Compiling Search & Change Rules into Subsequential Finite-State Transducers

Malek Azadegan
City University of New York, The Graduate Center
mazadegan@gc.cuny.edu

Abstract

Search & Change (S&C) is a procedural model of phonological rule application that is conceptually clear and linguistically motivated, but whose computational properties have not been fully characterized. This paper provides a formal specification of S&C within the framework of Logical Phonology, presents a linear-time algorithm for rule application with a proof of correctness, and gives a compilation procedure mapping S&C rules to a single transition structure that is subsequential in one scan orientation and reverse-subsequential in the other, situating S&C within a well-understood subclass of regular string-to-string functions with known learnability guarantees and algebraic characterizations, implying that S&C-definable mappings are learnable from positive input/output pairs and amenable to algebraic classification.

1 Introduction

A central goal in computational phonology is to characterize the computational nature of phonological patterns using theorems and proofs (Heinz and Idsardi, 2017). Search & Change (S&C; Dabbous et al., 2021, 2024) is a procedural model of phonological rule application that is well-suited to this treatment. S&C frames phonological processes explicitly: target segments initiate directional scans through a word, looking for triggers, and the target segment is transformed if the first trigger encountered satisfies a licensing condition.

Despite its explicit nature, S&C lacks a precise formal specification: the literature provides no rigorous definition of its data structures, no provably correct algorithm for computing derivations, and no characterization of the functions it expresses. Logical Phonology (LP; Bale et al., 2019; Reiss, 2021; Gorman and Reiss, 2025a), a research program that treats phonological objects as mathematical objects subject to explicit definition, provides the formal language needed to make S&C amenable to formal definition and proofs. Building on this framework, the paper proceeds in three steps. First, the underlying data structures, rule format, and execution semantics of S&C are defined. Second, a linear-time algorithm for rule application is presented with a proof of correctness. Third, a compilation procedure that maps S&C rules to subsequential or reverse-subsequential finite-state transducers is given, placing S&C within a well-understood subclass of regular string-to-string functions with known learnability guarantees and algebraic characterizations. With this roadmap in place, the next section lays out the formal preliminaries.

2 Preliminaries

S&C operates over phonological representations built from three primitives: valued features, segments, and natural classes (Dabbous et al., 2021). The following subsections define these primitives and establish the vocabulary used throughout the paper.

2.1 Segments as Sets of Valued Features

Given a universal, finite feature set $\mathcal{F}$ (Chomsky and Halle, 1968; Reiss and Volenec, 2022), a *valued feature* is a member of $\{+,-\}\times\mathcal{F}$. Following Bale et al. (2014, 2019), to abbreviate valued features, $+$VOICE and $-$BACK are written for $(+,\textsc{Voice})$ and $(-,\textsc{Back})$, respectively.

A *segment* x is a consistent set of valued features such that $\forall f \in \mathcal{F}\ \neg\big(+f \in x \wedge -f \in x\big)$. Allowing underspecification (Gorman and Reiss, 2025a), a segment can be defined as a total function $\sigma : \mathcal{F} \to \{\emptyset,+,-\}$, where $\sigma(f) = \emptyset$ indicates underspecification for feature f. The set-based and function-based representations are isomorphic: a total function σ can be represented as the set

$$\{(v,f) \in \{+,-\}\times\mathcal{F} \mid \sigma(f) = v\}.$$

Under this formulation, set consistency corresponds exactly to functionality: no segment can contain

Proceedings of the Society for Computation in Linguistics (SCiL) 2026, pages 171–179

both $+f$ and $-f$ because σ assigns at most one value to each feature. This lets us use whichever representation is more convenient below: total functions for defining the segment space $\Omega(\mathcal{F})$, and set operations for stating feature-changing transformations. The set of all segments is then defined as

$$\Omega(\mathcal{F}) \;=\; \{\, \sigma \mid \sigma : \mathcal{F} \to \{\emptyset, -, +\} \,\} \qquad (1)$$

where $|\Omega(\mathcal{F})| = 3^{|\mathcal{F}|}$. By convention, underspecified segments are denoted with capital letters (e.g., /U/). For example, the vowel /u/ might be represented as $\{+\textsc{Syllabic}, +\textsc{High}, +\textsc{Back}\}$, while an underspecified /U/ lacking a HIGH value could be $\{+\textsc{Syllabic}, +\textsc{Back}\}$. An inconsistent set such as $\{+\textsc{Back}, -\textsc{Back}\}$ violates the consistency condition and is not a valid segment.

2.2 Natural Classes as Sets of Segments

Natural classes are sets of segments defined by shared valued features (Bale and Reiss, 2015; Bale et al., 2019). For any feature specification $F \subseteq (\{+,-\} \times \mathcal{F})$,[1] the natural class defined by F is

$$\mathcal{N}(F) = \{x \in \Omega(\mathcal{F}) \mid F \subseteq x\}.$$

Conventionally, natural classes are written with square brackets, e.g. $[+\textsc{Syllabic}]$; $\mathcal{N}$-notation is used to make explicit that the class is quantified over $\Omega(\mathcal{F})$, which keeps later algorithms and proofs uniform.[2] Specificity is inclusion on specifications: $F_1 \supseteq F_2$ implies $\mathcal{N}(F_1) \subseteq \mathcal{N}(F_2)$ (Gorman and Reiss, 2025b), from which the definition of the universal natural class $\mathcal{N}(\emptyset) = \Omega(\mathcal{F})$ immediately follows.

2.3 Words as Strings of Segments

Words are strings of segments taken from $\Omega(\mathcal{F})$. Formally, $\Omega(\mathcal{F})^*$ is the free monoid over $\Omega(\mathcal{F})$ under concatenation, whose elements are finite sequences $\langle s_1, \ldots, s_n \rangle$ where each $s_i \in \Omega(\mathcal{F})$, with the empty string ϵ acting as the identity element (Heinz, 2018; Dabbous et al., 2021, 2024). In the examples below, IPA symbols are used as a convenient shorthand for segments as defined above; each symbol abbreviates one feature-valued segment. For example, /kæt/ denotes a word $\langle s_1, s_2, s_3 \rangle$ consisting of the segments written /k/, /æ/, and /t/.

[1] Under the identification of total valuations with their sets of non-empty feature assignments, each $x \in \Omega(\mathcal{F})$ may also be viewed as a subset of $(\{+,-\} \times \mathcal{F})$. Segments can therefore serve as feature specifications and thus as arguments to $\mathcal{N}$.

[2] For example, $[+\textsc{Syllabic}] = \mathcal{N}(\{+\textsc{Syllabic}\})$.

3 Search & Change

S&C is "a model of phonological computation characterized by a small set of parameters" (Dabbous et al., 2021). A S&C rule (Dabbous et al., 2021) is a quintuple $\langle \textsc{Inr}, \textsc{Trm}, \textsc{Dir}, \textsc{Out}, \textsc{Cnd} \rangle$ specifying a target class, a trigger class, a search direction, an output function, and a licensing condition (Dabbous et al., 2024).

3.1 Formalizing Search

The search domain is determined by INR, TRM, and DIR: INR and TRM are natural classes of search initiating and terminating segments, and $\textsc{Dir} \in \{\textsc{Left}, \textsc{Right}\}$ determines the search direction. Fix a word $w = \langle s_1, \ldots, s_n \rangle$ and a set TRM. For each position i, let R_i and L_i denote the sets of terminator indices strictly to the right and left of i, respectively. The search function Θ then returns the index of the nearest terminating segment in direction DIR: $\min R_i$ for rightward search and $\max L_i$ for leftward search, and is undefined when the relevant set is empty. The symbols $\downarrow$ and $\uparrow$ indicate definedness and undefinedness in the usual partial-function sense. For each position i in w:

$$\begin{aligned} R_i &= \{\, j \mid j > i \wedge s_j \in \textsc{Trm} \,\} \\ L_i &= \{\, j \mid j < i \wedge s_j \in \textsc{Trm} \,\} \\ \Theta_R(i) &= \begin{cases} \min R_i & R_i \neq \emptyset \\ \uparrow & \text{otherwise} \end{cases} \\ \Theta_L(i) &= \begin{cases} \max L_i & L_i \neq \emptyset \\ \uparrow & \text{otherwise} \end{cases} \end{aligned} \qquad (2)$$

When defined, $s_{\Theta_R(i)}$ or $s_{\Theta_L(i)}$ denotes the terminating segment for s_i under rightward or leftward search respectively.

3.2 Formalizing Change

While INR, TRM, and DIR determine the search domain, CND and OUT decide whether a target segment is eligible for change and what change applies. CND is a natural class and OUT is a function of type $\Omega(\mathcal{F}) \times \Omega(\mathcal{F}) \to \Omega(\mathcal{F})$.

3.2.1 The Condition

If $\Theta_R(i)$ or $\Theta_L(i)$ is defined, let s_j denote the terminating segment at that index. If $s_j \in \textsc{Cnd}$, the rule is licensed to apply OUT to s_i; otherwise s_i is unchanged.

3.2.2 The Output Function

OUT is defined using three operators over $\Omega(\mathcal{F})$ (Bale et al., 2014; Reiss, 2021; Gorman and Reiss, 2025b):

Subtraction. Removes valued features from segments (Bale et al., 2014):

$$A \setminus B = \{c\text{F} \mid c\text{F} \in A \wedge c\text{F} \notin B\} \quad (3)$$

Subtraction is essentially set difference: only valued features present in A but absent from B survive. It applies vacuously whenever B is empty or shares no elements with A. Consider

1. $\{+\text{F}, +\text{G}, -\text{H}\} \setminus \{+\text{G}\} = \{+\text{F}, -\text{H}\}$
2. $\{+\text{F}, -\text{G}\} \setminus \emptyset = \{+\text{F}, -\text{G}\}$
3. $\{+\text{F}, -\text{G}\} \setminus \{+\text{H}\} = \{+\text{F}, -\text{G}\}$.

Case 1 removes $+$G from A since it appears in B. Cases 2 and 3 are vacuous: in case 2 there is nothing to subtract; in case 3, $+\text{H} \notin A$, so the intersection $A \cap B$ is empty and nothing is removed.

Unification. Adds valued features without creating inconsistency (Bale et al., 2014):

$$A \sqcup B = A \cup \{c\text{F} \mid c\text{F} \in B \wedge -c\text{F} \notin A\} \quad (4)$$

Unlike plain set union, $\sqcup$ skips any feature in B for which A already carries a specification, preventing inconsistency; when A and B disagree on a feature, A takes priority, making $\sqcup$ an instance of priority union (Reiss, 2022) rather than unification in the logical or type-theoretic sense. The term *unification* is retained following Bale et al. (2014). The following cases illustrate the range of outcomes:

1. $\{-\text{F}\} \sqcup \{+\text{G}\} = \{-\text{F}, +\text{G}\}$
2. $\{-\text{F}\} \sqcup \{+\text{F}\} = \{-\text{F}\}$
3. $\{+\text{G}\} \sqcup \{+\text{G}\} = \{+\text{G}\}$

Case 1 succeeds: A has no specification for G, so $+$G is added freely. Cases 2 and 3 are vacuous: in case 2, A already contains the opposite value $-$F, so $+$F is blocked to prevent inconsistency; in case 3, A already contains the same value $+$G, so adding it again changes nothing.

Projection. Restricts a segment to a subset of its valued features (Codd, 1970). Viewing a segment s as a set of valued features $(v, f) \in \{+, -\} \times \mathcal{F}$, projection onto $F \subseteq \mathcal{F}$ retains only those pairs whose feature label f falls in F. Equivalently, it restricts the domain of s to F:

$$\pi_F(s) = s \cap (\{-, +\} \times F) \quad (5)$$

Projection returns the partial specification of a segment restricted to a given set of features, as if the remaining features were unspecified. Using $s = \{+\text{F}, +\text{G}, -\text{H}, +\text{J}\}$:

1. $\pi_{\{\text{G,H}\}}(s) = \{+\text{G}, -\text{H}\}$
2. $\pi_{\{\text{J}\}}(s) = \{+\text{J}\}$
3. $\pi_{\emptyset}(s) = \emptyset$

Case 1 keeps G and H while discarding F and J. Case 2 isolates a single feature. Case 3 projects onto the empty set, yielding the fully underspecified segment $\emptyset$.

Combining Operators. Unification implements feature-filling, while subtraction followed by unification models feature-changing (Bale et al., 2014, 2019). For instance, a devoicing rule uses $(s_i \setminus \{+\text{VOICE}\}) \sqcup \{-\text{VOICE}\}$, while feature copying uses $s_i \sqcup \pi_{\{\text{VOICE}\}}(s_j)$.

Operator Complexity. Since $\mathcal{F}$ is finite, segments can be represented as bitvectors of length $2|\mathcal{F}|$, with two bits per feature $f \in \mathcal{F}$ encoding the presence and polarity of its specification. Under this representation, subtraction, unification, and projection all reduce to bitwise operations and execute in $O(|\mathcal{F}|)$ time, which is constant since $\mathcal{F}$ is fixed.

3.3 The Search & Change Specification

Let $\Theta(i)$ denote $\Theta_R(i)$ or $\Theta_L(i)$ depending on DIR. Say that position i is *licensed* if

$$s_i \in \text{INR} \wedge \Theta(i)\downarrow \wedge\, s_{\Theta(i)} \in \text{CND}.$$

For every position i, the transformed segment is

$$s'_i = \begin{cases} \text{OUT}(s_i, s_{\Theta(i)}) & \text{if } i \text{ is licensed} \\ s_i & \text{otherwise} \end{cases} \quad (6)$$

For an input word $w = \langle s_1, \ldots, s_n \rangle$, the output word is $w' = \langle s'_1, \ldots, s'_n \rangle$, with each s'_i computed simultaneously from w.

Simultaneous Application. All s'_i are computed from the same input word w and applied at once, so intermediate changes cannot feed or bleed within a single rule application, following the Simultaneous Application Principle (Chomsky and Halle, 1968; Kenstowicz and Kisseberth, 1979). This assumption is inherited from the S&C framework itself (Dabbous et al., 2024) and is what makes the specification in Equation 6 well-defined: since all output segments are read from the same input, there is no possibility of conflicting licensed changes. Across rules, derivations proceed by explicit ordering (Chomsky and Halle, 1968; Kenstowicz and Kisseberth, 1979). Under ordered rule application, the output of one compiled transducer serves as the input to the next; thus a derivation corresponds to composition of the compiled transducers in rule order.

4 Worked Examples

Two case studies are presented. First, Georgian clear and dark lateral alternation (Kenstowicz and Kisseberth, 1979) as a local example; and second, Votic vowel harmony (Leduc et al., 2020) as a long-distance example.

4.1 Georgian Clear and Dark Laterals (Local)

Kenstowicz and Kisseberth (1979) observe that lateral "lightness" in Georgian is fully predictable: underlying dark laterals surface as clear before −Back vowels. Setting $\textsc{Trm} = \mathcal{N}(\emptyset)$ models adjacency, since the first segment found is always the immediately adjacent one. The S&C rule

$$R_1 = \langle \textsc{Inr}, \textsc{Trm}, \textsc{Dir}, \textsc{Out}, \textsc{Cnd} \rangle$$
$$\textsc{Inr} = \mathcal{N}(\{+\textsc{Lateral}, +\textsc{Dorsal}\})$$
$$\textsc{Trm} = \mathcal{N}(\emptyset)$$
$$\textsc{Dir} = \textsc{Right}$$
$$\textsc{Out} = \lambda s_i.\, \lambda s_j.\, (s_i \setminus \{+\textsc{Dorsal}\}) \sqcup \{-\textsc{Dorsal}\}$$
$$\textsc{Cnd} = \mathcal{N}(\{-\textsc{Back}\})$$

captures these alternations.

Case: /łeło/ 'goal' ↦ [leło]. *Search.* Let $w = \langle$ł, e, ł, o$\rangle$. Laterals s_1 and s_3 are in Inr; with $\textsc{Trm} = \mathcal{N}(\emptyset)$, every segment qualifies as a terminator, so search always stops at the segment immediately adjacent to the search initiator. Searching rightward: $\Theta_R(1) = 2$ (terminator s_2 = e) and $\Theta_R(3) = 4$ (terminator s_4 = o).

Change. Since s_2 = e $\in$ Cnd, position 1 is licensed and $s'_1 = \textsc{Out}(s_1, s_2) =$ l. Since s_4 = o $\notin$ Cnd, position 3 is not licensed and $s'_3 = s_3 =$ ł.

Result. /łeło/ ↦ [leło]. The second lateral is unchanged because its right neighbor is a +Back vowel, which does not satisfy Cnd; forms like /xoło/ surface unchanged for the same reason.

A subset of the Kenstowicz and Kisseberth (1979) dataset solved using R_1 is shown below. The first table gives four forms where the rule applies.

UR	/łeło/	/xeli/	/kłeba/	/cʰoli/
R_1	[leło]	[xeli]	[kleba]	[cʰoli]
SR	[leło]	[xeli]	[kleba]	[cʰoli]

These forms show the expected alternation of dark [ɬ] to clear [l] when the rule's local environment is satisfied.

The next table gives four forms where R_1 does not apply.

UR	/kała/	/xoło/	/łxena/	/kbiłs/
R_1	—	—	—	—
SR	[kała]	[xoło]	[łxena]	[kbiłs]

These forms confirm that outside that environment, the laterals surface unchanged.

4.2 Votic Vowel Harmony (Long-Distance)

Leduc et al. (2020) analyze suffix vowels in Votic as underspecified for Back, receiving a value from the nearest preceding non-/i/ vowel. Root vowels are fully specified; suffix vowels are underlyingly underspecified for Back and surface as one of the following alternants:

$$[\text{y}] \sim [\text{u}] \qquad [\text{e}] \sim [\text{ə}] \qquad [\text{æ}] \sim [\text{ɑ}]$$

Votic's inventory of fully specified vowels is:

	i	y	ɨ	u	e	ø	ə	o	æ	ɑ
High	+	+	+	+	−	−	−	−	−	−
Low	−	−	−	−	−	−	−	−	+	+
Back	−	−	+	+	−	−	+	+	−	+
Round	−	+	−	+	−	+	−	+	−	−

The following suffix vowels are underspecified for Back and correspond to the three alternation pairs seen earlier:

$$/\text{U}/ = \{+\textsc{High}, -\textsc{Low}, +\textsc{Round}\}$$
$$/\text{E}/ = \{-\textsc{High}, -\textsc{Low}, -\textsc{Round}\}$$
$$/\text{A}/ = \{-\textsc{High}, +\textsc{Low}, -\textsc{Round}\}$$

The resulting surface alternant is then determined by the Back value the suffix vowel receives.

A challenge noted by Leduc et al. (2020) is that "any vowel except /i/" does not form a natural class (Gorman and Reiss, 2025b). Formally, every natural class defined from a projection of /i/ contains at least one other segment:

$$\forall F \subseteq \mathcal{F}, \quad \mathcal{N}(\pi_F(\mathrm{i})) \neq \{\mathrm{i}\}.$$

To avoid this problem, Leduc et al. (2020) propose two copying rules plus a default rule:

$$F_{\text{NON-HIGH}} = \{+\textsc{Syllabic}, -\textsc{High}\}$$
$$F_{\text{HIGH-ROUND}} = \{+\textsc{Syllabic}, +\textsc{High}, +\textsc{Round}\}$$

R_2 copies Back from the nearest preceding non-high vowel:

$$R_2 = \langle \textsc{Inr}, \textsc{Trm}, \textsc{Dir}, \textsc{Out}, \textsc{Cnd} \rangle$$
$$\textsc{Inr} = \mathcal{N}(\{+\textsc{Syllabic}\})$$
$$\textsc{Trm} = \mathcal{N}(F_{\text{NON-HIGH}})$$
$$\textsc{Dir} = \textsc{Left}$$
$$\textsc{Out} = \lambda s_i.\, \lambda s_j.\, s_i \sqcup \pi_{\{\textsc{Back}\}}(s_j)$$
$$\textsc{Cnd} = \mathcal{N}(\emptyset)$$

R_3 and R_4 share Inr, Dir, and Cnd with R_2, differing only in Trm and Out:

$$R_3: \quad \textsc{Trm} = \mathcal{N}(F_{\text{HIGH-ROUND}})$$
$$\textsc{Out} = \lambda s_i.\, \lambda s_j.\, s_i \sqcup \pi_{\{\textsc{Back}\}}(s_j)$$
$$R_4: \quad \textsc{Trm} = \mathcal{N}(\emptyset)$$
$$\textsc{Out} = \lambda s_i.\, \lambda s_j.\, s_i \sqcup \{-\textsc{Back}\}$$

Note that here, the target class is intentionally broad: all syllabic segments are potential targets, but fully specified vowels remain unchanged, since adding a feature value to a fully specified segment through unification applies vacuously (Bale et al., 2014).

Applied in order, $R_2 \to R_3 \to R_4$ reproduces the full harmony pattern.

Case: /puhE/ 'tree.ILLAT' ↦ [puhə]. *Search.* Consider the word $w = \langle \mathrm{p}, \mathrm{u}, \mathrm{h}, \mathrm{E} \rangle$. The segments $s_2 = \mathrm{u}$ and $s_4 = \mathrm{E}$ are both members of $\textsc{Inr} = \mathcal{N}(\{+\textsc{Syllabic}\})$. Under R_3 ($\textsc{Trm} = \mathcal{N}(F_{\text{HIGH-ROUND}})$, $\textsc{Dir} = \textsc{Left}$):

1. $\Theta_L(2)$ is undefined (no high-round vowel precedes s_2).
2. $\Theta_L(4) = 2$ (terminator is $s_2 = \mathrm{u}$, which is a high-round vowel).

Change. $\textsc{Cnd} = \mathcal{N}(\emptyset)$, so position 4 is licensed.

1. $s'_4 = \textsc{Out}(s_4, s_2) = \mathrm{E} \sqcup \pi_{\{\textsc{Back}\}}(\mathrm{u}) = \mathrm{ə}$.

/E/ receives +Back from the preceding /u/, surfacing as [ə]. R_2 does not apply (no non-high vowel precedes /E/); R_3 bleeds R_4 by filling the underspecified vowel first, so R_4 applies only vacuously, since the relevant segment has already been specified as +Back at that point in the derivation.

Result. $\langle \mathrm{p}, \mathrm{u}, \mathrm{h}, \mathrm{E} \rangle \mapsto \langle \mathrm{p}, \mathrm{u}, \mathrm{h}, \mathrm{ə} \rangle$, i.e., /puhE/ surfaces as [puhə].

A subset of the dataset from Leduc et al. (2020) is shown below, illustrating the three-rule ordering.

UR	/vəttimisE/	/puhE/	/siliA/	/tyttærikkoA/
R_2	[vəttimisə]	—	—	[tyttærikkoɑ]
R_3	—	[puhə]	—	—
R_4	—	—	[siliæ]	—
SR	[vəttimisə]	[puhə]	[siliæ]	[tyttærikkoɑ]

The next section presents an explicit procedure for efficiently computing these mappings.

5 Algorithmic Implementation of Search & Change

The naive approach of evaluating the specification at each position requires a linear scan per position to compute $\Theta_R(i)$ or $\Theta_L(i)$, yielding quadratic worst case time complexity. The key observation is that many search initiating segments can share the same search terminating segment, so once a licensed terminator is found, it licenses all initiating segments in its scope. This enables a linear-time implementation of S&C computable in a single scan of the input string. This paper restricts attention to feature-changing and filling processes, for which no explicit word boundary symbols are required; boundary-sensitive processes, which require additional machinery, are left to future work.

5.1 Reverse Scan with Single Pointer

Scan the word in the direction opposite Dir, maintaining a licensor index j (initialized to -1). At each position k: first compute the output (apply Out if $s_k \in \textsc{Inr}$ and $j \neq -1$, otherwise copy s_k), and only then update j: set $j \leftarrow k$ if $s_k \in \textsc{Trm} \cap \textsc{Cnd}$, reset $j \leftarrow -1$ if $s_k \in \textsc{Trm} \setminus \textsc{Cnd}$, and leave j unchanged otherwise. The output step uses the value of j inherited from the previous position; updating j first would incorrectly allow s_k to license itself. For $\textsc{Dir} = \textsc{Left}$, we reverse the input first and reverse the output after. This follows the same general reversal insight used in harmony work by Heinz and Lai (2013).

The algorithm runs in linear time with only constant-time work per position. Its single integer

Input: word $w = \langle s_1, \ldots, s_n \rangle$, rule $R = \langle \textsc{Inr}, \textsc{Trm}, \textsc{Dir}, \textsc{Out}, \textsc{Cnd} \rangle$
Output: output word w'

if $\textsc{Dir} = \textsc{Left}$ **then**
  $w \leftarrow \textsc{Reverse}(w)$
$w' \leftarrow \epsilon$
$j \leftarrow -1$
for $k \leftarrow n$ **to** 1 **do**
  if $s_k \in \textsc{Inr} \wedge j \neq -1$ **then**
    $s'_k \leftarrow \textsc{Out}(s_k, s_j)$
  else
    $s'_k \leftarrow s_k$
  $w' \leftarrow s'_k \cdot w'$
  if $s_k \in \textsc{Trm} \wedge s_k \in \textsc{Cnd}$ **then**
    $j \leftarrow k$
  else if $s_k \in \textsc{Trm} \wedge s_k \notin \textsc{Cnd}$ **then**
    $j \leftarrow -1$
if $\textsc{Dir} = \textsc{Left}$ **then**
  $w' \leftarrow \textsc{Reverse}(w')$
return w'

Algorithm 1: Reverse Scan with Single Pointer

state makes compilation to finite-state transducers straightforward, as shown in Section 6.

Theorem 1. *Algorithm 1 computes exactly the mapping defined by the S&C specification (Equation 6).*

Proof. It suffices to prove correctness for $\textsc{Dir} = \textsc{Right}$; the $\textsc{Dir} = \textsc{Left}$ case follows by the reversal reduction. Define for each position i:

$$\textsc{Lic}(i) \iff \Theta_R(i)\downarrow \wedge\, s_{\Theta_R(i)} \in \textsc{Cnd},$$

$$\textsc{Idx}(i) = \begin{cases} \Theta_R(i) & \text{if } \textsc{Lic}(i) \\ -1 & \text{otherwise.} \end{cases}$$

We scan from $k = n$ down to 1, maintaining invariants (I1) $\forall m > k,\ s_m^{\text{alg}} = s_m^{\text{spec}}$ and (I2) $j = \textsc{Idx}(k)$.

Base case ($k = n$): no terminator lies strictly to the right of n, so $\textsc{Idx}(n) = -1$. The algorithm initializes $j \leftarrow -1$, so (I2) holds. (I1) is vacuously true.

Inductive step. Assume (I1) and (I2) hold before processing k. For the output at k: the algorithm applies $\textsc{Out}$ only when $s_k \in \textsc{Inr} \wedge j \neq -1$, which by (I2) is equivalent to $s_k \in \textsc{Inr} \wedge \textsc{Lic}(k)$, which is exactly the licensed case of Equation 6. In that case $j = \Theta_R(k)$ and the algorithm writes $s_k^{\text{alg}} = \textsc{Out}(s_k, s_{\Theta_R(k)})$, which matches the specification; otherwise $s_k^{\text{alg}} = s_k$. So (I1) is preserved. For j (to satisfy (I2) at $k-1$): if $s_k \in \textsc{Trm} \cap \textsc{Cnd}$, then $\textsc{Idx}(k-1) = k$ and the algorithm sets $j \leftarrow k$; if $s_k \in \textsc{Trm} \setminus \textsc{Cnd}$, then $\Theta_R(k-1) = k$ (since $s_k \in \textsc{Trm}$ is the nearest terminator), but $s_k \notin \textsc{Cnd}$ makes $\textsc{Lic}(k-1)$ false, so $\textsc{Idx}(k-1) = -1$. Every $i < k$ whose nearest terminator is also k is also unlicensed, so the algorithm correctly carries $j \leftarrow -1$ leftward until the next $\textsc{Trm}$ segment is reached; if $s_k \notin \textsc{Trm}$, then $\textsc{Idx}(k-1) = \textsc{Idx}(k)$ and j is unchanged (since k is not a candidate terminator, $\Theta_R(k-1)$ skips it and returns the same nearest terminator as $\Theta_R(k)$). Therefore (I2) is preserved. By induction, $w^{\text{alg}} = w^{\text{spec}}$. □

6 Compiling S&C Rules into Finite-State Transducers

Algorithm 1 is not itself a finite-state operation, as the reversal step requires reading the full input string before producing output, which is incompatible with streaming computation. What maps onto an FST is not the algorithm as written, but the control structure that it exposes: a single integer state j and a case split on each possible input symbol. The value $j = -1$ corresponds to a single *false* state q_f, and each possible licensor that j can be assigned to corresponds to a *true* state q_σ where $\sigma \in \textsc{Trm} \cap \textsc{Cnd}$. Since control updates and output choices are determined by finitely many (state, input) pairs, all behaviors can be precomputed into transitions. Algorithm 2 gives the full compilation procedure. The compiled transition structure is the same regardless of scan orientation, but its induced transduction depends on how it is traversed: in one orientation it is subsequential, while in the opposite orientation it yields the corresponding reverse-subsequential mapping via conjugation by reversal (Payne, 2017; Heinz, 2018).

Theorem 2. *Let T be the transducer produced by Algorithm 2 with $F_I = \mathcal{F}$. Assuming a fixed rule direction, T computes the same mapping as the corresponding instance of Algorithm 1. The opposite directional mapping is obtained by conjugation with reversal.*

Proof. The transducer states correspond exactly to the original procedure's control values: q_f corresponds to $j = -1$ and q_σ to $j = \sigma$. On input x in state q, both computations emit

$$y = \begin{cases} \textsc{Out}(x, \sigma) & q = q_\sigma \wedge x \in \textsc{Inr} \\ x & \text{otherwise,} \end{cases}$$

Input: feature set $F_I \subseteq \mathcal{F}$, rule $R = \langle \textsc{Inr}, \textsc{Trm}, \textsc{Dir}, \textsc{Out}, \textsc{Cnd} \rangle$ over $\Omega(F_I)$
Output: transition structure $T = \langle Q, \Sigma, \Gamma, I, F, \delta \rangle$ whose induced transduction depends on scan orientation

$\Sigma \leftarrow \Omega(F_I)$
$\Gamma \leftarrow \Omega(F_I)$
$Q \leftarrow \{q_f\} \cup \{q_\sigma \mid \sigma \in \textsc{Trm} \cap \textsc{Cnd}\}$
$I \leftarrow \{q_f\}$
$F \leftarrow Q$
$\delta \leftarrow \emptyset$
foreach $x \in \Sigma$ **do**
  if $x \in \textit{Trm} \cap \textit{Cnd}$ **then**
    $\delta(q_f, x) \leftarrow (q_x, x)$
  else
    $\delta(q_f, x) \leftarrow (q_f, x)$
foreach $\sigma \in \textit{Trm} \cap \textit{Cnd}$ **do**
  foreach $x \in \Sigma$ **do**
    if $x \in \textit{Inr}$ **then**
      $y \leftarrow \textsc{Out}(x, \sigma)$
    else
      $y \leftarrow x$
    if $x \in \textit{Trm} \cap \textit{Cnd}$ **then**
      $\delta(q_\sigma, x) \leftarrow (q_x, y)$
    else if $x \in \textit{Trm} \setminus \textit{Cnd}$ **then**
      $\delta(q_\sigma, x) \leftarrow (q_f, y)$
    else
      $\delta(q_\sigma, x) \leftarrow (q_\sigma, y)$
return $\langle Q, \Sigma, \Gamma, I, F, \delta \rangle$

Algorithm 2: Compilation of an S&C Rule into an FST

and update the control value by

$$q \leftarrow \begin{cases} q_x & x \in \textsc{Trm} \cap \textsc{Cnd} \\ q_f & x \in \textsc{Trm} \setminus \textsc{Cnd} \\ q & x \notin \textsc{Trm}. \end{cases}$$

Algorithm 2 defines exactly one transition per (state, input) pair encoding this case split. By induction on prefix length (base case: both start in q_f; inductive step: both case splits coincide at each position), state and output agree throughout. Hence both computations agree on all strings in $\Omega(\mathcal{F})^*$. □

6.1 Space Complexity

If Trm and Cnd are defined by consistent bundles fixing a set $V_{TC} \subseteq \mathcal{F}$ of features, then $|\textsc{Trm} \cap \textsc{Cnd}| = 3^{|\mathcal{F}|-|V_{TC}|}$, giving

$$|Q| = 1 + 3^{|\mathcal{F}|-|V_{TC}|}, \qquad |\delta| = |Q| \cdot 3^{|\mathcal{F}|}.$$

This is finite in principle but infeasible at realistic scales. For example, with $|\mathcal{F}| = 30$ and $|V_{TC}| = 5$:

$$|Q| = 1 + 3^{25} = 847{,}288{,}609{,}444,$$
$$|\delta| = |Q| \cdot 3^{30} \approx 1.74 \times 10^{26}.$$

The next subsection shows how to compress these generated FSTs by retaining only those state and transition distinctions that are relevant to the rule's behavior.

6.2 Compressing Generated FSTs

Rules interact with only a subset $V \subseteq \mathcal{F}$ of features, where $V = V_{\text{cls}} \cup V_{\text{out}}$: features referenced in Inr, Trm, Cnd and those referenced inside Out. Compiling over the restricted alphabet

$$\Sigma_V = \{\, \pi_V(x) \mid x \in \Omega(\mathcal{F}) \,\}, \quad |\Sigma_V| = 3^{|V|},$$

yields a transducer $T_V : \Sigma_V^* \to \Sigma_V^*$.

Canonical representatives. Let the canonicalization map be

$$\kappa : \Omega(\mathcal{F}) \to \Sigma_V \quad \text{where} \quad \kappa(x) = \pi_V(x)\,.$$

The map κ identifies each segment with its equivalence class under indistinguishability with respect to the rule's visible feature set V. Overload κ to strings by

$$\kappa : \Omega(\mathcal{F})^* \to \Sigma_V^*$$
$$\kappa(\langle x_1, \ldots, x_n \rangle) = \langle \pi_V(x_1), \ldots, \pi_V(x_n) \rangle\,.$$

The pipeline is: project the input to V-features, transduce, then restore the non-V features from the original. Then

$$f(w) = R(T_V(\kappa(w)),\, w)\,,$$
$$R(T_V(\kappa(w)), w)_i = (w_i \setminus \kappa(w)_i) \cup T_V(\kappa(w))_i,$$

where R replaces the V-visible features of each input segment with those produced by T_V, leaving all non-V features untouched. The compressed state and arc counts are

$$|Q'| = 1 + 3^{|V|-|V_{TC}|}, \qquad |\delta'| = |Q'| \cdot 3^{|V|}.$$

Since $|V|$ is much smaller than $|\mathcal{F}|$ in typical analyses, this reduction can be dramatic. For the Votic harmony rule that copies Back from a preceding high-round vowel, the visible features are $V = \{\textsc{Syllabic}, \textsc{High}, \textsc{Round}, \textsc{Back}\}$, so $|V| = 4$ and $|\Sigma_V| = 3^4 = 81$. Since Trm fixes Syllabic, High, and Round, we have $|V_{TC}| = 3$, which yields just $|Q'| = 1 + 3^{4-3} = 4$ states and $|\delta'| = 4 \cdot 81 = 324$ arcs. The correctness of this pipeline is established by the following theorem.

Theorem 3. *The visible-feature pipeline* $f(w) = R(T_V(\kappa(w)), w)$ *computes the same mapping as the full S&C specification.*

Proof. First, since every feature referenced by INR, TRM, and CND lies in V, projection to V preserves all three membership tests: $\pi_V(x) = \pi_V(y)$ implies x and y agree on all valued features with base feature in V, so $x \in \mathcal{N}(A) \iff y \in \mathcal{N}(A)$ for any $A \subseteq (\{+,-\} \times V)$. Second, by Theorem 2 instantiated with feature set V, the transducer T_V computes the projected S&C mapping on Σ_V^*: it applies $\textsc{Out}(x, \sigma)$ at every licensed position and copies elsewhere, with the same control updates as the single-pass procedure over V-projected symbols. Since OUT references only features in V, this projected output is exactly $\pi_V(s'_i)$ at each position i. Third, reconstruction $R(T_V(\kappa(w)), w)_i = (w_i \setminus \kappa(w)_i) \cup T_V(\kappa(w))_i$ re-inserts the non-visible features from the original input, yielding the full output segment s'_i at every position. □

Corollary 4 (Implicit State Merging Under Projection)**.** *Compiling with feature set V rather than $\mathcal{F}$ implicitly merges states into V-projection equivalence classes. Since Algorithm 2 builds over $\Omega(F_I)$ for its given F_I, setting $F_I = V$ causes all membership tests for INR, TRM, and CND to be evaluated only over V-features, collapsing V-equivalent states without changing the realized transduction.*

6.3 Subsequentiality

For every state q and input symbol x, the compiler defines exactly one transition $\delta(q, x)$, making the resulting transducers sequential in the sense of Mohri (1997). Since every sequential transducer is a special case of a subsequential transducer (with empty final outputs), S&C rules compile to subsequential FSTs for a fixed scan orientation.

An important distinction to make is the one between ordinary left-to-right subsequentiality and right-to-left, or reverse, subsequentiality: a mapping is reverse-subsequential when it is realized by reversing the input, applying a subsequential transducer, and reversing the output back (Heinz, 2018; Payne, 2017). For the opposite orientation, the same transition structure induces the corresponding reverse-subsequential mapping by exactly this conjugation with reversal. Thus, the compiler builds a single transition structure, while the choice of scan orientation determines which induced transduction it realizes.

7 Conclusion

Previous work has shown that phonological patterns can be modeled using finite-state transducers (Kaplan and Kay, 1994; Mohri and Sproat, 1996; Mohri, 1997). Unlike subsequential transducers defined directly over string alphabets, S&C rules are stated at the level of phonological features and natural classes, making the linguistic motivation of each rule transparent and independently evaluable. This paper shows that these two perspectives are compatible: S&C rules can be compiled via an explicit construction procedure into a single transition structure whose induced transduction depends on scan orientation: in one orientation it is subsequential, and in the opposite orientation it yields the corresponding reverse-subsequential mapping via conjugation by reversal. Two broader consequences follow from these results.

Learnability. Transductions expressible by subsequential FSTs are learnable from positive input/output string pairs (Oncina et al., 1993; Heinz, 2018), in the sense of identification in the limit (Gold, 1967). S&C-defined mappings therefore fall within a learnable profile: for one orientation they are subsequential, and for the other they are obtained by conjugating a subsequential mapping with reversal. This suggests that the hypothesis-class complexity of S&C-definable mappings can be bounded by restricting learners to subsequential transductions together with reversal, rather than arbitrary string-to-string functions.

Algebraic classification. Regular languages admit classification into algebraic and logical subclasses (Straubing, 1994; Heinz, 2018), and analogous methods exist for deterministic FSTs (Lambert and Heinz, 2023). Since S&C rules compile to FSTs, S&C-modelable mappings can be situated within these classes. A natural next step is a typology of S&C rule schemas grounded in transducer classification, which would yield testable predictions about the computational profiles of phonological processes. Tools such as LTK (Lambert, 2026) could be used to examine which rule families compile to which FST classes systematically.

References

Alan Bale, Maxime Papillon, and Charles Reiss. 2014. Targeting underspecified segments: A formal analysis of feature-changing and feature-filling rules. *Lingua*, 148:240–253.

Alan Bale and Charles Reiss. 2015. *Phonology: A formal introduction*. The MIT Press.

Alan Bale, Charles Reiss, and David Ta-Chun Shen. 2019. Sets, rules and natural classes: [ ] vs. { }. *Loquens*, 6(2):e065.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row. Simultaneous rule application discussion on p. 344.

Edgar F. Codd. 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Rim Dabbous, Marjorie Leduc, Fatemeh Mousavi, Charles Reiss, Ta-Chun David, and Shen. 2021. Satisfying long-distance relationships (without tiers): A strictly anti-local approach to phonology. Unpublished manuscript.

Rim Dabbous, Marjorie Leduc, Charles Reiss, and David Shen. 2024. Locality is epiphenomenal: Adjacency is opaqueness. In *Proceedings of the Canadian Linguistics Association 2024*, Carleton University, Ottawa. Conference presentation.

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

Kyle Gorman and Charles Reiss. 2025a. Metaphony in substance free logical phonology. To appear in a special issue of *Phonology*. LingBuzz/008634 (v3, October 2025).

Kyle Gorman and Charles Reiss. 2025b. Natural class reasoning in segment deletion rules. Paper presented at NELS 56, to appear in the proceedings. LingBuzz/009501 (November 2025).

Jeffrey Heinz. 2018. *The Computational Nature of Phonological Generalizations*.

Jeffrey Heinz and William J. Idsardi. 2017. Computational phonology today. *Phonology*, 34(2):211–219.

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

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

Michael Kenstowicz and Charles Kisseberth. 1979. *Generative Phonology: Description and Theory*. Academic Press, New York. Simultaneous rule application discussion on p. 318.

Dakotah Lambert. 2026. The language toolkit. Computer software.

Dakotah Lambert and Jeffrey Heinz. 2023. An algebraic characterization of total input strictly local functions. *Proceedings of the Society for Computation in Linguistics*, 6(1):25–34.

Marjorie Leduc, Charles Reiss, and Veno Volenec. 2020. Votic vowel harmony in substance free logical phonology. Published in *The Oxford Handbook of Vowel Harmony*. LingBuzz/005339.

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

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. *Preprint*, arXiv:cmp-lg/9606026.

Jose Oncina, Pedro Garcia, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

Amanda Payne. 2017. All dissimilation is computationally subsequential. *Language*, 93(4):e353–e371.

Charles Reiss. 2021. Towards a complete logical phonology model of intrasegmental changes. *Glossa: A Journal of General Linguistics*, 6(1):107.

Charles Reiss. 2022. Priority union and feature logic in phonology. *Linguistic Inquiry*, 53(1):199–209.

Charles Reiss and Veno Volenec. 2022. Conquer primal fear: Phonological features are innate and substance-free. *Canadian Journal of Linguistics / Revue canadienne de linguistique*, 67(4):581–610.

Howard Straubing. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser.