Finite State

Rewriting Rules

Martin Kay

Stanford University and
The University of the Saarland

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**
- **Kaplan, Ronald M and Martin Kay. "Regular Models for Phonological Rules Systems. "** ***Computational Linguistics*** **vol 20, 1994 pp. 331-378.**

Rewriting Rules

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

No rerewriting

$x \rightarrow axb$ would map x to $a^n x b^n$

and, unless the rule were optional

$$n = \infty$$

C. Douglas Johnson

Formal Aspects of Phonological Description **(1972)**

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

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

tie}ing#
tie}ing#
tie}ing#
ti}ing#
ty}ing#

Application Direction—Examples

V: → V / V: C*

Gidabal

left to right

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

Slovak

right to left or simultaneous

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

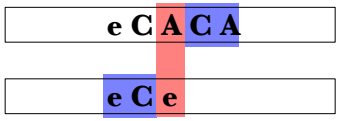
v o l + a: v + a m e

we call often

Direction of Application

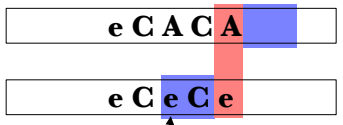
A → e / eC _

Left-to-right



A → e / eC _

Left-to-right

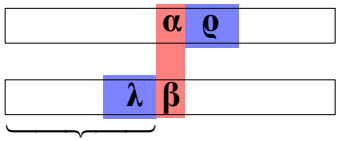


Context provided by this same rule!

α → β / λ _ ϱ

Left-to-right

Not yet affected by the rule

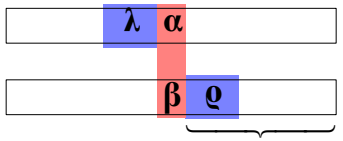


Already affected by the rule

α → β / λ _ ϱ

Right-to-left

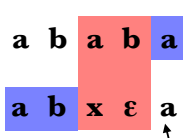
Already affected by the rule



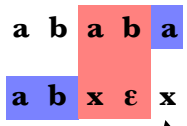
Not yet affected by the rule

a → x / ab _ a

Left-to-right



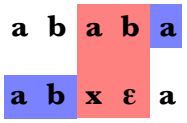
If the rule does not apply here



If the rule does apply here

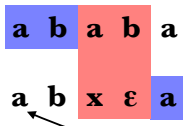
a → x / ab _ a

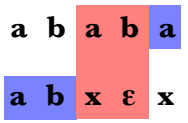
Left-to-right



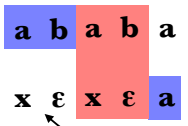
If the rule does not apply here

Right-to-left



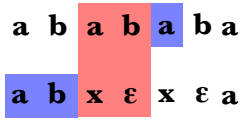


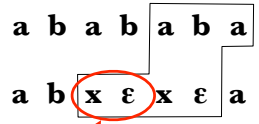
If the rule does apply here



a → x / ab _ a

Left-to-right

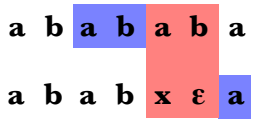


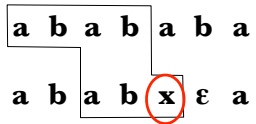


No match

a → x / ab _ a

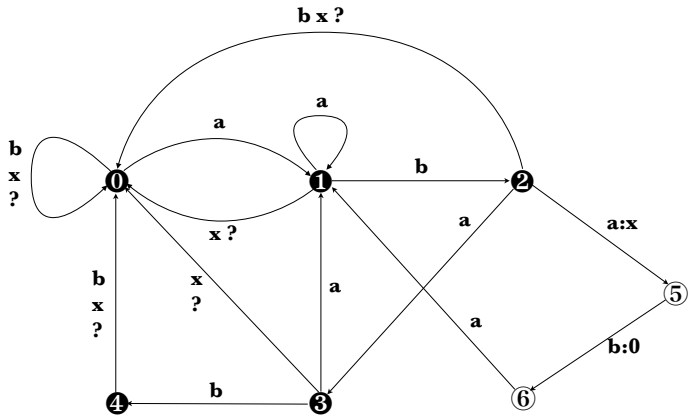
Right-to-left



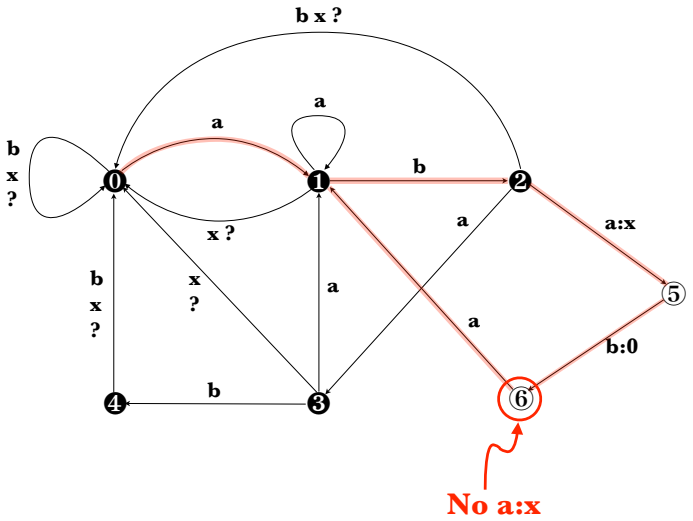


No match

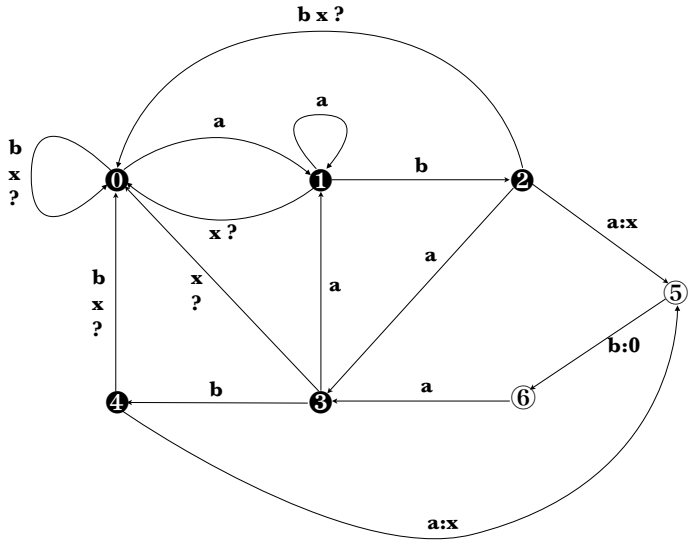
a b → x / ab _ a *Left-to-right*



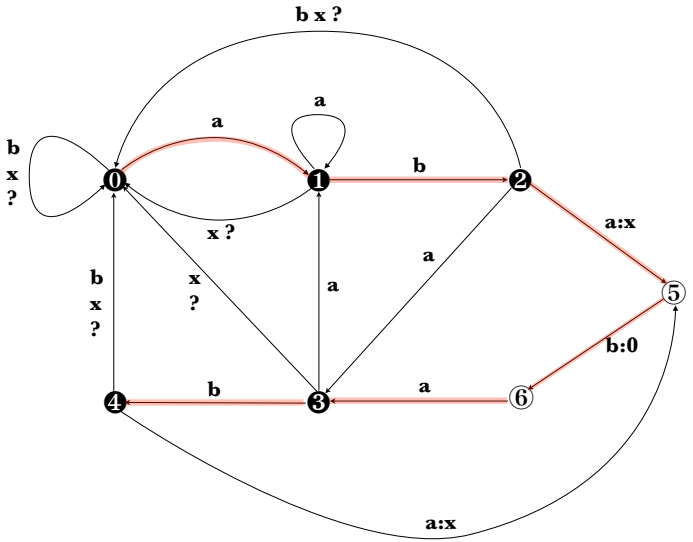
a b → x / ab _ a *Left-to-right*



a b → x / ab _ a *Right-to-left*

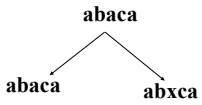


a b → x / ab _ a *Right-to-left*



Optional Rules

$Opt(a \rightarrow b\_)$

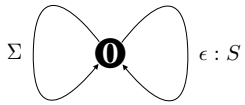


Most rules, and all optional rules, will produce at least one output string, perhaps just a copy of the input if the conditions for application are nowhere satisfied. Optional rules typically produce many outputs.

Special Operators

Intro

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



Insert S (**not in** Σ) **anywhere and everywhere.**

Example:

$\Sigma = \{a, b\}$ $S = \{\$\}$

Intro(S) contains $< a, a >$
$< a, \$a >$
$< a, a\$\$\$ >$
$< ab, \$a\$\$\$b\$\$ >$

Ignore

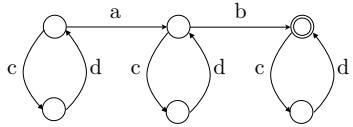
$$L_S =_{df} \text{Range}(\text{Id}(L) \circ \text{Intro}(S))$$

L **ignoring** S

The strings of L_S **differ from those of** L **in that occurrences of symbols in** S **may be freely interspersed.**

Includes only strings that would be in L **if some occurrences of symbols in** S **were ignored.**

ab_{cd} — ***ab*** **ignoring** ***cd***



If-P-then-S

If prefix then suffix

$\{x \mid$ for every partition x_1x_2 of x, if $x_1 \in L_1$ then $x_2 \in L_2\}$

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

If-S-then-P

If suffix then prefix

$\{x \mid$ for every partition x_1x_2 of x, if $x_2 \in L_2$ then $x_1 \in L_1\}$

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

Iff-S-then-P

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

Provisional definition

Keep going

Skip to application point

$$\text{Replace} = [\overbrace{\text{Id}(\Sigma^*)}\text{Opt}(\underbrace{\phi \times \psi}_{\textbf{Cartesian product}})]^*$$

Provisional definintion

$$\text{Replace} = [\text{Id}(\Sigma^*)\text{Opt}(\phi \times \psi)]^*$$

Center is optional because there may be no eligible substrings, or the rule may be optional.

Following does not work because contexts and centers can overlap

$$\text{Replace} = [\text{Id}(\Sigma^*)\text{Opt}(\text{Id}(\lambda)\phi \times \psi\text{Id}(\rho))]^*$$

Auxiliary Symbols

Introduce a < after each occurrence of the left context and > before each occurrence of the right context.

<i<N< >p<r <o< >b<a <>b<l<e<

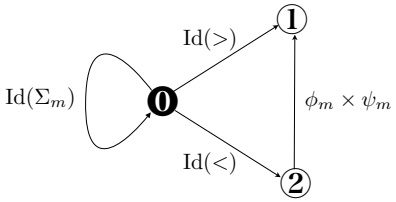
<i<N<t<r<a<c<t<a< >b<l<e<

Empty left context, so < everywhere

Reft context, so > before b **and** p

No Context

$$\text{Replace} = [\text{Id}(\Sigma_m^*)\text{Opt}(\overbrace{\text{Id}(<)}^{\text{empty left}}\phi_m \times \psi_m\overbrace{\text{Id}(>)}^{\text{empty right}})]^*$$



Suppose left context = λ=ab* and input is

... a b<a b<c ...

λ

λ

Context markers *inside* contexts!

Strategy

- **Put context markers everywhere**
- **Filter out the ones that are not where they should be.**

$$Prologue = Intro(m)$$

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

Strategy

- **Put context markers everywhere**
- **Filter out the ones that are not where they should be.**

$$Prologue = Intro(m)$$

$$\text{P-iff-S}(\Sigma^*\lambda, < \Sigma^*) \rightarrow \text{P-iff-S}(\Sigma^*_{<}\lambda_{<}, < \Sigma^*_{<})_{>}$$

Ignore *other* left markers

Ignore *all right* markers

Not so fast!

$$\text{P-iff-S}(\Sigma^*_< \lambda_<, < \Sigma^*_<)_>$$

but

$$(\lambda_< <) \in \lambda_<$$

so

$$\mathit{Leftcontext}(\lambda, l, r) = \mathit{P\text{-}iff\text{-}S}(\Sigma^*_l \lambda_l - \Sigma^*_l l, l\Sigma^*_l)_r$$

Begins with a marker

Does not end with a marker

$$\mathit{Leftcontext}(\lambda, l, r) = \mathit{P\text{-}iff\text{-}S}(\overbrace{\Sigma^*_l \lambda_l - \Sigma^*_l l}, \overbrace{l\Sigma^*_l})_r$$

Either ϕ or ψ could include ε.

$$a \rightarrow \epsilon / b\_$$

b a a ... a a a ➜ b

need more <'s

So replace a's with 0's

Then delete 0's

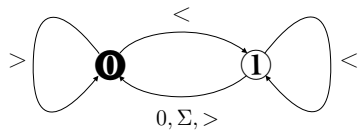
$$\phi^0 = \begin{cases} \phi & \text{if } \epsilon \notin \phi \\ (\phi - \epsilon) \cup 0 & \text{otherwise} \end{cases}$$

$$Replace = [Id(\Sigma^*_{m,0})\, Opt(Id(<)\phi^0_m \times \psi^0_m Id(>))]$$

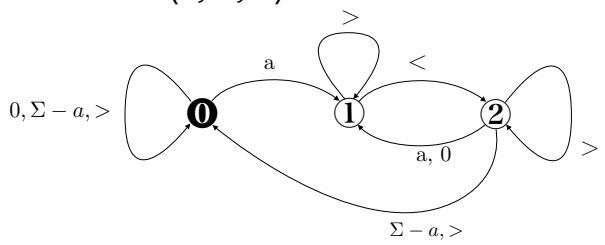
$$Leftcontext(\lambda, l, r) = P\text{-}iff\text{-}S(\Sigma^*_{l,0}\lambda_{l,0} - \Sigma^*_{l,0}l, l\Sigma^*_{l,0})$$

Empty left context

- **Contains at least one <**
- **Every 0, Σ and < followed by a <**



Leftcontext(a, <, >)



$$Rightcontext(\rho, l, r) = P\text{-}iff\text{-}S(\Sigma^*_{0,r}\rho^*_{r,0} - r\Sigma^*_{l,0}l, l\Sigma^*_{l,0})_l$$

$$Rightcontext(\rho, l, r) = Rev(Leftcontext(Rev(\rho), r, l))$$

Left-to-right optional rule

Prolog ∘
Id(RightContext(ρ, <, >)) ∘
Replace ∘
Id(LeftContext(λ, <, >)) ∘
Prolog^{-1}

Right-to-left optional rule

Prolog ∘
Id(LeftContext(λ, <, >)) ∘
Replace ∘
Id(RightContext(ρ, <, >)) ∘
Prolog^{-1}

Simultaneous optional rule

$$\text{Prolog} \circ \text{Id}(\text{RightContext}(\rho, <, >)) \cap \text{Id}(\text{LeftContext}(\lambda, <, >)) \circ \text{Replace} \circ \text{Prolog}^{-1}$$

Obligatory Rules

Reject string pairs containing sites where the conditions of application are met but the replacement is not carried out.

Proposal:

$$\overline{Id(\Sigma^*_{m,0})Id(<)\phi^0_m \times \overline{\psi^0_m}Id(>)(\Sigma^*_{m,0})}$$

- **Contains *complement* of a regular relation.**
- **We would have to *intersect* it with the replacement relation.**

Roles of Context brackets

- **Start (end) an application: $<_a$ $>_a$**
- **Ignored in identity portions: $<_i$ $>_i$**
- **In the center of an application: $<_c$ $>_c$**

When a rule is properly applied, the input side will contain a substring of the form

$$<_a \phi^0_{<_c>_c} >_a$$

Roles of Context brackets

$$<_a \phi^0_{<_c>_c} >_a$$

If an obligatory left-to-right rules fails to apply ϕ^0 will be marked $<_i$ instead of $<_a$.

From now on, we take < and > to be cover symbols:

$< = \{<_i, <_a, <_c\}$
$> = \{>_i, >_a, >_c\}$ and
$m = < \cup >$

$$Obligatory(\phi, l, r) = \overline{\Sigma^*_{m,0}\ l\ \phi^0_m\ r\ \Sigma^*_{m,0}}$$

Left-to-right

$$Obligatory(\phi, <_i, >)$$

Right-to-left

$$Obligatory(\phi, <, >_i)$$

One last time:

$$Replace = [Id(\Sigma^*_{<_i,>_i,0})Opt(Id(<_a)\phi^0_{<_c>_c} \times \psi^0_{<_c>_c}Id(>_a))]*$$

Left-to-right optional rule

Prolog ∘
Id(Obligatory(φ, <ᵢ, >**)** ∘
Id(RightContext(ρ, <, >**))** ∘
Replace ∘
Id(LeftContext(λ, <, >**))** ∘
Prolog^{-1}

Right-to-left optional rule

Prolog ∘
Id(Obligatory(φ, <, >ᵢ**)** ∘
Id(LeftContext(λ, <, >**))** ∘
Replace ∘
Id(RightContext(ρ, <, >**))** ∘
Prolog^{-1}