Using Recursive Programs for Phonological Analysis

Jane Chandlee & Adam Jardine

Presented by Alexa Agathos, Leo Cheng, and Logan Swanson

Introduction

This paper proposes a framework for using computational methods to do phonological analyses.

Chandlee and Jardine would like to use Boolean Monadic Recursive Schemes (BMRSs) to:

-explain observed computational properties of phonological patterns.

-capture phonological substance and linguistically significant generalizations.

Basics of a BMRS

-BMRSs consist of structures defined as logical predicates.

-Similarly to OT the predicates are constraint-like and able to be ranked

-They have an if … then … else syntax.

-The primitives of BMRSs are the boolean values of True (⊤) and False (⊥).

-There is also a finite set of monadic predicates P(t).

-These predicates take a single argument t and return true or false.

BMRS are evaluated locally which prevents computational overgeneration

Differs from the global evaluation strategy of Optimality Theory

BMRS captures both input and output based mappings

What is included as a BMRS Expression?

$\top$ and $\bot$ are expressions;

any predicate $P(t)$ is an expression;

if E_1, E_2, and E_3, are expressions, then

$\texttt{if } E_1 \texttt{ then } E_2 \texttt{ else } E_3$ is an expression

Nothing else is an expression.

Evaluate E1,
If it is true, then E returns the result of evaluating E2.
If it is false, then E returns the result of evaluating E3.

Licensing and Blocking in BMRS

An expression is considered to be a licensing structure when it allows something to surface on the output form.

An expression is considered to be a blocking structure when it prevents a certain feature from surfacing on the output form.

Defining BMRS Formally, Input Feature Predicates

Given the set of features $\mathbf{F}$ = {[F], [G], ..., [Z]} and boundary symbols {⋊,⋉}

Input Feature Predicates $\mathbf{I}$ = {$[F]_i(t)$, $[G]_i(t)$, ..., $[Z]_i(t)$, $⋊_i(t)$,$⋉_i(t)$}, where t is the argument (segment, syllable, etc) >> values are T or F

Output Feature Predicates $\mathbf{O}$ = {$[F]_o(t)$, $[G]_o(t)$, ..., $[Z]_o(t)$, $⋊_o(t)$,$⋉_o(t)$}

E.g. Input /tɛd/

⋊ t ɛ d ⋉ $[son]_i(2)$ = F $[son]_i(3)$ = T $[son]_i(1)$ = F $[vowel]_i(3)$ = T $[vowel]_i(2)$ = F

1 2 3 4 5 $⋊_i(1)$ = T $⋊_i(2)$ = F

Predecessor p(x) p(4) = 3 i.e. predecessor of /d/ is /ɛ/

Successor s(x) s(4) = 5 i.e. successor of /d/ is ⋉

$[son]_i(p(4))$ = ? $[son]_i(p(p(4)))$ = ? $⋊_i(p(2))$ = ? $⋊_i(p(p(2)))$ = ?

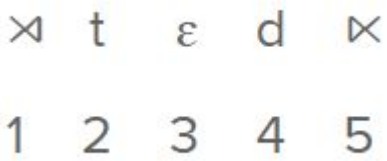
Building Logical Expressions if...then...else

So far: T/F are values and Predicates P(t) after evaluation also gives T/F

Now, we can build these into logical expressions (again gives T/F values)

$$\text{If } E_1 \text{ then } E_2 \text{ else } E_3$$

E.g. voiced obstruent in the input



if $[\text{son}]_i(x)$ then F else $[\text{voi}]_i(x)$

Conveniently, we can give a shorthand to the above statement

$\left[\begin{smallmatrix}-\text{son}\\+\text{voi}\end{smallmatrix}\right]_i(x)$:= if $[\text{son}]_i(x)$ then F else $[\text{voi}]_i(x)$ ~ similar to the familiar feature matrix

Another notation, specifying the position (e.g. word-final voiced obstruent)

$\left[\begin{smallmatrix}-\text{son}\\+\text{voi}\end{smallmatrix}\right]_i \rtimes_i(x)$:=if $\left[\begin{smallmatrix}-\text{son}\\+\text{voi}\end{smallmatrix}\right]_i x)$ then $\rtimes_i(s(x))$ else F

Defining Output Feature Predicates

Where the mapping takes place, not just a shorthand

Basic structure is the same as building complex input predicates

$[voi]_o(x)$ = if $\left[\begin{smallmatrix}-son\\+voi\end{smallmatrix}\right]_i \rtimes_i(x)$ then F else $[voi]_i(x)$ F==blocking, T==licensing

Markedness/blocking faithfulness

The entire mapping will consist of all individual maps for all features

Suppose only [voi] feature changes, we need also

$[son]_o(x) = [son]_i(x)$, $[cor]_o(x) = [cor]_i(x)$, $[lab]_o(x) = [lab]_i(x)$ and

out(x) = if $\rtimes_i(x)$ then F else accounts for deletion

if $\ltimes_i(x)$ then F else T $\rtimes_o(t), \ltimes_o(t)$? Out?

Recursion

More powerful side O-O recursion, potentially unbounded

E.g. Shambaa, underlying H tone spreading to the penultimate syllable

$$/\sigma\acute{\sigma}\sigma\sigma\sigma/ \rightarrow \sigma\acute{\sigma}\acute{\sigma}\sigma\sigma \rightarrow \sigma\acute{\sigma}\acute{\sigma}\acute{\sigma}\sigma \rightarrow [\sigma\acute{\sigma}\acute{\sigma}\acute{\sigma}\sigma]$$

1 2 3 4 5

$\acute{\square}_i(2)$ = T $\acute{\square}_i(3)$ = F $\text{final}_i(5)$ = T $\text{final}_i(4)$ = F $\text{final}_i(x) = \texttt{if } \ltimes_i(s(x)) \texttt{ then } \top \texttt{ else } \bot$

$$\begin{aligned}
\sigma_o(x) &= \sigma_i(x) \\
\acute{\square}_o(x) &= \texttt{if } \text{final}_i(x) \texttt{ then } \bot \texttt{ else} \\
&\quad \texttt{if } \acute{\square}_o(p(x)) \texttt{ then } \top \texttt{ else} \\
&\quad \acute{\square}_i(x)
\end{aligned}$$

Order matters, need to know the value of the previous syllable
Still subsequential (regular + deterministic)

Case Study 1: Interaction of Stress & Length in Hixkaryana

Stress in Hixkaryana is predictable.

There is an iterative, iambic pattern with long vowels in the stressed syllables.

There is a nonfinality condition which prevents this iteration from reaching the end of the word aka the final syllable is **not** stressed.

In disyllabic words the first syllable is stressed because of this condition.

Stress in Open Syllables

Hixkaryana (Cariban; Kager 1999)

[to.ró:.no]	'small bird'	LLL	↦	[LH́L]
[ne.mó:.ko.tó:.no]	'it fell'	LLLLL	↦	[LH́LH́L]
[a.tʃó:.wo.wo]	'wind'	LLLL	↦	[LH́LL]
[kʷá:.ja]	'red and green macaw'	LL	↦	[H́L]

Stress in Closed Syllables

Closed syllables are treated as heavy and receive stress

[ák.ma.tá:.ri]	'branch'	HLLL	↦	[ÁLÁL]
[tóh.ku.rʲé:.ho.na]	'to Tohkurye'	HLLLL	↦	[ÁLÁLL]
[nák.ɲóh.játʃ.ke.ná:.no]	'they were burning it'	HHHLLL	↦	[ÁÁÁLÁL]
[kʰa.ná:.níh.no]	'I taught you'	LLHL	↦	[LÁÁL]
[mi.há:.na.níh.no]	'you taught him'	LLLHL	↦	[LÁLÁL]

BMRS Analysis of Hixkaryana Facts

Need to define:

When a syllable is heavy in the output ($H_o(x)$)

When a syllable is light in the output ($L_o(x)$)

When a syllable is stressed in the output $\acute{\square}_o(x)$

Defining Predicates

$$\sigma_{\mathrm{i}}(x) \;=\; \texttt{if } \mathrm{L}_{\mathrm{i}}(x) \texttt{ then } \top \texttt{ else } \mathrm{H}_{\mathrm{i}}(x)$$

If syllable is light in the input, then True, else the syllable is heavy in the input

We also need to define when a syllable is potentially in a clash or lapse.

$$\begin{aligned}
\mathrm{clash}(x) &= \texttt{if } \sigma_{\mathrm{i}}(x) \texttt{ then } \acute{\Box}_{\mathrm{o}}(p(x)) \texttt{ else } \bot \\
\mathrm{lapse}(x) &= \texttt{if } \acute{\Box}_{\mathrm{o}}(p(x)) \texttt{ then } \bot \texttt{ else} \\
&\qquad \texttt{if } \sigma_{\mathrm{i}}(p(x)) \texttt{ then } \sigma_{\mathrm{i}}(x) \texttt{ else } \bot
\end{aligned}$$

We also need to define a predicate to identify the first syllable of disyllabic words.

$$\mathrm{only}_{\mathrm{i}}(x) \;=\; \texttt{if } \mathrm{final}_{\mathrm{i}}(s(x)) \texttt{ then } \mathrm{initial}_{\mathrm{i}}(x) \texttt{ else } \bot$$

Ranking of Predicates for Stressed Syllables

$$
\begin{aligned}
\acute{\square}_{\mathrm{o}}(x) \quad = \quad & \texttt{if } \mathrm{only}_{\mathrm{i}}(x) \texttt{ then } \top \texttt{ else} && \textbf{Licensing} \\
& \texttt{if } \mathrm{final}_{\mathrm{i}}(x) \texttt{ then } \bot \texttt{ else} && \textbf{Blocking} \\
& \texttt{if } \mathrm{H}_{\mathrm{i}}(x) \texttt{ then } \top \texttt{ else} && \textbf{Licensing} \\
& \texttt{if } \mathrm{initial}_{\mathrm{i}}(x) \texttt{ then } \bot \texttt{ else} && \textbf{Blocking} \\
& \texttt{if } \mathrm{clash}(x) \texttt{ then } \bot \texttt{ else} && \textbf{Blocking} \\
& \mathrm{lapse}(x) && \textbf{Licensing}
\end{aligned}
$$

Motivations for blocking or licensing

-**Only$_i$(x)** is **licensing** to account for disyllabic word

-**Final$_i$(x)** is **blocking** in order to prevent stress from appearing on final syllable

-**H$_i$(x)** is **licensing** to allow for weight-to-stress principle

-**Initial$_i$(x)** as a **blocking** structure allows for the iambic pattern

-**Clash(x)** as a **blocking** structure and **lapse(x)** as a **licensing** structure allows for the left-to-right binary stress pattern

Definition for output weight of syllables

Any syllable that receives stress becomes heavy.

$$\mathrm{H_o}(x) \;=\; \texttt{if}\ \acute{\Box}_\mathrm{o}(x)\ \texttt{then}\ \top\ \texttt{else}\ \mathrm{H_i}(x)$$

If the syllable is stressed then it is True for $H_o(x)$

Else: Any heavy syllable in the input becomes heavy in the output.

$$\mathrm{L_o}(x) \;=\; \texttt{if}\ \acute{\Box}_\mathrm{o}(x)\ \texttt{then}\ \bot\ \texttt{else}\ \mathrm{L_i}(x)$$

If the syllable is stressed then it is False for $L_o(x)$

Any light syllable in the input becomes light in the output

Input /LLLLL/

	⋊	L	L	L	L	L	⋉
	1	2	3	4	5	6	7
$\text{initial}_{\text{i}}(x)$	⊥	⊤	⊥	⊥	⊥	⊥	⊥
$\text{clash}(x)$	⊥		⊥	⊤	⊥	⊤	⊥
$\text{lapse}(x)$	⊥		⊤		⊤		⊥
$\acute{\square}_{\text{o}}(x)$	⊥	⊥	⊤	⊥	⊤	⊥	⊥

Initial$_{\text{i}}$(x) and clash(x) are blocking structures, so when they evaluate as true, the syllable won't receive stress in the output. (2,4,6)

When licensing structure lapse(x) evaluates as true, the syllable gets stress in the output. (3,5)

Input /LLLL/

	⋊ 1	L 2	L 3	L 4	L 5	⋉ 6
$\text{final}_i(x)$	⊥	⊥	⊥	⊥	⊤	⊥
$\text{H}_i(x)$	⊥	⊥	⊥	⊥		⊥
$\text{initial}_i(x)$	⊥	⊤	⊥	⊥		⊥
$\text{clash}(x)$	⊥		⊥	⊤		⊥
$\text{lapse}(x)$	⊥		⊤			⊥
$\acute{\square}_o(x)$	⊥	⊥	⊤	⊥	⊥	⊥

Final_i(x), initial_i(x), and clash(x) are blocking structures so when they evaluate as true the syllable is not stressed in the output. (2,4,5)

Lapse(x) is licensing so when True the syllable gets stress in the output. (3)

Input /HLLLL/

	⋊ 1	H 2	L 3	L 4	L 5	L 6	⋉ 7
$\text{final}_i(x)$	⊥	⊥	⊥	⊥	⊥	⊤	⊥
$\text{H}_i(x)$	⊥	⊤	⊥	⊥	⊥		⊥
$\text{initial}_i(x)$	⊥		⊥	⊥	⊥		⊥
$\text{clash}(x)$	⊥		⊤	⊥	⊤		⊥
$\text{lapse}(x)$	⊥			⊤			⊥
$\acute{\square}_o(x)$	⊥	⊤	⊥	⊤	⊥	⊥	⊥

$\text{Final}_i(x)$ blocks stress from output of 6

$\text{H}_i(x)$ is evaluated as true for 2, this licenses stress in the output

Clash(x) is true for 3 and 5, blocking stress

Lapse(x) licenses stress in output for 4

Input /HHHLLL/

	⋊ 1	H 2	H 3	H 4	L 5	L 6	L 7	⋉ 8
$\text{final}_i(x)$	⊥	⊥	⊥	⊥	⊥	⊥	⊤	⊥
$\text{H}_i(x)$	⊥	⊤	⊤	⊤	⊥	⊥		⊥
$\text{initial}_i(x)$	⊥				⊥	⊥		⊥
$\text{clash}(x)$	⊥				⊤	⊥		⊥
$\text{lapse}(x)$	⊥					⊤		⊥
$\acute{\square}_o(x)$	⊥	⊤	⊤	⊤	⊥	⊤	⊥	⊥

$\text{Final}_i(x)$ blocks element 7 from receiving stress

$\text{H}_i(x)$ licenses 2, 3, and 4 to receive stress

Clash(x) blocks element 5 from receiving stress

Lapse(x) licenses 6 to receive stress

Input /LL/

	⋊	L	L	⋉
	1	2	3	4
$\text{only}_\text{i}(x)$	⊥	⊤	⊥	⊥
$\text{final}_\text{i}(x)$	⊥		⊤	⊥
$\text{H}_\text{i}(x)$	⊥			⊥
$\text{initial}_\text{i}(x)$	⊥			⊥
$\acute{\square}_\text{o}(x)$	⊥	⊤	⊥	⊥

$\text{only}_\text{i}(x)$ being true licenses stress on syllable 2

$\text{final}_\text{i}(x)$ being true blocks stress on syllable 3

Input /LLHL/

	⋊ 1	L 2	L 3	H 4	L 5	⋉ 6
$\text{only}_i(x)$	⊥	⊥	⊥	⊥	⊥	⊥
$\text{final}_i(x)$	⊥	⊥	⊥	⊥	⊤	⊥
$\text{H}_i(x)$	⊥	⊥	⊥	⊤		⊥
$\text{initial}_i(x)$	⊥	⊤	⊥			⊥
$\text{clash}(x)$	⊥		⊥			⊥
$\text{lapse}(x)$	⊥		⊤			⊥
$\acute{\square}_o(x)$	⊥	⊥	⊤	⊤	⊥	⊥
$\text{H}_o(x)$	⊥	⊥	⊤	⊤	⊥	⊥
$\text{L}_o(x)$	⊥	⊤	⊥	⊥	⊤	⊥
Output		L	Ĥ	Ĥ	L	

$\text{Final}_i(x)$ being true blocks stress in element 5

$\text{H}_i(x)$ being true licenses stress in element 4

$\text{initial}_i(x)$ being true blocks stress in element 3

lapse(x) being true licenses stress in element 2

Case Study 2: Elsewhere Condition Effects

Strict Substructure Ordering Theorem (SSOT):

For any ranking of structures in a BMRS definition, whenever $STRUCT_j(x)$ implies $STRUCT_i(x)$ but the converse is not true, if i < j in order, then $STRUCT_j(x)$ will never evaluate to T and thus never take effect.

E.g. $[voi]_o(x)$ = if $V\underline{C}V_i(x)$ then T else

if $C_i(x)$ then F else

$[voi]_i(x)$

$V\underline{C}V_i(x) \Rightarrow C_i(x)$

where $V\underline{C}V_i(x)$ indicates an intervocalic consonant, $C_i(x)$ indicates a consonant

How SSOT predicts EC effect

English Vowel Shortening/Lengthening

a. shortening $\bar{V} \rightarrow \breve{V} \,/\, \underset{\substack{| \\ (\,\acute{\sigma}}}{\underline{\quad}}\ C_0\ \underset{\substack{| \\ \sigma)}}{V}$

b. lengthening $\breve{V} \rightarrow \bar{V} \,/\, \underset{\substack{| \\ (\,\acute{\sigma}}}{\underline{\quad}}\ C\ \underset{\substack{| \\ \sigma)}}{i}\ V$

a.

	/radical/	/radial/
Length.	—	(rādi)al
Short.	(rădi)cal	(rădi)al
Output	rădical	*rădial

b.

	/radical/	/radial/
Short.	(rădi)cal	(rădi)al
Length.	—	(rādi)al
Output	rădical	rādial

brhd$_i$(x) ~ head of binary branching

[-hi]CiV$_i$(x) , brhd&[-hi]CiV$_i$(x)

[long]$_o$(x) = if brhd&[-hi]CiV$_i$(x) then T else >> lengthening

if brhd$_i$(x) then F else >> shortening

[long]$_i$(x)

if lengthening applies, by SSOT, shortening will not apply; in order for shortening to apply, lengthening can not apply

Case Study 3: Typology of Repairs for *NT

- Many (though not all) languages avoid NT-type sequences, and different languages employ different repair strategies to avoid these sequences:
  - Deletion
  - (de) nasalization
  - (de) voicing
- BMRS can elegantly generate these typologies by altering the licensing/blocking relationships between structures and features
  - The combinatorics are restricted by the inherent logic of BMRS, since not all structures will meaningfully block or licence all features

Inherent Logic: Meaningful Structures

$$\texttt{if } X \texttt{ then } B \texttt{ else } Y$$

X is **meaningful** iff:

- It is NOT the case that (B is **T** and X ➔ Y)

  AND

- It is NOT the case that (B is **F** and X ➔ ¬Y)

Examples (non-meaningful structures):

meaningless

$$\mathtt{out}(x) \;=\; \mathtt{if}\ \underline{\mathrm{NC}}_i(x)\ \mathtt{then}\ \top\ \mathtt{else}\ [\mathrm{nas}]_i(x)$$

$= [\mathrm{nas}]_i(x)$

meaningful

$$\begin{aligned}\mathtt{out}(x) \;=\; & \mathtt{if}\ \underline{\mathrm{NC}}_i(x)\ \mathtt{then}\ \top\ \mathtt{else} \\ & \mathtt{if}\ [\mathrm{nas}]_i(x)\ \mathtt{then}\ \bot\ \mathtt{else} \\ & \top\end{aligned}$$

Typologies: Nasal Deletion

$$\begin{aligned} \texttt{out}(x) &= \texttt{if } \underline{\text{NC}}_{\circ\,\text{i}}(x) \texttt{ then } \bot \texttt{ else } \top \\ [\text{nas}]_{\text{o}}(x) &= [\text{nas}]_{\text{i}}(x) \\ [\text{voi}]_{\text{o}}(x) &= [\text{voi}]_{\text{i}}(x) \end{aligned}$$

/Nbaya/	[mbaya]	'CLASS 9/10-bad'
/Nkubwa/	[kubwa]	'CLASS 9/10-big'

(Bantu; Choti 2015)

<u>N</u>T(x) **blocks** out(x)

Typologies: Consonant Voicing

$$
\begin{aligned}
\texttt{out}(x) &= \top \\
[\text{nas}]_{\text{o}}(x) &= [\text{nas}]_{\text{i}}(x) \\
[\text{voi}]_{\text{o}}(x) &= \texttt{if } \underline{\text{NC}}_{\text{i}}(x) \texttt{ then } \top \texttt{ else } [\text{voi}]_{\text{i}}(x)
\end{aligned}
$$

➡

/kampa/ [kamba] ‘yours’

(Quechua; Orr 1962)

NT(x) **licenses** voi(x)

Full Paradigm

	$\underline{\text{NC}}_{\circ}$		$\text{NC}_{\underline{\circ}}$	
	licensing	blocking	licensing	blocking
out	(meaningless)	nasal deletion (Swahili)	(meaningless)	consonant deletion (Indonesian)
$[\text{nas}]_{\circ}$	(meaningless)	denasalization (Mandar)	nasalization (Konjo)	(meaningless when $[-\text{son}] \rightarrow \neg[\text{nas}]$)
$[\text{voi}]_{\circ}$	(meaningless when $[\text{nas}] \rightarrow [\text{voi}]$)	nasal devoicing (Ndonga (?) , Pokomo (?))	voicing (Quechua)	(meaningless)

Table 18: Summary of *$\text{NC}_{\circ}$ typology

Long-Distance Processes

- Although the previous examples have been output-local, BMRS can also capture truly long-distance transformations, like harmonies
  - Notably, because of the limits placed on recursion, BMRS can only do unbounded search in one direction
  - Unlike some other approaches to harmony processes that we have seen, BMRS do not make reference to tiers, instead using recursion to perform unbounded search for the relevant types of segments
- In contrast to the output-local transformations, modeling these types of processes requires at least one explicitly recursive predicate that is part of the system of equations, but not itself an output predicate

Conclusion

- BMRS offer formal logical representations that are intuitive from a phonological perspective
- They maintain appropriate computational restrictiveness: all and only the **subsequential** functions can be represented this way
- BMRS can capture the interactions of multiple generalizations, enabling them to capture conflicting constraints or pressures, Elsewhere Condition Effects, and typologies of repairs for marked structures

Additional Materials: Place Assimilation

$$
\begin{aligned}
[\mathrm{lab}]_{\mathrm{o}}(x) \;=\; & \texttt{if } \underline{[\mathrm{nas}]}[\mathrm{lab}]_{\mathrm{i}}(x) \texttt{ then } \top \texttt{ else} \\
& \texttt{if } \underline{[\mathrm{nas}]}[\mathrm{cor}]_{\mathrm{i}}(x) \texttt{ then } \bot \texttt{ else} \\
& \texttt{if } \underline{[\mathrm{nas}]}[\mathrm{dor}]_{\mathrm{i}}(x) \texttt{ then } \bot \texttt{ else} \\
& [\mathrm{lab}]_{\mathrm{i}}(x)
\end{aligned}
$$

Additional Materials: Recursive Helper

$$
\begin{aligned}
\sigma_i(x) &= \texttt{if } \mathrm{L}_i(x) \texttt{ then } \top \texttt{ else } \mathrm{H}_i(x) \\
\mathrm{clash}(x) &= \texttt{if } \sigma_i(x) \texttt{ then } \acute{\Box}_o(p(x)) \texttt{ else } \bot \\
\mathrm{lapse}(x) &= \texttt{if } \acute{\Box}_o(p(x)) \texttt{ then } \bot \texttt{ else} \\
&\quad\ \texttt{if } \sigma_i(p(x)) \texttt{ then } \sigma_i(x) \texttt{ else } \bot
\end{aligned}
$$

$$
\begin{aligned}
\acute{\Box}_o(x) &= \texttt{if } \mathrm{only}_i(x) \texttt{ then } \top \texttt{ else} \\
&\quad\ \texttt{if } \mathrm{final}_i(x) \texttt{ then } \bot \texttt{ else} \\
&\quad\ \texttt{if } \mathrm{H}_i(x) \texttt{ then } \top \texttt{ else} \\
&\quad\ \texttt{if } \mathrm{initial}_i(x) \texttt{ then } \bot \texttt{ else} \\
&\quad\ \texttt{if } \mathrm{clash}(x) \texttt{ then } \bot \texttt{ else} \\
&\quad\ \mathrm{lapse}(x)
\end{aligned}
$$

Output-local

$$
\begin{aligned}
\text{follows-}[\mathrm{nas}]_i(x) &= \texttt{if } \times_i(x) \texttt{ then } \bot \texttt{ else} \\
&\quad\ \texttt{if } [\mathrm{nas}]_i(p(x)) \texttt{ then } \top \texttt{ else} \\
&\quad\ \text{follows-}[\mathrm{nas}]_i(p(x))
\end{aligned}
$$

$$
\mathrm{N...}\underline{\mathrm{L}}_i(x) = \texttt{if } [\mathrm{lat}]_i(x) \texttt{ then follows-}[\mathrm{nas}]_i(x) \texttt{ else } \bot
$$

$$
\begin{aligned}
[\mathrm{nas}]_o(x) &= \texttt{if } \mathrm{N...}\underline{\mathrm{L}}_i(x) \texttt{ then } \top \texttt{ else} \\
&\quad\ [\mathrm{nas}]_i(x) \\
[\mathrm{lat}]_o(x) &= \texttt{if } [\mathrm{nas}]_o(x) \texttt{ then } \bot \texttt{ else} \\
&\quad\ [\mathrm{lat}]_i(x)
\end{aligned}
$$

Long-Distance