Phonological Processes as Modal Transductions

Tatevik Yolyan[1,2], **Jesse Comer**[1]
[1]University of Pennsylvania,
[2]Rutgers University, New Brunswick
Correspondence: tatevik.yolyan@rutgers.edu

Abstract

This paper argues in favor of a fundamentally new perspective on phonology via *modal logic*. We show that the class of total *Boolean Monadic Recursive Schemes* (**BMRS**), used in computational modeling of phonological processes (Bhaskar et al., 2020; Chandlee and Jardine, 2021), is equivalent in expressive power to the well-studied *modal μ-calculus*. As a corollary of this result, we obtain an alternative proof that order-preserving **BMRS** transductions capture the class of rational functions, which have been posited as a complexity bound on natural language phonological grammars.

1 Introduction

Rational functions are word-to-word maps which are realized by non-deterministic one-way finite-state transducers (FSTs).[1] Within phonological theory, the Rational Hypothesis argues that phonological maps in natural language fall within the restrictive class of rational functions (Kaplan and Kay, 1994; Johnson, 1972). We take the Rational Hypothesis as the starting point for this paper, and present a new formal perspective on rational functions and consequently, phonological maps. The conceptual map for this paper is presented in Figure 1; the relationship between phonological maps and finite state transducers that is central to the Rational Hypothesis constitutes the edge labeled (a).

(1) **The Rational Hypothesis:** Natural language phonological maps are rational.

The Rational Hypothesis provides a formal characterization of what kinds of *computations* are attested in natural language phonological grammars, and moreover predicts what kinds of computations *cannot* be realized as a phonological map in *any* natural language.

[1]In-depth discussion of rational functions can be found in (Berstel, 2007) and (Filiot, 2015).

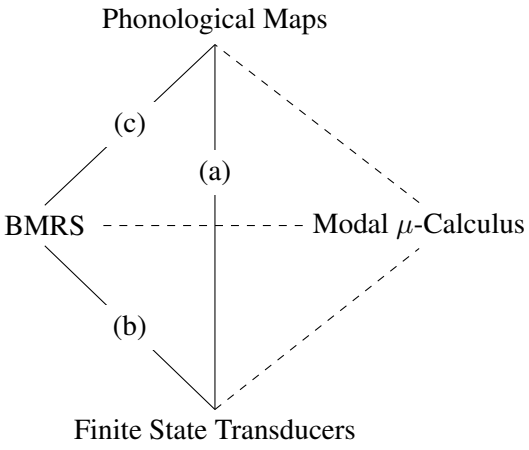

Figure 1: Conceptual map of this paper. (a): Kaplan and Kay (1994); Johnson (1972); (b): Bhaskar et al. (2020, 2023); (c): Chandlee and Jardine (2021); Yolyan (2025b,c). This paper discusses the relationships labeled with dashed lines.

It is worth noting that the rational relations are often referred to as 'regular' in computational phonology literature, and the Rational Hypothesis is better-known as the **Subregular Hypothesis** (e.g. Heinz, 2018). Though the terms 'rational' and 'regular' get conflated, it is important to keep the following distinction in mind: the *regular* relations are those which can be expressed with a *two-way* non-deterministic FST, and are therefore more expressive than the rational functions.

In recent work, Bhaskar et al. (2020, 2023) introduced Boolean Monadic Recursive Schemes (BMRS), a programming language interpreted over words, as a *logical characterization* of the rational functions. More specifically, they show that the so-called *well-defined, order-preserving* **BMRS** interpretations capture precisely the class of rational functions on strings. Their work constitutes the edge labeled (b) in Figure 1. Chandlee and Jardine (2021) present **BMRS** as an alternative framework for representing phonologi-

Proceedings of the Society for Computation in Linguistics (SCiL) 2026, pages 549–559

cal maps because **BMRS** programs can be used to formally represent both the computational nature of phonological maps as well as relevant *phonological generalizations*. Subsequent work arguing for the merits of **BMRS** for phonological theory is also found in (Yolyan, 2025b,c). This relationship constitutes edge (c) of Figure 1.

This paper presents a *modal logic perspective* on rational transductions and phonological maps. More specifically, we present the modal μ-calculus, which is an extension of propositional modal logic with a *fixed point operator*, which permits recursion and self-reference in formulas (Bradfield and Stirling, 2006). There are a few motivating factors for pursuing a modal perspective. First, the modal μ-calculus is a logic with a well-understood relationship to automata and formal languages. A detailed discussion of the relationship between the modal μ-calculus and alternating tree automata can be found in Emerson and Jutla (1991); Wilke (2001). In the setting of words, the modal perspective provides an alternative logical framework for expressing phonological transductions. Second, the μ-calculus employs a least fixed point semantics for evaluating recursive formulas. Recursive formulas are also used in **BMRS** to express long-distance and iterative string maps. Establishing a connection between the semantics of these two separate formalisms means that existing results on the formal properties of the modal μ-calculus can provide new insights into the **BMRS** framework. Third, the modal perspective may provide new insights on computational learning of phonological maps. A significant consequence of the Rational Hypothesis is that it restricts the learning space for phonological maps. This restricted space has led to advances in learning FST representations of rational functions, such as Chandlee (2014); Jardine et al. (2014); Oncina et al. (1993). More recent work from Yolyan (2025a) presents an approach to learning phonological maps represented as **BMRS** programs. There have also been recent advances on learning μ-calculus formulas, such as Koudijs (2022); ten Cate and Koudijs (2024). A formal connection between modal formulas and **BMRS** programs may ultimately lead to new insights that are valuable for learning.

With these motivating factors in mind, this paper makes several contributions which make up the three dotted lines in Figure 1. The main result of this paper is that the modal μ-calculus and total **BMRS** define the same unary properties over words. More precisely, for each output predicate of a total **BMRS** program, there is a modal μ-calculus formula which is true at all and only indices in a word for which that output predicate is true. An immediate consequence of our main result is that it is possible to represent rational functions as order-preserving interpretations of μ-calculus formulas, which have not been pursued in previous modal logic literature. This paper can moreover be viewed as an extension of previous work by Graf (2010), who uses modal logic to express well-formedness constraints on surface forms in phonology. The fixed point operator in the modal μ-calculus is the crucial formal component that makes it possible to lift modal logic formulas that represent surface form constraints to modal μ-calculus formulas which, given the formal results of this paper, represent phonological maps. An additional potential contribution of this paper is that it may be of interest to modal logicians, as it provides a new way to express modal μ-calculus expressions and connects modal logic to programming languages.

The structure of this paper is as follows. Section 2 gives a high-level presentation of how FSTs, **BMRS**, and modal logic are used to represent phonological maps. This section provides the motivation for this work, and the intuition for why a modal logic perspective is warranted. A more formal presentation of **BMRS** and the modal μ-calculus follow in Sections 3 and 4. The main result of this paper is presented in Section 5, followed by concluding remarks in Section 6.

2 Phonological Transductions

This section presents two examples of phonological maps to illustrate how FSTs, **BMRS**, and modal logic can be used to represent phonological maps. We present first a non-recursive example in (2) in order to provide an informal introduction to the three frameworks. We then consider an iterative process in (5) to demonstrate the evaluation of recursive predicates. Formal details of BMRS and modal logic follow in the subsequent two sections.

Consider the vowel nasalization map in (2), in which a vowel becomes [+nasal] when it precedes a nasal sound. The FST representation of this map is presented in Figure 2 with features N for nasals and V for vowels. The three states of this FST represent three possible sounds in the input. For ex-

ample, the machine transitions to the state labeled $+-$ when it reads an input that is [+V,−N]. The FST reads the input from right to left. The highlighted transition [+V,−N]:[+V,+N] from state $+-$ to $-+$ represents the fact that a [−N] vowel becomes [+N] when the previous sound that was read in the input was [+N].

(2) Vowel Nasalization
V → [+nasal] /__[+nasal].

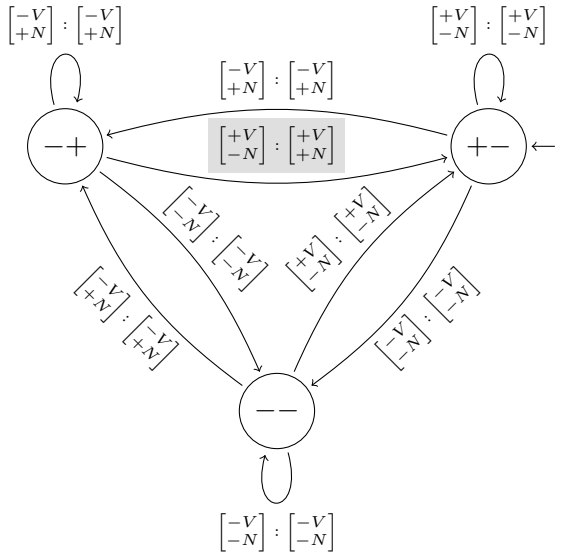

Figure 2: FST representation of vowel nasalization, where V and N represent vowel and nasal sounds.

The **BMRS** formalism provides a framework for expressing phonological maps as quantifier-free logical transductions. These transductions operate on relational structures called *words*.

Definition 1. *Given a set of unary predicate symbols Σ (called a* signature*), a Σ-word is a structure $\boldsymbol{w} = \langle w, s^w, p^w, (\sigma^w)_{\sigma\in\Sigma}\rangle$, where w is a finite set of indices; s^w and p^w denote (partial) successor and predecessor functions on w, undefined only at the last and first indices, respectively; and each σ^w is a subset of w. A Σ-string is a Σ-word with exactly one predicate true at each index. We write $\boldsymbol{Words}_\Sigma$ to denote the class of Σ-words.*

When logical transductions are adapted for representing phonological maps, the unary predicates in the word represent phonological features. Within the BMRS framework in particular, the predicates σ^w are represented with Boolean-valued predicates such that $\sigma^w(x) = \top$ if and only if $x \in \sigma^w$. Logical transductions represent maps in the following way: for every predicate σ^w of a word **w**, the logical transduction defines a new word **w′** with predicates σ' (called *output predicates*). The BMRS representation of vowel nasalization is given in (3) with predicates N and V, for nasal and vowel sounds. The two output predicates N' and V' define an output string by specifying which indices of the output string are nasals and vowels, respectively.

(3) BMRS representation of vowel nasalization
$V'(x) \quad = V(x)$
$N'(x) \quad =$ **if** $V(x)$ **then** $N(\mathrm{s}(x))$ **else** $N(x)$[2]

Note that **BMRS** programs are expressed using a `if...then...else` syntax. The output predicate $N'(x)$ is evaluated as follows. For every index x, if x is a vowel in the input word, then x is [+N] in the output word if and only if index $x+1$ of the input word is [+N]. Otherwise, x is [+N] in the output word if and only if it is [+N] in the input word. Consider an underlying-surface form pair of words (/bæn/,[bæ̃n]). This input-output relationship is realized by the transduction in Figure 3.

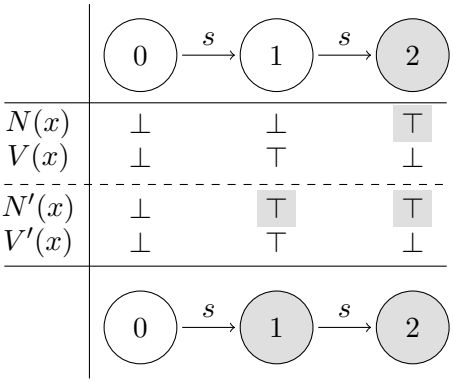

Figure 3: Representation of vowel nasalization as a logic transduction over words with features N and V. The highlighted nodes in the input and output words represent the [+N] elements of the words.

In its full generality, modal logic operates over relational structures called *labeled transition systems* which are very similar to words with two main distinctions that we highlight here. First, the relational structures have relations between nodes

[2]This formula does not account for the edges of words (i.e. when $\mathrm{s}(x)$ is not defined). More precisely, we should set

$$N'(x) = \textbf{if } V(x) \textbf{ then } (\textbf{if } \max(x) \textbf{ then } N(x) \textbf{ else } N(\mathrm{s}(x))) \textbf{ else } N(x),$$

where $\max(x) = \top$ if and only if x is the final index of a word.

(called *accessibility relations*) rather than successor and predecessor functions. Second, we use *modalities* to refer to neighboring indices. For example, given a model with accessibility relation R, the modal expression $\Diamond P$ evaluates to $\top$ at some node x if and only if there is some node y such that $R(x, y)$ and P holds at y. If the relation R is the graph of the successor function (i.e. $R = \{(x, y) \mid y = \mathrm{s}(x)\}$) then $\Diamond P$ evaluates to $\top$ at some index x of a word if and only if $P(\mathrm{s}(x))$ holds. With this equivalence in mind, the BMRS formulas in (3) can be equivalently expressed as the modal formulas in (4), which we refer to as a *modal transduction*. Note that the formulas in (4) do not have variables because the $\Diamond$ modality makes it possible to refer to adjacent indices without a variable.[3]

(4) Modal representation of vowel nasalization

$$V' = V$$
$$N' = (V \wedge \Diamond N) \vee N$$

We consider now an *iterative* nasalization process. In (2), a single vowel becomes nasalized when it is followed by a nasal; no other sounds in a word are affected. (5) presents a vowel nasalization process from Warao in which vowels become nasalized when they are preceded by a nasal sound, and this process iterates through a word until a voiceless stop suspends the nasalization process.

(5) Nasal Spreading in Warao (Osborn, 1966)

a.	esoha-ya	'He pours'
b.	**nãõ-ỹã**	'He saw'
c.	**nãõ**-te	'He will come'
d.	ho**nĩw̃ã**ku-hae	'It is a turtle'
e.	pa**nãpanã-h̃ãẽ**	'It is a porpoise'

The **BMRS** representation of the nasal spreading process in Warao is presented in (6), where the predicate T is a shorthand for voiceless stops, and $\min$ is the predicate which is true only at index 0. Notably, the output predicate N' is *recursive*.

(6) BMRS representation of nasal spreading

$$N'(x) = \textbf{if } N(x) \textbf{ then } \top \textbf{ else}$$
$$\textbf{if } T(x) \textbf{ then } \bot \textbf{ else}$$
$$\textbf{if } \min(x) \textbf{ then } \bot \textbf{ else}$$
$$\mathrm{N}'(px)$$

[3]The modal formulas also use the fact that conjunction and disjunction can be expressed as `if...then...else` expressions. For example, $p \wedge q$ can be rewritten as `if` p `then` q `else` $\bot$.

Figure 4 shows the evaluation of the recursive predicate N' for the input-output pair of words (/naote/, [nãõte]). This predicate is evaluated as follows. At index 0, N' evaluates to $\top$ because N holds at index 0. At index 3, N' evaluates to $\bot$ because T holds at index 3. For the remaining indices, $N'(x)$ cannot be evaluated to $\top$ or $\bot$ because $\mathrm{N}'(px)$ does not have a valuation. However, a second run through the `if...then...else` expression yields valuations for indices 1 and 4 because $\mathrm{N}'(0)$ and $\mathrm{N}'(3)$ have valuations. At a third run of the expression, we also get a valuation for index 2. Beyond this point, further iterations of $N'(x)$ will not yield any changes; the final row in Figure 4 is the *least fixed point* of $N'(x)$.

input	n	a	o	t	e
index	0	1	2	3	4
predicates	N	V	V	T	V
$\mathrm{N}'(x)$	$\top$			$\bot$	
$\mathrm{N}'(x)$	$\top$	$\top$		$\bot$	$\bot$
$\mathrm{N}'(x)$	$\top$	$\top$	$\top$	$\bot$	$\bot$
output	n	ã	õ	t	e

Figure 4: Evaluation of the recursive predicate $N'(x)$.

The modal formula which expresses the recursive predicate in (6) is presented in (7). This formula uses the backward modality $\blacklozenge$, where $\blacklozenge P$ evaluates to $\top$ at some node y if and only if there is some node x such that $R(x, y)$ and P holds at x. Intuitively, the backward modality is the analog of the predecessor function over words. The recursive aspect of the formula is handled by the modal operator μX which scopes over $N \vee (\neg T \wedge \blacklozenge X)$ and binds the variable X. The evaluation of this formula is analogous to the evaluation of the recursive BMRS formula in the following sense: the expression $N \vee (\neg T \wedge \blacklozenge X)$ gets re-evaluated iteratively until reaching a fixed point.

(7) Modal representation of nasal spreading

$$N' = \mu X.(N \vee (\neg T \wedge \blacklozenge X))$$

Figure 5 shows the evaluation of the modal μ-calculus expression in (7). We start with X being false at all indices, and evaluate the formula $N \vee (\neg T \wedge \blacklozenge X)$. This formula is true at index 0 (because N is true at 0), and false everywhere else. This becomes the new semantics for X. In other words, X becomes true at index 0 and remains false everywhere else, as shown in the second row of the table. We then evaluate the for-

input	n	a	o	t	e
index	0	1	2	3	4
predicates	N	V	V	T	V
X	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
X	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
X	$\top$	$\top$	$\bot$	$\bot$	$\bot$
X	$\top$	$\top$	$\top$	$\bot$	$\bot$
output	n	ã	õ	t	e

Figure 5: Evaluation of the μX operator.

mula $N \vee (\neg T \wedge \blacklozenge X)$ again with the new truth values for X. This time, $\blacklozenge X$ holds at index 1 and therefore $N \vee (\neg T \wedge \blacklozenge X)$ is true at index 1. The evaluation of X changes, as shown in row 2, with the addition of a $\top$ evaluation for X at index 1. The process repeats again, and now the formula $N \vee (\neg T \wedge \blacklozenge X)$ is true at index 2, and so X becomes true at index 2. In this way, the set of indices where X is true grows with each iteration until reaching a fixed point, where further evaluations of $N \vee (\neg T \wedge \blacklozenge X)$ will not yield any new indices where X holds. The key observation is that the least fixed point of the modal formula in (7) and the least fixed point of the BMRS formula in (6) evaluate to exactly the same thing.

The natural relationship between BMRS and the modal μ-calculus is reminiscent of early work from Chandlee and Jardine (2019), who use a quantifier-free logic equipped with a least fixed point operator to model phonological maps. The modal logic perspective is yet another quantifier-free perspective on the same idea.

Ultimately, this section showed three formal representations of phonological maps. The examples here, though informal, point to an intuitive equivalence between **BMRS** formulas and modal μ-calculus formulas when we restrict our structures to words. Intuitively, we see that references to the successor and predecessor functions can instead be expressed using forward and backward modalities, while recursion in **BMRS** is captured by applications of the fixed point operator.

3 Boolean Monadic Recursive Schemes

To define **BMRS** formally, we fix a single *index variable* x and a countably infinite set $\mathcal{F}$ of unary predicate symbols. **BMRS** terms T and expressions E are defined as in the following recursive

$$\frac{}{\mathbf{w}, i \vdash x \to i} \qquad \frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash \mathrm{p}(T) \to v - 1} \text{ if } v > 0$$

$$\frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash \mathrm{s}(T) \to v + 1} \text{ if } v < |w| - 1$$

Figure 6: Derivation system for denotations of terms. The judgment $\mathbf{w}, i \vdash T \to v$ means that term T denotes index v when evaluated at index i in word $\mathbf{w}$.

grammar, where $f \in \mathcal{F}$.

$$\begin{aligned} T &::= x \mid \mathrm{p}(T) \mid \mathrm{s}(T) \\ E &::= \top \mid \bot \mid \min(T) \mid \max(T) \mid \\ &\quad\; f(T) \mid \textbf{if } E \textbf{ then } E \textbf{ else } E \end{aligned}$$

A **BMRS** *program* P has the form

$$\begin{aligned} f_1(x) &:= E_1 \\ &\vdots \\ f_n(x) &:= E_n, \end{aligned}$$

where $f_1, \ldots, f_n \in \mathcal{F}$ are pairwise distinct and $E_1, \ldots, E_n$ are **BMRS** expressions. We refer to $f_j(x) := E_j$ as a *rule* of the program; a *rule head* for the program is an element of $\{f_1, \ldots, f_n\}$. The *input signature*[4] of P is the set $\mathrm{sig}(P)$ containing all $f \in \mathcal{F}$ which appear in some E_j but which do not appear in $\{f_1, \ldots, f_n\}$.

Terms are *index-valued*; a term T in a word $\mathbf{w}$ at index i *denotes* the index j if the judgment $\mathbf{w}, i \vdash T \to j$ is provable according to the derivation system given in Figure 6. Expressions of a **BMRS** program are *Boolean*-valued. An expression occurring in a program P may be evaluated at an index in a $\mathrm{sig}(P)$-word according to the derivation system in Figure 7. The judgment $\mathbf{w}, i \vdash_P E \to b$ indicates that expression E evaluates to $b \in \{\textbf{true}, \textbf{false}\}$ in $\mathrm{sig}(P)$-word $\mathbf{w}$ at index i under program P.

Note that evaluating an expression depends on the program, while evaluating a term does not. One reason for this is the *unfolding rule* for rule heads f_j, which depends on the rule of the form $f_j := E_j$ in program P (if it exists):

$$\frac{\mathbf{w}, i \vdash T \to v \qquad \mathbf{w}, v \vdash_P E_j \to b}{\mathbf{w}, i \vdash_P f_j(T) \to b}$$

[4] Earlier work on **BMRS** distinguishes explicitly between the input and output signatures; here, we *infer* these from the "free symbols" occurring in the program.

Another reason is the *lookup rule* for elements of $\mathrm{sig}(P)$, which requires that the unary predicate σ being evaluated at a term T is defined in the word:

$$\frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash_P \sigma(T) \to \mathbf{true}} \; (v \in \sigma^{\mathbf{w}})$$

Avoiding confusion over whether the unfolding or lookup rule can be applied to a given unary predicate is why an expression E in program P can *only* be evaluated in a $\mathrm{sig}(P)$-word.

It is worth noting that, by the rules in Figure 6, not every term has a denotation. For example, if i is the largest index in $\mathbf{w}$, then no judgment of the form $\mathbf{w}, i \vdash \mathrm{s}(x) \to j$ is derivable. Similarly, expressions are also not always guaranteed to evaluate; some programs P contain expressions E for which neither $\mathbf{w}, i \vdash_P E \to \mathbf{true}$ nor $\mathbf{w}, i \vdash_P E \to \mathbf{false}$ is derivable according to the rules in Figure 7. As an example, observe that in the program containing only the rules $f(x) := g(x)$ and $g(x) := f(x)$, both $f(x)$ and $g(x)$ do not evaluate at any index in any word. Intuitively, this is because, as f and g are both rule heads (i.e., not in the signature of the program), only the unfolding rule applies. Attempting to construct a proof of some judgment of the form $\mathbf{w}, i \vdash_P f(x) \to b$ leads to an infinite unfolding. In other words, this is a "non-halting" program.

An expression E is *defined* at index i in $\mathbf{w}$ with respect to P if $\mathbf{w}, i \vdash_P E \to b$ for some $b \in \{\mathbf{true}, \mathbf{false}\}$. A **BMRS** program P is *total* on words (resp. strings) if, for every $\mathrm{sig}(P)$-word (resp. string) $\mathbf{w}$, every rule head f of P, and every index $i \in w$, the expression $f(x)$ is defined at i in $\mathbf{w}$ with respect to P.

A **BMRS** *interpretation* is a pair (P, Γ), where Γ is a subset of the rule heads of the program, called the *output signature*. The *input signature* of the interpretation is just the input signature of the program P.

An interpretation (P, Γ) induces a logical transduction $[\![P]\!]_\Gamma : \mathbf{Words}_{\mathrm{sig}(P)} \to \mathbf{Words}_\Gamma$ where $[\![P]\!]_\Gamma(\mathbf{w})$ is the Γ-word with domain w, successor function s^w, and predecessor function p^w, such that

$$\gamma^{[\![P]\!]_\Gamma(\mathbf{w})} := \{i \in w \mid \mathbf{w}, i \vdash_P \gamma(x) \to \mathbf{true}\}$$

for each $\gamma \in \Gamma$. For brevity, we omit the notion of *copy sets* used in (Courcelle, 1994; Courcelle and Engelfriet, 2012; Bhaskar et al., 2020, 2023), which permit interpretations to stretch the length of the input word. However, all results in this paper can be extended to the more general case.

It is shown in (Bhaskar et al., 2023, Theorem 6) that (1) the logical transductions induced by well-defined order-preserving[5] **BMRS** interpretations are rational functions, and (2) every rational function is induced by some well-defined order-preserving **BMRS** interpretation. Fact (1) follows from the fact that order-preserving **BMRS** interpretations can easily be simulated by order-preserving monadic second-order logic (**MSO**) interpretations, which are known to express exactly the rational functions (Filiot, 2015, Theorem 4). Fact (2) is more difficult to show; in (Bhaskar et al., 2023), the authors proceed by solving the *syntactic composition problem* for order-preserving **BMRS** interpretations. The syntactic composition problem is interesting in its own right, but its solution does not yield the simplest proof of fact (2). In Section 5, we show that formulas of the modal μ-calculus express exactly the unary properties expressible by total **BMRS** programs. Since the order-preserving interpretations considered here are determined by their output predicates, this expressive equivalence result constitutes an alternative proof of fact (2).

4 Modal Logic

The *basic modal language* (**ML**) is an extension of propositional logic with *modalities* (Blackburn et al., 2001). In the context of modal logic, unary predicate symbols like those in $\mathcal{F}$ are called *proposition letters* or *variables*. For brevity, rather than describing the formal semantics of modal logic over arbitrary labeled transition systems, as was alluded to in Section 2, we instead interpret modal formulas directly over words. Formulas are evaluated at indices, or equivalently at *pointed words* $(\mathbf{w}, i)$. The semantics of atomic predicates, $\min$, $\max$, conjunction, disjunction, and negation are defined in the natural way, and we set

$$\begin{aligned} \mathbf{w}, i \models \Diamond\varphi \iff & \; s^w(i) \text{ is defined and} \\ & \mathbf{w}, s^w(i) \models \varphi; \\ \mathbf{w}, i \models \blacklozenge\varphi \iff & \; p^w(i) \text{ is defined and} \\ & \mathbf{w}, p^w(i) \models \varphi. \end{aligned}$$

The modal μ-calculus (Kozen, 1983) extends modal logic with fixed point operators, with for-

[5] The interpretations defined in the present paper are also *order-preserving*, as in (Bhaskar et al., 2023).

$$\frac{}{\mathbf{w}, i \vdash_P \top \to \mathbf{true}} \qquad \frac{}{\mathbf{w}, i \vdash_P \bot \to \mathbf{false}} \qquad \frac{\mathbf{w}, i \vdash T \to 0}{\mathbf{w}, i \vdash_P \min(T) \to \mathbf{true}} \qquad \frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash_P \min(T) \to \mathbf{false}}\ (v > 0)$$

$$\frac{\mathbf{w}, i \vdash T \to |w| - 1}{\mathbf{w}, i \vdash_P \max(T) \to \mathbf{true}} \qquad \frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash_P \max(T) \to \mathbf{false}}\ (v < |w| - 1) \qquad \frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash_P \sigma(T) \to \mathbf{true}}\ (v \in \sigma^{\mathbf{w}})$$

$$\frac{\mathbf{w}, i \vdash T \to v}{\mathbf{w}, i \vdash_P \sigma(T) \to \mathbf{false}}\ (v \notin \sigma^{\mathbf{w}}) \qquad \frac{\mathbf{w}, i \vdash T \to v \qquad \mathbf{w}, v \vdash_P E_j \to b}{\mathbf{w}, i \vdash_P f_j(T) \to b}\ (1 \le j \le n)$$

$$\frac{\mathbf{w}, i \vdash_P E \to \mathbf{true} \qquad \mathbf{w}, i \vdash_P E' \to b}{\mathbf{w}, i \vdash_P \mathbf{if}\ E\ \mathbf{then}\ E'\ \mathbf{else}\ E'' \to b} \qquad \frac{\mathbf{w}, i \vdash_P E \to \mathbf{false} \qquad \mathbf{w}, i \vdash_P E'' \to b}{\mathbf{w}, i \vdash_P \mathbf{if}\ E\ \mathbf{then}\ E'\ \mathbf{else}\ E'' \to b}$$

Figure 7: Semantics for **BMRS** expressions over a program P. The judgment $\mathbf{w}, i \vdash_P E \to b$ means expression E evaluates to $b \in \{\mathbf{true}, \mathbf{false}\}$ at index i in word $\mathbf{w}$ under P.

mulas generated by the recursive grammar

$$\varphi := \alpha \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \nabla\varphi \mid \mu X.\varphi,$$

where $\alpha \in \mathcal{F} \cup \{\min, \max, \top, \bot\}$ and $\nabla \in \{\Diamond, \blacklozenge\}$. In formulas of the form $\mu X.\varphi$, we require X to occur only positively in φ, that is, under the scope of an even number of negation symbols. Given a word $\mathbf{w}$ with domain w and a set $U \subseteq w$, let $\mathbf{w}[X \mapsto U]$ be the word obtained from $\mathbf{w}$ by interpreting X as U. The formula $\mu X.\varphi$ induces an operator $F^{\varphi}_{\mathbf{w},X} : \mathcal{P}(w) \to \mathcal{P}(w)$ by

$$F^{\varphi}_{\mathbf{w},X}(U) = \{i \in w \mid \mathbf{w}[X \mapsto U], i \models \varphi\}.$$

By positivity, $F^{\varphi}_{\mathbf{w},X}$ is monotone, and so it has a least fixed point $\mathrm{lfp}(F^{\varphi}_{\mathbf{w},X})$ by the Knaster-Tarski theorem (Knaster, 1928; Tarski, 1955). We define

$$\mathbf{w}, i \models \mu X.\varphi \iff i \in \mathrm{lfp}(F^{\varphi}_{\mathbf{w},X}).$$

We use an equivalent vectorial presentation for our proofs in Section 5. A vectorial formula χ is a finite system of equations

$$X_1 = \theta_1, \quad \ldots, \quad X_n = \theta_n$$

together with a *designated variable* $X_o \in \{X_1, \ldots, X_n\}$. We refer to the X_i as *recursive variables* and assume that they are pairwise distinct. The right-hand side of each equation is generated by the recursive grammar

$$\theta ::= \alpha \mid \theta \wedge \theta \mid \theta \vee \theta \mid \neg\theta \mid \nabla\theta$$

where $\alpha \in \mathcal{F} \cup \{\min, \max, \top, \bot\}$ and $\nabla \in \{\Diamond, \blacklozenge\}$. Elements of $\mathcal{F} \setminus \{X_1, \ldots, X_n\}$ in the right-hand side of some equation in χ are *free*, and the set of free variables of χ is denoted by $\mathrm{sig}(\chi)$. As above, recursive variables must occur only positively in the right-hand sides which define them.

Let $\mathbf{w}$ be a $\mathrm{sig}(\chi)$-word with domain w, and let $\vec{X}$ denote the tuple $X_1, \ldots, X_n$. For tuples $\vec{U} = (U_1, \ldots, U_n) \in (\mathcal{P}(w))^n$, we write $\mathbf{w}[\vec{X} \mapsto \vec{U}]$ for the $(\mathrm{sig}(\chi) \cup \{X_1, \ldots, X_n\})$-word obtained from $\mathbf{w}$ by interpreting each X_j as U_j. The system χ induces a monotone operator $F^{\chi}_{\mathbf{w}} : (\mathcal{P}(w))^n \to (\mathcal{P}(w))^n$ by setting

$$[\![\theta_j]\!]_{\mathbf{w},\vec{U}} = \{i \in w \mid \mathbf{w}[\vec{X} \mapsto \vec{U}], i \models \theta_j\}, \quad \text{and}$$
$$F^{\chi}_{\mathbf{w}}(\vec{U}) = \left([\![\theta_1]\!]_{\mathbf{w},\vec{U}}, \ldots, [\![\theta_n]\!]_{\mathbf{w},\vec{U}}\right),$$

where the satisfaction relation on pointed words in the definition of $[\![\theta_j]\!]_{\mathbf{w},\vec{U}}$ is ordinary modal satisfaction over the expanded word, with $\min$ and $\max$ interpreted in the natural way. Since recursive variables occur only positively in the right-hand sides of equations, $F^{\chi}_{\mathbf{w}}$ is monotone. Thus, by the Knaster-Tarski theorem (Knaster, 1928; Tarski, 1955), it has a least fixed point

$$\mathrm{lfp}(F^{\chi}_{\mathbf{w}}) = (U^*_1, \ldots, U^*_n).$$

Then, for each recursive variable X_j, we write $\mathbf{w}, i \models_{\chi} X_j$ if and only if $i \in U^*_j$. The formula χ itself is satisfied at i if and only if its designated variable is satisfied there:

$$\mathbf{w}, i \models \chi \iff \mathbf{w}, i \models_{\chi} X_o.$$

We write $\mu\mathbf{ML}^{\mathbf{f}}$ for the full vectorial modal μ-calculus. We write $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ for the syntactic fragment of $\mu\mathbf{ML}^{\mathbf{f}}$ which omits negation. Note that, by Bekić's theorem (Bekić, 1984), the ordinary nested syntax for the μ-calculus is equivalent to the vectorial μ-calculus. Thus our choice of syntax is only made for ease of proof, and does not impact our results.

5 Translating μ-calculus to BMRS

In this section, we prove the main containment needed for our equivalence result: every $\mu\mathbf{ML}^{\mathbf{f}}$ formula can be expressed by a **BMRS** program that halts on every input. The proof first works over strings. We establish an automata-theoretic normal form for $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ and then translate the resulting *directed* formulas into total **BMRS** programs. Together with the containment of total **BMRS** predicates in **MSO** and the equivalence of $\mu\mathbf{ML}^{\mathbf{f}}_{+}$, $\mu\mathbf{ML}^{\mathbf{f}}$, and **MSO** on strings (Arenas et al., 2011, Corollary 3.3), this yields the desired equivalence. The passage from equivalence over strings to equivalence over words is obtained by the standard powerset-alphabet reduction, yielding the following main theorem.

Theorem 2. *For every $\mu\mathbf{ML}^{\mathbf{f}}$ formula φ, there exists a total* **BMRS** *program P with output predicate $X_\varphi \in \mathcal{F}$ such that, for all $sig(\varphi)$-words $\boldsymbol{w}$ and indices $i \in w$, we have*

$$\boldsymbol{w}, i \models \varphi \quad \Longleftrightarrow \quad \boldsymbol{w}, i \vdash_P X_\varphi(x) \to \textbf{\textit{true}}.$$

We first prove the string-level containment of $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ in total **BMRS** by passing through a *directed* normal form for $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ formulas, which we now define.

The *dependency graph* of $(\langle X_i = \theta_i \rangle_{i \le n}, X_o)$ has nodes $X_1, \ldots, X_n$ and an edge $X_r \to X_s$ if and only if X_s occurs in θ_r. An SCC C is *backward* if, for every $X_i \in C$, every occurrence in θ_i of a variable from C is under some backward modality and under no forward modality; C is *forward* if the analogous condition holds with forward and backward reversed.

Definition 3 (Directed vectorial formulas)**.** *A vectorial formula is* directed *if every SCC of its dependency graph is backward or forward. We write* **dir-**$\mu\mathbf{ML}^{\mathbf{f}}_{+}$ *for the class of negation-free directed vectorial formulas.*

Lemma 4. *For every $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ formula φ, there exists a* **dir-**$\mu\mathbf{ML}^{\mathbf{f}}_{+}$ *formula ψ_φ with $sig(\varphi) = sig(\psi_\varphi)$ and, for all $sig(\varphi)$-strings $\boldsymbol{w}$ and indices $i \in w$, we have*

$$\boldsymbol{w}, i \models \varphi \quad \textit{if and only if} \quad \boldsymbol{w}, i \models \psi_\varphi.$$

Proof. Let φ be a $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ formula with $\mathrm{sig}(\varphi) = A$. Let $A^\bullet$ be the marked alphabet which contains, for each $a \in A$, both a and a marked copy $a^\bullet$ (Filiot et al., 2019, Sec. 6.1.3 and Ex. 6.10). Given a pointed A-string $(\mathbf{w}, i)$, let $\mathbf{w}^{\bullet i}$ be the corresponding $A^\bullet$-string obtained by marking the i-th symbol. By the equivalence of $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ and **MSO** on strings (Arenas et al., 2011, Corollary 3.3) and the Büchi-Elgot-Trakhtenbrot theorem (Büchi, 1960), there is a DFA $\mathcal{A} = (Q, q_0, \delta, F)$ over $A^\bullet$ such that $\mathbf{w}, i \models \varphi$ if and only if $\widehat{\delta}(q_0, \mathbf{w}^{\bullet i}) \in F$, where $\widehat{\delta}$ denotes the usual extension of δ to strings.

Using the transition function of $\mathcal{A}$, form a vectorial formula ψ_φ with variables L_q and R_q for each $q \in Q$, and designated variable X_φ. The L_q equations use only the backward modality and record the state of $\mathcal{A}$ before the current position; the R_q equations use only the forward modality and record whether the suffix beginning at the current position is accepted from state q; and the equation for X_φ combines these summaries with the current marked symbol. We omit the routine finite disjunctions spelling out these equations from δ. By construction, $\widehat{\delta}(q_0, \mathbf{w}^{\bullet i}) \in F$ if and only if $\mathbf{w}, i \models \psi_\varphi$. Thus we conclude that $\mathbf{w}, i \models \psi_\varphi$ if and only if $\mathbf{w}, i \models \varphi$.

Clearly ψ_φ is negation-free and $\mathrm{sig}(\psi_\varphi) = A$. Every SCC is contained among the L_q's, among the R_q's, or is $\{X_\varphi\}$; these are backward, forward, and vacuous, respectively. Hence $\psi_\varphi \in$ **dir-**$\mu\mathbf{ML}^{\mathbf{f}}_{+}$. □

Definition 5 (Agreement)**.** *Let P be a* **BMRS** *program, let E be a* **BMRS** *expression, let φ be a $\mu\mathbf{ML}^{\mathbf{f}}$ formula, let $\boldsymbol{w}$ be a $sig(\varphi)$-word, and let $i \in w$. We say that E* agrees with *φ at i in $\boldsymbol{w}$ if E is defined at i in $\boldsymbol{w}$ with respect to P, and $\boldsymbol{w}, i \models \varphi$ if and only if $\boldsymbol{w}, i \vdash_P E \to \textbf{\textit{true}}$.*

Definition 6. *Given a formula $\chi \in$* **dir-**$\mu\mathbf{ML}^{\mathbf{f}}_{+}$ *of the form $(\langle X_i = \theta_i \rangle_{i \le n}, X_o)$, for each right-hand side formula φ, define a* **BMRS** *expression $\mathrm{tr}(\varphi)$ with free variable x by*

$$\begin{aligned}
\mathrm{tr}(b) &:= b, \qquad (b \in \{\top, \bot\}),\\
\mathrm{tr}(X) &:= X(x), \qquad (X \in \mathcal{F}),\\
\mathrm{tr}(\min) &:= \min(x), \qquad \mathrm{tr}(\max) := \max(x),\\
\mathrm{tr}(\varphi \wedge \psi) &:= \textbf{\textit{if}}\ \mathrm{tr}(\varphi)\ \textbf{\textit{then}}\ \mathrm{tr}(\psi)\ \textbf{\textit{else}}\ \bot,\\
\mathrm{tr}(\varphi \vee \psi) &:= \textbf{\textit{if}}\ \mathrm{tr}(\varphi)\ \textbf{\textit{then}}\ \top\ \textbf{\textit{else}}\ \mathrm{tr}(\psi),\\
\mathrm{tr}(\Diamond\varphi) &:= \textbf{\textit{if}}\ \max(x)\ \textbf{\textit{then}}\ \bot\ \textbf{\textit{else}}\ \mathrm{tr}(\varphi)[s(x)],\\
\mathrm{tr}(\blacklozenge\varphi) &:= \textbf{\textit{if}}\ \min(x)\ \textbf{\textit{then}}\ \bot\ \textbf{\textit{else}}\ \mathrm{tr}(\varphi)[p(x)].
\end{aligned}$$

Here, $E[T]$ denotes substitution of the **BMRS** *term T for the free variable x in E. The translated* **BMRS** *program P_χ has output predicate X_o and contains the rule $X_j(x) := \mathrm{tr}(\theta_j)$ for*

each equation $X_j = \theta_j$. Thus the rule heads of P_χ are exactly the recursive variables of χ and $sig(P_\chi) = sig(\chi)$.

Remark 7 (Compositionality of the translation)**.** *The following fact is immediate by induction on φ. Let χ be as above, let P_χ be its* **BMRS** *translation, let* ***w*** *be a $sig(\chi)$-string, and let $i \in w$. Suppose the following atomic correspondence for $\mathrm{tr}(\varphi)$ holds: whenever $Y(T)$ occurs in $\mathrm{tr}(\varphi)$, where Y is a rule head of P_χ and $\boldsymbol{w}, i \vdash T \to h$, then the expression $Y(T)$ is defined at i in* ***w*** *with respect to P_χ, and*

$$\boldsymbol{w}, i \vdash_{P_\chi} Y(T) \to \textbf{\textit{true}} \quad \iff \quad \boldsymbol{w}, h \models_\chi Y.$$

Then $\mathrm{tr}(\varphi)$ is defined at i in ***w*** *with respect to P_χ, and agrees with φ at i in* ***w****.*

Theorem 8. *For every $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ formula φ, there exists a* **BMRS** *program P that is total on strings and has output predicate $X_\varphi \in \mathcal{F}$ such that, for all $sig(\varphi)$-strings* ***w*** *and indices $i \in w$, we have*

$$\boldsymbol{w}, i \models \varphi \quad \iff \quad \boldsymbol{w}, i \vdash_P X_\varphi(x) \to \textbf{\textit{true}}.$$

Proof. Let $\varphi \in \mu\mathbf{ML}^{\mathbf{f}}_{+}$, and choose by Lemma 4 a formula $\chi = (\langle X_i = \theta_i \rangle_{i \le n}, X_o) \in$ **dir-**$\mu\mathbf{ML}^{\mathbf{f}}_{+}$ which is equivalent to φ on strings. Let P_χ be its **BMRS** translation. Let $C_1, \ldots, C_k$ be the SCCs of the dependency graph of χ, topologically ordered so that if $X \in C_j$ and a recursive variable Y occurs in θ_X, then $Y \in C_\ell$ for some $\ell \le j$.

We prove by strong induction on j that every rule head in $C_1 \cup \cdots \cup C_j$ is defined and agrees with the corresponding recursive variable of χ at every index of every $\mathrm{sig}(\chi)$-string. Fix C_j, and assume inductively that

$$\begin{aligned} &\text{for every } \ell < j, \text{ every } Y \in C_\ell, \\ &\text{every } \mathrm{sig}(\chi)\text{-string } \mathbf{w}, \text{ and every index} \\ &h \in w,\ Y(x) \text{ agrees with } Y \text{ at } h \text{ in } \mathbf{w}. \end{aligned} \tag{IH1}$$

Assume that C_j is backward. Fix a $\mathrm{sig}(\chi)$-string $\mathbf{w}$, and proceed by induction on indices i from left to right. Fix an index i, and assume inductively that

$$\begin{aligned} &\text{for every } m < i \text{ and every } Y \in C_j,\ Y(x) \\ &\text{agrees with } Y \text{ at } m \text{ in } \mathbf{w}. \end{aligned} \tag{IH2}$$

Now fix $X \in C_j$, and write $X = \theta_X$ for its equation in χ. We verify the atomic correspondence hypothesis of Remark 7 for $\mathrm{tr}(\theta_X)$ at i. Let $Y(T)$ be an occurrence in $\mathrm{tr}(\theta_X)$, where Y is a rule head of P_χ, and suppose $\mathbf{w}, i \vdash T \to h$. If $Y \in C_j$, then by directionality of C_j and the definition of tr, $T = \mathrm{p}^r(x)$ for some $r > 0$, so $h < i$. Hence (IH2) gives that $Y(x)$ agrees with Y at h in $\mathbf{w}$. Thus $Y(T)$ is defined at i and $\mathbf{w}, i \vdash_{P_\chi} Y(T) \to$ **true** if and only if $\mathbf{w}, h \models_\chi Y$. If $Y \notin C_j$, then by the ordering of the SCCs we have $Y \in C_\ell$ for some $\ell < j$, and the same conclusion follows from (IH1). Thus the atomic correspondence hypothesis of Remark 7 holds, and therefore $\mathrm{tr}(\theta_X)$ is defined at i in $\mathbf{w}$ with respect to P_χ, and agrees with θ_X at i in $\mathbf{w}$.

Since P_χ contains the rule $X(x) := \mathrm{tr}(\theta_X)$, and $X = \theta_X$ is an equation of χ, it follows that $X(x)$ agrees with X at i in $\mathbf{w}$. This closes the induction on indices for the backward component C_j. The case where C_j is forward is analogous, using induction on $|w| - i + 1$. Thus the induction on j gives totality and agreement between all recursive variables of χ and the corresponding rule head of P_χ at all indices of $\mathbf{w}$. In particular, since χ has designated variable X_o, for every $\mathrm{sig}(\chi)$-string $\mathbf{w}$ and every index $i \in w$, $X_o(x)$ agrees with χ at i in $\mathbf{w}$. Taking $P = P_\chi$ and renaming X_o as X_φ, and using equivalence of χ and φ on strings, we have $\mathbf{w}, i \models \varphi \iff \mathbf{w}, i \vdash_P X_\varphi(x) \to$ **true**. □

Proof of Theorem 2. With respect to the class of strings, Theorem 8 shows that every $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ formula can be expressed by a total **BMRS** program. Conversely, a straightforward encoding shows that every total **BMRS** predicate over strings is definable in **MSO**. By (Arenas et al., 2011, Corollary 3.3), **MSO**, $\mu\mathbf{ML}^{\mathbf{f}}$, and $\mu\mathbf{ML}^{\mathbf{f}}_{+}$ are expressively equivalent over strings. Hence $\mu\mathbf{ML}^{\mathbf{f}}$ and total **BMRS** are expressively equivalent over strings. The passage from strings to words is by the standard powerset-alphabet reduction, whose details we omit. □

6 Concluding Remarks

We have shown that total Boolean Monadic Recursive Schemes are equivalent to the modal μ-calculus on words, yielding an alternative proof that order-preserving **BMRS** interpretations capture the rational functions. Because the learning and model-theoretic properties of modal logic are well studied, this connection may benefit phonology; conversely, the programming-language perspective of **BMRS** may also inform modal logic and its applications.

References

Marcelo Arenas, Pablo Barceló, and Leonid Libkin. 2011. Regular languages of nested words: fixed points, automata, and synchronization. *Theory of Computing Systems*, 49(3):639–670.

Hans Bekić. 1984. Definable operations in general algebras, and the theory of automata and flowcharts. In C. B. Jones, editor, *Programming Languages and Their Definition: H. Bekič (1936–1982)*, pages 30–55. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jean Berstel. 2007. *Transductions and Context-Free Languages*. Vieweg+Teubner Verlag Wiesbaden.

Siddharth Bhaskar, Jane Chandlee, and Adam Jardine. 2023. Rational functions via recursive schemes. *Preprint*, arXiv:2302.03074.

Siddharth Bhaskar, Jane Chandlee, Adam Jardine, and Christopher Oakden. 2020. Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In *International conference on language and automata theory and applications*, pages 157–169. Springer.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. 2001. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Julian Bradfield and Colin Stirling. 2006. *Modal Mu-Calculi*, chapter 12. Elsevier.

J. Richard Büchi. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92.

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

Jane Chandlee and Adam Jardine. 2019. Quantifier-free least fixed point functions for phonology. In *Proceedings of the 16th Meeting on the Mathematics of Language (MOL)*, pages 50–62. Association for Computational Linguistics.

Jane Chandlee and Adam Jardine. 2021. Computational universals in linguistic theory: Using recursive programs for phonological analysis. *Language*, 97(3):485–519.

Bruno Courcelle. 1994. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126(1):53–75.

Bruno Courcelle and Joost Engelfriet. 2012. Monadic second-order transductions. In *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, chapter 7, pages 505–577. Cambridge University Press.

E. A. Emerson and C. S. Jutla. 1991. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, page 368–377, USA. IEEE Computer Society.

Emmanuel Filiot. 2015. Logic-automata connections for transformations. In *Indian conference on logic and its applications*, pages 30–57. Springer.

Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. 2019. Logical and algebraic characterizations of rational transductions. *Logical Methods in Computer Science*, Volume 15, Issue 4:16.

Thomas Graf. 2010. Logics of phonological reasoning. Master's thesis, University of California, Los Angeles.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry M. Hyman and Frans Plank, editors, *Phonological Typology*, volume 23 of *Phonology and Phonetics*, pages 126–195. De Gruyter Mouton, Berlin, Boston.

Adam Jardine, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Very efficient learning of structured classes of subsequential functions from positive data. In *The 12th International Conference on Grammatical Inference*, volume 34, pages 94–108.

Douglas Johnson. 1972. *Formal aspects of phonological description*. De Gruyter.

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

Bronisław Knaster. 1928. Un théorème sur les fonctions d'ensembles. *Annales de la Société Polonaise de Mathématique*, 6:133–134.

Raoul Koudijs. 2022. Learning modal formulas via dualities. Master's thesis, Universiteit van Amsterdam.

Dexter Kozen. 1983. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982.

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

Henry A. Osborn. 1966. Warao i: Phonology and morphophonemics. *International Journal of American Linguistics*, 32(2).

Alfred Tarski. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309.

Balder ten Cate and Raoul Koudijs. 2024. Characterising modal formulas with examples. *ACM Transactions on Computational Logic*, 25(2).

Thomas Wilke. 2001. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of The Belgian Mathematical Society*, 8:359–391.

Tatevik Yolyan. 2025a. A framework for learning phonological maps as logical transductions. In *Proceedings of the 18th Meeting on the Mathematics of Language*, page 44–58. Association for Computational Linguistics.

Tatevik Yolyan. 2025b. A logical characterization of weak determinism as simultaneous application. *Journal of Logic, Language and Information*.

Tatevik Yolyan. 2025c. *Phonological Expressivity and Learning via Boolean Monadic Recursive Schemes*. Ph.D. thesis, Rutgers The State University of New Jersey, School of Graduate Studies.