An Algebraic Characterization of Total Input Strictly Local Functions

Dakotah Lambert
Université Jean Monnet Saint-Étienne, CNRS
Institut d Optique Graduate School
Laboratoire Hubert Curien UMR 5516
F-42023, Saint-Étienne, France
dakotahlambert@acm.org

Jeffrey Heinz
Stony Brook University
Department of Linguistics
Institute for Advanced Computational Science
jeffrey.heinz@stonybrook.edu

Abstract

This paper provides an algebraic characterization of the total input strictly local functions. Simultaneous, noniterative rules of the form A→B/C__D, common in phonology, are definable as functions in this class whenever CAD represents a finite set of strings. The algebraic characterization highlights a fundamental connection between input strictly local functions and the simple class of definite string languages, as well as connections to string functions studied in the computer science literature, the definite functions and local functions. No effective decision procedure for the input strictly local maps was previously available, but one arises directly from this characterization. This work also shows that, unlike the full class, a restricted subclass is closed under composition. Additionally, some products are defined which may yield new factorization methods.

1 Introduction

The strictly local languages are those in which membership is decidable by the substrings up to some fixed width k of its words (McNaughton and Papert, 1971; Rogers and Pullum, 2011). Such languages are useful in the description of phonotactic patterns. Edlefsen et al. (2008) demonstrated that 75% of the patterns in the StressTyp2 database of stress patterns (Goedemans et al., 2015) are strictly local for k less than or equal to 6, reminiscent of Miller's Law on working memory, that an average person can hold roughly seven plus or minus two objects in short-term working memory (Miller, 1956). Even $k \leqslant 3$ suffices to capture nearly half of the patterns (see also Rogers and Lambert, 2019).

Chandlee et al. (2014) define the input strictly local functions to extend this notion to maps. They provide an efficient learner identifying functions in this class in the limit from positive data alone, using polynomial time and space. These mappings describe phonologically natural processes in which the output associated with a particular input symbol is uniquely determined by some local context around that symbol. Evidencing this naturality, 95 percent of maps in the P-Base database of phonological patterns (Mielke, 2008) lie in this class (Chandlee and Heinz, 2018). Related to this are the output strictly local maps, in which the output contributed by an input symbol is determined by the most recent symbols in the previous output (Chandlee et al., 2015).

One aspect of the study of formal languages is a deep connection between logic, automata, and algebra (Pin, 1997). Many classes of formal languages are characterized by decidable properties of an algebraic structure associated with each language in the class. The connection between algebraic structures and string languages can be extended to string-to-string maps based on the transducers that generate them (Filiot et al., 2016; Lambert, 2022).

This paper is structured as follows. Deterministic (sometimes called "unambiguous") finite-state acceptors (DFA) and transducers as well as the algebraic structures they induce are described in section 2. The formal definition of input strictly local maps is provided in section 3. The primary result, an algebraic characterization of this class, is given there alongside the polytime decision algorithm that it induces. This section also draws the connection to research in computer science which studied

Proceedings of the Society for Computation in Linguistics (SCiL) 2023, pages 25-34.
Amherst, Massachusetts, June 15-17, 2023

these functions under different names. Next in section 4 we discuss closure properties and an algorithm for composing transducers. We demonstrate that input strictly local functions are not closed under composition, but a subclass of them is so closed. Other operations under which the full class, perhaps with extensions, is closed are discussed in section 5. We conclude with discussion of these results in section 6.

2 Structures and Machines

A **semigroup** is a set S closed under some binary operation $\cdot$ (often denoted by adjacency) which is associative: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$. Given some finite alphabet Σ, the set of all nonempty sequences made up of those letters forms a semigroup with the concatenation operation. This is the **free semigroup** generated by Σ. A **monoid** is a semigroup in which there exists some element e such that for all x, $e \cdot x = x \cdot e = x$. Typically this identity element is represented by 1. The free semigroup generated by Σ can be adapted to the **free monoid** generated by Σ by including the empty sequence (denoted by λ), the identity for concatenation. The free semigroup and free monoid generated by Σ are often denoted Σ^+ and Σ^*, respectively.

A formal language L over Σ is some subset of this free monoid. Two useful equivalence relations can be defined based on L. **Nerode equivalence** is defined such that $a \overset{N}{\sim} b$ iff for all $v \in \Sigma^*$ it holds that $av \in L \Leftrightarrow bv \in L$ (Nerode, 1958). This is often called the Myhill-Nerode equivalence relation, as the well-known Myhill-Nerode theorem states that a language is regular iff its set of equivalence classes is finite. However, Myhill used a finer partition to achieve the same result: **Myhill equivalence** is defined such that $a \overset{M}{\sim} b$ iff for all $u \in \Sigma^*$, $ua \overset{N}{\sim} ub$; alternatively, for all $u, v \in \Sigma^*$, $uav \in L \Leftrightarrow ubv \in L$ (Rabin and Scott, 1959). Being a coarser partition, $\overset{N}{\sim}$ can never define more classes than $\overset{M}{\sim}$, and the number of classes defined by $\overset{M}{\sim}$ is in the worst case exponential in that defined by $\overset{N}{\sim}$ (Holzer and König, 2004), so finiteness in one translates to the other.

2.1 Illustrating Nerode and Myhill Relations

Consider the example language over $\{a, b, c\}$ consisting of all and only those words that do not contain an ab substring. Consider which classes must exist. $\llbracket ab \rrbracket$ is the set of words containing an ab substring. These are rejected, and no suffix can save them. So if $x, y \in \llbracket ab \rrbracket$, for all v it holds that $xv \notin L$ and $yv \notin L$. All of these words are related, and distinct from any accepted words. But the accepted words partition into two classes: words that end in a ($\llbracket a \rrbracket$) and others ($\llbracket \lambda \rrbracket$). The former are rejected after adding a b suffix, while the latter remain accepted after adding a b suffix. No suffix distinguishes words within these classes, so the three can define a minimal DFA for L (Nerode, 1958), as will be discussed shortly.

There are then at least three Myhill classes. But some classes split. An a prefix distinguishes the strings a and ba, and this generalizes. The class of accepted words ending in a splits to two: words ending in a that begin with b ($\llbracket ba \rrbracket$) and other words ending in a ($\llbracket a \rrbracket$). The other class of accepted words splits to three: words beginning in b ($\llbracket b \rrbracket$), nonempty words not beginning in b ($\llbracket c \rrbracket$), and the empty word ($\llbracket \lambda \rrbracket$). An a prefix distinguishes the first of these from the other two, while the a_b circumfix distinguishes the last from $\llbracket c \rrbracket$. The six $\overset{M}{\sim}$ classes are $\llbracket \lambda \rrbracket$, $\llbracket a \rrbracket$, $\llbracket b \rrbracket$, $\llbracket c \rrbracket$, $\llbracket ba \rrbracket$, and $\llbracket ab \rrbracket$.

The $\overset{N}{\sim}$ classes may have ill-defined concatenation. If $u \overset{N}{\sim} u'$, then $uv \overset{N}{\sim} u'v$ (it is a left congruence), but it may be that $v \overset{N}{\sim} v'$ while $uv \overset{N}{\nsim} uv'$ (it is not a right congruence). In the current example, $b \overset{N}{\sim} c$ but $ab \overset{N}{\nsim} ac$. In contrast, $\overset{M}{\sim}$ is compatible with concatenation (it is a congruence): if $u \overset{M}{\sim} u'$ and $v \overset{M}{\sim} v'$, it follows that $uv \overset{M}{\sim} u'v'$. That means these equivalence classes form the elements of a submonoid of Σ^*. The quotient monoid $\Sigma^*/\overset{M}{\sim}$ (these equivalence classes under concatenation) is the **syntactic monoid** of L. If L is a regular language, then this is the smallest monoid which can be used as a DFA accepting L (Rabin and Scott, 1959). The **syntactic semigroup** is $\Sigma^+/\overset{M}{\sim}$.

A string language is rational if and only if it has finitely many Myhill classes (Rabin and Scott, 1959). A **variety** of finite semigroups is a class closed under subsemigroup, quotients and finitary direct product (Pin, 1997). This implies closure under Boolean operations. Eilenberg's theorem states that these varieties uniquely define subclasses of rational languages (Eilenberg and Schützenberger, 1976). As we explain later they can also characterize subclasses of rational functions, such as the input strictly local functions.

2.2 String Acceptors

A DFA is a five-tuple $\langle \Sigma, Q, \delta, q_0, F \rangle$ where Σ is a finite alphabet, Q a finite state set, $\delta : \Sigma \times Q \to Q$

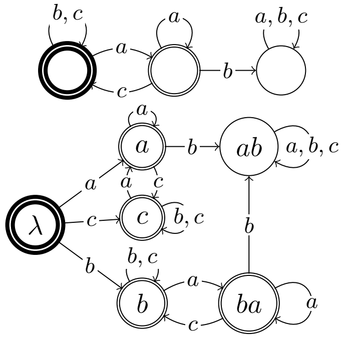

Figure 1: A DFA forbidding ab substrings induced by $\overset{N}{\sim}$ (above) and $\overset{M}{\sim}$ (below). States are labeled by class representatives. Doubly circled states are accepting and extra thick borders designate initial states.

a transition function, q_0 an initial state, and F a set of accepting states. A word is read one symbol at a time. If computation is in state q, the remaining string is σw, and $\delta(\sigma, q) = r$, then after one step, the computation will be in state r with remaining string w. Given the equivalence classes under $\overset{N}{\sim}$ or $\overset{M}{\sim}$, we can construct such an acceptor. Σ is the alphabet, Q the set of equivalence classes, $\delta(\sigma, q)$ the equivalence class of $q\sigma$, q_0 whichever class contains the empty sequence, and F the set of equivalence classes containing accepted words. Hopcroft and Ullman (1979) discuss a dynamic programming algorithm to reduce an arbitrary DFA to that induced by $\overset{N}{\sim}$. Another procedure, which will be described later, derives the Myhill relation from this form.

Figure 1 shows the acceptors induced by $\overset{N}{\sim}$ and $\overset{M}{\sim}$ for the example language over $\{a, b, c\}$ in which no word contains an ab substring. That induced by $\overset{M}{\sim}$ is a right Cayley graph of the syntactic monoid (see Zelinka, 1981), augmented with information about whether classes are accepting.

2.3 String-to-String Transducers

Oncina et al. (1993) discuss one method of generalizing these acceptors into functions. A **sequential transducer** is a five-tuple $\langle \Sigma, \Delta, Q, \delta, q_0 \rangle$, where Σ is the alphabet of the input, Δ that of the output, Q a finite set of states, $\delta : \Sigma \times Q \to \Delta^* \times Q$ a transition function, and q_0 an initial state. This behaves like an acceptor, where all strings in the domain are accepted and every edge traversed appends to an accumulating output. Sequential functions are total. A **subsequential transducer** generalizes this by associating outputs with states (Oncina et al., 1993); if an input word ends in state q, the output receives the suffix associated with q. The function σ mapping states to suffixes is added as a sixth element: $\langle \Sigma, \Delta, Q, \delta, q_0, \sigma \rangle$. The names and order of these components here are not the same as those used in the original work, but seem to have become commonly used in later work. Adding another element, a string prefixed to all output strings, adds nothing because it could be added to each edge out of q_0 and to that state's output. So in this work, this universal prefix π will be assumed: $\langle \Sigma, \Delta, Q, \delta, q_0, \pi, \sigma \rangle$. This change leaves most definitions unaffected.

Bruyère and Reutenauer (1999) argue that the subsequential notion is more deserving of the status as the basic object, and refer to such functions as simply sequential, a practice followed by Lombardy and Sakarovitch (2006), among others. A subsequential machine is equivalent to a sequential machine over a larger alphabet that includes explicit boundary symbols, and a well-formed version of the latter can be rewritten as the former. Given this bijection, the remainder of this work will follow this recent notational trend.

Sequentiality may depend on the direction in which the input is read. Iterative regressive harmony patterns cannot be described by left-to-right sequential functions as they admit unbounded delay between seeing a harmonizing symbol and finding the trigger that determines its surface form (Heinz and Lai, 2013; Mohri, 1997). However, this process can be expressed as a right-to-left sequential function. This is equivalent to reversing the output of a left-to-right transducer applied to the input's reversal. Or one could say the machine reads the string from right to left, prefixing to the output. If SQ is the sequential class, we denote the left-to-right class →SQ and its right-to-left variant ←SQ, with the arrow indicating directionality.

The **longest common prefix** (denoted lcp) of a set of strings S is the unique string u such that u is a prefix of every string in S and that u is longer than every other string u' which prefixes every string in S. A transducer is **onward** if it emits output as early as it can: for all states p, $\operatorname{lcp}(\{y \in \Delta^* \colon \delta(a, p) = \langle y, q \rangle\} \cup \{\sigma(p)\}) = \lambda$. The Nerode equivalence relation extends naturally to functions by means of the **tails** of input strings. The set of tails of x in a function f, $T_f(x)$, is defined as follows:

$$T_f(x) = \{\langle y, v \rangle \colon f(xy) = \operatorname{lcp}(f(x\Sigma^*))v\}.$$

Two strings are related iff their tails are equal. We write this relation as $\overset{N}{\sim}$, emphasizing connection to

Nerode equivalence for string sets. A transducer in canonical form is onward and has one state per $\overset{N}{\sim}$ class. A two-sided extension generalizes Myhill equivalence. The **contexts** of x in f are as follows:

$$C_f(x) = \{\langle w, y, v\rangle \colon f(wxy) = \mathrm{lcp}(f(wx\Sigma^*))v\}.$$

The subset where $w = \lambda$ is essentially equivalent to $T_f(x)$, so the $\overset{M}{\sim}$ relation derived from C forms, as with string sets, a refinement of $\overset{N}{\sim}$.

2.4 Monoids from Canonical Machines

The canonical form of a machine derives states from the $\overset{N}{\sim}$ relation. The $\overset{M}{\sim}$ relation accounts for the influence of prefixes. So, to construct a machine over $\overset{M}{\sim}$ from a canonical machine, i.e. to construct a right Cayley graph of its associated monoid, we look to see where each input symbol takes each of the states. In other words, what is the action of each symbol over the states? This is the transition congruence (Filiot et al., 2016). McNaughton and Papert (1971) use this same construction.

Consider the automata of Figure 1. Assign an arbitrary number to each state of the automaton induced by $\overset{N}{\sim}$: $[\![\lambda]\!]$ is 1, $[\![a]\!]$ is 2, and $[\![ab]\!]$ is 3. Denote by $\langle x, y, z\rangle$ the function mapping 1 to x, 2 to y, and 3 to z. The identity function, $\langle 1, 2, 3\rangle$, corresponds to λ. From there, a, b, and c act as $\langle 2, 2, 3\rangle$, $\langle 1, 3, 3\rangle$ and $\langle 1, 1, 3\rangle$, respectively. The complete structure extends from these. Consider ab: this first applies the a mapping, then applies that of b to its result. So $\langle 1, 2, 3\rangle$ maps first to $\langle 2, 2, 3\rangle$ by a then to $\langle 3, 3, 3\rangle$ by b. By the same process, we find that $aa = ca = a$, $ac = cb = cc = c$, $bb = bc = b$ and ba is a new state $\langle 2, 3, 3\rangle$. Extending $ab = \langle 3, 3, 3\rangle$ and $ba = \langle 2, 3, 3\rangle$, we find $ab{\cdot}a = ab{\cdot}b = ab{\cdot}c = ba{\cdot}b = ab$, $ba{\cdot}a = ba$ and finally $ba \cdot c = b$. Iteration generated no new states, so the process is complete. This conforms to the structure shown in Figure 1, whose Cayley graph is shown at the top in Table 1.

Note that the complement of the language forbidding ab substrings – the language of words with ab substrings – shares the same syntactic semigroup. This holds in general: an automaton and its complement share the same algebraic structure, as state parity is independent from the actions of transitions. It follows that classes defined purely by semigroup properties must be closed under complement.

Now consider the transducer of Figure 2. This transducer is a representation of intervocalic voicing, a phonological process where voiceless obstruents become voiced between vowels. As a phonological rule this is T→D/V__V. For example, this transducer maps the string TVTVD to TVDVD.

	a	b	c	ab	ba
a	a	ab	c	ab	ab
b	ba	b	b	ab	ba
c	a	c	c	ab	a
ab	ab	ab	ab	ab	ab
ba	ba	ab	b	ab	ab

	T	V	D	VT
T	D	V	D	VT
V	VT	V	D	VT
D	D	V	D	VT
VT	D	V	D	VT

Table 1: The Cayley table for the syntactic semigroups in Figure 1 (above) and Figure 2 (below).

The transducer above is in canonical form, where each state represents one $\overset{N}{\sim}$ class. State 2 is all those strings that end in V, state 3 those ending in VT, and state 1 all others. The five actions are the identity $\langle 1, 2, 3\rangle$ corresponding to λ, $\langle 1, 1, 1\rangle$, $\langle 1, 3, 1\rangle$, and $\langle 2, 2, 2\rangle$ corresponding to D, T, and V, respectively, and finally $\langle 3, 3, 3\rangle$ for VT. One can verify that for each class, some context distinguishes its words from words in each other class, and that no context distinguishes words within a class. For example, a V__V context separates λ and T, as for λ the following V contributes V alone while for T it contributes DV. Technically, $\langle \mathrm{V}, \mathrm{V}, \mathrm{V}\rangle \in C(\lambda)$ while $\langle \mathrm{V}, \mathrm{V}, \mathrm{DV}\rangle \in C(\mathrm{T})$, but by determinism the triples are unique in their first two components. A VT__λ context separates λ and D, as the λ contributes T to the former but λ to the latter. That no context distinguishes strings within a class is guaranteed by the construction. The Cayley graph corresponding to the monoid in Figure 2 is shown at bottom in Table 1.

This construction appears to discard output information, but it is recoverable. Outputs may be compatibly assigned to the states and edges and the result used as a transducer. Its structure is the same as that of the string language in which all words end in "VT". This notion of structural equivalence gives rise to a deep theory of function complexity.

2.5 Definite Algebraic Structure

A string language L is **definite** if can be defined by a finite set X of permitted suffixes: $L = \{wv :$

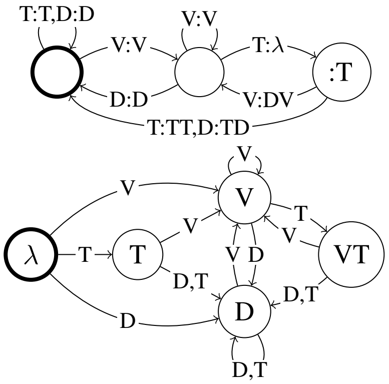

Figure 2: Transducer and monoid for "T becomes D directly between two V".

$w \in \Sigma^*, v \in X\}$ (Perles et al., 1963). The class of definite languages is denoted $\mathcal{D}$. Because X is a finite set it holds some longest string of length n. Whether a string belongs to L can be decided by examining its last n symbols. Such languages are called n-definite. More generally, as the canonical acceptor for a definite language processes strings, the states correspond to strings in Σ^n representing the most recent history. In the sense of Jurafsky and Martin (2008) then, the state space of definite languages is Markovian.

The definite languages were one of the early classes of formal languages to be given an algebraic characterization (Brzozowski and Simon, 1973; Brzozowski and Fich, 1984). Many algebraic structures are defined in terms of idempotents. An element e of a monoid is **idempotent** if $e \cdot e = e$. As an example, the idempotents of the syntactic semigroups shown in 1 are $\{a, b, c, ab\}$ and $\{\mathrm{V}, \mathrm{D}, \mathrm{VT}\}$, respectively. Denote by E the set of idempotents.

An algebraic property characterizes exactly the definite languages (Brzozowski and Simon, 1973; Brzozowski and Fich, 1984). Syntactic semigroups of definite languages have the property that for all $e \in E$, $x \in S$, it holds that $xe = e$, often written $Se = e$ with universal quantification left implicit.

The string language which forbids ab substrings is not definite. This follows from the algebraic characterization and from the Cayley table for this language in Table 1. While b is an idempotent (since $b \cdot b = b$), $a \cdot b = ab \neq b$. Thus $Se \neq e$.

For intervocalic voicing it holds that $Se = e$ for all its idempotents $e \in \{\mathrm{V}, \mathrm{D}, \mathrm{VT}\}$. One verifies this by examining their columns in the Cayley table in Table 1. As its minimal transducer processes input, the most recently read symbols fix its state.

The syntactic semigroups such that $Se = e$ form the variety $\mathcal{D}$ (Brzozowski and Simon, 1973; Brzozowski and Fich, 1984). It follows they are closed under subsemigroup, quotients, finite direct products, and thus the Boolean operations. This variety has played a key role in developing an algebraic theory of recognizable languages (Straubing, 1985).

3 Input Strictly Local Functions

Chandlee et al. (2014) define input strictly local transducers by a restriction on the tails, inducing a canonical structure. A function is input strictly local iff for some natural number k, the function is definable by a sequential transducer whose states are labeled by $\Sigma^{<k}$, q_0 is the state labeled by λ, and edges are of the form $\delta(a, q) = \langle w, \mathrm{Suff}^{k-1}(qa)\rangle$. The suffix function is defined as expected:

$$\mathrm{Suff}^n(w) = \begin{cases} \lambda & \text{if } n \leqslant 0, \\ w & \text{if } |w| \leqslant n, \\ v & \text{if } w = uv \text{ for } u \in \Sigma^*, v \in \Sigma^n. \end{cases}$$

This canonical form is a monoid. The operation $u \cdot v = \mathrm{Suff}^{k-1}(u \cdot v)$ is associative, and λ is the identity. Let f be a function, $\overrightarrow{S}$ and $\overleftarrow{S}$ be the semigroups of the left-to-right and right-to-left transducers associated with f, respectively, and e range over idempotents of the appropriate semigroup.

Theorem 1. *The following are equivalent:*

- *f is a total input strictly local function*
- *f is $\rightarrow \mathcal{D}$: $\overrightarrow{S}e = e$*
- *f is $\leftarrow \mathcal{D}$: $\overleftarrow{S}e = e$*

Proof. The nonidentity idempotent elements of this monoid are Σ^{k-1}, as if $x \in \Sigma^{k-1}$ we have $x = \mathrm{Suff}^{k-1}(x) = \mathrm{Suff}^{k-1}(xx)$ and if $x \in \Sigma^{<k-1}$ we instead have $x \neq \mathrm{Suff}^{k-1}(xx)$. If $x \in \Sigma^{k-1}$ we have that $\mathrm{Suff}^{k-1}(ux) = x$ for all $u \in \Sigma^*$, so for all elements s it holds that $s \cdot x = x$. In other words, $Se = e$ for all idempotent elements e in the syntactic semigroup (which excludes the identity). This is the property characterizing definite languages, defined by a set of permitted suffixes (Brzozowski and Simon, 1973; Brzozowski and Fich, 1984).

The directionality statement follows from the fact that input strictly local functions are not directional (Chandlee and Heinz, 2018). □

The canonical form of an input strictly local transducer is the same as that of a definite string language (Perles et al., 1963). Both are defined by the k most recent symbols encountered fixing the state, with no long-distance effects. Indeed, this class has been discussed as the definite (Krohn et al., 1967; Stiffler, 1973) or local (Vaysse, 1986) functions decades before Chandlee et al. (2014) introduced them to linguists as input strictly local.

We invoke this characterization of input strictly local functions as definite structures to provide an effective decision procedure for the class. First, it is converted to a canonical sequential form by the algorithm of Mohri (1997). If conversion fails, the map is certainly not in the class, as it is not even sequential. Otherwise, the syntactic semigroup is constructible by the algorithms shown in section 2 (McNaughton and Papert, 1971). Finally one needs only to check that for each idempotent e and each element s, $se = e$. Recall that the identity is in the semigroup iff it is reachable by a nonempty string.

Strictly local string languages follow the same structure but additionally allow transition to a rejecting sink in lieu of some otherwise expected transitions. These changes do not necessarily retain the algebraic structure, but a semigroup can be regenerated by the usual method. Accounting for whether a factor in some fixed set has ever occurred admits some long-distance dependency.

4 Composing Functions

Closure properties provide important insight into classes of languages. An intersection-closed class admits new patterns satisfying its properties defined by coöccurrence of patterns in that class. Many subregular classes are so closed, and learning a strictly piecewise pattern as a coöccurrence of constraints has proven more effective than learning a single pattern (Heinz and Rogers, 2013). (Pseudo)varieties of finite semigroups are closed under finitary products, subsemigroups, and quotients (Eilenberg and Schützenberger, 1976). Intersections and unions of automata are computed from a product, extracting the reachable subsemigroup and minimizing the result by a quotient. Automata share structure with their complements, so varieties define classes closed under Boolean operations. The property defining definite languages, that $Se = e$ for all idempotents e, yields a variety, $\mathbf{D}$, of finite semigroups. These languages are then Boolean-closed.

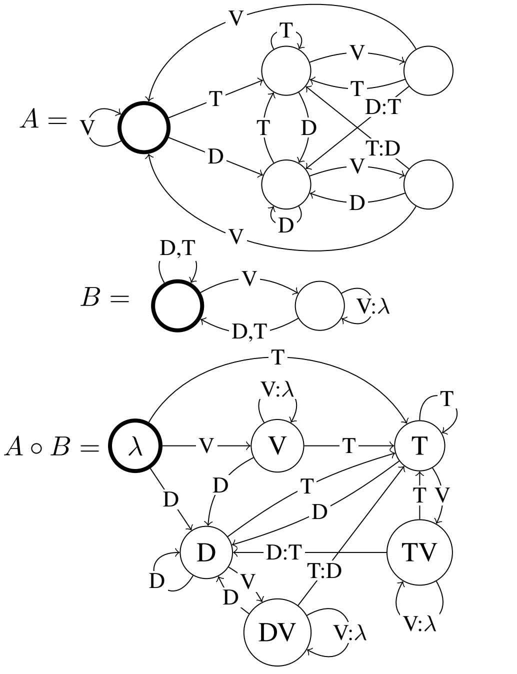

Figure 3: ISL is not composition-closed.

If coöccurring factors are a basic unit of string languages, composed rules might be a basic unit of functions. Let δ^* denote the transitive closure of δ:

$$\delta^*(w,x) = \begin{cases} \langle\lambda, x\rangle & w = \lambda \\ \langle uv, y\rangle & w = aw', a \in \Sigma, \\ & \langle u, x'\rangle = \delta(a,x), \\ & \langle v, y\rangle = \delta^*(w', x') \end{cases}$$

Then if $f = \langle\Delta, \Gamma, Q_f, \delta_f, q_{0f}, \pi_f, \sigma_f\rangle$ and $g = \langle\Sigma, \Delta, Q_g, \delta_g, q_{0g}, \pi_g, \sigma_g\rangle$, the composition $f \circ g$ computes the result of applying f to the output of g. This composition is effectively constructible (Mohri, 1997). A construction is as follows:

$$\begin{aligned} f \circ g &= \langle\Sigma, \Gamma, Q_f \times Q_g, \delta_\circ, q_i, \pi_f\alpha, \sigma_\circ\rangle \\ \langle\alpha, r\rangle &= \delta_f^*(\pi_g, q_{0f}) \\ q_i &= \langle r, q_{0g}\rangle \\ \delta_\circ(a, \langle m,n\rangle) &= \Big\{\langle w, \langle s,t\rangle\rangle \colon \delta_g(a,n) = \langle u,t\rangle, \\ &\qquad \delta_f^*(u,m) = \langle w,s\rangle\Big\} \\ \sigma_\circ(\langle m,n\rangle) &= \sigma_f(\delta_f^*(m, \sigma_g(n))_1) \end{aligned}$$

This composition is not a direct product in the algebraic sense. The state space is the product space, but the action is not the natural pointwise action defining the direct product. Thus, composition closure is not free and in general does not hold.

The transducers shown in Figure 3 exhibit this nonclosure for definite functions. The first, A, is simultaneous application of two rules, D→T/TV__ and T→D/DV__, a voicing assimilation across a single vowel. Then B is a vowel-span truncation: V→∅/V__. By applying B and then A, the context in which T or D changes becomes unboundedly long. The strings V^n and DV^n have the same n-suffix for any n, but a suffixed T contributes a T to the first and a D to the latter. The two have differing tails, failing to satisfy input strict locality. In semigroup terms, V is idempotent as V and VV lie in the same class, but DVV is DV and not V. Thus $Se \neq e$ and the function is not definite. In fact, the resulting monoid is not even locally a semilattice (locally testable, Brzozowski and Simon, 1973) nor $\mathscr{J}$-trivial (piecewise testable, Simon, 1975). It is everywhere-idempotent, which in string languages would imply definability in two-variable first-order logic of general precedence alone (Brzozowski and Fich, 1984; Kufleitner and Weil, 2010).

One subclass of definite functions is composition closed: that where only bounded spans may delete.

Theorem 2. *If f and g are definite functions and if all input sequences of length k to g are guaranteed to produce nonempty output, then $f \circ g$ is definite.*

Proof. If f is m-definite, g is n-definite, and input sequences of g are guaranteed to contribute nonempty output after at most k symbols, then after mk input symbols, g must have produced at least m intermediate output symbols. This fixes the state in f. The state of g is fixed after n or more symbols. So the degree of definiteness of $f \circ g$ is at most the greater of n and mk. □

Corollary 1. *The subclass of definite functions deleting only bounded spans is composition closed.*

Proof. If f and g are definite and guarantee nonempty output after reading at most k and ℓ symbols, respectively, then $f \circ g$ yields nonempty output after reading at most ℓk symbols. The composition remains in this subclass. □

The machines of Figure 3 do not compose to a definite machine because unbounded spans of V delete, collapsing to just V no matter their length.

Many phonologically relevant patterns lie in this subclass, including some that optimality theory has struggled to analyze (Chandlee et al., 2018). Interconsonantal schwa deletion, intervocalic voicing, and word-final devoicing are each computed by transducers where all input sequences of length two contribute nonempty output. In this order, their composition is shown in Figure 4, and it is definite of degree four: every string of length three synchronizes the machine. Moreover all length four input sequences produce nonempty output.

Readers familiar with the literature on local functions may recall results that seem stronger than our Theorem 2. For example, Sakarovitch (2009, p. 664) states that if g is a proper local function of degree d and f a local function of degree d', then the composition $f \circ g$ is local of degree $d + d'$. In that work, a proper local function is one in which no deletion occurs. This is a more restrictive constraint than our own, as we allow for deletion in bounded spans. Similarly, Vaysse (1986, p. 168) states that the composition of any local function f of degree d and any local function g of degree d' is local of degree $d + d' - 1$. This, however, takes place in a richer setting in which transitions not only might append symbols to the output, but also might delete previous symbols. Neither previous result is directly applicable here.

5 Other Kinds of Operations

Although a subclass of the definite functions is closed under composition, the class as a whole is not. In general, function composition does not preserve algebraic properties. This section discusses a general type of machine that unifies transducers, DFAs, weighted automata, and more. Operations on these general automata that behave like products will preserve algebraic properties and allow complex systems to be factored in an algebraically natural way. Some such operations are shown here.

The outputs of a transducer influence its semigroup structure only by preventing state merges. Mohri (Lothaire, 2005) describes a more general notion of a transducer whose outputs are elements in some semiring rather than mere strings. Sequential transducers are input-deterministic, so the operation combining paths is unnecessary. We can think about machines whose output lies in some monoid. Standard transducers satisfy this property: if the output alphabet is Δ then the output monoid is Δ^* under concatenation.

Consider then a system in which the output monoid is not Δ^* but regular languages over Δ. The form of the output is irrelevant, but for concreteness suppose we are dealing with DFAs over Δ. A definite transducer may be translated directly into this form by replacing the output strings with a DFA accepting that string alone, with one state more

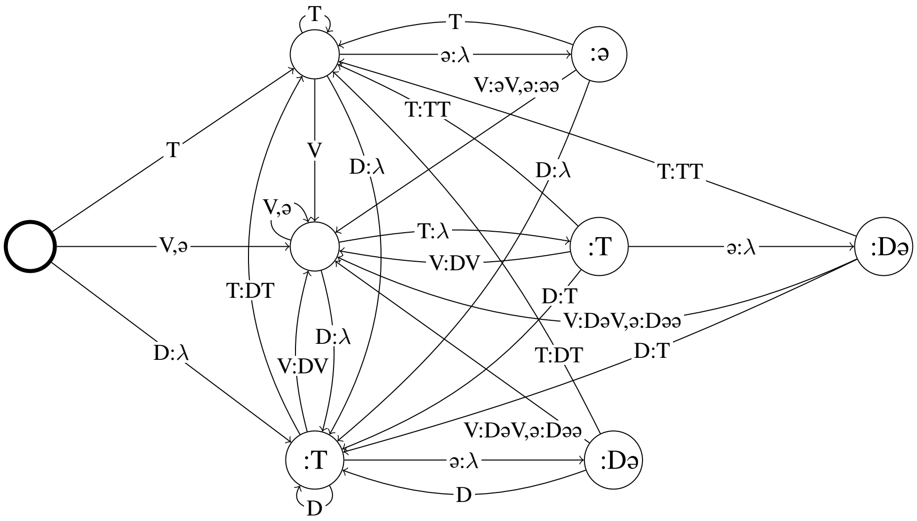

Figure 4: Interconsonantal schwa deletion, then intervocalic voicing, then word-final devoicing, all composed.

than the string's length. We define three distinct products over this structure: pointwise evaluation, union, and parallel application with preference. In the following discussion, machines are defined with an output monoid in place of an output alphabet.

If we have $f = \langle \Sigma, \Delta^*, Q_f, \delta_f, q_{0f}, \pi_f, \sigma_f \rangle$ and $g = \langle \Sigma, \Gamma^*, Q_g, \delta_g, q_{0g}, \pi_g, \sigma_g \rangle$, we define their pointwise evaluation product, $f \odot g$, as follows:

$$f \odot g = \big\langle \Sigma, \Delta^* \times \Gamma^*, Q_f \times Q_g, \delta_{f \odot g}, \langle q_{0f}, q_{0g} \rangle, \langle \pi_f, \pi_g \rangle, \sigma_{f \odot g} \big\rangle,$$

where $\sigma_{f \odot g}(\langle q, r \rangle) = \langle \sigma_f(q), \sigma_g(r) \rangle$, pointwise application of suffixing, and if $\delta_f(a, q) = \langle u, q' \rangle$ and $\delta_g(a, r) = \langle v, r' \rangle$ then $\delta_{f \odot g}(a, \langle q, r \rangle) = \langle \langle u, v \rangle, \langle q', r' \rangle \rangle$. The operation is pointwise concatenation: $\langle a, b \rangle \cdot \langle c, d \rangle = \langle ac, bd \rangle$. The pair that $f \odot g$ derives from an input w juxtaposes the result of applying f to w or that of applying g to w.

Let $\mathscr{A}_X$ represent the DFAs over alphabet X. If we have $f = \langle \Sigma, \mathscr{A}_\Delta, Q_f, \delta_f, q_{0f}, \pi_f, \sigma_f \rangle$ and $g = \langle \Sigma, \mathscr{A}_\Gamma, Q_g, \delta_g, q_{0g}, \pi_g, \sigma_g \rangle$, we define the unioned product of f and g, $f \boxdot g$ as follows:

$$f \boxdot g = \big\langle \Sigma, \mathscr{A}_{\Delta \cup \Gamma}, Q_f \times Q_g, \delta_{f \boxdot g}, \langle q_{0f}, q_{0g} \rangle, \{\pi_f, \pi_g\}, \sigma_{f \boxdot g} \big\rangle,$$

where $\sigma_{f \boxdot g}(\langle q, r \rangle) = \{\sigma_f(q), \sigma_g(r)\}$, the union of the outputs of the two suffixing functions, and if $\delta_f(a, q) = \langle u, q' \rangle$ and $\delta_g(a, r) = \langle v, r' \rangle$ then $\delta_{f \boxdot g}(a, \langle q, r \rangle) = \langle u \cup v, \langle q', r' \rangle \rangle$. Every input symbol admits choice, applying either f or g.

For homogeneous functions we have a final operation: apply both at once, outputting from the left machine if it changes the input, else from the right machine. Let $f = \langle \Sigma, \Sigma^*, Q_f, \delta_f, q_{0f}, \pi_f, \sigma_f \rangle$ and $g = \langle \Sigma, \Sigma^*, Q_g, \delta_g, q_{0g}, \pi_g, \sigma_g \rangle$, and define this change-preferring product as follows:

$$f \Diamond g = \big\langle \Sigma, \Sigma^*, Q_f \times Q_g, \delta_{f \Diamond g}, \langle q_{0f}, q_{0g} \rangle, \pi_{f \Diamond g}, \sigma_{f \Diamond g} \big\rangle,$$

where $\pi_{f \Diamond g}$ is equal to π_f unless that is λ in which case it is equal to π_g, and similarly $\sigma_{f \Diamond g}(\langle q, r \rangle)$ is equal to $\sigma_f(q)$ unless that is λ in which case it is equal to $\sigma_g(r)$, and finally if $\delta_f(a, q) = \langle u, q' \rangle$ and $\delta_g(a, r) = \langle v, r' \rangle$ then $\delta_{f \Diamond g}(a, \langle q, r \rangle) = \langle w, \langle q', r' \rangle \rangle$, where $w = v$ if $u = a$ or else $w = u$. For two processes that do not affect one another, this is algebraic-property preserving composition.

These combinators are built on the product construction that Rabin and Scott (1959) and Hopcroft and Ullman (1979) use for unions or intersections of DFAs. The transition semigroup of the result is the product of those of the inputs. Definite machines are defined by a variety, and so are product closed, which means the $\odot$, $\boxdot$, and $\Diamond$ combinators yield definite machines from definite inputs.

Consider then that deletion of schwa between two consonants is definite, defined by the rule ə→∅/C__C. This is a definite function, by the construction used by Chandlee (2014). The identity function is also definite, having but a single state. Applying $\boxdot$ yields the (definite) deterministic rational relation of Figure 5 implementing optional interconsonantal schwa deletion. Some deterministic rational relations have been studied (Beros and de la Higuera, 2016), and this algebraic perspective

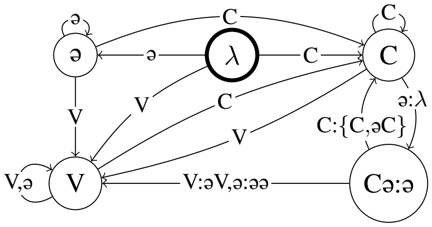

Figure 5: Optional schwa deletion between consonants.

offers a general mechanism for dealing with them.

6 Conclusion

Input strictly local maps suffice to describe a large portion of phonological processes. They are definite functions (Krohn et al., 1967; Stiffler, 1973). They are local functions (Vaysse, 1986; Sakarovitch, 2009). Given a minimal sequential finite-state transducer representing a mapping, we showed that it is decidable in time polynomial in the size of the transition semigroup of the machine whether the process is input strictly local: all idempotents must be right zeros. Using this characterization, we have shown that this class of functions cannot be closed under composition, but that this closure does hold for a restricted subclass in which deletion may occur only in bounded spans.

In these functions, only a local context around a symbol can influence its output. They do not exhibit the long-distance effects that strictly local string languages allow, where a single factor might cause computation to fall into a sink state for the remainder of the run. Definite languages are all strictly local, but so are, say, reverse definite languages. These have the opposite characterization, defined by semigroups where $eS = e$. These are the functions where $\mathrm{Pref}^{k-1}(u) = \mathrm{Pref}^{k-1}(v)$ implies $u \overset{N}{\sim} v$, that $T(u) = T(v)$.

Current research involves exploring the function analogues of some of the other classes that correspond to subregular string languages, such as these reverse definite functions, and classifying natural language patterns accordingly (Lambert, 2022). Additional lines of future research include better understanding how algebraic properties can fuel grammatical inference of string functions (de la Higuera, 2010), and the factorization of string functions into component parts along the lines of Rogers and Lambert (2019).

References

Achilles Beros and Colin de la Higuera. 2016. A canonical semi-deterministic transducer. *Fundamenta Informaticae*, 146(4):431–459.

Véronique Bruyère and Christophe Reutenauer. 1999. A proof of Choffrut's theorem on subsequential functions. *Theoretical Computer Science*, 215(1–2):329–335.

Janusz Antoni Brzozowski and Faith Ellen Fich. 1984. On generalized locally testable languages. *Discrete Mathematics*, 50:153–169.

Janusz Antoni Brzozowski and Imre Simon. 1973. Characterizations of locally testable events. *Discrete Mathematics*, 4(3):243–271.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–503.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language*, pages 112–125, Chicago, USA. Association for Computational Linguistics.

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

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

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. 2008. Deciding strictly local (SL) languages. In *Proceedings of the 2008 Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pages 66–73.

Samuel Eilenberg and Marcel-Paul Schützenberger. 1976. On pseudovarieties. *Advances in Mathematics*, 19(3):413–418.

Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. 2016. First-order definability of rational transductions: An algebraic approach. In *LICS '16: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 387–396. Association for Computing Machinery.

R. W. N. Goedemans, Jeffrey Heinz, and Harry van der Hulst. 2015. StressTyp2.

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

Jeffrey Heinz and James Rogers. 2013. Learning subregular classes of languages with factored deterministic automata. In *Proceedings of the 13th Meeting on the Mathematics of Language*, pages 64–71, Sofia, Bulgaria. Association for Computational Linguistics.

Markus Holzer and Barbara König. 2004. On deterministic finite automata and syntactic monoid size. *Theoretical Computer Science*, 327(3):319–347.

John Edward Hopcroft and Jeffrey David Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, 2nd edition. Prentice-Hall, Upper Saddle River, NJ.

Kenneth Krohn, Richard Mateosian, and John Rhodes. 1967. Methods of the algebraic theory of machines: Decomposition theorem for generalized machines; Properties preserved under series and parallel compositions of machines. *Journal of Computer and System Sciences*, 1(1):55–85.

Manfred Kufleitner and Pascal Weil. 2010. On the lattice of sub-pseudovarieties of **DA**. *Semigroup Forum*, 81:243–254.

Dakotah Lambert. 2022. *Unifying Classification Schemes for Languages and Processes with Attention to Locality and Relativizations Thereof*. Ph.D. thesis, Stony Brook University.

Silvain Lombardy and Jacques Sakarovitch. 2006. Sequential? *Theoretical Computer Science*, 356(1–2):224–244.

M. Lothaire. 2005. *Applied Combinatorics on Words*. Cambridge University Press, New York.

Robert McNaughton and Seymour Aubrey Papert. 1971. *Counter-Free Automata*. MIT Press.

Jeff Mielke. 2008. *The Emergence of Distinctive Features*. Oxford University Press, New York, NY.

George Armitage Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97.

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

Anil Nerode. 1958. Linear automaton transformations. In *Proceedings of the American Mathematical Society*, volume 9, pages 541–544. American Mathematical Society.

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

Micha A. Perles, Michael Oser Rabin, and Eliahu Shamir. 1963. The theory of definite automata. *IEEE Transactions on Electronic Computers*, 12(3):233–243.

Jean-Éric Pin. 1997. Syntactic semigroups. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, pages 679–746. Springer-Verlag, Berlin.

Michael Oser Rabin and Dana Scott. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125.

James Rogers and Dakotah Lambert. 2019. Extracting Subregular constraints from Regular stringsets. *Journal of Language Modelling*, 7(2):143–176.

James Rogers and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20(3):329–342.

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press.

Imre Simon. 1975. Piecewise testable events. In Helmut Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer-Verlag, Berlin.

Price Stiffler, Jr. 1973. Extension of the fundamental theorem of finite semigroups. *Advances in Mathematics*, 11(2):159–209.

Howard Straubing. 1985. Finite semigroup varieties of the form $V * D$. *Journal of Pure and Applied Algebra*, 36:53–94.

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

Bohdan Zelinka. 1981. Graphs of semigroups. *Časopis pro pěstování matematiky*, 106(4):407–408.