A weak but flexible logic for modeling phonological processes

Adam Jardine
Rutgers University

FLANN
March 23, 2026

Collaborators

Siddharth Bhaskar
(U. of Southern Denmark)

Jane Chandlee
(Haverford)

Tatevik Yolyan
(Rutgers)

Overview

Overview

Boolean monadic recursive schemes (BMRS) [1]

- Restricted fragment of recursive programs[2]
- Capture the subsequential functions[3] on strings
- Useful for describing phonological processes[4]
- Current work focusing on learning[5]

[1] (Bhaskar et al. 2020)

[2] (Moschovakis 2019)

[3] (Schützenberger 1977; Mohri 1997)

[4] (Chandlee and Jardine 2021)

[5] (Yolyan 2025)

What is phonology?

Phonology as functions

English plural

"-z"		**"-s"**	
[piz]	‘peas’	[k^{h}læsps]	‘clasps’
[t^{h}oʊz]	‘toes’	[mɪts]	‘mitts’
[dɑlz]	‘dolls’	[bloʊks]	‘blokes’
[dɑgz]	‘dogs’	[k^{h}ɑfs]	‘coughs’
[læbz]	‘labs’		

Phonology as functions

English plural

"-z"		**"-s"**	
[piz]	‘peas’	[k^{h}læsps]	‘clasps’
[t^{h}oʊz]	‘toes’	[mɪts]	‘mitts’
[dɑlz]	‘dolls’	[bloʊks]	‘blokes’
[dɑgz]	‘dogs’	[k^{h}ɑfs]	‘coughs’
[læbz]	‘labs’		

Analysis:

- **Underlying representation (UR)** of plural: /-z/
- A **process** mapping URs to **surface representations (SRs)**: (/piz/,[piz]), (/dɑlz/,[dɑlz]), (/k^{h}læspz/,[k^{h}læsps]), (/mɪtz/,[mɪts]), …

Goals of Generative phonology

(/piz/,[piz]), (/dɑlz/,[dɑlz]), (/k^{h}læspz/,[k^{h}læsps]),
(/mɪtz/,[mɪts]), ...

- How do we capture phonological processes?
    - SPE[6]: [-sonorant] → [-voice] / [-voice] __
    - OT[7]: Agree in voicing ≫ Don't change voicing
- What is a *possible* phonological process?

[6] Sound Pattern of English (Chomsky and Halle 1968)

[7] Optimality Theory (Prince and Smolensky 1993)

Goals of Generative phonology

High-tone spreading in Shambaa (Odden 1982)

[ku-ʃunt^{h}-a]	‘to wash’	$(/\sigma\sigma\sigma/, [\sigma\sigma\sigma])$
[ku-ɣoʃo-a]	‘to do’	$(/\sigma\sigma\sigma\sigma/, [\sigma\sigma\sigma\sigma])$
[ku-tʃí-ʃúnt^{h}-a]	‘to wash it’	$(/\sigma\acute{\sigma}\sigma\sigma/, [\sigma\acute{\sigma}\acute{\sigma}\sigma])$
[ku-ví-ɣóʃó-a]	‘to do them’	$(/\sigma\acute{\sigma}\sigma\sigma\sigma/, [\sigma\acute{\sigma}\acute{\sigma}\acute{\sigma}\sigma])$
[ku-ɣoʃo-a-ɣoʃo-a]	‘to do repeatedly’	$(/\sigma\sigma\sigma\sigma\sigma\sigma\sigma/, [\sigma\sigma\sigma\sigma\sigma\sigma\sigma])$
[ku-tʃí-ɣóʃó-á-ɣóʃó-a]	‘to do it repeatedly’	$(/\sigma\acute{\sigma}\sigma\sigma\sigma\sigma\sigma\sigma/, [\sigma\acute{\sigma}\acute{\sigma}\acute{\sigma}\acute{\sigma}\acute{\sigma}\acute{\sigma}\sigma])$

Goals of Generative phonology

Long-distance consonant harmony in Kikongo

[sos-ila]	‘searched for’	(/sos-ila/,	[sos-ila])
[sakid-ila]	‘congratulate for’	(/sakid-ila/,	[sakid-ila])
[ku-toːt-ila]	‘to harvest for’	(/ku-toːt-ila/,	[ku-toːt-ila])
[ku-kin-ina]	‘to dance for’	(/ku-kin-ila/,	[ku-kin-ina])
[ku-dumuk-ina]	‘to jump for’	(/ku-dumuk-ila/,	[ku-dumuk-ina])
[ku-dumuk-is-ina]	‘to make jump for’	(/ku-dumuk-is-ila/,	[ku-dumuk-is-ina])

Goals of Generative phonology

- Why?
  - ✓ High tone spreads to second-to-last syllable
  - × High tone spreads to center syllable

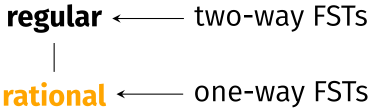
Phonology and computation

What kinds of functions are phonological processes?

- Heinz (2007);Heinz (2018): Phonological generalizations have a *computationally* restrictive character

What kinds of functions are phonological processes?

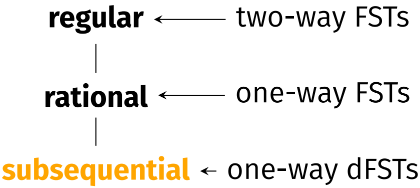
- Johnson (1972);Kaplan and Kay (1994): Phonological rules are **rational**



regular ← two-way FSTs

rational ← one-way FSTs

What kinds of functions are phonological processes?



regular ← two-way FSTs

rational ← one-way FSTs

subsequential ← one-way dFSTs

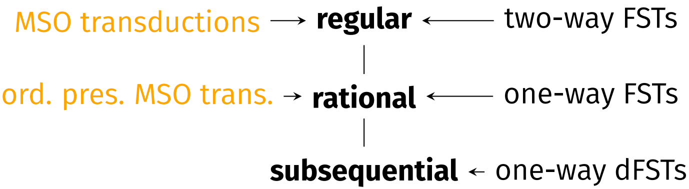
- Mohri (1997);Heinz and Lai (2013):
  Phonological processes are **subsequential**

What kinds of functions are phonological processes?

- Heinz (2018): Connections between logic and FLT[8] are useful for phonology
    - Ex: features, syllable structure, etc.

[8] Büchi (1960), McNaughton and Papert (1971), et. seq.

What kinds of functions are phonological processes?



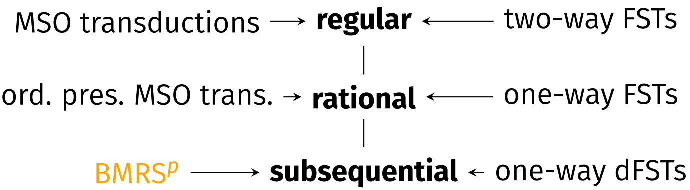
MSO transductions → **regular** ← two-way FSTs

ord. pres. MSO trans. → **rational** ← one-way FSTs

subsequential ← one-way dFSTs

Filiot and Reynier (2016)

What kinds of functions are phonological processes?



MSO transductions → **regular** ← two-way FSTs

ord. pres. MSO trans. → **rational** ← one-way FSTs

BMRS^p → **subsequential** ← one-way dFSTs

Bhaskar et al. (2020)

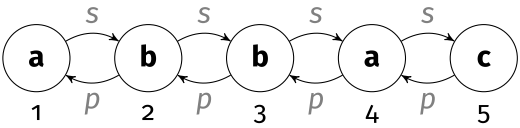
Boolean Monadic Recursive Schemes

BMRS

- Associate strings in Σ^* with structures of the form

$$\langle D; P_\sigma \text{ for } \sigma \in \Sigma, p, s \rangle$$

- E.g., *abbac* would be



where $P_a = \{1, 4\}$, $P_b = \{2, 3\}$, $P_c = \{c\}$.

BMRS

- $\mathtt{X}$ are index variables; $\mathtt{F}$ are function variables
- Syntax:

$$\begin{aligned} T \rightarrow \quad & \mathtt{x} \mid \top \mid \bot \mid \mathtt{f}(T) \mid s(T) \mid p(T) \mid \\ & \sigma(T) \mid \mathtt{if}\ T\ \mathtt{then}\ T\ \mathtt{else}\ T \end{aligned}$$

  where:

  - $\mathtt{x} \in \mathtt{X}$, $\mathtt{f} \in \mathtt{F}$, $\sigma \in \Sigma$
  - $\mathtt{x}$, $s(T)$ and $p(T)$ are index-typed, all others boolean
  - For $\sigma(T)$ and $\mathtt{f}(T)$, T must be index-typed
  - For $\mathtt{if}\ T\ \mathtt{then}\ T\ \mathtt{else}\ T$, each T is boolean-typed

BMRS

- Syntax:

$$\begin{aligned} T \to\ & \mathtt{x} \mid \top \mid \bot \mid \mathtt{f}(T) \mid s(T) \mid p(T) \mid \\ & \sigma(T) \mid \mathtt{if}\ T\ \mathtt{then}\ T\ \mathtt{else}\ T \end{aligned}$$

- Boolean predicates with one free variable, e.g.

$$\mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ b(p(\mathtt{x}))\ \mathtt{else}\ \bot$$

- Assume $\mathtt{f}(T)$ and $\sigma(T)$ return $\bot$ if T is undefined (at end or beginning of word)

BMRS

- A recursive, boolean, monadic **scheme** (or *program*) pairs $\vec{\mathtt{f}} = (\mathtt{f}_1, \ldots, \mathtt{f}_n)$ with definitions:

$$\begin{gathered}\mathtt{f}_1(\mathtt{x}) = T_1(\vec{\mathtt{f}}, \mathtt{x}) \\ \vdots \\ \mathtt{f}_n(\mathtt{x}) = T_n(\vec{\mathtt{f}}, \mathtt{x})\end{gathered}$$

  where each T_i is boolean.

- The semantics is given by least-fixed semantics of recursive programs

BMRS

An example

- Let $\Sigma = \{a, b, c\}$ and let $\vec{f} = (\mathtt{a}, \mathtt{b}, \mathtt{c})$.

$$\begin{aligned}
\mathtt{a}(\mathtt{x}) &= \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x}) \\
\mathtt{b}(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\
&\qquad\quad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot \\
\mathtt{c}(\mathtt{x}) &= c(\mathtt{x})
\end{aligned}$$

BMRS

An example

$$\mathtt{a}(\mathtt{x}) = \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathtt{b}(\mathtt{x}) = {} & \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\ & \quad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathtt{c}(\mathtt{x}) = c(\mathtt{x})$$

input: *a* *b* *a* *a* *c* *a*

BMRS

An example

$$\mathrm{a}(\mathtt{x}) = \mathtt{if}\ \mathrm{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathrm{b}(\mathtt{x}) = {} & \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\ & \qquad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathrm{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathrm{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
$\mathrm{a}(\mathtt{x})$						
$\mathrm{b}(\mathtt{x})$						
$\mathrm{c}(\mathtt{x})$						

BMRS

An example

$$\texttt{a}(\texttt{x}) = \texttt{if}\ \texttt{b}(\texttt{x})\ \texttt{then}\ \bot\ \texttt{else}\ a(\texttt{x})$$

$$\begin{aligned}\texttt{b}(\texttt{x}) = {} & \texttt{if}\ b(\texttt{x})\ \texttt{then}\ \top\ \texttt{else} \\ & \quad \texttt{if}\ a(\texttt{x})\ \texttt{then}\ \texttt{b}(p(\texttt{x}))\ \texttt{else}\ \bot\end{aligned}$$

$$\texttt{c}(\texttt{x}) = c(\texttt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
$\texttt{a}(\texttt{x})$	$\top$					
$\texttt{b}(\texttt{x})$	$\bot$					
$\texttt{c}(\texttt{x})$	$\bot$					

BMRS

An example

$$\mathtt{a}(\mathtt{x}) = \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathtt{b}(\mathtt{x}) = {} & \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\ & \quad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathtt{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
$\mathtt{a}(\mathtt{x})$	$\top$	$\bot$				
$\mathtt{b}(\mathtt{x})$	$\bot$	$\top$				
$\mathtt{c}(\mathtt{x})$	$\bot$	$\bot$				

BMRS

An example

$$\mathrm{a}(\mathtt{x}) = \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathrm{b}(\mathtt{x}) = {} & \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\ & \qquad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathrm{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
a(x)	$\top$	$\bot$				
b(x)	$\bot$	$\top$	$\top$			
c(x)	$\bot$	$\bot$				

BMRS

An example

$$\mathtt{a}(\mathtt{x}) = \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathtt{b}(\mathtt{x}) = \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else}\\ \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathtt{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
$\mathtt{a}(\mathtt{x})$	$\top$	$\bot$	$\bot$			
$\mathtt{b}(\mathtt{x})$	$\bot$	$\top$	$\top$			
$\mathtt{c}(\mathtt{x})$	$\bot$	$\bot$	$\bot$			

BMRS

An example

$$\mathtt{a}(\mathtt{x}) = \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x})$$

$$\begin{aligned}\mathtt{b}(\mathtt{x}) = {} & \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\ & \qquad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot\end{aligned}$$

$$\mathtt{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
$\mathtt{a}(\mathtt{x})$	$\top$	$\bot$	$\bot$	$\bot$		
$\mathtt{b}(\mathtt{x})$	$\bot$	$\top$	$\top$	$\top$		
$\mathtt{c}(\mathtt{x})$	$\bot$	$\bot$	$\bot$	$\bot$		

BMRS

An example

$$\begin{aligned}
\mathtt{a}(\mathtt{x}) &= \mathtt{if}\ \mathtt{b}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ a(\mathtt{x}) \\
\mathtt{b}(\mathtt{x}) &= \mathtt{if}\ b(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\
&\qquad\quad \mathtt{if}\ a(\mathtt{x})\ \mathtt{then}\ \mathtt{b}(p(\mathtt{x}))\ \mathtt{else}\ \bot \\
\mathtt{c}(\mathtt{x}) &= c(\mathtt{x})
\end{aligned}$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
a(x)	⊤	⊥	⊥	⊥	⊥	⊤
b(x)	⊥	⊤	⊤	⊤	⊥	⊥
c(x)	⊥	⊥	⊥	⊥	⊤	⊥

BMRS

An example

$$\mathtt{a}(\mathtt{x}) = \texttt{if } \mathtt{b}(\mathtt{x}) \texttt{ then } \bot \texttt{ else } a(\mathtt{x})$$

$$\begin{aligned}\mathtt{b}(\mathtt{x}) = {} & \texttt{if } b(\mathtt{x}) \texttt{ then } \top \texttt{ else} \\ & \quad \texttt{if } a(\mathtt{x}) \texttt{ then } \mathtt{b}(p(\mathtt{x})) \texttt{ else } \bot\end{aligned}$$

$$\mathtt{c}(\mathtt{x}) = c(\mathtt{x})$$

input:	*a*	*b*	*a*	*a*	*c*	*a*
a(x)	⊤	⊥	⊥	⊥	⊥	⊤
b(x)	⊥	⊤	⊤	⊤	⊥	⊥
c(x)	⊥	⊥	⊥	⊥	⊤	⊥
output:	a	b	b	b	c	a

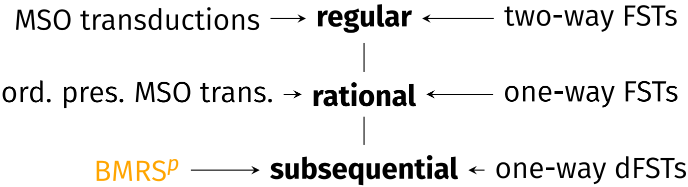
BMRS

- Fix an **output alphabet** Γ. We can include a function symbol in $\vec{\mathtt{f}}$ for each $\gamma \in \Gamma$.

$$
\begin{gathered}
\gamma_1(\mathtt{x}) = T_1(\vec{\mathtt{f}}, \mathtt{x}) \\
\vdots \\
\gamma_k(\mathtt{x}) = T_k(\vec{\mathtt{f}}, \mathtt{x}) \\
\mathtt{f}_1(\mathtt{x}) = T_{k+1}(\vec{\mathtt{f}}, \mathtt{x}) \\
\vdots \\
\mathtt{f}_n(\mathtt{x}) = T_{k+n}(\vec{\mathtt{f}}, \mathtt{x})
\end{gathered}
$$

- We can interpret this BMRS as a transduction from $\Sigma^* \rightarrow \Gamma^*$

BMRS - Expressivity



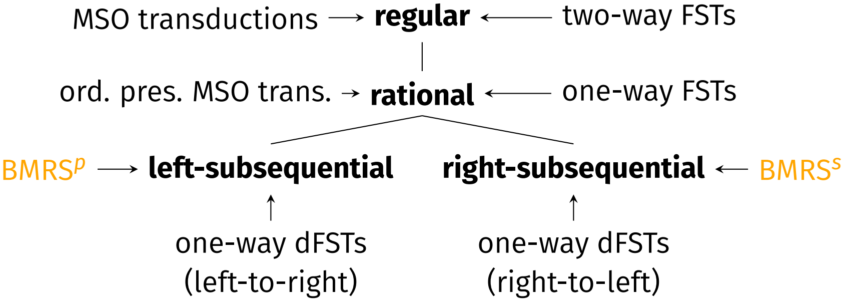
MSO transductions → **regular** ← two-way FSTs

ord. pres. MSO trans. → **rational** ← one-way FSTs

BMRS^p → **subsequential** ← one-way dFSTs

Bhaskar et al. (2020)

BMRS - Expressivity



MSO transductions → **regular** ← two-way FSTs

ord. pres. MSO trans. → **rational** ← one-way FSTs

BMRS^p → **left-subsequential**

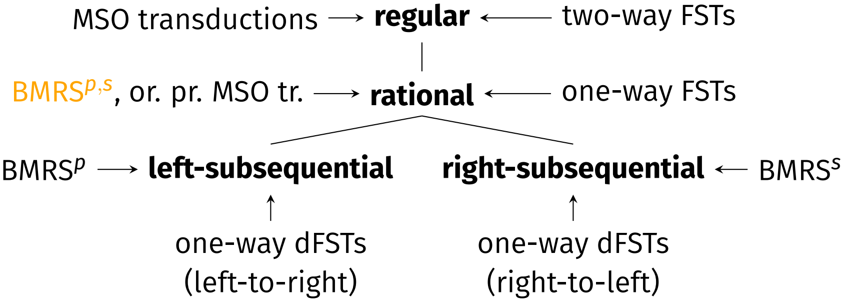
right-subsequential ← BMRS^s

one-way dFSTs (left-to-right)

one-way dFSTs (right-to-left)

Bhaskar et al. (2020)

BMRS - Expressivity



MSO transductions → **regular** ← two-way FSTs

BMRS^{p,s}, or. pr. MSO tr. → **rational** ← one-way FSTs

BMRS^{p} → **left-subsequential**

right-subsequential ← BMRS^{s}

one-way dFSTs (left-to-right) → **left-subsequential**

one-way dFSTs (right-to-left) → **right-subsequential**

Bhaskar and Jardine (in prep.)

BMRS and phonology

Shambaa spreading[9]

$$
\begin{gathered}
/\sigma\sigma\sigma\sigma\sigma/ \mapsto [\sigma\sigma\sigma\sigma\sigma] \\
/\sigma\acute{\sigma}\sigma\sigma\sigma/ \mapsto [\sigma\acute{\sigma}\acute{\sigma}\acute{\sigma}\sigma]
\end{gathered}
$$

$$
\begin{aligned}
\acute{\sigma}'(\mathtt{x}) &= \mathtt{if}\ \acute{\sigma}(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else} \\
&\qquad\quad \mathtt{if}\ \mathtt{last}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ \acute{\sigma}'(p(\mathtt{x})) \\
\sigma'(\mathtt{x}) &= \mathtt{if}\ \acute{\sigma}'(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ \top
\end{aligned}
$$

[9] We can include `last`(T) to input signature without increasing expressivity

BMRS and phonology

$$\begin{aligned}\acute{\sigma}'(\mathtt{x}) &= \mathtt{if}\ \acute{\sigma}(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else}\\ &\qquad \mathtt{if}\ \mathtt{last}(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ \acute{\sigma}'(p(\mathtt{x}))\\ \sigma'(\mathtt{x}) &= \mathtt{if}\ \acute{\sigma}'(\mathtt{x})\ \mathtt{then}\ \bot\ \mathtt{else}\ \top\end{aligned}$$

- Captures ’Elsewhere Condition” effects (Chandlee and Jardine 2021; Jardine and Oakden 2025)
- Even long-distance phonological processes are ‘myopic’ (Wilson 2003, 2006; though c.f. McCollum et al. 2020)

Learning

Learning

- dFSTs are effectively learnable (Oncina, García, and Vidal 1993)
- Yolyan (2025): fragments of BMRS^p are learnable

$$\begin{array}{ccc} a & b & \boxed{\begin{array}{c} a \\ \downarrow \\ b \end{array}} \end{array}$$

$$\begin{array}{cc} a & b \end{array}$$

To conclude

To conclude

- BMRS are a flexible, yet well-behaved model for computing string transductions
  - BMRS^p $\rightarrow$ left-subsequential
  - BMRS^s $\rightarrow$ right-subsequential
  - BMRS^{p,s} $\rightarrow$ rational
- They are useful for describing phonology
- There is a lot to learn about learning BMRS
- Also BMRS and B-RASP?

Thank you!

References I

Bhaskar, Siddharth, Jane Chandlee, Adam Jardine, and Christopher Oakden. 2020. “Boolean Monadic Recursive Schemes as a Logical Characterization of the Subsequential Functions.” In *LATA 2020*, 157–69. Lecture Notes in Computer Science. Springer.

Büchi, J. Richard. 1960. “Weak Second-Order Arithmetic and Finite Automata.” *Zeitschrift Für Mathematische Logik Und Grundlagen Der Mathmatik* 6: 66—92.

Chandlee, Jane, and Adam Jardine. 2021. “Computational Universals in Linguistic Theory: Using Recursive Programs for Phonological Analysis.” *Language* 93: 485–519.

Chomsky, Noam, and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row.

Filiot, Emmanuel, and Pierre-Alain Reynier. 2016. “Transducers, Logic, and ALgebra for Functions of Finite Words.” *ACM SIGLOG News* 3 (3): 4–19.

Heinz, Jeffrey. 2007. “The Inductive Learning of Phonotactic Patterns.” PhD thesis, UCLA.

References II

———. 2018. “The Computational Nature of Phonological Generalizations.” In *Phonological Typology*, edited by Larry Hyman and Frans Plank, 126–95. Phonetics and Phonology. De Gruyter Mouton.

Heinz, Jeffrey, and Regine Lai. 2013. “Vowel Harmony and Subsequentiality.” In *Proceedings of the 13th Meeting on Mathematics of Language*, edited by Andras Kornai and Marco Kuhlmann. Sofia, Bulgaria.

Jardine, Adam, and Christopher Oakden. 2025. “Computing Process-Specific Constraints.” *Linguistic Inquiry*, no. 56, 4: 807–16. https://doi.org/10.1162/ling_a_00510.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. Mouton.

Kaplan, Ronald, and Martin Kay. 1994. “Regular Models of Phonological Rule Systems.” *Computational Linguistics* 20: 331–78.

McCollum, Adam G., Eric Baković, Anna Mai, and Eric Meinhardt. 2020. “Unbounded Circumambient Patterns in Segmental Phonology.” *Phonology* 37 (2): 215–55. https://doi.org/10.1017/S095267572000010X.

References III

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

Mohri, Mehryar. 1997. "Finite-State Transducers in Language and Speech Processing." *Computational Linguistics* 23 (2): 269–311.

Moschovakis, Yiannis N. 2019. *Abstract Recursion and Intrinsic Complexity*. Vol. 48. Lecture Notes in Logic. Cambridge University Press.

Odden, David. 1982. "Tonal Phenomena in Kishambaa." *Studies in African Linguistics* 13 (2): 177–208.

Oncina, José, Pedro García, and Enrique Vidal. 1993. "Learning Subsequential Transducers for Pattern Recognition Tasks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (May): 448–58.

Prince, Alan, and Paul Smolensky. 1993. "Optimality Theory: Constraint Interaction in Generative Grammar." *Rutgers University Center for Cognitive Science Technical Report* 2.

Schützenberger, Marcel Paul. 1977. "Sur Une Variante Des Fonctions Séquentielles." *Theoretical Computer Science* 4: 47–57.

Wilson, Colin. 2003. "Analyzing Unbounded Spreading with Constraints: Marks, Targets, and Derivations."

References IV

———. 2006. “Unbounded Spreading Is Myopic.”

Yolyan, Tatevik. 2025. “Phonological Expressivity and Learning via Boolean Monadic Recursive Schemes.” PhD thesis, Rutgers University.