Intro to Finite-State Transducers

Jean Mark Gawron
San Diego State
January 23, 2016

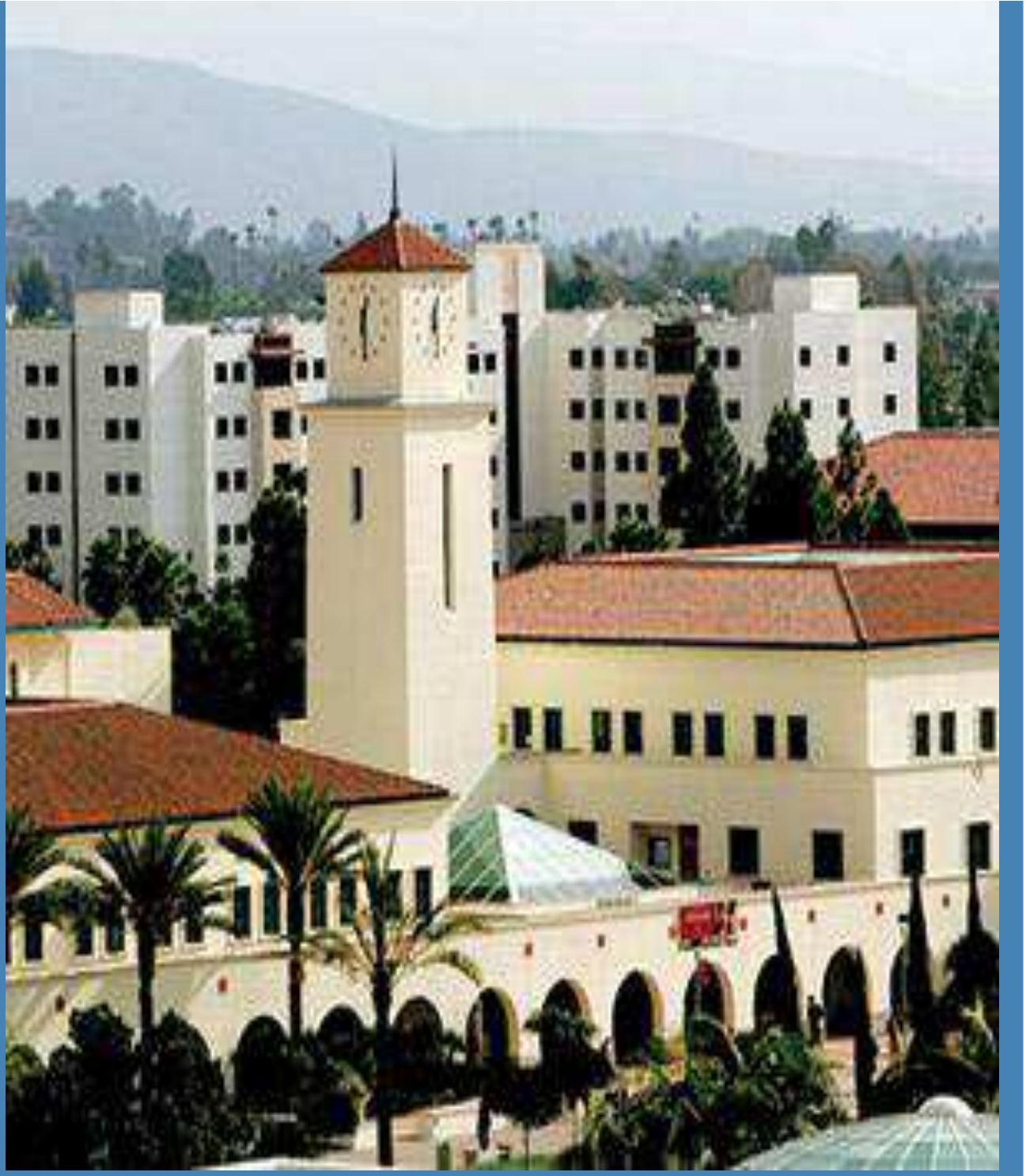
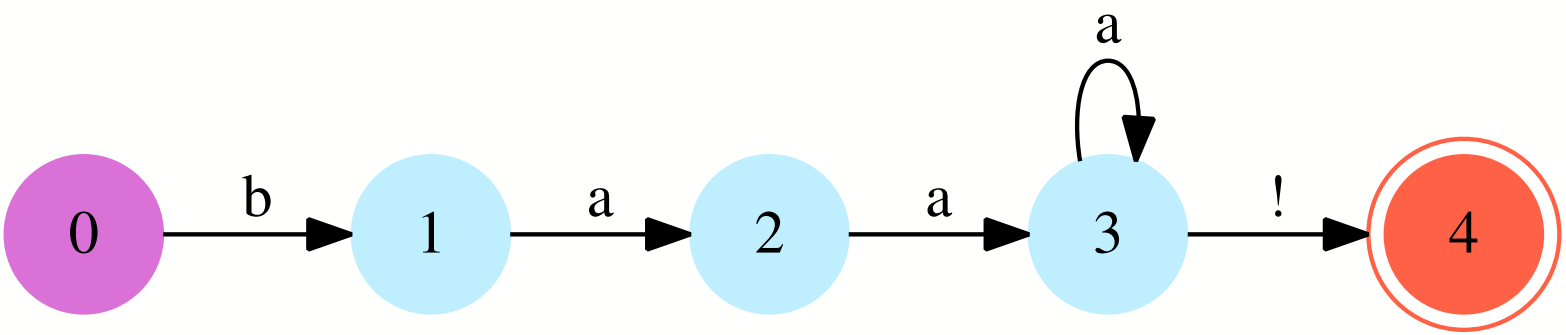


Table of contents

FSAs

An FSA defines a set of strings.

It is primarily used to accept and reject strings.

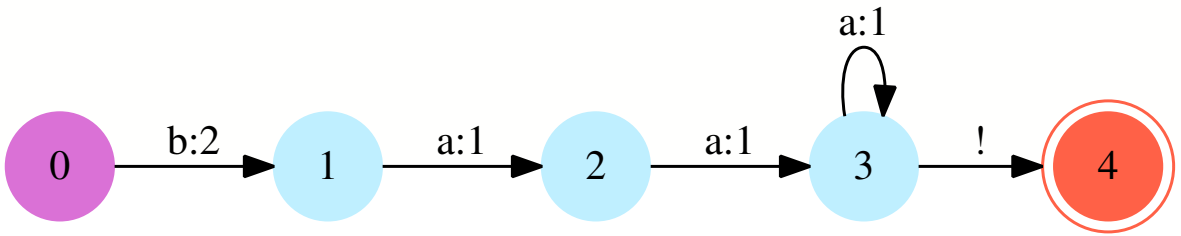


Sheep Language

FSTs

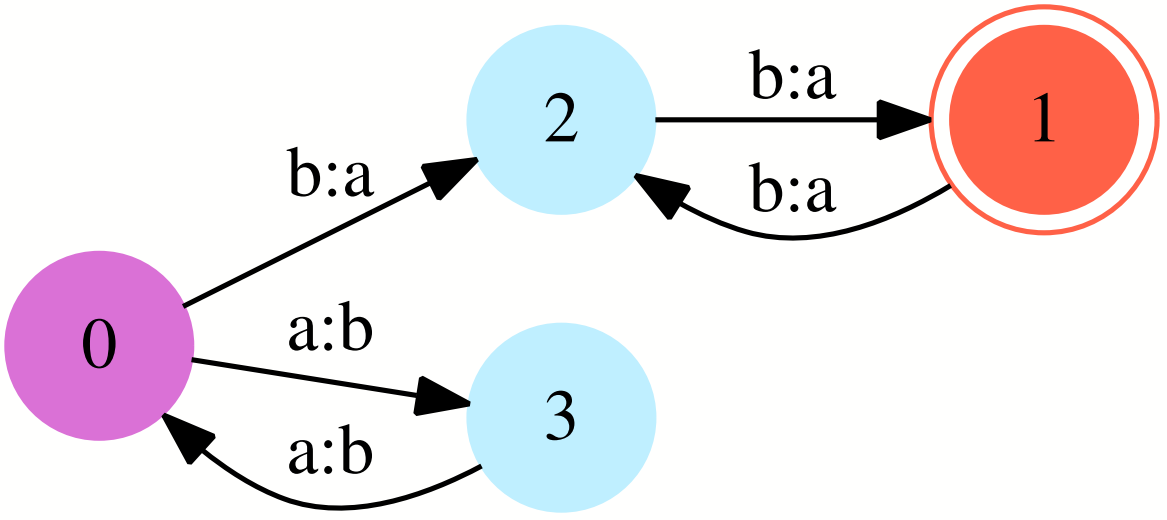
Instead of defining sets of strings, FSTs defines sets of PAIRS of strings. A set of pairs in set theory is called a **relation**.

FSTs define relations on strings. Not all string relations can be captured by FSTs (as we'll see). The set that can be captured is called the **regular relations**.



Sheep Language

AB swapper



AB swapper

Given a string that consist of an even number of a's followed by an even number of b's, it will swap the a's and b's.

Upper/Lower alphabets & languages

1. Two alphabets in the sheep language transducer:

$$\begin{array}{ll} \text{Upper} & \{a, b, !\} \\ \text{Lower} & \{1, 2, !\} \end{array}$$

2. In the A and B swapper. the upper and lower alphabets are the same.
3. An FST also has two languages, an upper and lower language. In the sheep language transducer, the upper language is $baa^{+}!$, and the lower language is $21^{+}!$.
4. The FST defines a correspondence between members of the lower and upper alphabets, so there's also a set of **feasible pairs**, giving the possible pairings of upper-alphabet symbols with lower-alphabet symbols ({a:1, b:2, !:!}).

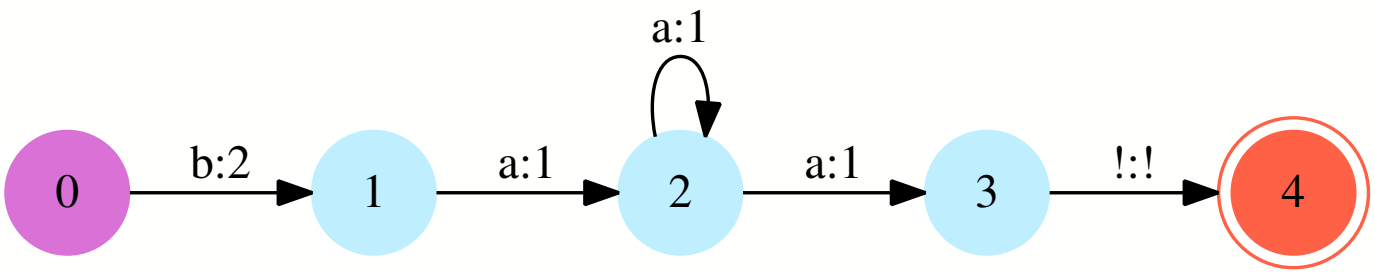
An FSA with two tapes

We call the two tapes the upper and lower tapes.

1. As **recognizers**, FSTs read both tapes and check to see that the strings on the two tapes stand in the rleation described by the machine.
2. As **analyzers**, they read the lower tape and find all possible upper strings that stand in the relation described by the machine. In our terms they find all analyses of a surface string.
3. As **generators**, they read the upper tape and find all possible lower strings that stand in the relation described by the machine. In our terms they find all possible realizations of an underlying string.

Nondeterminism

AS with FSAs, FSTs can be deterministic or non-deterministic.



Sheep Language

ϵ-transitions

ϵ:a	An "a" on the lower tape that does not corresponds to anything on the upper tape (insert an "a" on the lower tape).
a:ϵ	The "a" on the upper tape does not correspond to anything on the lower tape. (omit the "a" on the lower tape)
ϵ : ϵ	Jump to another state without reading either tape.

Any of these types of transitions is quite problematic if it loops from state *i* to state *i*. Why?

Closure properties

From Jurafsky and Martin, Ch 2, we learned FSAs are closed under:

1. Union
2. Intersection
3. Concatenation
4. Complementation
5. Reversal

FSTs have only a a subset of these closure properties:

1. Union
2. Concatenation

Closure example

$L_1 = \{\langle a^n, b^* c^n \rangle \mid n > 0\}$ A relation that pairs n underlying a's with any number of b's followed by exactly n c's.

$L_2 = \{\langle a^n, b^n c^* \rangle \mid n > 0\}$ A relation that pairs n underlying a's with exactly n b's followed by any number of c's.

$$L_1 \cap L_2 = \{\langle a^n, b^n c^n \rangle \mid n > 0\}$$

The lower language is not regular, so the relation cannot be regular.

Chomsky Hierarchy

Type	Name	Automaton	Rule Type
0	Recursive enumerable	Turing	Any rewrite rule
1	Context sensitive	LBND Turing	$A \rightarrow B\ C\_\_\_\_ D$
2	Context-Free	ND PDA	$A \rightarrow BC \ldots$
3	Regular	FSA	$A \rightarrow aB$, $A \rightarrow Ba$

Examples

$(ab)^n$	Regular
$a^n b^n$	Context-free
XX^R	Mirror-language: Context-free
$a^n b^n c^n$	Context-sensitive

Phonological/morphological rewrite rules

$A \rightarrow B \ C\_\_\_\_ D$

$$\epsilon \rightarrow ab \backslash \_\_\_\_ b$$

Applied to input ab, generates $a^n b^n$.

Johnson's 1972 insight

The rule generates a regular language **only if** it is allowed to reapply endlessly in the same position. But phonologists never had this interpretation in mind. Rules always drift rightward in string after every application.

Rule application regime

in-place	Drift right
a ↑ b	a ↑ b
aa ↑ bb	aab ↑ b
aaa ↑ bbb	aabab ↑ b
aaaa ↑ bbbb	aababab ↑ b

The drifting application regime produces the regular language $a(ab)^n b$

Johnson's result

Johnson showed that if the rules were only used under the drifting application regime, the languages produced were regular. (Also see Karttunen 1993)

Rules describe (regular) relations

A $\rightarrow$ B C____ D
CAD $\rightarrow$ CBD

In	Out	
cad	cbd	yes
cad	cad	no
cae	cae	yes
cae	cbe	yes

Approximate rule translation

a $\rightarrow$ b \ L ___ R
Id(Σ^*) (Opt(Id(L) $a \times b$ ID(R)))*
Kaplan and Kay 1994

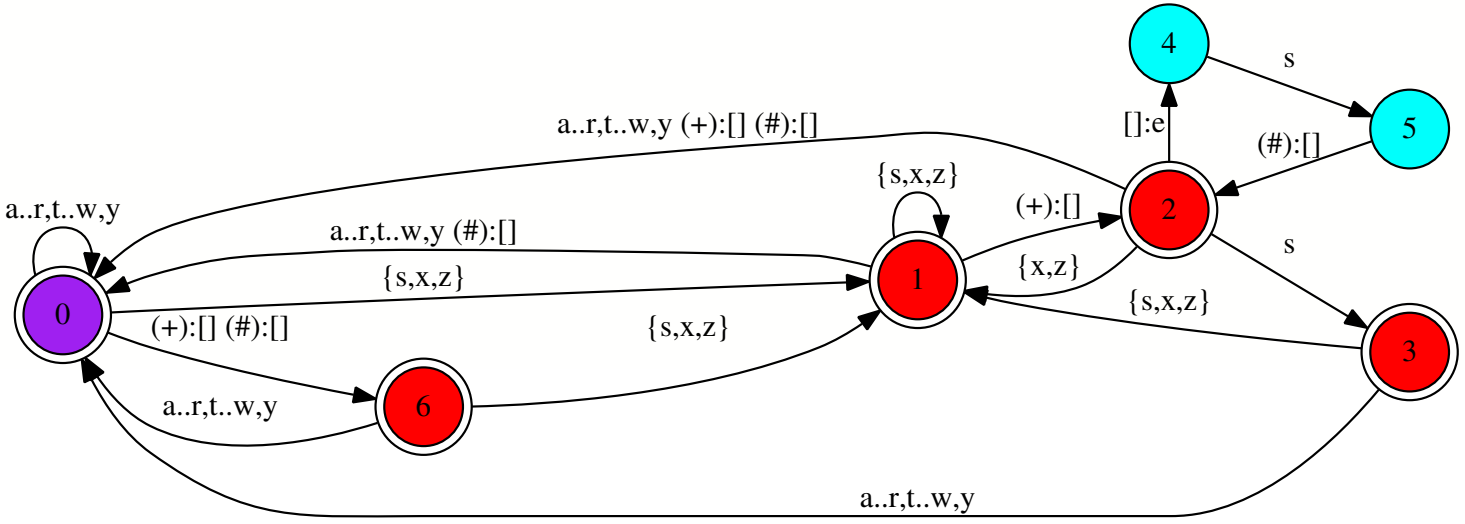
Rules are just FSTs

Consequence

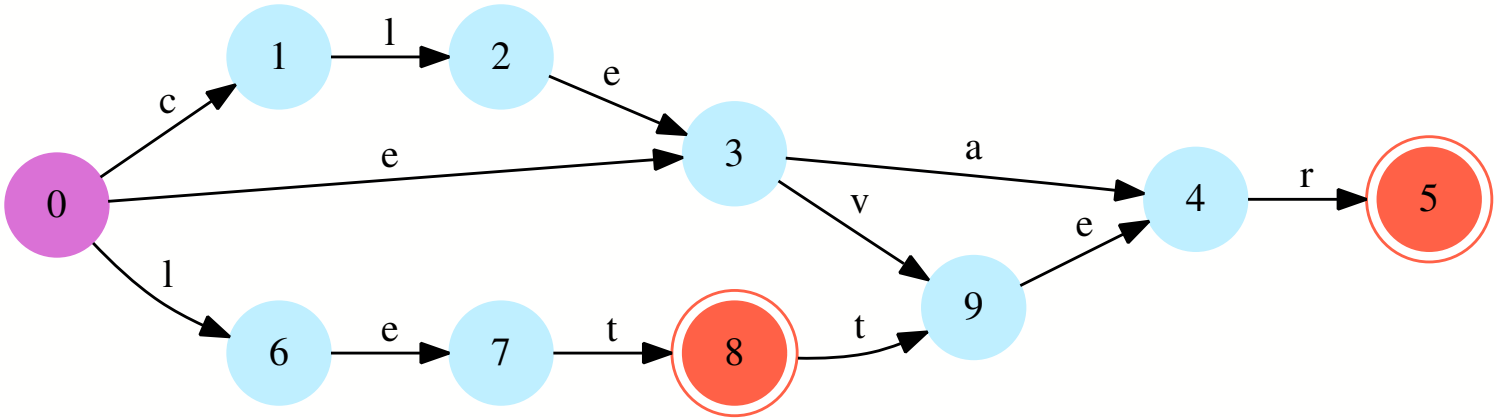
Rules can be compiled into FSTs. Rule FSTs can be combined together into bigger FSTs that enforce all the rules. The rule applications can be ordered (linguists have shown this is necessary).

E-insertion revisited

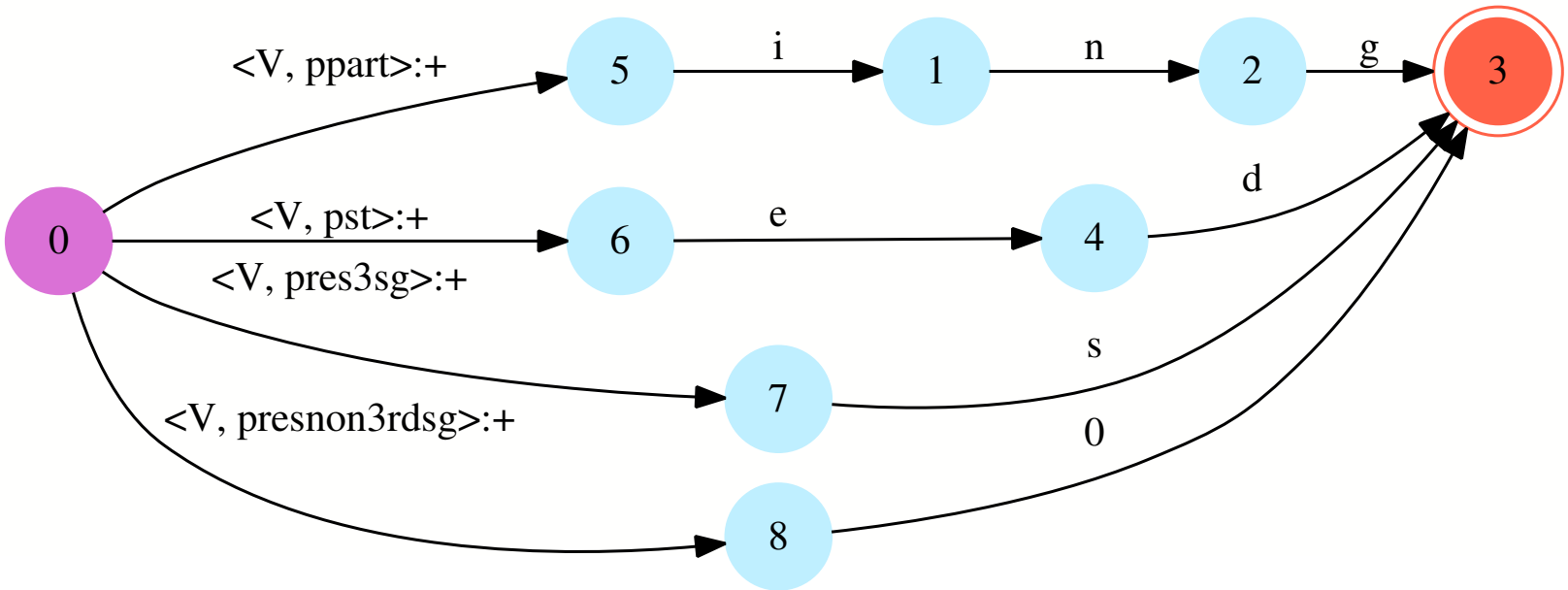
$$\epsilon \rightarrow e \backslash \hat{} : \epsilon \_\_\_\_ s \#$$



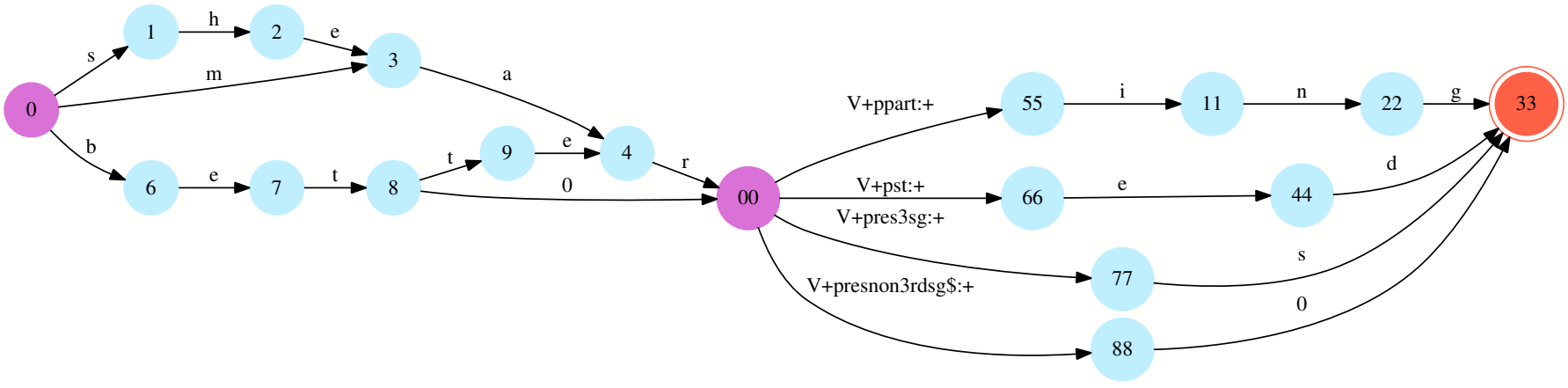
Word trie



Suffixes



Concatenation



Does not enforce spelling rules.

Composition of relations

Example

accurate	inaccurate
definite	indefinite
competent	incompetent
patient	impatient
mature	immature

Prefix will be *iN*.

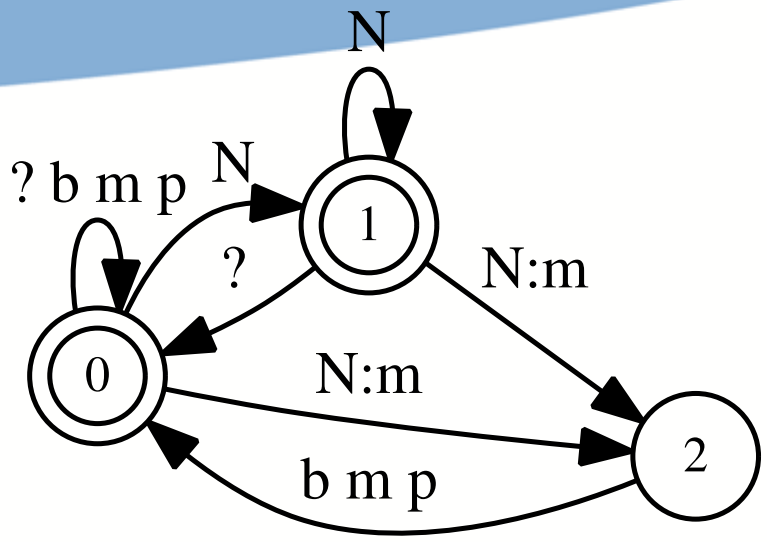
Order dependency

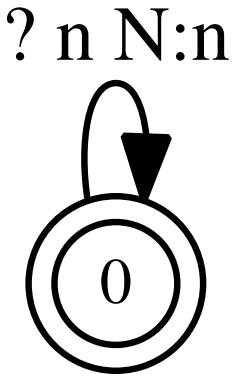
$$\text{R1.}\quad \text{N} \rightarrow \text{m} \setminus \_\_ \left\{ \begin{array}{c} m \\ b \\ p \end{array} \right\}$$

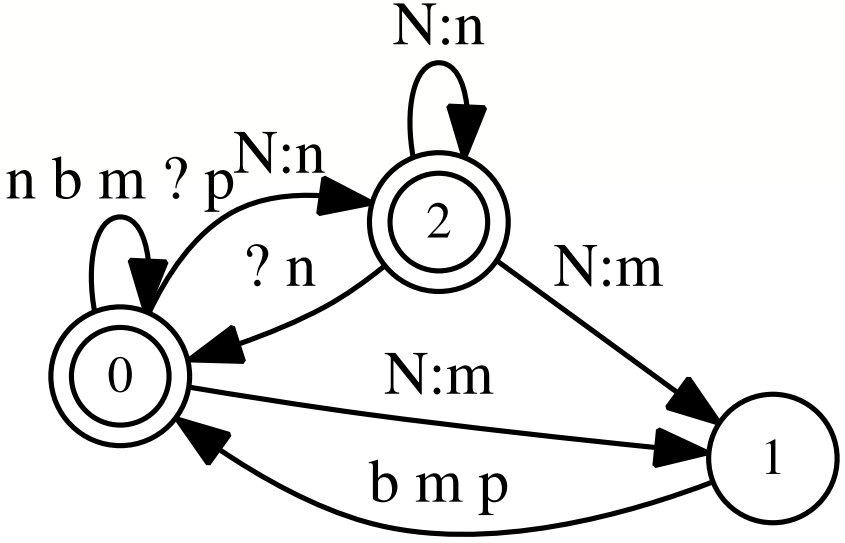
$$\text{R2.}\quad \text{N} \rightarrow \text{n}$$

What we need is R1 $\circ$ R2

Composition of Transducers II





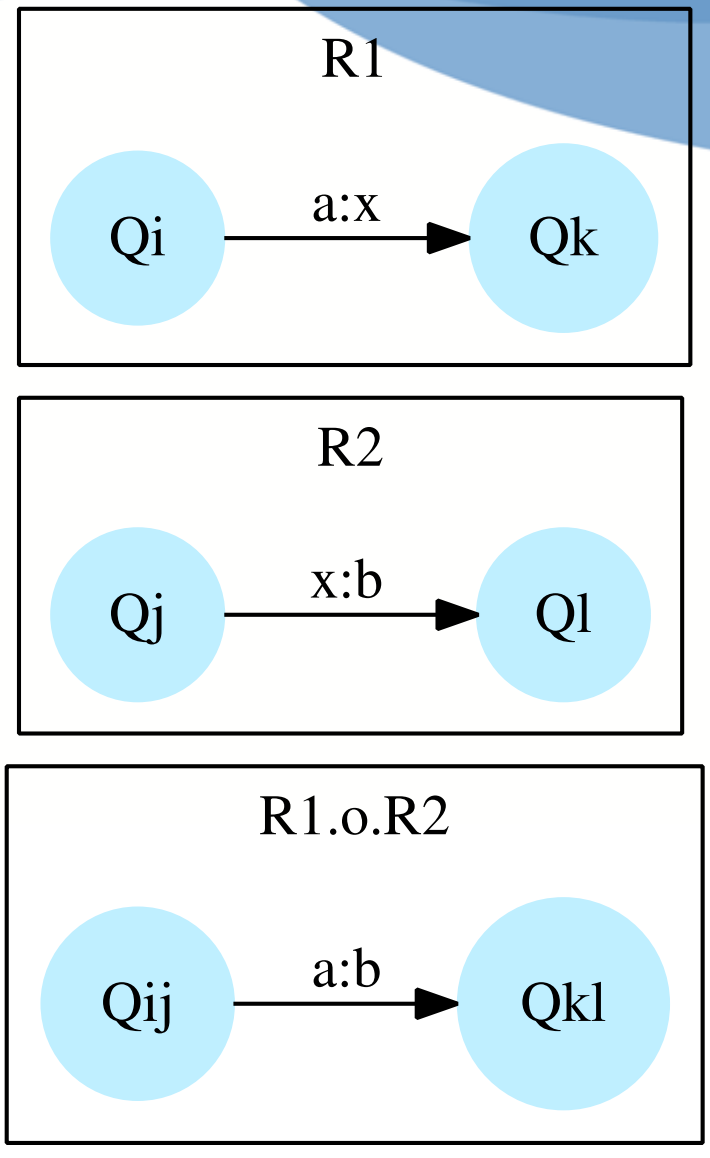


Composition of Transducers II

Composition of two relations

T1	Apfel:alma
T2	alma:apple
T2∘T1	Apfel:apple
R1	iNpatient:impatient
R1	iNdefinite:iNdefinite
R2	iNdefinite:indefinite
R2∘R1	iNpatient:impatient
R2∘R1	iNdefinite:indefinite

In R2∘R1, in effect, the lower tape of R1 becomes the upper tape of R2 (R1 output R1 becomes R2 input).



Rule Composition

Composition: Final step to full lexicon

Combining concatenated lexicon & spelling rules

We **compose** the entire concatenated lexicon with the spelling rules. Concatenated *fox+s* becomes the input to e-insertion, emerging as *fox+es*. This is why the spelling rules have to allow all strings not falling in a rule's domain to pass through unchanged.

Summary

1. Each part of the lexicon starts out as a transducer.
2. We **concatenate** the prefix, root, and suffix transducers. In practice there are different transducers for different parts of speech (noun, verb, adjective, noun suffix, verb suffix, adjective suffix), and different morphological classes (regular and irregular verbs).
3. In theory, we **compose** the concatenated lexicon with the spelling rule transducer (composition of all spelling rules). In practice, this gets very big, and we apply the rules in parallel when possible. (Regular relations are closed under serial composition: Kaplan & Kay 1994)
4. Implementation of FSTs being used in this class (Karttunen 2000).

Bibliography I

Johnson, C. Douglas. 1972.
Formal aspects of phonological description.
The Hague: Mouton.
University of California at Berkeley Ph Dissertation 1970.

Kaplan, Ronald M., and Martin Kay. 1994.
Regular models of phonological rule systems.
Computational Linguistics 20(3):331–378.

Karttunen, K. R. Beesley L. 2000.
Finite-State Morphology: Xerox Tools and Techniques.
Cambridge University Press.

Karttunen, Lauri. 1993.
Finite-state constraints.
In J. Goldsmith (Ed.), *The Last Phonological Rule*, 173–194.
University of Chicago Press.