

Search and Copy/Change with Boolean Monadic Recursive Schemes AMP 2026

Scott Nelson & Matthew King

University of Illinois Urbana-Champaign

October 9, 2026



Talk Overview

We propose here boolean monadic recursive schemes (BMRSs) and demonstrate their utility for phonological analysis.

As we will show and discuss, this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance.

...Thus, the BMRS formalism enables a theory of phonology that captures both intensional and computational generalizations about phonology in a way that is unavailable to previous frameworks.

Computational universals in linguistic theory: Using recursive programs for phonological analysis (Chandlee and Jardine, 2021)

Talk Overview

We propose here boolean monadic recursive schemes (BMRSs) and demonstrate their utility for phonological analysis.

As we will show and discuss, this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance.

...Thus, the BMRS formalism enables a theory of phonology that captures both intensional and computational generalizations about phonology in a way that is unavailable to previous frameworks.

Computational universals in linguistic theory: Using recursive programs for phonological analysis (Chandlee and Jardine, 2021)

Talk Overview

We propose here boolean monadic recursive schemes (BMRSs) and demonstrate their utility for phonological analysis.

As we will show and discuss, this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance.

...Thus, the BMRS formalism enables a theory of phonology that captures both intensional and computational generalizations about phonology in a way that is unavailable to previous frameworks.

Computational universals in linguistic theory: Using recursive programs for phonological analysis (Chandlee and Jardine, 2021)

Talk Overview

We propose here boolean monadic recursive schemes (BMRSs) and demonstrate their utility for phonological analysis.

As we will show and discuss, this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance.

...Thus, the BMRS formalism enables a theory of phonology that captures both intensional and computational generalizations about phonology in a way that is unavailable to previous frameworks.

Computational universals in linguistic theory: Using recursive programs for phonological analysis (Chandlee and Jardine, 2021)

Talk Overview

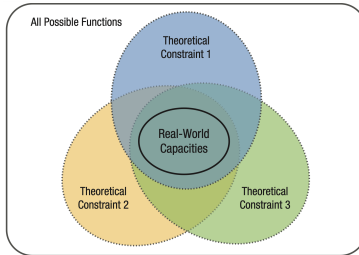
We propose here boolean monadic recursive schemes (BMRSs) and demonstrate their utility for phonological analysis.

As we will show and discuss, this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance.

*...Thus, the BMRS formalism enables a theory of phonology that captures both **intensional** and computational generalizations about phonology in a way that is unavailable to previous frameworks.*

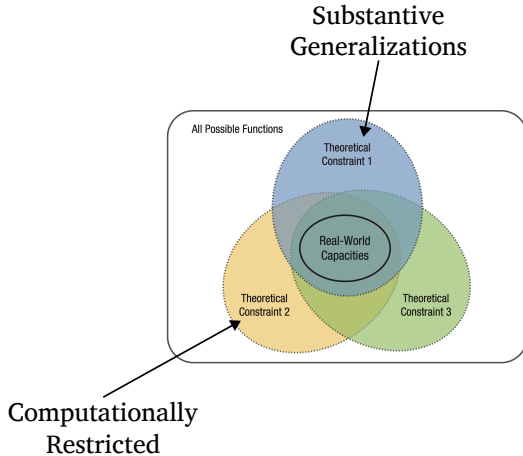
Computational universals in linguistic theory: Using recursive programs for phonological analysis (Chandlee and Jardine, 2021)

Talk Overview



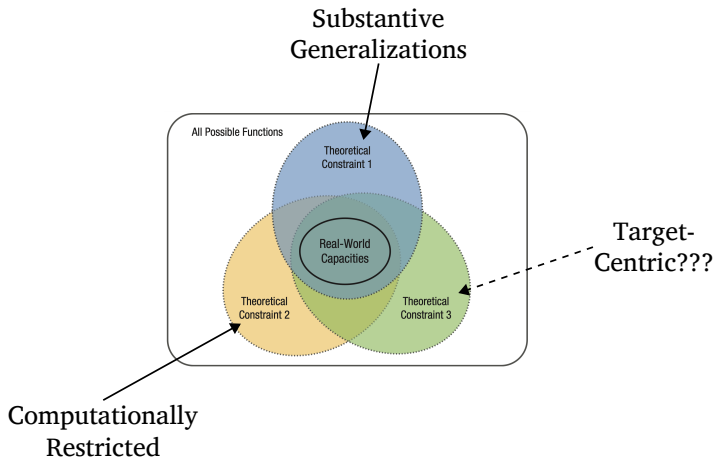
van Rooij and Baggio (2021)

Talk Overview



Chandlee and Jardine (2021)

Talk Overview



Nevins (2005, 2010); Mailhot and Reiss (2007); Samuels (2009); Dabbous et al. (2021); Burness et al. (2024)

Background

How to determine the [back] value of the final vowel?

p	u	l	ʁ	A	→	p	u	l	ʁ	ɑ
C	V	C	C	V		C	V	C	C	V
	[+back]			[+/-back]			[+back]			[+back]
t	y	r	d	A	→	t	y	r	d	æ
C	V	C	C	V		C	V	C	C	V
	[-back]			[+/-back]			[-back]			[-back]

[pul-ʁɑ] ‘money-DAT’, [tyr-dæ] ‘type-LOC’

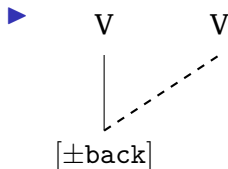
Uyghur; Mayer (2021, p. 23)

Traditional Formal Analyses

► Rules:

$$\text{► } [+syl] \rightarrow [\alpha back] / \left[\begin{array}{c} +syl \\ \alpha back \end{array} \right] C_0 _$$

► Autosegmental Spreading:



► OT Constraints:

► Agree $\gg$ Ident(back)

Source-centric vs. Target-centric

- ▶ Does the trigger “give” its value to the target?
 - ▶ Autosegmental spreading
- ▶ Or does the target “take” its value from the trigger?
 - ▶ Search Copy/Change
 - ▶ BMRS

Nevins (2005, 2010); Mailhot and Reiss (2007); Samuels (2009); Dabbous et al. (2021); Burness et al. (2024)

Source-centric vs. Target-centric

- ▶ Does the trigger “give” its value to the target?
 - ▶ Autosegmental spreading
- ▶ Or does the target “take” its value from the trigger?
 - ▶ Search Copy/Change
 - ▶ BMRS

Burness et al. (2024, p. 271)

“**What if**, instead of harmony being driven by an imperative for fully specified vowels to propagate their values onto unspecified values, **harmony was driven by an imperative for ... vowels to seek out values in order to become fully specified?**”

Nevins (2005, 2010); Mailhot and Reiss (2007); Samuels (2009); Dabbous et al. (2021); Burness et al. (2024)

- ▶ Vowel (dis)harmony is a search procedure such that morphemes which undergo harmony are lexically underspecified for the harmonizing feature and need to search for a donor to provide a value.

Nevins (2005, 2010); Burness et al. (2024)

```
1: myVals =  $V$ 
2: myPosition =  $P$ 
3: myFeatsneeded =  $F$ 
4: myConditionalRequirements( $F$ ) =  $R$ 
5: Direction =  $\delta$ 
6: while  $F$  is not empty do
7:   Go in direction  $\delta$  and update  $P$ 
8:   if  $R$  is true of  $P$  then:
9:     Copy  $Val(P, f)$  to  $V$ 
10:    Remove  $f$  from  $F$ 
11:   else:
12:     exit
```

Nevins (2010)

Search & Copy

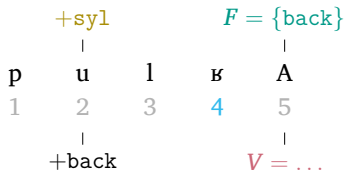
```
1: myVals = V = ...
2: myPosition = P = 5
3: myFeatsneeded = F = {back}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P,f) to V
10:    Remove f from F
11:   else:
12:     exit
```

	+syl			F = {back}
p	u	l	ɾ	A
1	2	3	4	5
	+back			V = ...

Nevins (2010)

Search & Copy

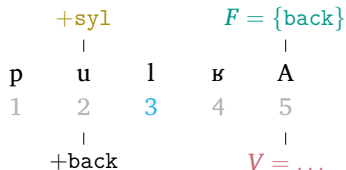
```
1: myVals = V = ...
2: myPosition = P = 4
3: myFeatsneeded = F = {back}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P,f) to V
10:    Remove f from F
11:   else:
12:     exit
```



Nevins (2010)

Search & Copy

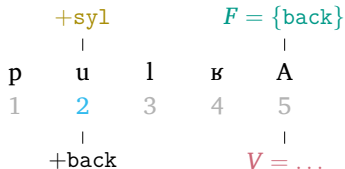
```
1: myVals = V = ...
2: myPosition = P = 3
3: myFeatsneeded = F = {back}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P,f) to V
10:    Remove f from F
11:   else:
12:     exit
```



Nevins (2010)

Search & Copy

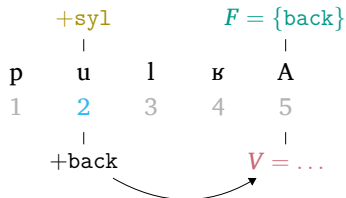
```
1: myVals = V = ...
2: myPosition = P = 2
3: myFeatsneeded = F = {back}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P,f) to V
10:    Remove f from F
11:   else:
12:     exit
```



Nevins (2010)

Search & Copy

```
1: myVals = V = ...
2: myPosition = P = 2
3: myFeatsneeded = F = {back}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P,f) to V
10:    Remove f from F
11:   else:
12:     exit
```



Nevins (2010)

Search & Copy

```
1: myVals = V = ..., +back
2: myPosition = P = ...
3: myFeatsneeded = F = {}
4: myConditionalRequirements(F) = R = +syl
5: Direction =  $\delta$  = LEFT
6: while F is not empty do
7:   Go in direction  $\delta$  and update P
8:   if R is true of P then:
9:     Copy Val(P, f) to V
10:    Remove f from F
11:   else:
12:     exit
```

	+syl			F = {}
p	u	l	ɸ	ɑ
1	2	3	4	5
	+back			V = ..., +back

Nevins (2010)

Search & Change

- ▶ The copy aspect has been changed to more generally have a trigger induce any general type of *change* on the target.
- ▶ Phonological rules are parameterized search procedures that require targets to search for triggers (which may be arbitrarily far away) and then undergo some type of change if and only if the trigger has some additional set of properties.

Mailhot and Reiss (2007); Dabbous et al. (2021)

- ▶ SEARCH
 - ▶ INR:
 - ▶ TRM:
 - ▶ DIR:
- ▶ CHANGE
 - ▶ INPUT:
 - ▶ OUTPUT:
 - ▶ CONDITION:

Dabbous et al. (2021)

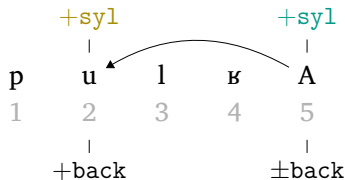
Search & Change

► SEARCH

- INR: +syl
- TRM: +syl
- DIR: LEFT

► CHANGE

- INPUT: INR
- OUTPUT: back(TRM)
- CONDITION: n/a



Dabbous et al. (2021)

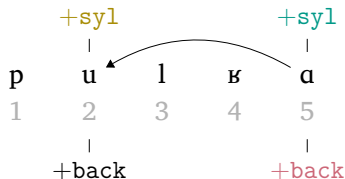
Search & Change

► SEARCH

- INR: +syl
- TRM: +syl
- DIR: LEFT

► CHANGE

- INPUT: INR
- OUTPUT: back(TRM)
- CONDITION: n/a



Dabbous et al. (2021)

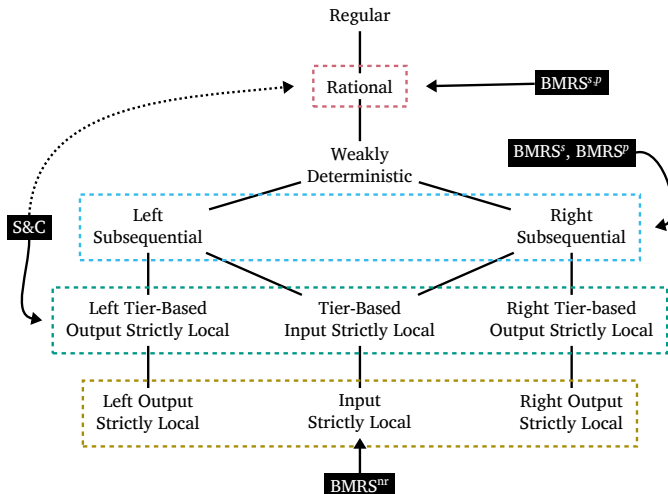
Boolean Monadic Recursive Schemes

$\phi_{\text{back}}(\mathbf{x}) :=$ IF $\text{init}(\mathbf{x})$
 THEN $\text{back}(\mathbf{x})$
 ELSE $\phi_{\text{back}}(\mathbf{p}(\mathbf{x}))$
 $\phi_f(\mathbf{x}) := f(\mathbf{x}) \mid f \in \mathcal{F} \setminus \text{back}$

	1		2	
<i>Input:</i>	$\mathbf{p}$	$\mathbf{u}_{+b}$	$\mathbf{l}$	$\mathbf{A}_{\pm b}$
$\text{back}(\mathbf{x})$		$\top$		$\top / \perp$
$\text{syl}(\mathbf{x})$		$\top$		$\top$
$\text{init}(\mathbf{x})$		$\top$		$\perp$
$\phi_{\text{back}}(\mathbf{x})$		$\top$		$\top$
$\phi_{\text{syl}}(\mathbf{x})$		$\top$		$\top$
<i>Output:</i>	$\mathbf{p}$	$\mathbf{u}_{+b}$	$\mathbf{l}$	$\mathbf{a}_{+b}$

Bhaskar et al. (2020); Chandlee and Jardine (2021); Bhaskar et al. (2023)
 See also Nelson and Baković (2026); Nelson and King (2026)

Subregular Function Classes

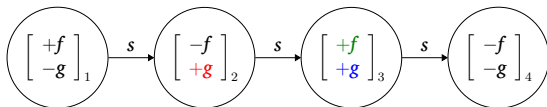


Andersson et al. (2020); Bhaskar et al. (2020, 2023); Burness et al. (2024)

Boolean Monadic Recursive Schemes

BMRS Overview

- ▶ BMRS define string-to-string functions as subregular logical transductions and correspond with different classes of subregular functions.
- ▶ A string is a structure $\mathbf{s}$ over a model signature $\mathbf{S} = \langle f \mid f \in \mathcal{F}, p(x), s(x) \rangle$
 - ▶ f are unary **labeling** relations drawn from a set $\mathcal{F}$
 - ▶ $s(x)$ and $p(x)$ are successor and predecessor **ordering** functions.



Bhaskar et al. (2020); Chandlee and Jardine (2021); Bhaskar et al. (2023)

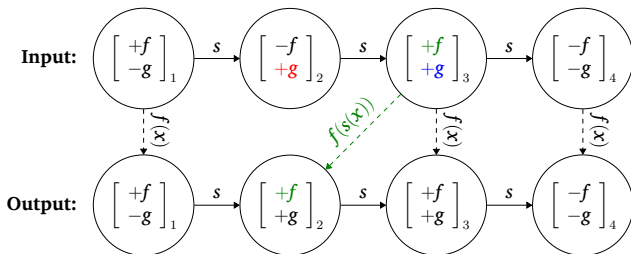
BMRS Overview

- ▶ A BMRS program ρ is a collection of recursive functions $f_1, \dots, f_n$ where each function has output type boolean and a single argument (monadic).
- ▶ BMRS functions are built using IF ... THEN ... ELSE ... syntax.
- ▶ A subset of these functions are labeled as $\phi_f(x)$ and indicate that domain element x carries property f in the output.
- ▶ Following a suggestion by Chandlee and Jardine (2021), we loosen the boolean restriction and extend these functions to have type $\text{index} \rightarrow \{0, -, +\}$.

Bhaskar et al. (2020); Chandlee and Jardine (2021); Bhaskar et al. (2023)

BMRS Overview

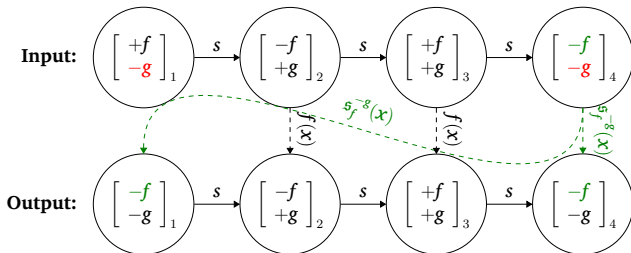
- ▶ A local function contains no recursive calls.
- ▶ Assimilation of feature f between local $+g$ segments:
 - ▶ $\phi_f(x) := \text{IF } \mathbf{g(x)} \approx + \wedge \mathbf{g(s(x))} \approx + \text{ THEN } \mathbf{f(s(x))} \text{ ELSE } \mathbf{f(x)}$



Bhaskar et al. (2020); Chandlee and Jardine (2021); Bhaskar et al. (2023)

BMRS Overview

- ▶ A non-local function uses recursive calls.
- ▶ Non-local harmony for feature f between $-g$ segments:
 - ▶ $\phi_f(x) := \text{IF } g(x) \approx - \text{ THEN } \mathfrak{s}_f^{-g}(x) \text{ ELSE } f(x)$
 - ▶ $\mathfrak{s}_f^{-g}(x) := \text{IF edge}(x) \text{ THEN } f(x) \text{ ELSE IF } g(s(x)) \approx - \text{ THEN } f(s(x)) \text{ ELSE } \mathfrak{s}_f^{-g}(s(x))$



Search & Copy/Change as BMRS Programs

Nevins style S&C

$$\phi_{f_i}(\mathbf{x}) := \text{IF } \iota_i(\mathbf{x}) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x})$$
$$\iota_i(\mathbf{x}) := \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp$$
$$\mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) := \text{IF edge}(\mathbf{x}) \text{ THEN def ELSE}$$
$$\text{IF } \psi(\delta(\mathbf{x})) \text{ THEN}$$
$$\text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE def}$$
$$\text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(\mathbf{x}))$$

- The primary predicate is $\phi_{f_i}(\mathbf{x})$ which checks if an initiator is seen, runs the search procedure ($\mathfrak{s}_{f_i}^{\iota}$) if so, and makes no change if not.

Nevins style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } s_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN def} \\ &\quad \text{IF } \psi(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE def} \\ &\quad \text{ELSE } s_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- An element being an initiator is determined by the second predicate ($\iota_i(\mathbf{x})$) which states that an element initiates search for the feature f_i if it is underspecified for that feature.

Nevins style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } s_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN } \text{def} \text{ ELSE} \\ &\quad \text{IF } \psi(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE } \text{def} \\ &\quad \text{ELSE } s_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- If an element is on an edge, it has nowhere to search and therefore automatically returns a default value.

Nevins style S&C

$$\begin{aligned}\phi_{f_i}(x) &:= \text{IF } \iota_i(x) \text{ THEN } s_{f_i}^t(x) \text{ ELSE } f_i(x) \\ \iota_i(x) &:= \text{IF } f_i(x) \approx 0 \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^t(x) &:= \text{IF edge}(x) \text{ THEN def} \\ &\quad \text{IF } \psi(\delta(x)) \text{ THEN} \\ &\quad \quad \text{IF } R_i(\delta(x)) \text{ THEN } f_i(\delta(x)) \text{ ELSE def} \\ &\quad \text{ELSE } s_{f_i}^t(\delta(x))\end{aligned}$$

- Otherwise, the functions looks to the adjacent element in the string. If the adjacent element satisfies the termination requirement established by ψ , the predicate moves forward with checking if that character has the required features for copying, R_i .

Nevins style S&C

$$\phi_{f_i}(\mathbf{x}) := \text{IF } \iota_i(\mathbf{x}) \text{ THEN } s_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x})$$
$$\iota_i(\mathbf{x}) := \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp$$
$$s_{f_i}^{\iota}(\mathbf{x}) := \text{IF edge}(\mathbf{x}) \text{ THEN def ELSE}$$
$$\text{IF } \psi(\delta(\mathbf{x})) \text{ THEN}$$
$$\text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE def}$$
$$\text{ELSE } s_{f_i}^{\iota}(\delta(\mathbf{x}))$$

- If so, f_i is copied over to the initiating element, otherwise it takes the default value for f_i .

Nevins style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } s_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN } \text{def} \text{ ELSE} \\ &\quad \text{IF } \psi(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE } \text{def} \\ &\quad \text{ELSE } s_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- If the adjacent element is not a terminator, the predicate recurses until a terminator is found or the procedure reaches the end of the string.

Nevins style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } s_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } f_i(\mathbf{x}) \approx 0 \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN def} \\ &\quad \text{IF } \psi(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } R_i(\delta(\mathbf{x})) \text{ THEN } f_i(\delta(\mathbf{x})) \text{ ELSE def} \\ &\quad \text{ELSE } s_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- In these functions, δ stands for the successor or predecessor functions which determine the direction of recursion/“search”. edge corresponds to initial or final segments depending on the direction.

Concordia style S&C

$$\begin{aligned}\phi_{f_i}(x) &:= \text{IF } \iota_i(x) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(x) \text{ ELSE } f_i(x) \\ \iota_i(x) &:= \text{IF } P(x) \text{ THEN } \top \text{ ELSE } \perp \\ \mathfrak{s}_{f_i}^{\iota}(x) &:= \text{IF edge}(x) \text{ THEN } f_i(x) \text{ ELSE} \\ &\quad \text{IF term}(\delta(x)) \text{ THEN} \\ &\quad \quad \text{IF cond}(\delta(x)) \text{ THEN } v \text{ ELSE } f_i(x) \\ &\quad \text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(x))\end{aligned}$$

- The Concordia School version of Search and Change can be captured with BMRS in a similar manner.

Concordia style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } P(\mathbf{x}) \text{ THEN } \top \text{ ELSE } \perp \\ \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN } f_i(\mathbf{x}) \text{ ELSE} \\ &\quad \text{IF } \text{term}(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } \text{cond}(\delta(\mathbf{x})) \text{ THEN } \mathbf{v} \text{ ELSE } f_i(\mathbf{x}) \\ &\quad \text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- Importantly, the initiator elements for this procedure do not need to be underspecified.

Concordia style S&C

$$\begin{aligned}\phi_{f_i}(x) &:= \text{IF } \iota_i(x) \text{ THEN } s_{f_i}^{\iota}(x) \text{ ELSE } f_i(x) \\ \iota_i(x) &:= \text{IF } P(x) \text{ THEN } \top \text{ ELSE } \perp \\ s_{f_i}^{\iota}(x) &:= \text{IF edge}(x) \text{ THEN } f_i(x) \text{ ELSE} \\ &\quad \text{IF term}(\delta(x)) \text{ THEN} \\ &\quad \quad \text{IF cond}(\delta(x)) \text{ THEN } v \text{ ELSE } f_i(x) \\ &\quad \text{ELSE } s_{f_i}^{\iota}(\delta(x))\end{aligned}$$

- Instead, the initiators are defined with a set of features on a domain element ($P(x)$), which trigger a search in direction (δ) for a terminating element (term) which has further conditions (cond) similarly to Nevins.

$$\begin{aligned}\phi_{f_i}(x) &:= \text{IF } \iota_i(x) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(x) \text{ ELSE } f_i(x) \\ \iota_i(x) &:= \text{IF } P(x) \text{ THEN } \top \text{ ELSE } \perp \\ \mathfrak{s}_{f_i}^{\iota}(x) &:= \text{IF edge}(x) \text{ THEN } f_i(x) \text{ ELSE} \\ &\quad \text{IF } \text{term}(\delta(x)) \text{ THEN} \\ &\quad \quad \text{IF } \text{cond}(\delta(x)) \text{ THEN } v \text{ ELSE } f_i(x) \\ &\quad \text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(x))\end{aligned}$$

- If these conditions are met, the initiating element takes a feature value for f according to a change procedure.

Concordia style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } P(\mathbf{x}) \text{ THEN } \top \text{ ELSE } \perp \\ \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN } f_i(\mathbf{x}) \text{ ELSE} \\ &\quad \text{IF } \text{term}(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } \text{cond}(\delta(\mathbf{x})) \text{ THEN } \nu \text{ ELSE } f_i(\mathbf{x}) \\ &\quad \text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- Unlike the Nevins version, there is no requirement that the change be related directly to the terminator. This can be seen on line 5 in blue where we represent change as ν which stands for a feature $\nu \in \{+, -, 0\}$.

Concordia style S&C

$$\begin{aligned}\phi_{f_i}(\mathbf{x}) &:= \text{IF } \iota_i(\mathbf{x}) \text{ THEN } \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) \text{ ELSE } f_i(\mathbf{x}) \\ \iota_i(\mathbf{x}) &:= \text{IF } P(\mathbf{x}) \text{ THEN } \top \text{ ELSE } \perp \\ \mathfrak{s}_{f_i}^{\iota}(\mathbf{x}) &:= \text{IF edge}(\mathbf{x}) \text{ THEN } f_i(\mathbf{x}) \text{ ELSE} \\ &\quad \text{IF } \text{term}(\delta(\mathbf{x})) \text{ THEN} \\ &\quad \quad \text{IF } \text{cond}(\delta(\mathbf{x})) \text{ THEN } \mathbf{v} \text{ ELSE } f_i(\mathbf{x}) \\ &\quad \text{ELSE } \mathfrak{s}_{f_i}^{\iota}(\delta(\mathbf{x}))\end{aligned}$$

- The other major difference between the two approaches is what happens when a search fails.

BMRS Implementations Side-by-Side

$$\phi_{f_i}(x) := \text{IF } \iota_i(x) \text{ THEN } s_{f_i}^L(x) \text{ ELSE } f_i(x)$$
$$\iota_i(x) := \text{IF } f_i(x) \approx 0 \text{ THEN } \top \text{ ELSE } \perp$$
$$s_{f_i}^L(x) := \text{IF edge}(x) \text{ THEN } \text{def} \text{ ELSE}$$
$$\text{IF } \psi(\delta(x)) \text{ THEN}$$
$$\text{IF } R_i(\delta(x)) \text{ THEN } f_i(\delta(x)) \text{ ELSE } \text{def}$$
$$\text{ELSE } s_{f_i}^L(\delta(x))$$
$$\phi_{f_i}(x) := \text{IF } \iota_i(x) \text{ THEN } s_{f_i}^L(x) \text{ ELSE } f_i(x)$$
$$\iota_i(x) := \text{IF } P(x) \text{ THEN } \top \text{ ELSE } \perp$$
$$s_{f_i}^L(x) := \text{IF edge}(x) \text{ THEN } f_i(x) \text{ ELSE}$$
$$\text{IF } \text{term}(\delta(x)) \text{ THEN}$$
$$\text{IF } \text{cond}(\delta(x)) \text{ THEN } v \text{ ELSE } f_i(x)$$
$$\text{ELSE } s_{f_i}^L(\delta(x))$$

- Structurally identical, but subtle differences in what can initiate plus what to do if a search fails.

Discussion & Conclusion

Mailhot and Reiss (2007)

“The basic idea of a procedural approach is that the grammar should specify, not what is well-formed or ill-formed, but how to map an input to an output, as an explicitly characterized function.”

- ▶ Logical transductions are essentially **declarative programming languages** which use constraints to determine whether or not a change needs to be made.
- ▶ BMRS is therefore a declarative model (i.e., not procedural) and yet still gives an explicitly characterized function.

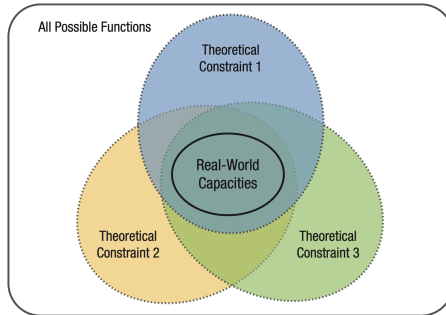
Nevens (2010)

“The no-lookahead property of the Search procedure seems to require a target-initiated, procedural search, which cannot obviously be replicated in declarative models.”

- ▶ Furthermore, BMRS is declarative and still can clearly “search”.
- ▶ Even without the specific implementations shown today, BMRS is still target-initiated.
- ▶ Search is the result of recursion, but even this is just an implementation of a fixed-point operator.

Chandlee and Jardine (2019); Yolyan and Comer (2026)

Variability/Optionality



- What other theoretical constraints are important and can we relate them to BMRS?

- ▶ How might we integrate insights about phonological variation into BMRS?
- ▶ **Idea 1:** translate BMRS into some type of predicate logic and use weighted MSO.
- ▶ **Idea 2:** alter BMRS semantics in a way similar to weighted MSO.

Lee (2026); Nelson (2026); Heinz et al. (Forthcoming)

Summary

- ▶ BMRS has been proposed as a general framework for formal phonological analysis.
- ▶ Target-centric “search” can be directly implemented with BMRS.
- ▶ This may be indicative of some type of formal convergence.

Thank You!

Bibliography I

- Andersson, S., Dolatian, H., and Hao, Y. (2020). Computing vowel harmony: The generative capacity of search & copy. In Proceedings of the Annual Meetings on Phonology.
- Bhaskar, S., Chandlee, J., and Jardine, A. (2023). Rational functions via recursive schemes. arXiv preprint arXiv:2302.03074.
- Bhaskar, S., Chandlee, J., Jardine, A., and Oakden, C. (2020). Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In Language and Automata Theory and Applications: 14th International Conference, LATA 2020, Milan, Italy, March 4–6, 2020, Proceedings 14, pages 157–169. Springer.
- Burness, P., McMullin, K., and Nevins, A. (2024). Revisiting locality in vowel harmony. In Ritter, N. A. and van der Hulst, H., editors, The Oxford Handbook of Vowel Harmony, chapter 22, pages 269–290. Oxford University Press.
- Chandlee, J. and Jardine, A. (2019). Quantifier-free least fixed point functions for phonology. In Proceedings of the 16th Meeting on the Mathematics of Language, pages 50–62.
- Chandlee, J. and Jardine, A. (2021). Computational universals in linguistic theory: Using recursive programs for phonological analysis. Language, 97(3):485–519.
- Dabbous, R., Marjori, L., Mousavi, F., Reiss, C., and Ta-Chun Shen, D. (2021). Satisfying long-distance relationships (without tiers): A strictly anti-local approach to phonology. <https://ling.auf.net/lingbuzz/006329>.
- Heinz, J., Dolatian, H., and Strother-Garcia, K., editors (Forthcoming). Doing Phonology Computationally. Oxford University Press.
- Lee, A. (2026). A computational analysis of malay affixation morphophonology. Bachelor's thesis, University of Illinois Urbana-Champaign.
- Mailhot, F. and Reiss, C. (2007). Computing long-distance dependencies in vowel harmony. Biolinguistics, 1:028–048.
- Mayer, C. (2021). Issues in Uyghur backness harmony: Corpus, experimental, and computational studies. PhD thesis, University of California, Los Angeles.
- Nelson, S. (2026). Phonological knowledge, weighted logic, and the competence/performance distinction. Manuscript, UIUC.
- Nelson, S. and Baković, E. (2026). A BMRS analysis of Trojan (and transparent) vowels in Hungarian vowel harmony. Proceedings of the Linguistic Society of America, 11(1).
- Nelson, S. and King, M. (2026). Computing tier-based functions using Boolean Monadic Recursive Schemes. Manuscript, UIUC.

Bibliography II

- Nevins, A. (2005). Conditions on (dis) harmony. PhD thesis, Massachusetts Institute of Technology.
- Nevins, A. (2010). Locality in vowel harmony. MIT Press.
- Samuels, B. D. (2009). The structure of phonological theory. Harvard University.
- van Rooij, I. and Baggio, G. (2021). Theory before the test: How to build high-verisimilitude explanatory theories in psychological science. Perspectives on Psychological Science.
- Yolyan, T. and Comer, J. (2026). Phonological processes as modal transductions. In Proceedings of the Society for Computation in Linguistics 2026, pages 549–559.