *A*. This condition is met in the 1SG form below, and accordingly, the *A* surfaces as *e*. The condition is not met in the 3SG form, however, because the *a* to the right of *A* is [−HIGH].

(16) *Bidirectional Vowel Harmony in Woleaian (Howard, 1972)*

	'drinking object'
Independent form	/ülüm/ ⇝ [ü:l]
1SG	/ülüm-A-ji/ ⇝ [ülümej]
3SG	/ülüm-A-la/ ⇝ [ülümal]

Nevins (2010) proposes the following S&C analysis of the Woleaian bidirectional harmony process.

(17) *S&C Analysis of Woleaian Bidirectional Harmony (Nevins, 2010)*

a. From A, SEARCH left and right for targets γ_1 and γ_2 matching the feature specification [±HIGH].

b. From γ_1 and γ_2, COPY [+HIGH] to A only if *both* γ_1 and γ_2 match the specification [+HIGH]. Otherwise, assign A the default value of [−HIGH].

In this analysis, the directionality parameter δ specifies that the SEARCH triggered by *A* should be conducted in both directions. When the *A* is equidistant from two appropriate donors, the COPY step is successful only if both donors contribute the same feature value to the recipient. Otherwise, the *A* receives a default surface form *a*.

Nevins's (2010) analysis poses two problems for the subsequential criteria. Firstly, the streaming condition cannot be satisfied if the value of the recipient's [±HIGH] feature depends on donors on both sides of the recipient. As previously mentioned, Gainor et al. (2012) have already shown that bidirectional vowel harmony processes cannot be modeled as subsequential functions for this reason, though they can be modeled as weakly deterministic functions.

Secondly, Nevins (2010) is ambiguous in terms of whether the two donors need to be *equidistant* to the recipient in order for SEARCH to terminate. If equidistance is required, then SEARCH may fail to find two suitable targets if none are equidistant to the recipient. Alternatively, SEARCH may terminate early if a target is found on one side of the input even if no target has yet been found on the other side of the input. In either case, equidistance violates the finite-stateness condition. Intuitively, this is because enforcing equidistance requires counting the distance between each potential target and the trigger and comparing the two distances. Since counting requires the state to take on infinitely many possible values, it is not compatible with finite-stateness.

6.2 *Multiple Harmony Processes in One Language* We have shown that individual spreading processes modeled by unidirectional S&C are TISL. However, since the TISL functions are not closed under composition (Sakarovitch, 2009; Chandlee et al., 2018), two harmony processes may combine to produce a non-TISL process when they are applied to an underlying form one after the other.

To illustrate, consider the interaction between backness harmony in Turkish with *roundness harmony*, illustrated in (18).

(18) *Backness and Roundness Harmony in Turkish*

	'stalk'	'end'
SG	/sap/ ⇝ [sap]	/son/ ⇝ [son]
PL	/sap-lAr/ ⇝ [sap-lar]	/son-lAr/ ⇝ [son-lar]
GEN	/sap-In/ ⇝ [sap-ɨn]	/son-In/ ⇝ [son-un]
PL.GEN	/sap-lAr-In/ ⇝ [sap-lar-ɨn]	/son-lAr-In/ ⇝ [son-lar-un]

The genitive suffix *-In* contains an underspecified vowel *I* bearing the features $[\circ\text{BACK}, -\text{HIGH}, \circ\text{ROUND}]$. In the singular, the *I* in *-In* receives its backness and roundness features from the root vowel. In the plural, *I* gets its backness feature from the plural suffix and its roundness feature from the root vowel. Any monolithic implementation of the two harmony processes as a single function must violate the tier-based strict locality condition. This is because the backness features of *I* depend on *A*, so *A* must be part of the tier set T; however, the roudness features of *I do not* depend on *A*, so *A* cannot be part of the tier set T. This is clearly a contradiction.

7 Conclusion

In analyzing the expressive power of S&C within the framework of subregular phonology, we have given a TISL implementation of SEARCH and COPY in their unidirectional mode of application. Our analysis enhances Gainor et al.'s (2012) previous results in two ways. Firstly, our TISL characterization of S&C is tighter than their subregular characterization, and therefore our analysis consitutes a more precise hypothesis regarding the range of possible vowel harmony patterns. Secondly, applying our analysis to the S&C framework rather than to individual vowel harmony processes improves the genrality of Gainor et al.'s (2012) result, since our analysis implies that vowel harmony patterns not included in Nevins's (2010) typology are TISL or right-TISL as long as they can be analyzed within unidirectional S&C.

An interesting corollary of our analysis is the observation that the TISL implementation of S&C is more computationally efficient than the procedure described by Mailhot & Reiss's (2007) formulation of S&C. Although Mailhot & Reiss and others give conceptual arguments for thinking of S&C in those terms, our observation suggests that searches triggered by targets may not be the most computationally natural way to implement vowel harmony.

Our work raises the following questions for future inquiry.

1. Can bidirectional harmony processes be analyzed without resorting to non-finite-state means? Does the bidirectional vowel harmony process of Woleaian require equidistance?

2. Are there formal restrictions on how S&C rules may be combined with one another? Is there a natural formal description of the class of transformations obtained by composing vowel harmony processes with one another?

3. How does S&C compare formally to other accounts of vowel harmony, such as simultaneous and iterated rule-based accounts? Are there length-preserving TISL functions that cannot be analyzed within S&C?

References

Aksënova, Alëna & Aniello De Santo (2018). Strict locality in morphological derivations. *Proceedings of the 53rd meeting of Chicago Linguistics Society (CLS 53)*.

Aksënova, Alëna & Sanket Deshmukh (2018). Formal restrictions on multiple tiers. *Proceedings of the Society for Computation in Linguistics*, 64–73.

Aksënova, Alëna, Thomas Graf & Sedigheh Moradi (2016). Morphotactics as tier-based strictly local dependencies. *Proceedings of the 14th sigmorphon workshop on computational research in phonetics, phonology, and morphology*, 121–130.

Baek, Hyunah (2018). Computational representation of unbounded stress: Tiers with structural features. *Proceedings of the Chicago Linguistics Society*, vol. 53, 13–24.

Beesley, Kenneth R. & Lauri Karttunen (2003). *Finite-State Morphology*. CSLI Studies in Computational Linguistics, CSLI Publications, Stanford, CA.

Burness, Phillip & Kevin McMullin (2019). Efficient learning of output tier-based strictly 2-local functions. *Proceedings of the 16th Meeting on the Mathematics of Language*, 78–90.

Chandlee, Jane (2014). *Strictly Local Phonological Processes*. PhD Dissertation, University of Delaware, Newark, DE.

Chandlee, Jane, Rémi Eyraud & Jeffrey Heinz (2015). Output Strictly Local Functions. Kuhlmann, Marco, Makoto Kanazawa & Gregory M. Kobele (eds.), *Proceedings of the 14th Meeting on the Mathematics of Language*, Association for Computational Linguistics, Chicago, IL, 112–125.

Chandlee, Jane, Jeffrey Heinz & Adam Jardine (2018). Input Strictly Local opaque maps. *Phonology* 35:2, 171–205.

Culy, Christopher (1985). The complexity of the vocabulary of Bambara. *Linguistics and philosophy* 8, 345–351.

De Santo, Aniello (2018). Extending tsl to account for interactions of local and non-local constraints. *Proceedings of the Society for Computation in Linguistics*, vol. 1.

De Santo, Aniello & Thomas Graf (2019). Structure sensitive tier projection: Applications and formal properties. *Formal Grammar. Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, vol. 11668.

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

Dolatian, Hossep & Jeffrey Heinz (2019). Reduplication with finite-state technology. *Proceedings of the 53rd Annual Meeting of the Chicago Linguistics Society*.

Engelfriet, Joost (2015). Two-way pebble transducers for partial functions and their composition. *Acta Informatica* 52:7-8, 559–571.

Engelfriet, Joost & Sebastian Maneth (2002). Two-way finite state transducers with nested pebbles. *International Symposium on Mathematical Foundations of Computer Science*, Springer, 234–244.

Gainor, Brian, Regine Lai & Jeffrey Heinz (2012). Computational Characterizations of Vowel Harmony Patterns and Pathologies. *Proceedings of the 29th West Coast Conference on Formal Linguistics*, Cascadilla Proceedings Project, Somerville, MA, 63–71.

Gold, E Mark (1967). Language identification in the limit. *Information and Control* 10:5, 447–474.

Graf, Thomas (2017). The power of locality domains in phonology. *Phonology* 34:2, 385–405.

Graf, Thomas & Connor Mayer (2018). Sanskrit n-retroflexion is input-output tier-based strictly local. *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Association for Computational Linguistics, Brussels, Belgium, 151–160.

Hao, Yiding (2020). Metrical grids and generalized tier projection. *Proceedings of the Society for Computation in Linguistics*, vol. 3.

Hao, Yiding & Samuel Andersson (2019). Unbounded Stress in Subregular Phonology. *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Association for Computational Linguistics, Florence, Italy, 135–143.

Hao, Yiding & Dustin Bowers (2019). Action-Sensitive Phonological Dependencies. *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Association for Computational Linguistics, Florence, Italy, 218–228.

Heinz, Jeffrey (2010). String Extension Learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Uppsala, Sweden, 897–906.

Heinz, Jeffrey (2018). The computational nature of phonological generalizations. Hyman, Larry M. & Frans Plank (eds.), *Phonological Typology*, no. 23 in Phonology and Phonetics, De Gruyter Mouton, Berlin, Germany, 126–195.

Heinz, Jeffrey & Regine Lai (2013). Vowel Harmony and Subsequentiality. *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, Association for Computational Linguistics, Sofia, Bulgaria, 52–63.

Heinz, Jeffrey, Chetan Rawal & Herbert G Tanner (2011). Tier-based strictly local constraints for phonology. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, Association for Computational Linguistics, 58–64.

Howard, Irwin (1972). *A Directional Theory of Rule Application in Phonology*. PhD Dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Jardine, Adam (2016). Learning tiers for long-distance phonotactics. *Proceedings of the 6th Conference on Generative Approaches to Language Acquisition North America (GALANA 2015)*, 60–72.

Jardine, Adam & Jeffrey Heinz (2016). Learning tier-based strictly 2-local languages. *Transactions of the Association for Computational Linguistics* 4, 87–98.

Jardine, Adam & Kevin McMullin (2017). Efficient learning of tier-based strictly k-local languages. *International Conference on Language and Automata Theory and Applications*, Springer, 64–76.

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

Karakaş, Ayla (2020). An ibsp description of sanskrit /n/-retroflexion. *Proceedings of the Society for Computation in Linguistics*, vol. 3.

Mailhot, Frédéric & Charles Reiss (2007). Computing Long-Distance Dependencies in Vowel Harmony. *BIOLINGUISTICS* 1, 028–048.

Mayer, Connor & Travis Major (2018). A challenge for tier-based strict locality from uyghur backness harmony. *International Conference on Formal Grammar 2018*, Springer, Berlin/Heidelberg, 62–83.

McMullin, Kevin & Gunnar Olafur Hansson (2016). Long-distance phonotactics as tier-based strictly 2-local languages. *Proceedings of the Annual Meetings on Phonology*, vol. 2.

McMullin, Kevin & Gunnar Olafur Hansson (2019). Inductive learning of locality relations in segmental phonology. *Laboratory Phonology: Journal of the Association for Laboratory Phonology* 10:1.

McMullin, Kevin James (2016). *Tier-based locality in long-distance phonotactics: learnability and typology*. Ph.D. thesis, University of British Columbia.

McMullin, Kevin, Alëna Aksënova & Aniello De Santo (2019). Learning phonotactic restrictions on multiple tiers. *Proceedings of the Society for Computation in Linguistics*, vol. 2, 377–378.

McNaughton, Robert & Seymour A. Papert (1971). *Counter-Free Automata*. No. 65 in Research Monograph, MIT Press, Cambridge, MA.

Moradi, Sedigheh, Alëna Aksënova & Thomas Graf (2019). The computational cost of generalizations: An example from micromorphology. *Proceedings of the Society for Computation in Linguistics*, vol. 2, 367–368.

Nevins, Andrew (2005). *Conditions on (Dis)Harmony*. PhD Dissertation, Massachusetts Institute of Technology.

Nevins, Andrew (2010). *Locality in vowel harmony*, vol. 55. Mit Press.

Raney, George N. (1958). Sequential Functions. *Journal of the Association for Computing Machinery* 5:2, 177–180.

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

Sakarovitch, Jacques (2009). *Elements of Automata Theory*. Cambridge University Press, Cambridge, United Kingdom.

Samuels, Bridget D. (2009a). Structure & specification in harmony. *Proceedings of the Thity-Eighth Annual Meeting of the North East Linguistics Society*, GLSA Publications, Amherst, MA, vol. 2, 283–296.

Samuels, Bridget D. (2009b). *The Structure of Phonological Theory*. PhD Dissertation, Harvard University, Cambridge, MA.

Samuels, Bridget D (2011). *Phonological architecture: A biolinguistic perspective*. Oxford Studies in Biolinguistics, Oxford University Press.

Schützenberger, M. P. (1977). Sur une variante des fonctions sequentielles. *Theoretical Computer Science* 4:1, 47–57.

Vaysse, Odile (1986). Addition molle et fonctions p-locales. *Semigroup Forum* 34:1, 157–175.