Some further properties of BMRS transductions

Chris Oakden[*], Siddharth Bhaskar[†], and Adam Jardine[*]

DRAFT
October 10, 2020

1 Introduction

A key result due to Engelfriet and Hoogeboom (2001) establishes the relationship between monadic second-order (MSO) logical transduction and regular functions on strings (those describable by two-way finite state transducers), echoing earlier foundational studies of logic-automata connections with formal languages (Elgot, 1961; Büchi, 1960; Trakhtenbrot, 1961). Logical transductions are interpretations in which alphabet labels on output string positions and order on output string indices are defined by unary and binary predicates in the logical language of the input strings. These interpretations are relativized over a copy set; every output element is a copy of an input index, and its properties—its label and relative order—are determined by which predicates are satisfied by input strings at a particular index. Thus the output string may be longer than the input string.

Subsequent work has identified further connections between sub-MSO logics and subregular function classes. For example, Bhaskar et al. (2020) provide a logical characterization of the subsequential functions, a sub-class of the rational functions with particular relevance to grammatical inference (Oncina et al., 1993), speech and language processing (Mohri, 1997), and computational theories of natural language phonology (Heinz, 2018; Heinz and Lai, 2013). In particular, they define boolean monadic recursive schemes (BMRSs), a restriction on recursive program schemes (Moschovakis, 2019). BMRSs recursively define a set of unary functions that take an input string index and return a boolean value. When these functions are assigned to symbols in an output alphabet, the system expresses a logical transduction over strings, such that the output string is defined by the symbols whose functions return a true value at each input index. Bhaskar et al. (2020) show that BMRS string transductions defined with predecessor and successor functions describe exactly the left- and right-subsequential functions, respectively.

In this paper, we expand on Bhaskar et al. (2020)'s result by investigating further properties of BMRS transductions, their connection to regular functions, and their applications in natural language phonology. We introduce a syntactic operator over BMRS transductions that is equivalent to function composition. The scope of this operator is limited to length-preserving transductions—that is, where input and output string lengths are identical. Using this operator,

[*]Linguistics Department, Rutgers University; chris.oakden@rutgers.edu, adam.jardine@rutgers.edu
[†]Datalogisk Institut, Københavns Universitet, Copenhagen 2100, Denmark; sbhaskar@di.ku.dk

we offer a proof showing that BMRS transductions without predecessor/successor restrictions describe exactly the regular functions. Additionally, we demonstrate applications of BMRS composition to modeling phonological process interactions. For example, it models interactions of serially-ordered rewrite rules in a rule-based formalism (Chomsky and Halle, 1968) given the equivalence of ordering and function composition (Kaplan and Kay, 1994).

This paper is structured as follows. Sections 2 establishes notation and definitions for strings, and Section 3 establishes BMRSs and BMRS transductions. Section 4 defines a composition operator for strict, length-preserving transductions, and proves that it is equivalent to function composition. Section 5 utlilizes this operator to prove that BMRS transductions (with no restrictions on predecessor and successor functions) describe exactly the regular functions. In Section 6, we discuss the applications of a BMRS composition operator to modeling interactions in natural language phonology and identify areas for future work. Section 6 concludes.

2 Preliminaries

We assume standard set-theoretic and formal language-theoretic notation. An *alphabet* Σ is a finite set of symbols; let Σ^* indicate all strings of any length of symbols in Σ. For a string $w = \sigma_1 ... \sigma_n$ we write $w[i]$ for σ_i (note that we start counting with 1) and $|w|$ for n (i.e., the length of w).

For an alphabet Σ we identify a string with a *model* $\langle D; \sigma_1, ..., \sigma_{|\Sigma|}, p, s \rangle$ where U is the universe of string indices (canonically $\{1, ..., n\}$ for a string of length n), σ for each $\sigma \in \Sigma$ is a unary relation containing the set of positions labeled with σ, and p and s are the *predecessor* and *successor* functions, respectively, on the indices of the string. By convention we say that $p(1)$ and $s(n)$ (for a string of length n) are undefined. We will often use $x - i$ and $x + i$ as abbreviations for i applications of p and i applications of s, respectively, to x. We abuse types somewhat and refer to strings and their models interchangeably. Similarly, we can associate the signature of strings over Σ with Σ itself.

3 Boolean monadic recursive schemes and string transductions

3.1 Boolean monadic recursive schemes

Syntax. BMRSs are a restriction on the *recursive programs* of Moschovakis (2019) that operate over these structures. The syntax and semantics are as follows. First, we assume two countably infinite sets, a set $\mathtt{X}$ of *index variables* and set $\mathtt{F}$ of *recursive function names*. Each recursive function name $\mathtt{f} \in \mathtt{F}$ comes with an arity n and output type; here the arity of each $\mathtt{f} \in \mathtt{F}$ is 1 (hence *monadic*) and its type is either `index` or `boolean`.

Assuming string models over Σ *terms* are then defined by the following grammar.

$$T \rightarrow \ \top \mid \bot \mid \mathtt{x} \text{ (for } \mathtt{x} \in \mathtt{X}) \mid \mathtt{f}(T) \text{ (for } \mathtt{f} \in \mathtt{F}) \mid \sigma(T) \text{ (for } \sigma \in \Sigma) \mid \\ p(T) \mid s(T) \mid T = T \mid \mathtt{if}\ T\ \mathtt{then}\ T\ \mathtt{else}\ T \tag{1}$$

The terms $\top$ and $\bot$ are of type `boolean` and any variable term $\mathtt{x}$ is of type `index`; all other terms inherit their type inductively from one of these terms. Terms are *well-formed* if

they conform to the usual typing rules: for terms $\mathtt{f}(T)$, $\sigma(T)$, $p(T)$, or $s(T)$ the type of T must be `index`, for a term $T_1 = T_2$ the type of T_1 and T_2 must be the same, and for a term `if` T_1 `then` T_2 `else` T_3 the type of T_1 must be `boolean` and T_2 and T_3 must be of the same type. We only consider well-formed terms and thus henceforth refer to them just as terms.

A *program* consists of a tuple $(\mathtt{f}_1, ..., \mathtt{f}_k)$ of `boolean`-typed function names paired with k terms of the form $\mathtt{f}_i(\mathtt{x}) = T_i$ where T_i is a `boolean` term that contains a single index variable `x` and whose recursive function variables are drawn from $\mathtt{f}_1, ..., \mathtt{f}_k$. We write a program

$$\begin{aligned} \mathtt{f}_1(\mathtt{x}) &= T_1(\mathtt{x}, \vec{\mathtt{f}}) \\ &\vdots \\ \mathtt{f}_k(\mathtt{x}) &= T_k(\mathtt{x}, \vec{\mathtt{f}}) \end{aligned}$$

where $\vec{\mathtt{f}} = \mathtt{f}_1, ...\mathtt{f}_k$, to indicate that these properties hold. When clear from context we omit $\vec{\mathtt{f}}$.

Semantics. Given a string model with universe U and $B = \{\bot, \top\}$, a term $T_i(\mathtt{x}, \vec{\mathtt{f}})$ denotes a monotone functional $T_1(x, \vec{f})$ that takes as arguments elements in U and partial functions from U to B and returns some value in B. The semantics of a program over functions $(\mathtt{f}_1, ..., \mathtt{f}_k)$ is the least-fixed point solution $(\overline{f}_1, ..., \overline{f}_k)$ of the system of equations

$$\begin{aligned} f_1(x) &= T_1(x, \vec{f}) \\ &... \\ f_k(x) &= T_k(x, \vec{f}) \end{aligned}$$

Such a least-point solution is always guaranteed to exist (Moschovakis, 2019).

It will be useful to illustrate how to generate this solution with a single equation (for which the notation is much cleaner). The equation $f(x) = T(x, f)$ has the least-fixed point solution

$$\overline{f} = \bigcup_{k \in \mathbb{N}} \{\overline{f}^k\}$$

where $\overline{f}^0$ is the totally undefined function, and $\overline{f}^{k+1} = T(x, \overline{f}^k)$. Here, 'least' refers to the subfunction relation (i.e., the order $\sqsubseteq$ defined as $f \sqsubseteq g$ iff $f(x) = g(x)$ wherever $f(x)$ is defined). These functions of the form $\overline{f}^k$ are called the *iterates*.

The following is crucial in relating BMRS systems of equations to regular functions on strings.

Remark 1 *A BMRS system of equations can be reduced to a series of definitions in MSO formulae.*

For $\vec{\mathtt{f}} = (\mathtt{f}_1, ..., \mathtt{f}_k)$ we take a series $\mathtt{X} = (\mathtt{X}_1, ..., \mathtt{X}_k)$ of monadic second-order variables. For each term $T_i(\mathtt{x}, \vec{\mathtt{f}})$ in the system of equations create a MSO formula $\phi_i(\mathtt{x}, \vec{\mathtt{X}})$ by replacing each instance of $\mathtt{f}_j(T)$ in T_i with $\mathtt{X}_j(T)$. Then replace each definition $\mathtt{f}_i(\mathtt{x}) = T_i(\mathtt{x}, \vec{f})$ with

$$\mathtt{f}_i(\mathtt{x}) = (\exists \vec{\mathtt{X}}\, \forall \mathtt{y}) \left[\left(\bigwedge_{1 \leq j \leq k} (\phi_j(\mathtt{y}, \vec{\mathtt{X}}) \rightarrow \mathtt{X}_j(\mathtt{y})) \right) \wedge \mathtt{X}_i(\mathtt{x}) \right].$$

3.2 BMRS transductions

We can define a string function $f : \Sigma^* \to \Gamma^*$ via a BMRS *interpretation* based on the techniques of (Courcelle, 1994; Engelfriet and Hoogeboom, 2001). Fix a *copy set* $C = \{1, \ldots, n\}$ as some initial segment of the natural numbers. A BMRS transduction is thus a program $\mathcal{T}$ of Σ-terms defining the recursive functions $\vec{\mathtt{f}} = (\gamma_1^1, \ldots, \gamma_1^n, \gamma_2^1, \ldots, \gamma_{|\Gamma|}^n, \mathtt{f}_1, \ldots, \mathtt{f}_k)$; that is, with a boolean function $\gamma_i^c(\mathtt{x})$ for each $\gamma_i^c \in \Gamma \times C$, some set of boolean functions $\mathtt{f}_1(\mathtt{x}), ..., \mathtt{f}_k(\mathtt{x})$. That is, we define the sets $\gamma \in \Gamma$ of the output Γ-string in terms of the input Σ-string.

The semantics of $\mathcal{T}$ given an input string $w \in \Sigma^*$ with a universe U as follows. For each $u \in U$, we output a copy u^c of u if and only if there is exactly one $\gamma \in \Gamma$ for $c \in C$ such that $\gamma^c(\mathtt{x}) \in T$ evaluates to $\top$ when $\mathtt{x}$ is mapped to d. We fix the order of these output copies to be derived from the order on C and the order on D induced by the input predecessor function p. This output order can be represented explicitly with a series of output (`index`-typed) predecessor functions p^c for each $c \in C$ defined as below:

$$\begin{aligned} p^1(\mathtt{x}) &= \mathtt{if}\ \mathtt{out}^n(p(x))\ \mathtt{then}\ p(\mathtt{x})\ \mathtt{else}\ p^c(p(\mathtt{x})) \\ p^{i+1}(\mathtt{x}) &= \mathtt{if}\ \mathtt{out}^i(\mathtt{x})\ \mathtt{then}\ \mathtt{x}\ \mathtt{else}\ p^i(\mathtt{x}) \end{aligned}$$

where $\mathtt{out}^c(T) = \gamma_1^c(T) \vee \gamma_2^c(T) \vee ... \vee \gamma_n^c(T)$. As this output predecessor function is fixed it is not necessary to specify as part of a transduction.

This semantics of T thus defines a string transduction $t = t(T)$ where for a string $w \in \Sigma^*$ of length ℓ, $t(w) = u_0 u_1 ... u_\ell$, where each $u_i = \gamma_1 ... \gamma_r$ if and only if for each γ_j, $1 \leq j \leq r$, γ_j is the unique symbol in Γ for $j \in C$ such that $\gamma_j^j(\mathtt{x})$ evaluates to $\top$ when $\mathtt{x}$ is assigned to i in the structure of w. An example is given in Example 1; it describes a function from strings in $\Sigma = \{0, 1\}$ to strings in $\Gamma = \{a, b\}$ such that 0 maps to a single a and 1 maps to two bs.

Example 1 *Let $\mathcal{T}$ be a transduction with a copy set $C = \{1, 2\}$ from strings over $\Gamma = \{0, 1\}$ to strings over $\Delta = \{a, b\}$ be defined as*

$$\begin{aligned} a^1(x) &= 0(x) \\ a^2(x) &= \bot \\ b^1(x) &= 1(x) \\ b^2(x) &= 1(x) \end{aligned}$$

The following shows how this maps 10010 to bbaabba.

Input:	1	0	0	1	0
Copy 1:	*b*	*a*	*a*	*b*	*a*
Copy 2:	*b*			*b*	

Let BMRS^p be the transductions defined by some BMRS tranducer $\mathcal{T}$ in which the term $s(T)$ does not appear; and likewise BMRS^s the transductions in which the term $p(T)$ does not appear. BMRS^p describes exactly the left-subsequential functions, and BMRS^s describes exactly the right-subsequential functions (Bhaskar et al., 2020). Note that Bhaskar et al. (2020) use special boundary symbols to obtain this exact equivalence; we omit this technical detail here as it is not relevant to the present results.

4 Composition of length-preserving BMRS transductions

Let Γ and Σ be alphabets, and $\mathcal{T}$ be a BMRS^p transduction over a copy set $C = \{1, ..., n\}$ such that $t(\mathcal{T}) : \Sigma^* \to \Gamma^*$. We say $\mathcal{T}$ is *strict* iff for $t = t(\mathcal{T})$, $|t(w)| = n|w|$ for all $w \in \Sigma^*$. That is, $\mathcal{T}$ is strict iff for any index i in any string $w \in \Sigma^*$, exactly one $\gamma^c(i)$ evaluates to true. That is, no deletion occurs.

It is simpler to define and prove composition for strict transductions, so we establish this first. The reason for this is that the output precedence relation is equivalent to the following

$$\begin{aligned} p^1(\mathtt{x}) &\equiv p(\mathtt{x}) \\ p^{i+1}(\mathtt{x}) &\equiv \mathtt{x} \end{aligned}$$

because $\mathtt{out}^c(\mathtt{x})$ can always be assumed to be true. This greatly simplifies 'aligning' the outputs and inputs of two transductions. Furthermore, if $\mathcal{T}$ is strict and $C = \{1\}$, then $\mathcal{T}$ is *length-preserving*, and

$$p^1(\mathtt{x}) \equiv p(\mathtt{x}).$$

The composition of length-preserving transductions thus allows us to focus on the main thrust of composition without worrying about calculating indices, so we begin there.

Take two strict, length-preserving BMRS^p transductions $\mathcal{T}$ and $\mathcal{S}$, where $\mathcal{T}$ describes a function $t_1 : \Sigma^* \to \Delta^*$ with a set $\vec{\mathtt{f}}$ of recursive function symbols and a copy set $C = \{1\}$, and $\mathcal{S}$ describes $t_2 : \Delta^* \to \Gamma^*$ with a set $\vec{\mathtt{g}}$ of recursive function symbols and a copy set $D = \{1\}$. We aim to define a syntactic operation $\mathcal{T} \otimes \mathcal{S}$ that describes $t_1 \circ t_2 : \Sigma^* \to \Gamma^*$. This section gives a definition specific to length-preserving transductions.

Note that since both $\mathcal{T}$ and $\mathcal{S}$ are length-preserving, for both transductions $p^1(\mathtt{x}) = p(\mathtt{x})$; that is, the output predecessor function p^1 describes the same order over indices as the input predecessor function p. This means that function definitions in $\mathcal{S}$ can take indices directly from the output of $\mathcal{T}$.

We define $\mathcal{T} \otimes \mathcal{S}$ by replacing, for each definition in $\mathcal{S}$, each atomic term of the form $\delta(T)$ with $\delta^1(T)$. This makes the input boolean functions of $\mathcal{S}$ compatible with the output boolean functions of $\mathcal{T}$.

Let $\mathcal{S}'$ be $\mathcal{S}$ with these terms so replaced. Then $\mathcal{T} \otimes \mathcal{S} = \mathcal{T} \cup \mathcal{S}'$.

Example 2 *Let $\mathcal{T}$ be a transduction with a copy set $C = \{1\}$ from strings over $\Sigma = \{0, 1\}$ to strings over $\Delta = \{a, b\}$ be defined as*

$$\begin{aligned} a^1(\mathtt{x}) &= 0(\mathtt{x}) \\ b^1(\mathtt{x}) &= 1(\mathtt{x}) \end{aligned}$$

Then, e.g., $t_1(10010) = baaba$.

Let $\mathcal{S}$ be a transduction with a copy set $D = \{1\}$ from strings in $\Delta = \{a, b\}$ to strings in $\Gamma = \{c, d, e\}$ be defined as

$$\begin{aligned} c^1(\mathtt{x}) &= \textit{ if } e^1(\mathtt{x}) \textit{ then } \perp \textit{ else } a(\mathtt{x}) \\ d^1(\mathtt{x}) &= b(x) \\ e^1(\mathtt{x}) &= \textit{ if } a(\mathtt{x}) \textit{ then } b(\mathtt{x} - 1) \textit{ else } \perp \end{aligned}$$

Then, e.g., $t_2(baaba) = decde$.

Note that both $\mathcal{T}$ and $\mathcal{S}$ are strict and length-preserving. Then $\mathcal{S}'$ is

$$\begin{aligned} c^1(\mathtt{x}) &= \textit{ if } e^1(\mathtt{x}) \textit{ then } \perp \textit{ else } a^1(\mathtt{x}) \\ d^1(\mathtt{x}) &= b^1(\mathtt{x}) \\ e^1(\mathtt{x}) &= \textit{ if } a^1(\mathtt{x}) \textit{ then } b^1(\mathtt{x}-1) \textit{ else } \perp \end{aligned}$$

$\mathcal{T} \otimes \mathcal{S}$ simply puts $\mathcal{T}$ together with $\mathcal{S}'$. Let $t_{1,2} = t(\mathcal{T} \otimes \mathcal{S})$. Then $t_{1,2}(10010) = \textit{decde}$.

Theorem 1 (Composition of length-preserving transductions) $t(\mathcal{T} \otimes \mathcal{S}) = t(\mathcal{S}) \circ t(\mathcal{T})$.

Proof: Let $t_1 = t(\mathcal{T})$ and $t_2 = t(\mathcal{S})$ and let $t_{1,2} = t(\mathcal{T} \otimes \mathcal{S})$. Consider an arbitrary $w \in \Sigma^*$, and let $u \in \Delta^*$ be the string such that $u = t_1(w)$. We show that $t_2(u) = t_{1,2}(w)$.

Recall that by the definition of BMRS transductions, the universe U_u of positions in u are constructed from $U_w \times C$, where U_w is the universe of positions in w and C is the copy set of $\mathcal{T}$. The predecessor function p_u on u is likewise determined by p^1 as defined in $\mathcal{T}$. As noted, because $p^1(\mathtt{x}) \equiv p(\mathtt{x})$, then p_u is identical to the predecessor function p_w on w. We can thus establish a direct correspondence between the ith position in u with the ith position in w.

From this, it then follows for any $\delta \in \Delta$, for the input term $\delta(\mathtt{x}-k)$ in $\mathcal{S}$, $\delta(i-k)$ evaluates to $\top$ in u iff $\delta^1(i-k)$ evaluates to $\top$ in w for $\delta^1(\mathtt{x})$ in $\mathcal{T}$ (replacing $\mathtt{x}$ with $i-k$ in the latter case).

From the definition of $\otimes$, $\mathcal{T} \subseteq \mathcal{T} \otimes \mathcal{S}$, and so we can show by the induction on the definition of the least-fixed point solution for $\mathcal{S}$ that for any recursive function symbol $\mathtt{g} \in \vec{\mathtt{g}}$, for the function $\mathtt{g}_2(\mathtt{x})$ defined in $\mathcal{S}$, $\mathtt{g}_2(i)$ evaluates to $\top$ in u iff for the corresponding function $\mathtt{g}_{1,2}(\mathtt{x})$ defined in $\mathcal{T} \otimes \mathcal{S}$, $\mathtt{g}_{1,2}(i)$ evaluates to true in w.

We do so by recursing on the least-fixed point solutions of the definitions $\mathtt{g}_2(\mathtt{x}) = T_2$ in $\mathcal{S}$ and $\mathtt{g}_{1,2}(\mathtt{x}) = T_{1,2}$ in $\mathcal{T} \otimes \mathcal{S}$. That is, we consider $\overline{g_2}^k$ and $\overline{g_{1,2}}^k$ for the kth iterator for finding the least fixed point solutions for each system of equations. Recall that from the definition of $\otimes$, T_2 and $T_{1,2}$ are identical except that any term $\delta(\mathtt{x}-j)$ in T_2 has been replaced with $\delta^1(\mathtt{x}-j)$ in $T_{1,2}$.

When $k = 0$, it is trivially true as by definition, both $\overline{g_2}^0(x)$ and $\overline{g_{1,2}}^0(x)$ are undefined for any x. For the inductive step we look at the definitions of $\mathtt{g_2}^{k+1}$ take as a hypothesis that for any $\mathtt{h} \in \vec{\mathtt{g}}$, the iterators $\overline{h_2}^k$ and $\overline{h_{1,2}}^k$ for the functions $\mathtt{h}_2(\mathtt{x})$ defined in $\mathcal{S}$ and $\mathtt{h}_{1,2}(\mathtt{x})$ defined in $\mathcal{T} \otimes \mathcal{S}$ behave identically. That is, for any i, $\overline{h_2}^k(i) = \top$ in u iff $\overline{h_{1,2}}^k(i) = \top$ in w. Thus for any term $\mathtt{h}(\mathtt{x}-j)$ in T_2 and $T_{1,2}$, $\mathtt{h}(i-j)$ evaluates to $\top$ in u iff $\mathtt{h}(i-j)$ evaluates to $\top$ in w. As established above, for any term $\delta(\mathtt{x}-j)$ in T_2, $\delta(i-j)$ evaluates to $\top$ in u iff $\delta^1(i-j)$ evaluates to $\top$ in w for the corresponding term $\delta^1(\mathtt{x}-j)$ in $T_{1,2}$. As the predecessor functions are also identical, in T_2 and $T_{1,2}$ all of the atomic BMRS^p terms behave equivalently; thus by recursing on the semantics of terms $\overline{g_2}^{k+1}(i)$ evaluates to $\top$ in u iff $\overline{g_{1,2}}^{k+1}(i)$ evaluates to $\top$ in w. By induction on the definition of the least fixed point solutions we then have $\overline{g_2}(i)$ evaluates to $\top$ in u iff $\overline{g_{1,2}}(i)$ evaluates to $\top$ in w.

Thus, whenever the ith position of u is output as γ under $\mathcal{S}$, the ith position of w is output as γ under $\mathcal{T} \otimes \mathcal{S}$. Thus, $t_2(u) = t_{1,2}(w)$. □

5 BMRS=REG

Let $\text{BMRS}^{p,s}$ be the transductions defined with no restrictions on using p or s. We connect these to the regular functions using two results from the literature. First, the *regular* (REG) functions are those that are describable by order-preserving MSO transductions.

Theorem 2 (Filiot 2015) *A function f is in* REG *iff there is an order-preserving MSO transduction $\mathcal{T}$ such that $f = t(\mathcal{T})$.*

Second, regular functions are exactly those that can be decomposed into the composition of a left- and right-subsequential function.

Theorem 3 (Elgot and Mezei 1965) *Any regular function $f = f_\ell \circ f_r$ for one left- and one right-subsequential function f_ℓ and f_r, respectively.*

The following thus follows from these two theorems and the above results.

Theorem 4 $\text{BMRS}^{p,s} = \text{REG}$

Proof: For $\text{BMRS}^{p,s} \subseteq \text{REG}$, by Remark 1 we can convert any BMRS tranduction into an MSO transduction. The inclusion thus follows from Thm 2.

For $\text{REG} \subseteq \text{BMRS}^{p,s}$, consider the decomposition of a REG function f into left- and right-subsequential functions f_ℓ and f_r. By Bhaskar et al. (2020), f_ℓ is described by a BMRS^p transduction $\mathcal{T}_\ell$ and likewise f_r admits a BMRS^s transduction $\mathcal{T}_r$. The inclusion thus follows from the fact that $\mathcal{T}_\ell \otimes \mathcal{T}_r \in \text{BMRS}^{p,s}$ and that by Thm. 1, $f = t(\mathcal{T}_\ell \otimes \mathcal{T}_r)$. □

6 Discussion

A composition operator over length-preserving BMRS transductions is useful in modeling process interactions in natural language phonology. Given the connection between function composition and serial ordering in a rule-based framework, composing BMRS transductions that describe individual rewrite rules mirrors the ordering of those rules and interactions that arise from particular orderings. This section presents an idealized example using an alphabet $\Sigma = \{a, b, c, d\}$.

Rule-based theories of phonology represent phonological grammars as an ordered list of individual rewrite rules. Broadly speaking, rules can interact in one of four ways (McCarthy, 2007). Given two rules A, B such that A precedes B in a serial ordering,

a. A **feeds** B iff A creates additional inputs to B.

b. A **bleeds** B iff A eliminates potential inputs to B.

c. B **counterfeeds** A iff B creates additional inputs to A.

d. B **counterbleeds** A iff B eliminates additional inputs to A.

In what follows, we consider two rules A and B: A is of the form $a \to b \ / \ c\_\_$, and B is of the form $b \to d \ / \ b\_\_$. Each rule is described as a BMRS transduction, and composition of these transductions models a feeding or a counterfeeding relationship, depending on order.

Example 3 *Let $\mathcal{T}$ be a transduction with a copy set $C = \{1\}$ from strings over $\Sigma = \{a, b, c, d\}$ to strings over $\Delta = \Sigma = \{a, b, c, d\}$ be defined as*

$$
\begin{aligned}
a^1(\mathtt{x}) &= \mathtt{if}\ c(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x}) \\
b^1(\mathtt{x}) &= \mathtt{if}\ c(\mathtt{x}-1)\ \mathtt{then}\ a(\mathtt{x})\ \mathtt{else}\ b(\mathtt{x}) \\
c^1(\mathtt{x}) &= c(\mathtt{x}) \\
d^1(\mathtt{x}) &= d(\mathtt{x})
\end{aligned}
$$

Let $t_1 = t(\mathcal{T})$. Then, e.g., $t_1(acaba) = acbba$. This describes the application of rule A.

Let $\mathcal{S}$ be a transduction with a copy set $D = \{1\}$ from strings over $\Sigma = \{a, b, c, d\}$ to strings over $\Delta = \Sigma = \{a, b, c, d\}$ be defined as

$$
\begin{aligned}
a^2(\mathtt{x}) &= a(\mathtt{x}) \\
b^2(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ b(\mathtt{x}) \\
c^2(\mathtt{x}) &= c(\mathtt{x}) \\
d^2(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x}-1)\ \mathtt{then}\ b(\mathtt{x})\ \mathtt{else}\ d(\mathtt{x})
\end{aligned}
$$

Let $t_2 = t(\mathcal{S})$. Then, e.g., $t_2(acbba) = acbda$. This describes the application of rule B.

Then $\mathcal{T} \otimes \mathcal{S}$ is $\mathcal{T} \cup \mathcal{S}'$:

$$
\begin{aligned}
a^2(\mathtt{x}) &= a^1(\mathtt{x}) \\
b^2(\mathtt{x}) &= \mathtt{if}\ b^1(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ b^1(\mathtt{x}) \\
c^2(\mathtt{x}) &= c^1(\mathtt{x}) \\
d^2(\mathtt{x}) &= \mathtt{if}\ b^1(\mathtt{x}-1)\ \mathtt{then}\ b^1(\mathtt{x})\ \mathtt{else}\ d^1(\mathtt{x}) \\
a^1(\mathtt{x}) &= \mathtt{if}\ c(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x}) \\
b^1(\mathtt{x}) &= \mathtt{if}\ c(\mathtt{x}-1)\ \mathtt{then}\ a(\mathtt{x})\ \mathtt{else}\ b(\mathtt{x}) \\
c^1(\mathtt{x}) &= c(\mathtt{x}) \\
d^1(\mathtt{x}) &= d(\mathtt{x})
\end{aligned}
$$

Let $t_{1,2} = t(\mathcal{T} \otimes \mathcal{S})$, which, by Thm 1, is equivalent to $t(\mathcal{S}) \circ t(\mathcal{T})$. Then $t_{1,2}(acaba) = acbda$.

The composite transduction $\mathcal{T} \otimes \mathcal{S}$ is equivalent to the rule ordering A $<$ B, that is, where A applies first, and then B applies to the output of A. This is shown in a rule-based derivation in Figure 1. Crucially, the application of A creates the environment for the application of B, thus A feeds B.

Example 4 gives the opposite order of composition of the two transductions, and therefore is equivalent to the reverse ordering B $<$ A.

Example 4 *Let $\mathcal{T}$ and $\mathcal{S}$ be the transductions from Example 3, again such that $t_1 = t(\mathcal{T})$ and $t_2 = t(\mathcal{S})$.*

	/*acaba*/
Rule A: $a \to b$ / c__	*acbba*
Rule B: $b \to d$ / b__	*acbda*
	[*acbda*]

Figure 1: Derivation of *acaba* using rule order A < B

	/*acaba*/
Rule B: $b \to d$ / b__	*acaba*
Rule A: $a \to b$ / c__	*acbba*
	[*acbba*]

Figure 2: Derivation of *acaba* using rule order A < B

Then $\mathcal{S} \otimes \mathcal{T}$ *is* $\mathcal{S} \cup \mathcal{T}'$*:*

$$
\begin{aligned}
a^1(\mathtt{x}) &= \mathtt{if}\ c^2(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ a^2(\mathtt{x}) \\
b^1(\mathtt{x}) &= \mathtt{if}\ c^2(\mathtt{x}-1)\ \mathtt{then}\ a^2(\mathtt{x})\ \mathtt{else}\ b^2(\mathtt{x}) \\
c^1(\mathtt{x}) &= c^2(\mathtt{x}) \\
d^1(\mathtt{x}) &= d^2(\mathtt{x}) \\
a^2(\mathtt{x}) &= a(\mathtt{x}) \\
b^2(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x}-1)\ \mathtt{then}\ \bot\ \mathtt{else}\ b(\mathtt{x}) \\
c^2(\mathtt{x}) &= c(\mathtt{x}) \\
d^2(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x}-1)\ \mathtt{then}\ b(\mathtt{x})\ \mathtt{else}\ d(\mathtt{x})
\end{aligned}
$$

Let $t_{2,1} = t(\mathcal{S} \otimes \mathcal{T})$*, which, by Thm 1, is equivalent to* $t(\mathcal{T}) \circ t(\mathcal{S})$*. Then* $t_{1,2}(acaba) = acbba$*.*

The composition $\mathcal{S} \otimes \mathcal{T}$ is equivalent to the ordering B < A, where B applies (vacuously) to *acaba*, then A applies to its output to give *acbba*. Figure 2 provides a derivation of *acaba*; this is reflective of a counterfeeding relationship between rules B and A.

Thus, composition of length-preserving BMRS transductions provides a means to describe ordering relationships between rules in a rule-based framework of phonology.

In this paper, we define a composition operator for length-preserving BMRS transductions. A full definition would necessarily encompass composition of transductions like those in Example 1, which are neither length-preserving nor strict (as deletion is allowed). Such a definition would lay the groundwork for additional proofs regarding the relationship between a broader range of BMRS transductions and regular functions. Defining composition for non-length-preserving and (especially) non-strict transductions is considerably messier compared to the straight-forward composition of length-preserving transductions pursued here. This is due to difficulties in aligning input and output indices for transductions defined over copy sets of different sizes (and when deletion occurs). We therefore leave this task for future work.

7 Conclusion

This paper has defined a composition operator for strict, length-preserving BMRS transductions, and proved its equivalence to function composition. Using this operator, the paper has also shown that the full class of BMRS transductions describes the regular class of functions. It also has identified applications of BMRS composition to natural language phonology, namely a means to model phonological process interactions via rule ordering. The results introduced in this paper provide a foundation for further study of the connections between subclasses of regular functions and applications of the BMRS formalism to natural language phonology.

References

Bhaskar, S., Chandlee, J., Jardine, A., and Oakden, C. (2020). Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In Leporati, A., Martín-Vide, C., Shapira, D., and Zandron, C., editors, *Language and Automata Theory and Applications - LATA 2020*, Lecture Notes in Computer Science, pages 157–169. Springer.

Büchi, J. R. (1960). Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathmatik*, 6:66—92.

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

Courcelle, B. (1994). Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126:53–75.

Elgot, C. C. (1961). Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51.

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

Engelfriet, J. and Hoogeboom, H. J. (2001). MSO definable string transductions and two-way finite-state transducers. *ACM Transations on Computational Logic*, 2:216–254.

Filiot, E. (2015). Logic-automata connections for transformations. In *Logic and Its Applications (ICLA)*, pages 30–57. Springer.

Heinz, J. (2018). The computational nature of phonological generalizations. In Hyman, L. and Plank, F., editors, *Phonological Typology*, Phonetics and Phonology, chapter 5, pages 126–195. De Gruyter Mouton.

Heinz, J. and Lai, R. (2013). Vowel harmony and subsequentiality. In Kornai, A. and Kuhlmann, M., editors, *Proceedings of the 13th Meeting on Mathematics of Language*, Sofia, Bulgaria.

Kaplan, R. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20:331–78.

McCarthy, J. J. (2007). Derivations and levels of representation. In *The Cambridge Handbook of Phonology*, pages 99–117. Cambridge: Cambridge University Press.

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

Moschovakis, Y. N. (2019). *Abstract recursion and intrinsic complexity*, volume 48 of *Lecture Notes in Logic*. Cambridge University Press.

Oncina, J., García, P., and Vidal, E. (1993). Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458.

Trakhtenbrot, B. A. (1961). Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329.