

# **Finite-State Methods in Natural-Language Processing: Rewriting Rules**

**Ronald M. Kaplan**

**and**

**Martin Kay**

# Reference

- **Chomsky, Noam. and Morris Halle. *The Sound Pattern of English*. Harper Row, 1968**
- **Kenstowicz, Michael and Charles Kisseberth. *Generative Phonology: Description and Theory*. Academic Press, 1979**

# Spelling Conventions

**N** → m/ \_ +labial

**N** → n

**iN+tractable** → intractable

**iN+practical** → impractical

**iN** is the common negative prefix

— im before labial

— in otherwise

c.f. input → input

# Rewriting Rules

$$\alpha \rightarrow \beta / \lambda \_ \rho$$

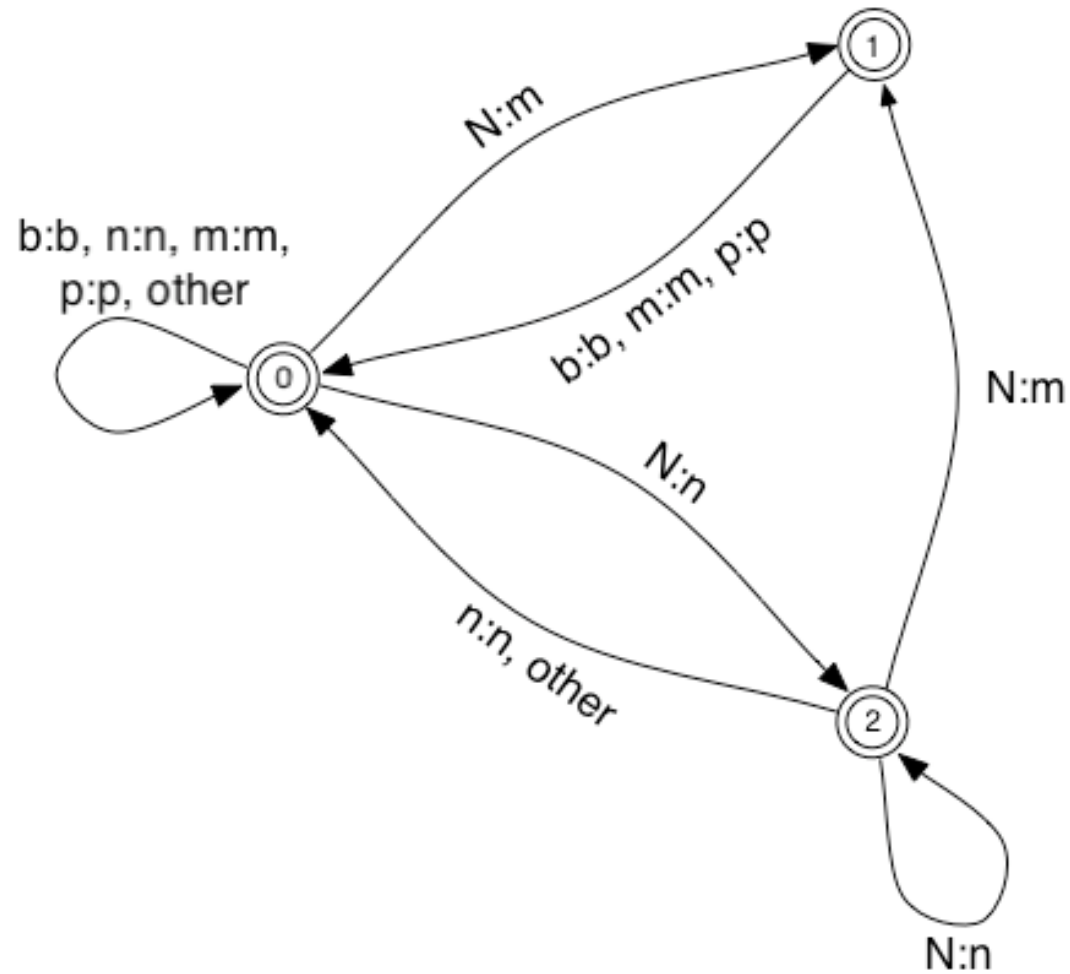
**No rerewriting!**

$$\varepsilon \rightarrow ab / a \_ b$$

would map  $ab$  into  $a^n b^n$

$\alpha \rightarrow \beta / \lambda \_ \rho$  is not the same as  $\lambda \alpha \rho \rightarrow \lambda \beta \rho$

# An in/im Transducer

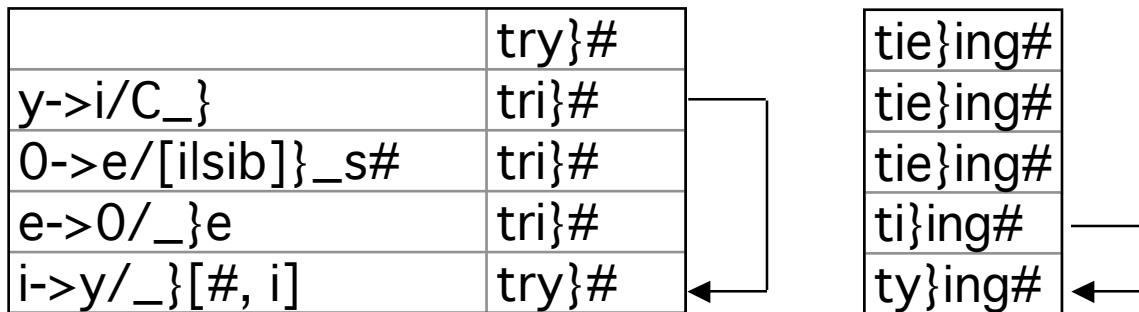


# Ordered rules

	try}#	try}s#	try}ed#	try}ing#
y->i/C_}	tri}#	tri}s#	tri}ed#	tri}ing#
0->e/[ilsib]}_s#	tri}#	tri}es#	tri}ed#	tri}ing#
e->0/_}e	tri}#	tri}es#	tri}ed#	tri}ing#
i->y/_}[#, i]	try}#	tri}es#	tri}ed#	try}ing#

	tie}#	tie}s#	tie}ed#	tie}ing#
y->i/C_}	tie}#	tie}s#	tie}ed#	tie}ing#
0->e/[ilsib]}_s#	tie}#	tie}s#	tie}ed	tie}ing#
e->0/_}e	tie}#	tie}es#	ti}ed#	ti}ing#
i->y/_}[#, i]	tie}#	tie}es#	ti}ed#	ty}ing#

# Feeding

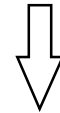
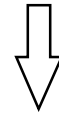


# Application Direction—Examples

$V: \rightarrow V / V: C^* \_$

left to right

## Gidabal

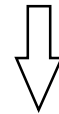


g u n u: m + b a: + d a: ng + b e: +  
 g u n u: m + b a + d a: ng + b e +

*is certainly right on the stump*

right to left  
or  
simultaneous

## Slovak



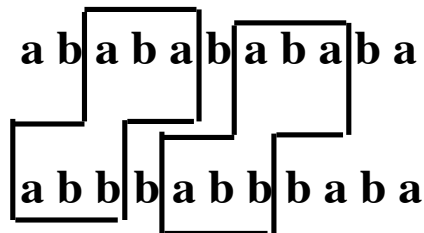
v o l + a: v + a: m e:  
 v o l + a: v + a m e

*we call often*

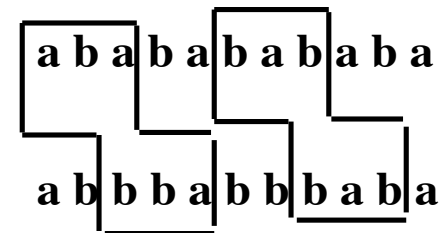
# Direction of Application

<b>a → b / ab _ ba</b>	
—Input	abababababa
—Output	
abbbabbbaba	<i>(Left to right)</i>
ababbbabbbba	<i>(Right to left)</i>
abbbbbbbba	<i>(Simultaneous)</i>

**Left to right**

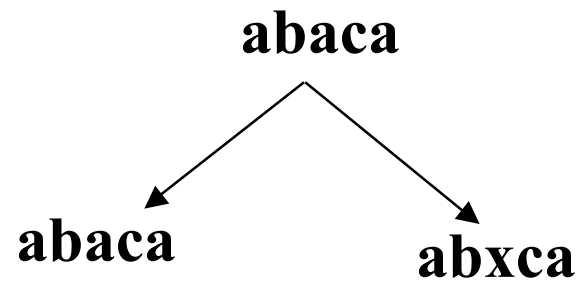


**Right to Left**



# Optional Rules

$a \rightarrow x / b$



# Special Operators

- $\text{Intro}(S) =_{\text{df}} [\text{Id}(\Sigma) \cup [\{\epsilon\} \times S]]^*$

Strings of  $\Sigma^*$  with members of the set  $S$  of symbols freely intersperced.  $\text{Intro}^{-1}(S)$  removes members of  $S$  if  $S$  and  $\Sigma$  are disjoint.

- $L_S =_{\text{df}} \text{Range}(\text{Id}(L) \circ \text{Intro}(S))$  *Ignore*

Strings that would be in  $L$  if some symbols in  $S$  were removed.

- $\text{If-P-then-S}(L_1, L_2)$

$$=_{\text{df}} \{x_1 x_2 \mid \text{if } x_1 \in L_1, \text{ then } x_2 \in L_2\} = \overline{\overline{L_1 L_2}}$$

- $\text{If-S-then-P}(L_1, L_2)$

$$= \overline{\overline{L_1 L_2}} = \text{If-P-then-S}(\overline{\overline{L_1}}, \overline{\overline{L_2}})$$

- $\text{P-iff-S}(L_1, L_2) = \text{If-P-then-S}(L_1, L_2) \cap$

$$\text{If-S-then-P}(L_1, L_2)$$

# Optional Replacement

$\alpha \rightarrow \beta$  (optional)

$Replace = [Id(\Sigma)^* Opt(\alpha \times \beta)]^*$

where  $Opt(R) = R \cup \{<\epsilon, \epsilon>\}$

# Context Restrictions—1

$Replace = [Id(\Sigma)^* Opt(Id(\lambda) \alpha \times \beta Id(\rho))]^*$

$B \rightarrow b / V \_ V$

$V B V B V$

$V b V B V$



Accepts this

$V B V B V$

$V b V b V$



but not this

... but both should be accepted

# Context Restrictions—2

*Put markers in the right places*

*Replace* =  $[Id(\Sigma)^* Opt(Id(<) \alpha_m \times \beta_m Id(>))]^*$

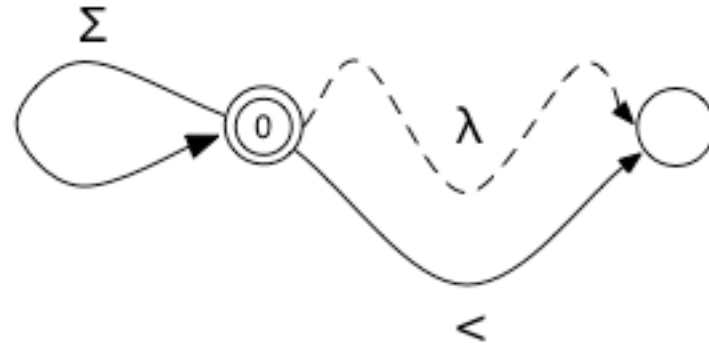
*Remove markers*

$X_m$  is  $X$  with  $m$ 's freely interspersed

# Left Context — 1

**Purpose: Eliminate brackets that are not preceded by an instance of the left context.**

$$LC(\lambda) = (\Sigma^* \lambda <)^* \Sigma^*$$



**Every < is preceded by λ.**

**By default, '<' ∉ Σ**

**But there can be λ without <**

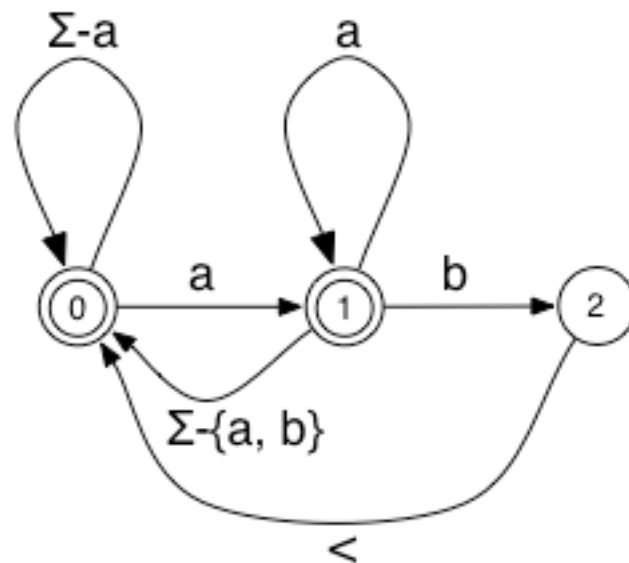
*Nondeterministic*

# Left Context — 2

But there can be  $\lambda$  without  $<$   
 $LC(\lambda) = P\text{-iff-}S(\Sigma^* \lambda, < \Sigma^*)$

$$\lambda = a b$$

$$LC(\lambda) = (\Sigma^* \lambda <)^* \Sigma^*$$



# Left Context — 3

$$\text{LC}(\lambda) = \text{P-iff-S}(\Sigma^* \lambda, < \Sigma^*)$$

What about  $\lambda$  -prefixes in  $\lambda$ ?

$\text{LC}(a(b))$  contains  $a <$

$a < b$

$a b <$

but not  $a < b <$

$$\text{LC}(\lambda) = \text{P-iff-S}((\Sigma^* \lambda) \prec, < \Sigma^* \prec)$$

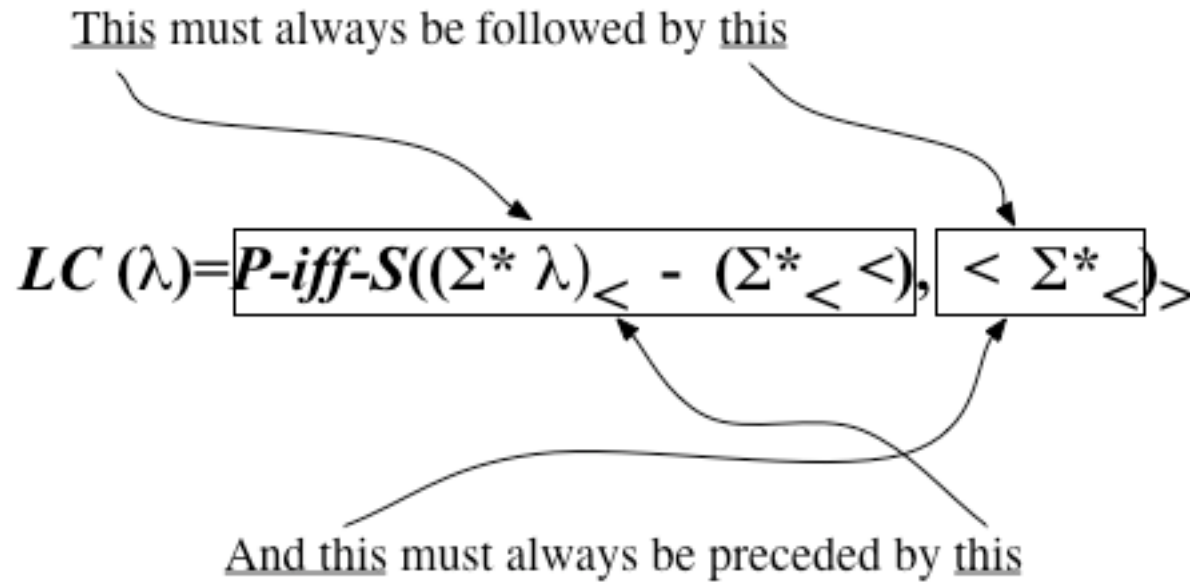
# Left Context — 4

$$LC(\lambda) = P\text{-iff-}S((\Sigma^* \lambda) \prec, \prec \Sigma^* \prec)$$

What if  $\varepsilon \in \lambda$ ?

Consider  $\{\varepsilon\} = \lambda$ . Every substring must be followed by  $\prec$ , including substrings that end in  $\prec$ . The language therefore contains no strings of finite length.

# Left Context—3

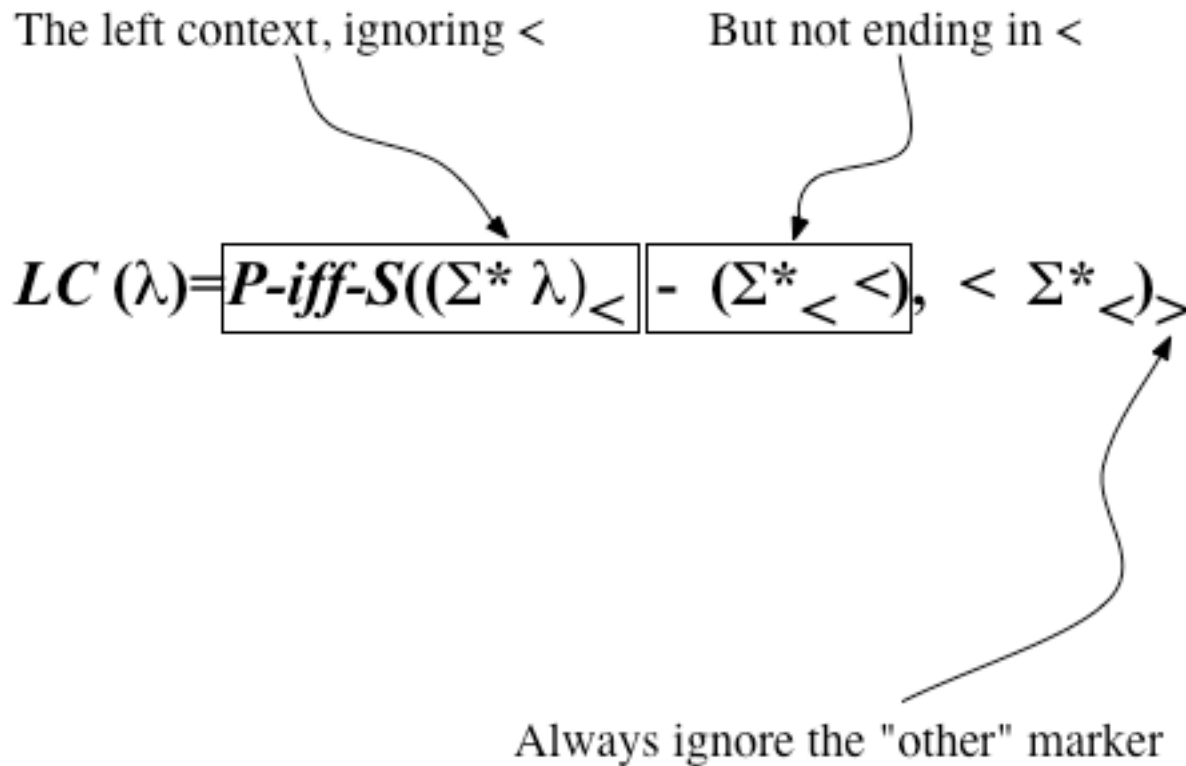


# Left Context—3

The left context, ignoring <                      But not ending in <

$$LC(\lambda) = \boxed{P\text{-iff-}S((\Sigma^* \lambda)_{<}} \mid \boxed{- (\Sigma^*_{< <})}, < \Sigma^*_{<})_{>}$$

# Left Context—3



# Left Context Examples

$$\lambda = abab$$

**xabxabab<xabab<ab<x**

$$\lambda = \varepsilon$$

**<a<b<c<d<e<f<g<**

# Left and Right Context

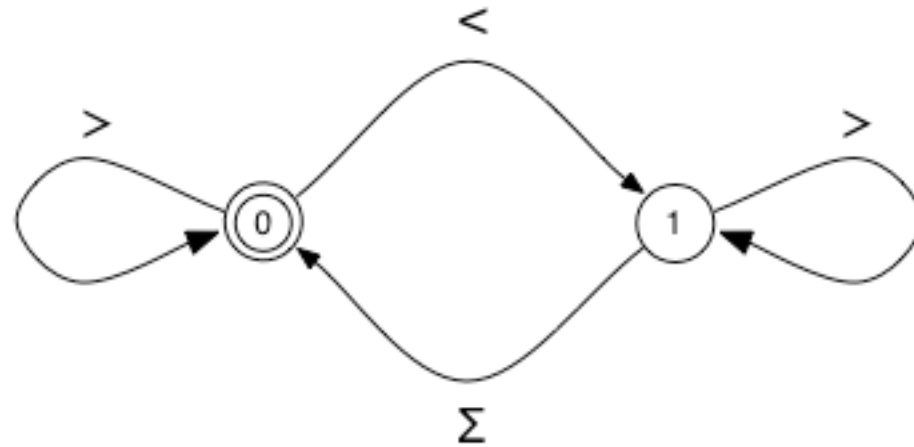
$LC(\lambda, \ell, r) = P\text{-iff-}S((\Sigma^* \lambda)_{\ell} - (\Sigma^*_{\ell} \ell), < \Sigma^*_{\ell})_r$

$LeftContext(\lambda) = LC(\lambda, <, >)$

$RightContext(\rho) = Reverse(LC(Reverse(\rho), >, <))$

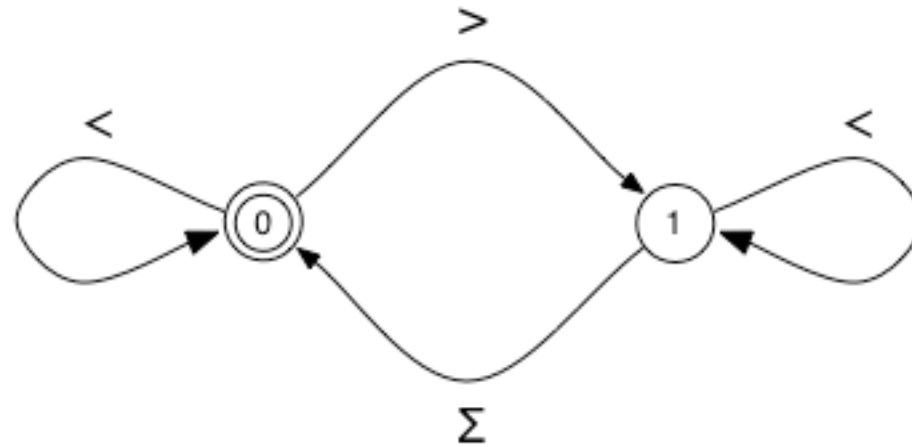
# The Empty Left Context

*LeftContext( $\epsilon$ )*



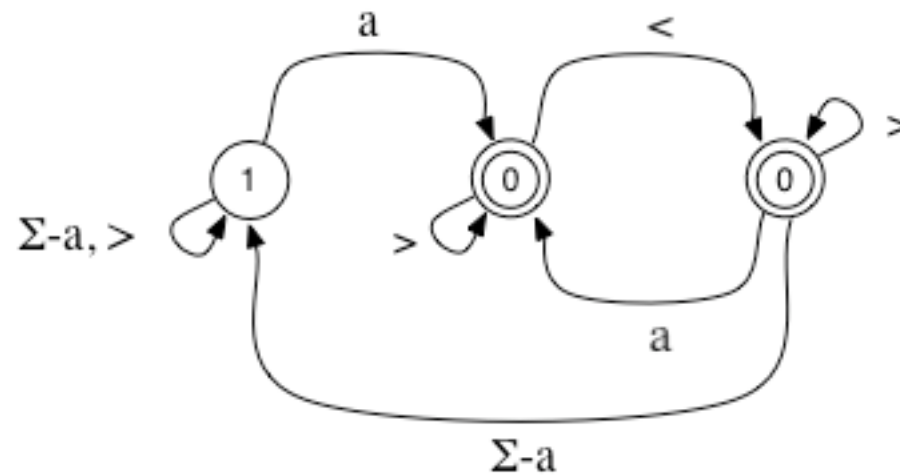
# The Empty Right Context

*RightContext( $\epsilon$ )*



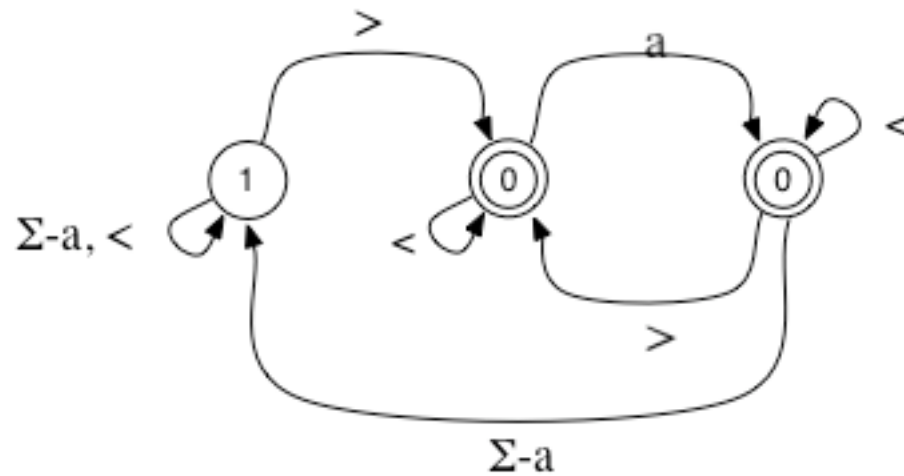
# One-character Left Context

*LeftContext(a)*



# One Character Right Context

*RightContext(a)*



$\alpha \rightarrow \beta / \lambda \_ \rho$     **Optional**

**Left to Right**

*Intro*({<, >}) ◦

*Id*(*RightContext*( $\rho$ )) ◦

*Replace*( $\alpha$ ,  $\beta$ ) ◦

*Id*(*LeftContext*( $\lambda$ )) ◦

*Intro*<sup>-1</sup>({<, >})

**Right to Left**

*Intro*({<, >}) ◦

*Id*(*LeftContext*( $\lambda$ )) ◦

*Replace*( $\alpha$ ,  $\beta$ ) ◦

*Id*(*RightContext*( $\rho$ )) ◦

*Intro*<sup>-1</sup>({<, >})

# Optional Simultaneous Application

*Intro*({<, >}) ◦

*Id*(*RightContext*( $\rho$ )  $\cap$  *LeftContext*( $\lambda$ )) ◦

*Replace*( $\alpha$ ,  $\beta$ ) ◦

*Intro*<sup>-1</sup>({<, >})

# Obligatory rules—Brackets

$\langle_a \rangle_a$       application

$\langle_i \rangle_i$       ignored

$\langle_c \rangle_c$       center

**Whenever a rule is properly applied, the input side of the replacement relation will contain a substring of the form**

$$\langle_a \alpha_{\langle_c \rangle_c} \rangle_a$$

**Notation:**

$$\langle = \{ \langle_a, \langle_i, \langle_c \}$$

$$\rangle = \{ \rangle_a, \rangle_i, \rangle_c \}$$

# Obligatory Rules

$$\text{Obligatory}(\phi, l, r) = \overline{\Sigma_{m,0}^* l \phi_m^0 r \Sigma_{m,0}^*}$$

Left to Right

*Prolog*

*Id(Obligatory( $\phi$ ,  $\langle_i, \rangle$ ))*

*Id(RightContext( $\rho, \langle, \rangle$ ))*

*Replace ( $\phi$ ,  $\psi$ )*

*Id(LeftContext( $\lambda, \langle, \rangle$ ))*

*Prolog<sup>-1</sup>*

Right to Left

*Prolog*

*Id(Obligatory( $\phi$ ,  $\langle, \rangle_i$ ))*

*Id(LeftContext( $\lambda, \langle, \rangle$ ))*

*Replace( $\phi$ ,  $\psi$ )*

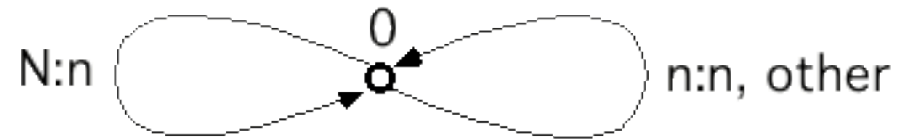
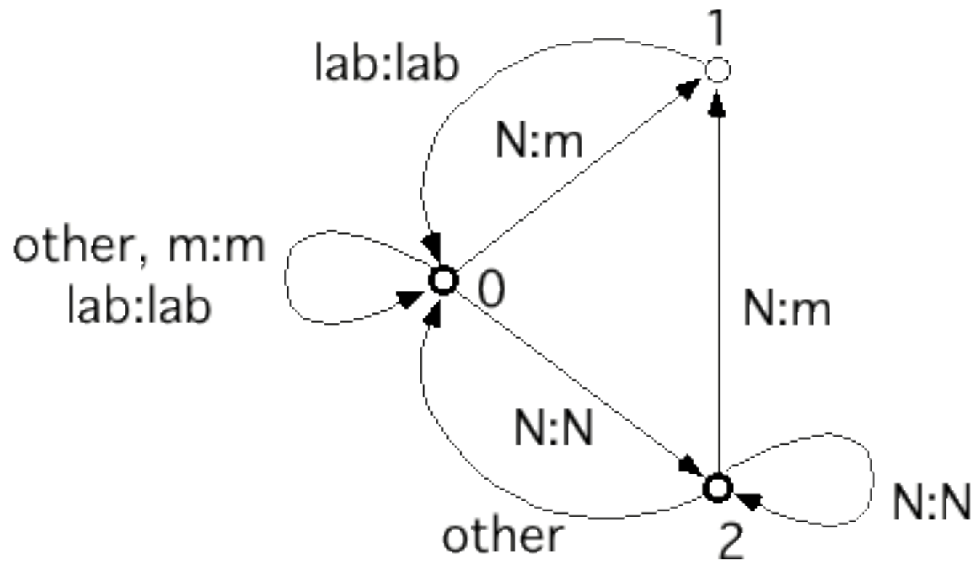
*Id(RightContext( $\rho, \langle, \rangle$ ))*

*Prolog<sup>-1</sup>*

# Other Topics

- **Boundary Contexts**
- **Features**
- **2-level Rules**

# Composition



i N t r a c t a b l e  
 0 0 2 0 0 0 0 0 0 0 0  
 i N t r a c t a b l e  
 0 0 0 0 0 0 0 0 0 0 0  
 i n t r a c t a b l e

i N p r a c t i c a l  
 0 0 1 0 0 0 0 0 0 0 0  
 i m p r a c t i c a l  
 0 0 0 0 0 0 0 0 0 0 0  
 i m p r a c t i c a l

# Lexical and Surface Forms

intractable → intractable

input → input

iNpractical → impractical

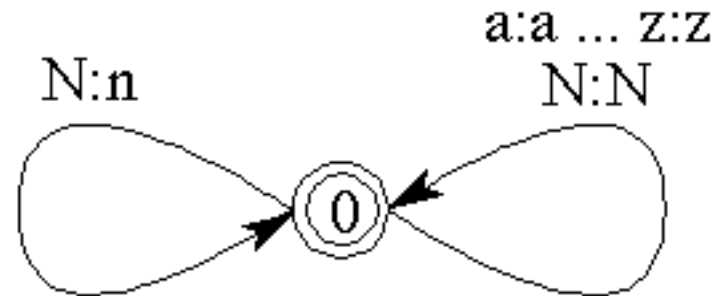
1. N → m / \_\_ [+labial]

2. N → n

*Archiphonemes*

# An Optional Rule

$N \rightarrow n$  (Optional)



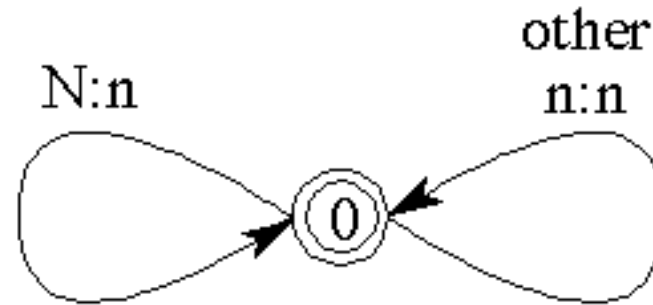
**Start state: 0**

**Usually leftmost in the diagram**

**Final states have double circles**

# An Obligatory Rule

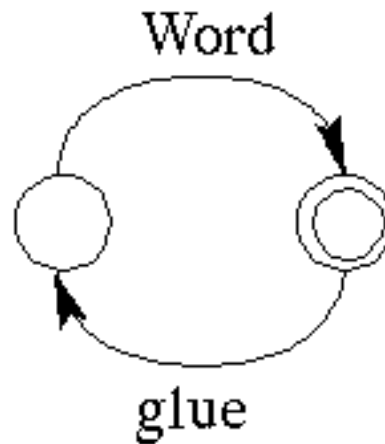
$N \rightarrow n$  (Obligatory)



**other = all matched pairs of symbols not otherwise mentioned on any transition.**

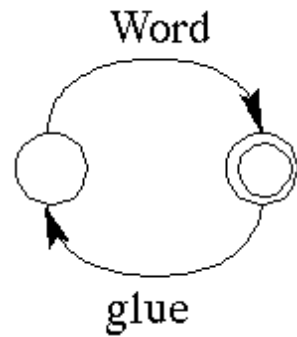
# Context

$N \rightarrow m / \_ [+labial]$  (Obligatory)

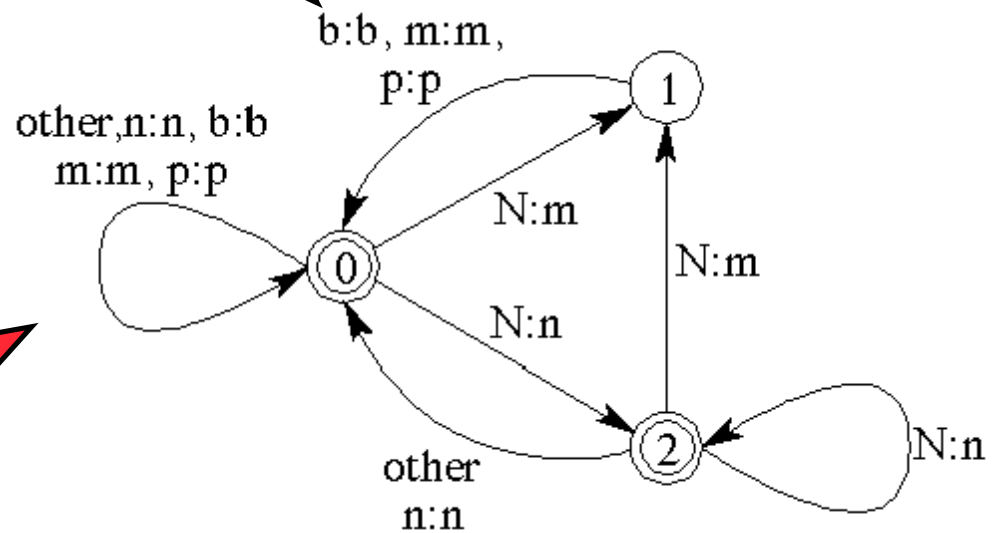
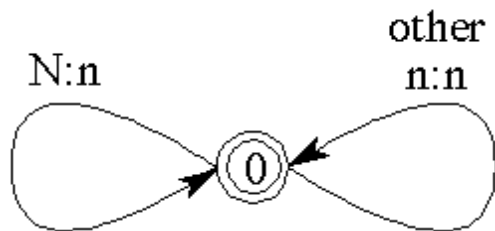


← No exit from this state except over a labial.

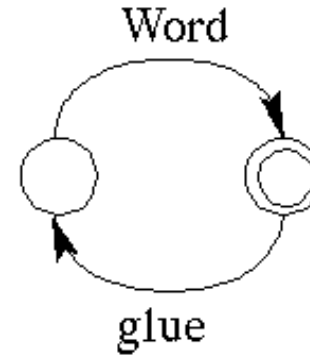
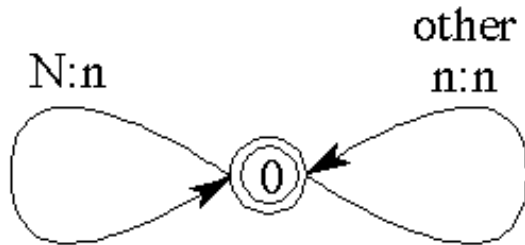
# Composition



**Rules 1 and 2**

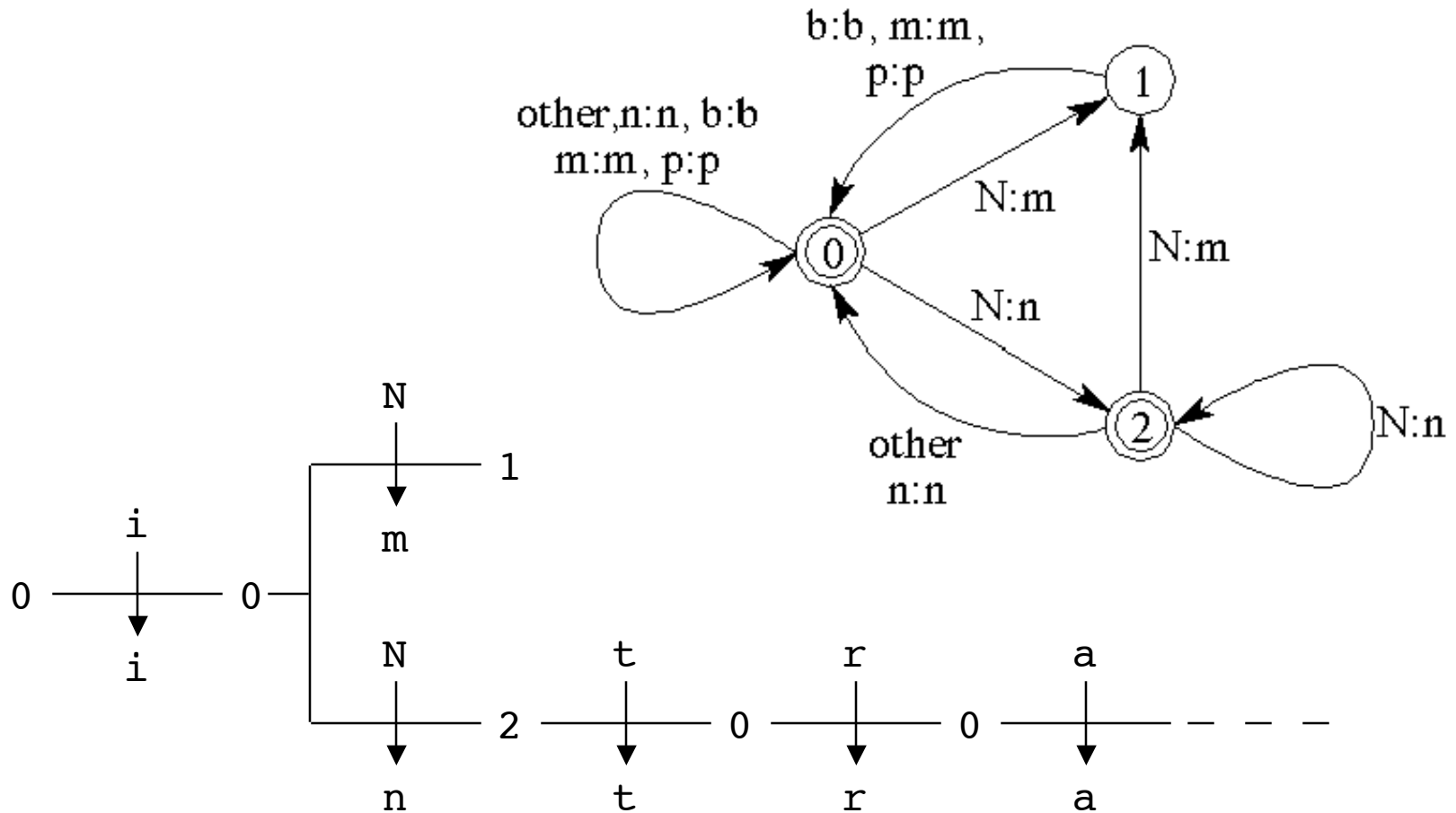


# Composition

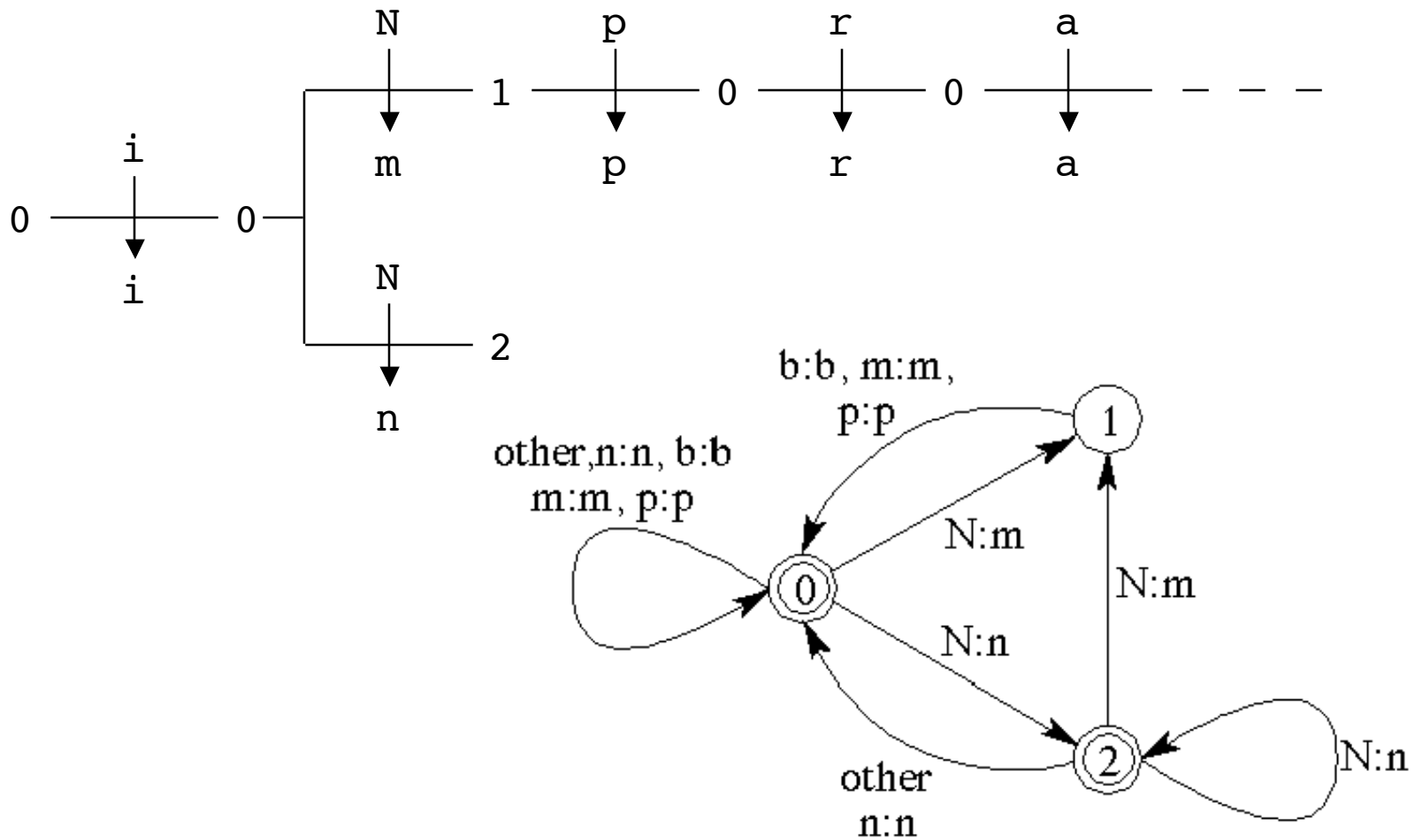


input	<i>m</i> -machine	output		
		input	<i>n</i> -machine	output
<b>N</b>	<b>labial follows</b>	<b>m</b>		<b>m</b>
<b>N</b>	<b>nonlabial follows</b>	<b>N</b>		<b>n</b>

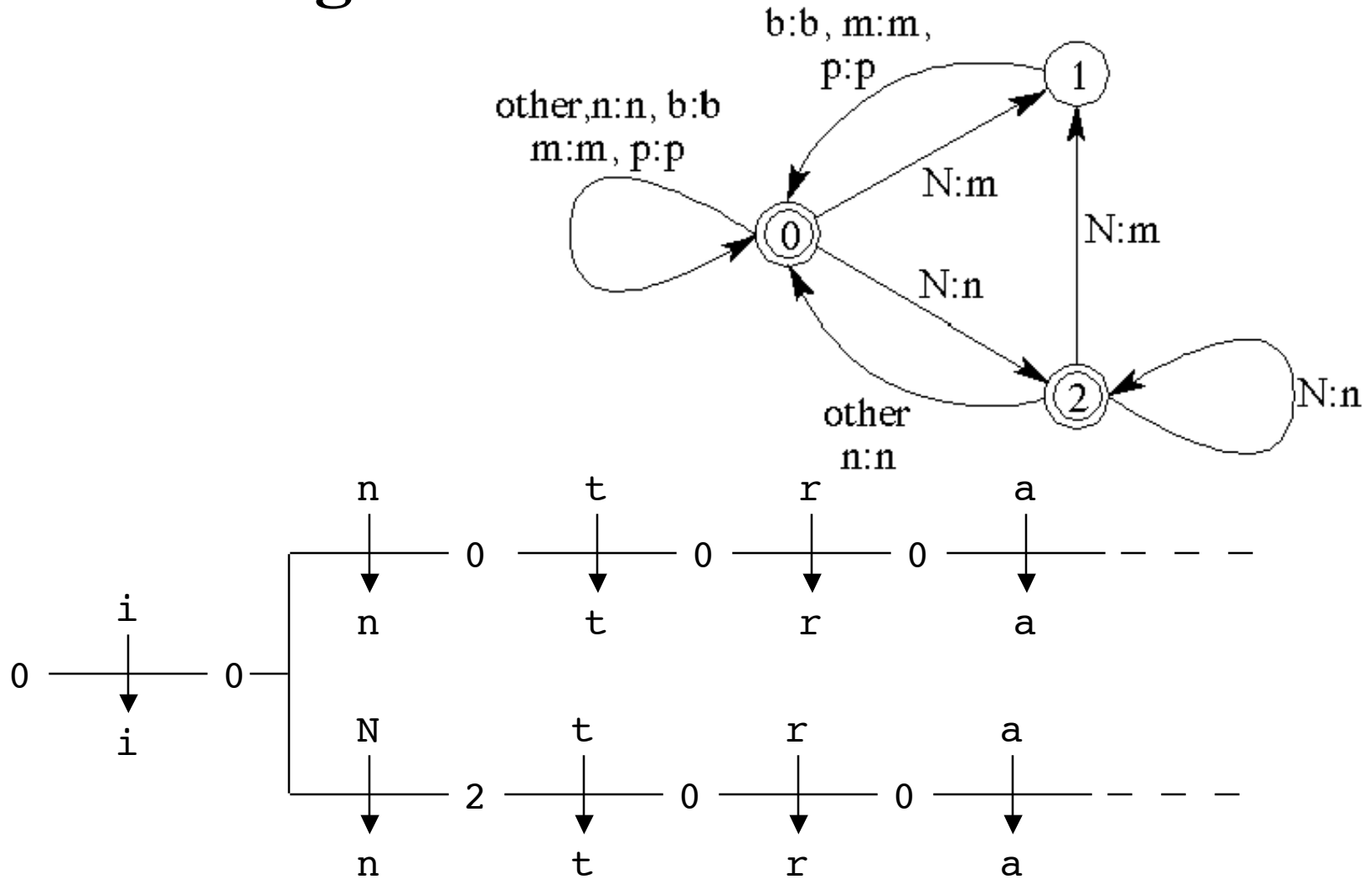
# Generation — “intractable”



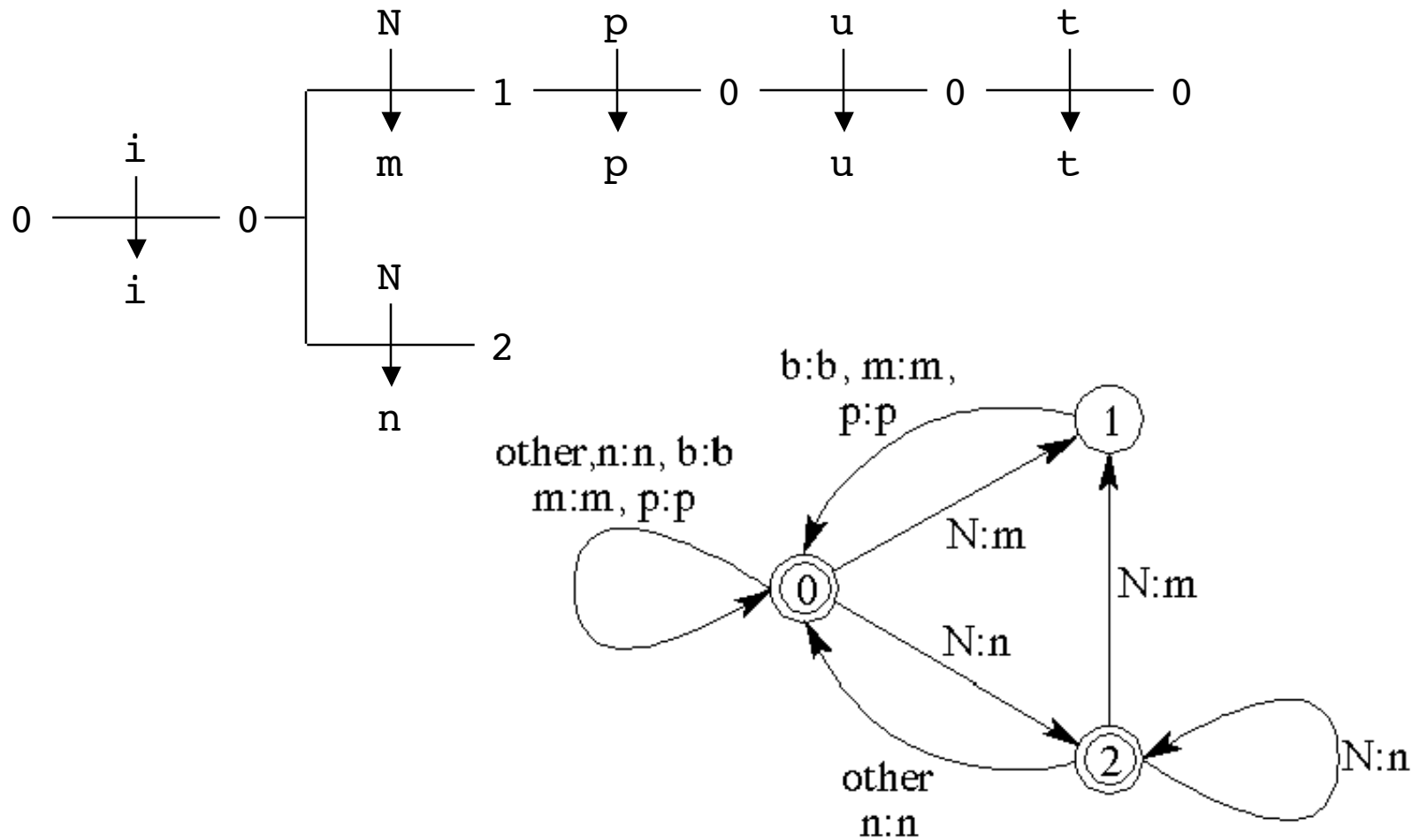
# Generation — “impractical”



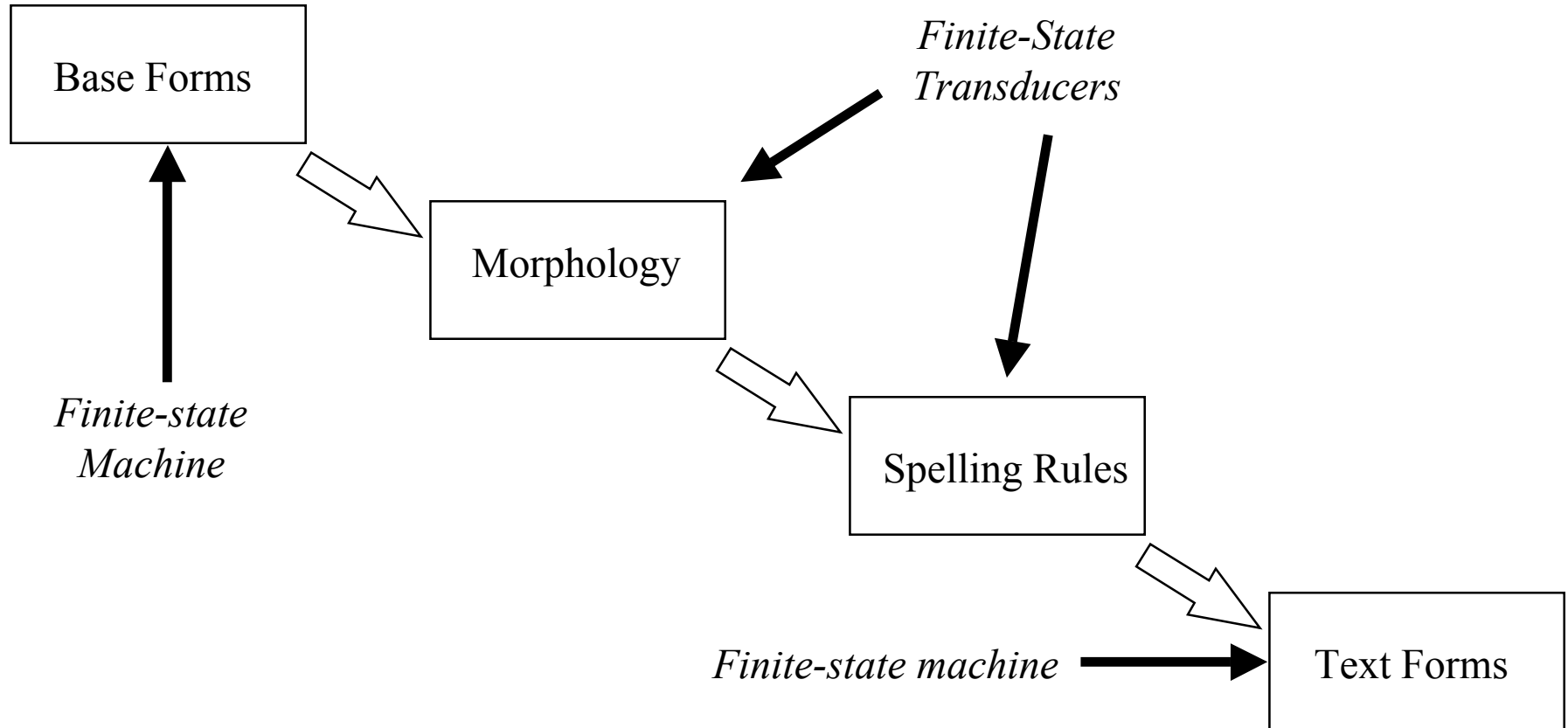
# Recognition — “intractable”



# Generation — “input”

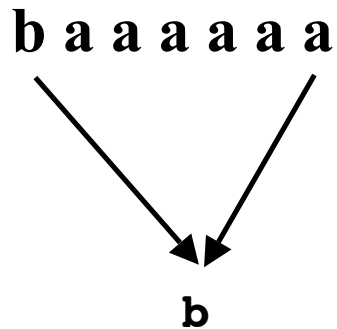


# A Word Transducer



# Context Reuse

$a \rightarrow \varepsilon / b \_$



$$\phi^0 = \begin{cases} \phi & \text{if } \varepsilon \notin \phi \\ [\phi - \varepsilon] \cup 0 & \text{otherwise} \end{cases}$$

**We need a separate left-context marker for each application of the rule. But they all have to be in the same place.  
Solution: Temporarily replace a with 0, and delete this later.**

# Epenthesis and Deletion

$$LC(\lambda, l, r, a) = P\text{-iff-}S((\Sigma^* \lambda)_{l,a} - (\Sigma^*_l l), l \Sigma^*_r)$$

$$\text{LeftContext}(\lambda) = LC(\lambda, <, \{>_i, >_c\}, >_a)$$

$$\text{RightContext}(\rho) = \text{Reverse}(LC(\text{Reverse}(\rho), >, \{<_i, <_c\}, <_a))$$

# Other Topics

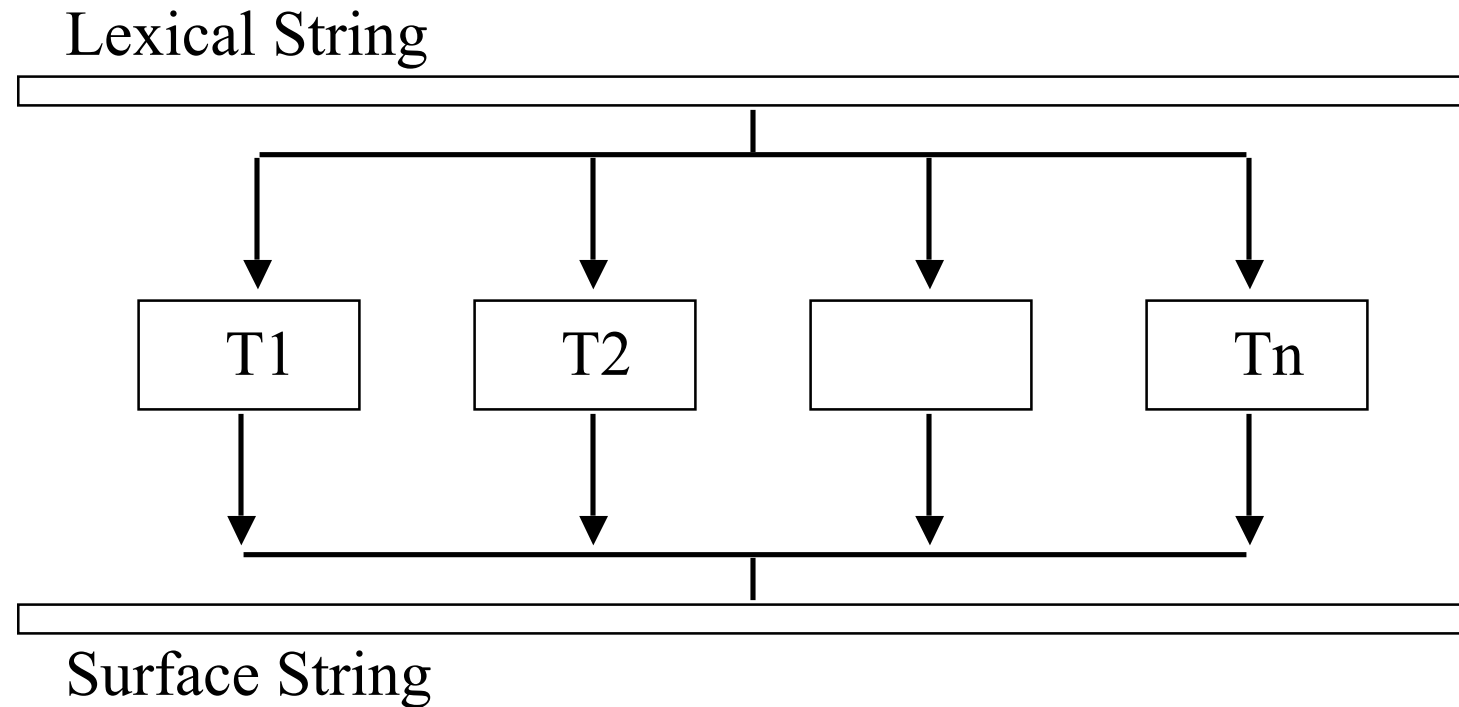
- **Boundary Contexts**
- **Batch Rules**
- **Features**

# Theorem

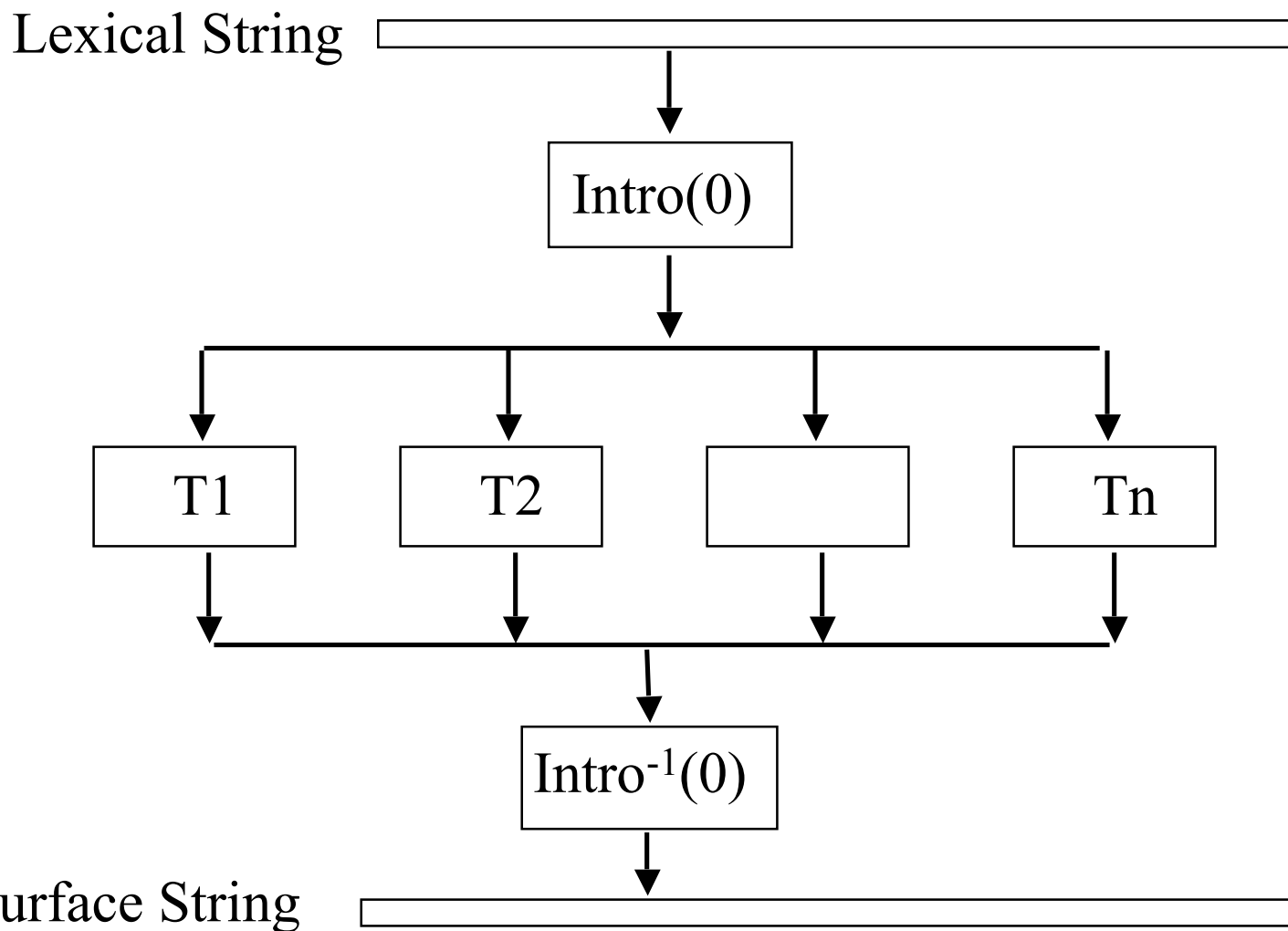
- **Every rewriting grammar denotes a regular relation.**
- **Every regular relation is denoted by some rewriting grammar.**

# Two-level Rule Systems

# Parallel Combination



# Parallel Combination



# Mathematically ...

$$\mathbf{Z} = \text{Intro}(\mathbf{0})$$

$$\mathbf{Z} \circ [\bigcap_i \mathbf{T}_i] \circ \mathbf{Z}^{-1} \quad (\text{Regular})$$

$$\neq \bigcap_i [\mathbf{Z} \circ \mathbf{T}_i \circ \mathbf{Z}^{-1}] \quad (\text{May not be regular})$$

# Spies

**s p y + s #**

**s p y 0 s 0**

**s p y + s #**

**s p i 0 s 0**

**s p y + 0 s #**

**s p y 0 e s 0**

**s p y + 0 s #**

**s p i + e s 0**

$\Pi = a:a \dots z:z, 0:e, y:i, +:0, #:0$

$Sib = \{s, x, z\}$

$0:e \iff (Sib: \text{or } y:) +: \_ s:s \#$

$y:y \iff \_ #: \text{or } +: i: \text{or } \Pi - \{+:. #: \}$

# Two-level Rules

$\tau \Rightarrow \lambda \_ \rho$

**Context restriction**

$\tau, \lambda, \rho$  in  $\Pi^*$

If a  $\tau$  pair-string appears,  $\lambda$  is before and  $\rho$  is after.

$\tau \Leftarrow \lambda \_ \rho$

**Surface Coercion**

If the first component of a pair-string appearing between  $\lambda$  and  $\rho$  is in  $\text{Domain}(\tau)$ , that pair-string must be in  $\tau$ .

$\tau \Leftrightarrow \lambda \_ \rho$

**Bicondition**

$\tau \Rightarrow \lambda \_ \rho$  and  $\tau \Leftarrow \lambda \_ \rho$

$\tau / \Leftarrow \lambda \_ \rho$

**Surface Prohibition**

If the first component of a pair-string appearing between  $\lambda$  and  $\rho$  is in  $\text{Domain}(\tau)$ , that pair-string must **NOT** be in  $\tau$ .

# Observations

- $\tau, \lambda$  and  $\rho$  in  $\Pi^*$  are same-length relations
- Therefore, closed under intersection and complementation
- Therefore, we can define

$$\text{If-P-then-S}(R_1, R_2) = \Pi^* - R_1 (\Pi^* - R_2)$$

$$\text{If-S-then-P}(R_1, R_2) = \Pi^* - (\Pi^* - R_1) R_2$$