

Syntax Done Right: What LING 250 Covers

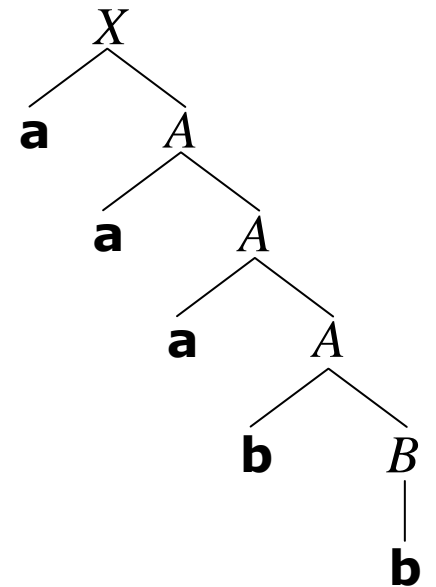
LING 106
March 25, 2009

Regular Languages: Review

- Good at very local effects: $\mathbf{a^*b^*}$, $\mathbf{a^*b(cd)^*}$
- Right Linear Grammars: $A \rightarrow xB$ or $A \rightarrow x$
- $G_1 = \langle T, N, S, R \rangle$, where
 - $T = \{\mathbf{a}, \mathbf{b}\}$
 - $N = \{X, A, B\}$
 - X is the start symbol, and

$$R = \left\{ \begin{array}{l} X \rightarrow \mathbf{aA} \\ A \rightarrow \mathbf{aA} \\ A \rightarrow \mathbf{bB} \\ B \rightarrow \mathbf{bB} \\ B \rightarrow \mathbf{b} \end{array} \right\}$$

$$\begin{aligned} X &\Rightarrow \\ \mathbf{aA} &\Rightarrow \\ \mathbf{a(aA)} &\Rightarrow \\ \mathbf{a(a(aA))} &\Rightarrow \\ \mathbf{a(a(a(bB)))} &\Rightarrow \\ \mathbf{a(a(a(b(b))))} &= \\ \mathbf{aaabb} & \end{aligned}$$



Context-Free Grammars: Review

- Context Free Grammars:

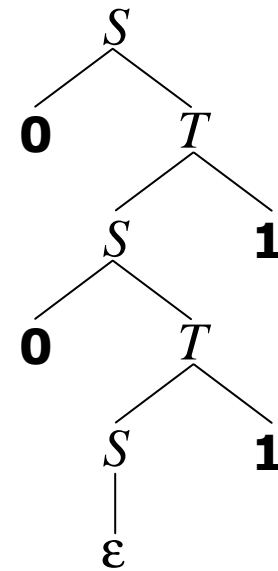
$$A \rightarrow Bx \text{ or } A \rightarrow xB \text{ or } A \rightarrow x$$

- e.g., $\mathbf{0^n 1^n}$

- Example:

- $S \rightarrow \varepsilon$
- $S \rightarrow \mathbf{0}T$
- $T \rightarrow S\mathbf{1}$

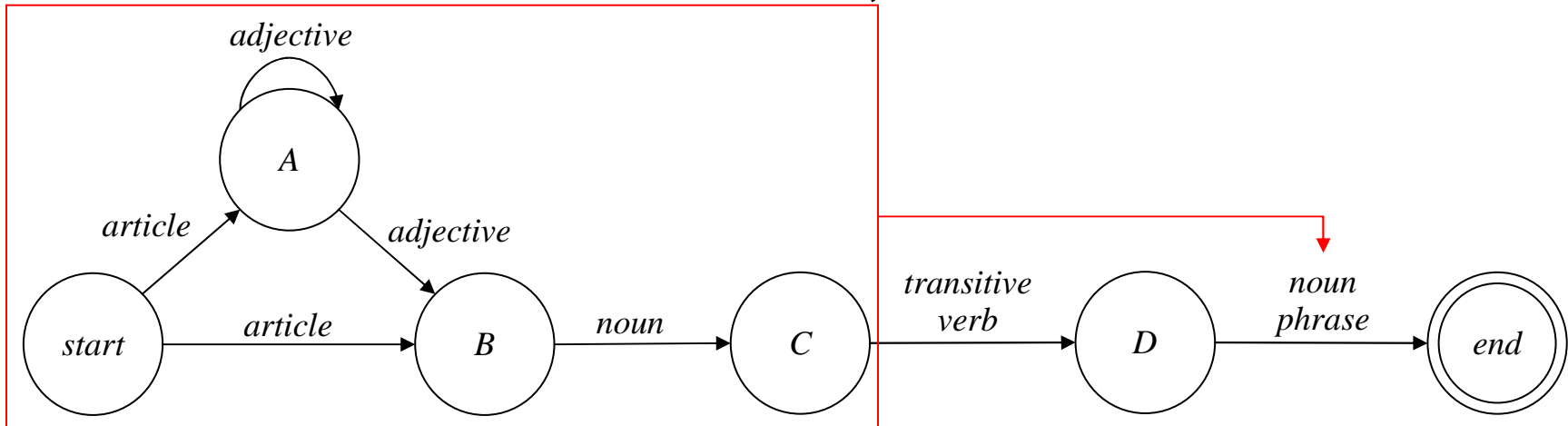
$$\begin{aligned} S &\Rightarrow \\ \mathbf{0}T &\Rightarrow \\ \mathbf{0}(S\mathbf{1}) &\Rightarrow \\ \mathbf{0}((\mathbf{0}T)\mathbf{1}) &\Rightarrow \\ \mathbf{0}((\mathbf{0}(S\mathbf{1}))\mathbf{1}) &\Rightarrow \\ \mathbf{0}((\mathbf{0}((\varepsilon)\mathbf{1}))\mathbf{1}) &= \\ \mathbf{0011} & \end{aligned}$$



Modeling Syntactic Strings: Things That Don't Work

- Storing every string in the language.
- Using very local contexts.
 - Long distance dependencies
 - Can “**is**” follow “**my parents**” in a string?
 - No, if it's part of **my parents is teachers**.
 - Yes, if it's part of **one of my parents is a teacher**.
 - No, if it's part of **the friends of one of my parents is teachers**.
 - Yes, if it's part of **the son of the friends of one of my parents is a teacher**.
 - etc.
 - Which is to say, we need to know where we are in the structure and what came before.
 - (Which is why FSAs don't work)

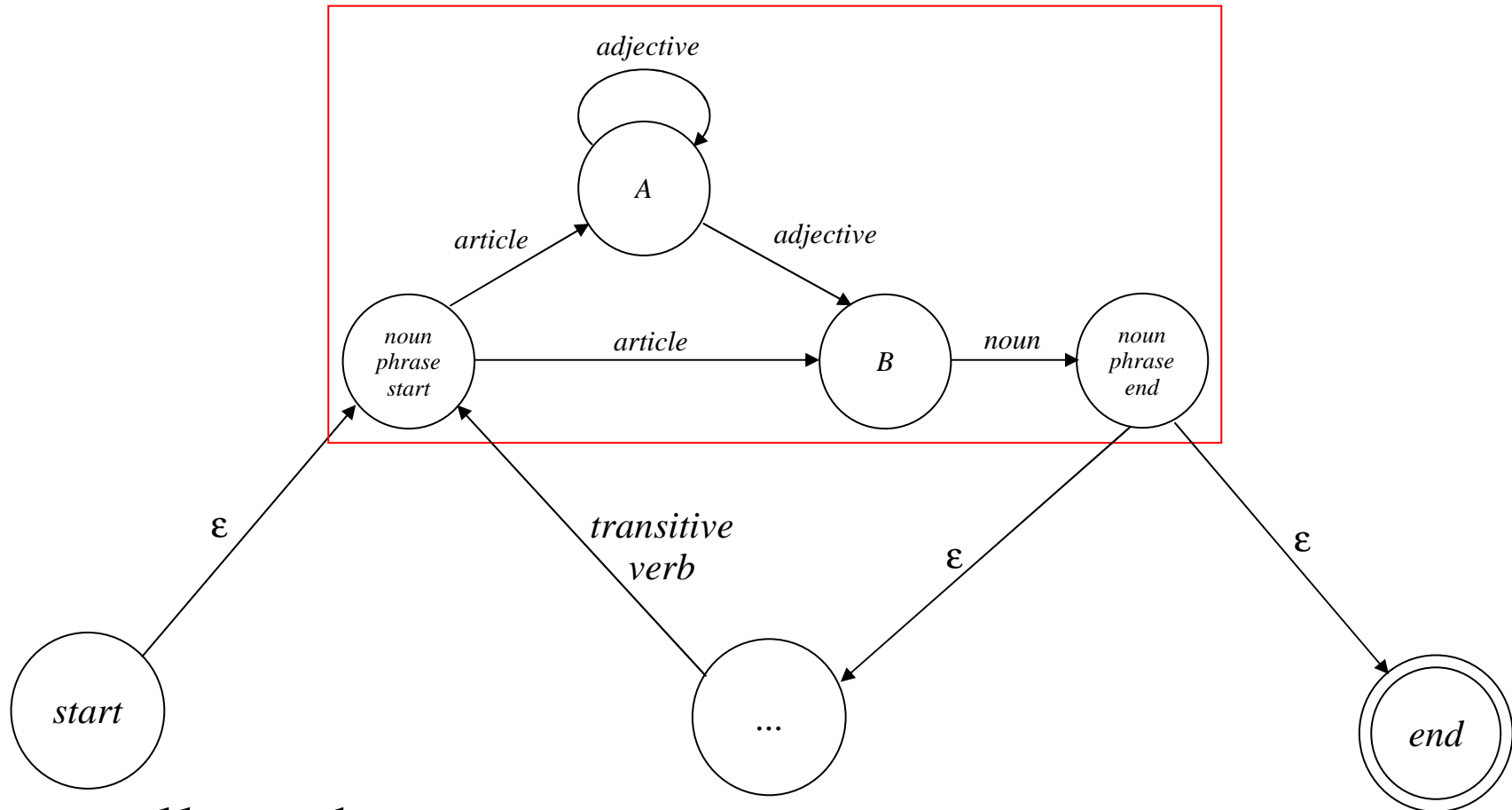
FSA, Redux



- We know, of course, that this doesn't work. Could it be on the right track?
- Consider:
 - **the dog saw a cat**
 - **a cat saw the dog**
 - **the happy dog saw every sad cat**
 - **every sad cat saw the happy dog**
- (think Harris's Conditions...)

FSA, Redux

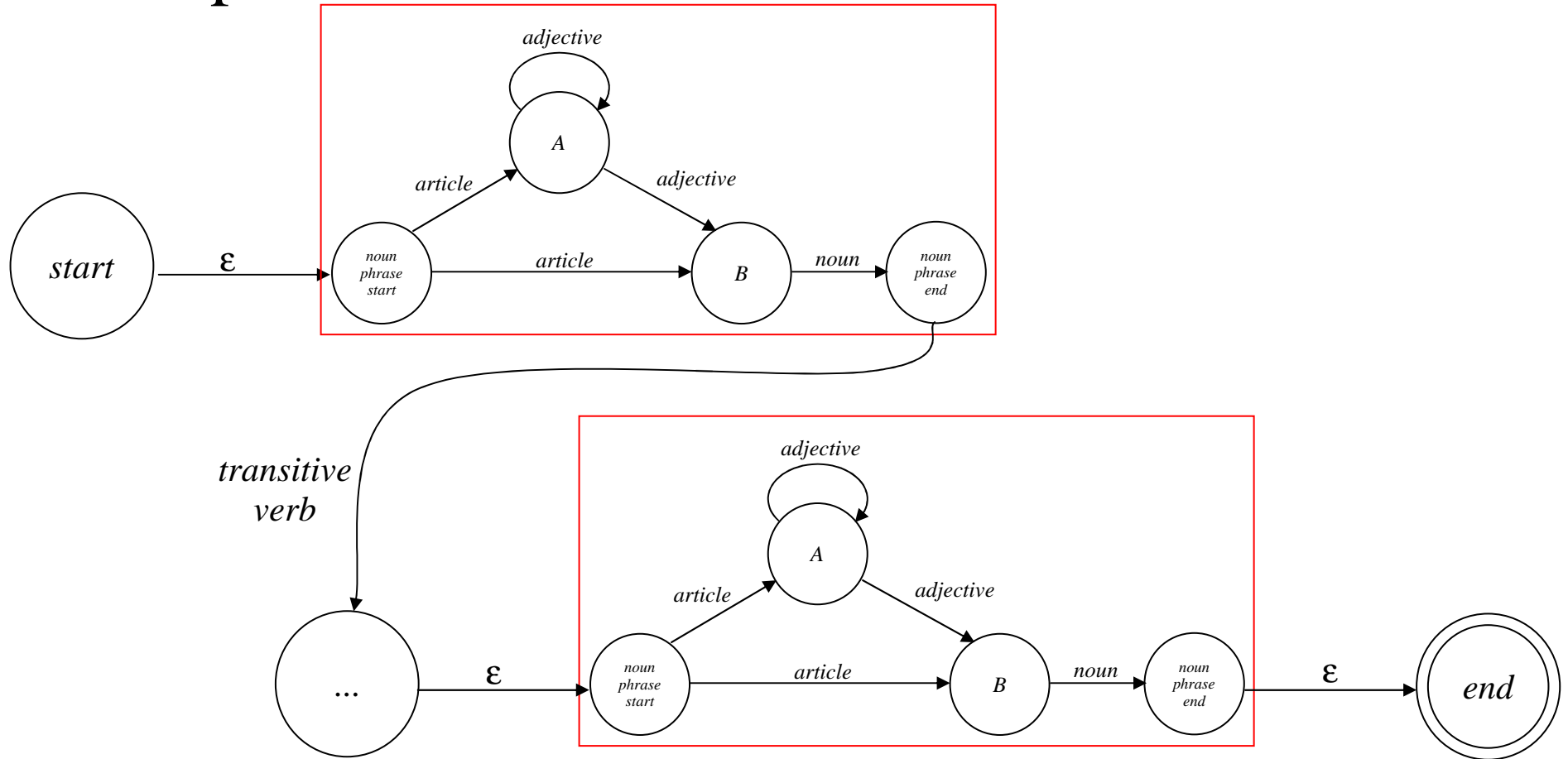
- Perhaps instead:



- ...well, perhaps not.

FSA, Redux

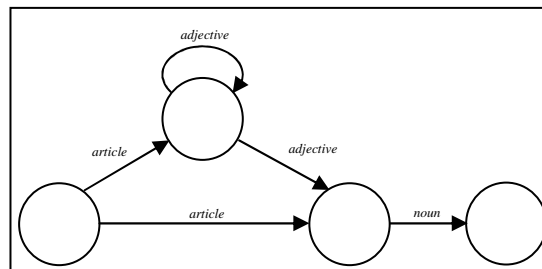
- Perhaps *instead* instead:



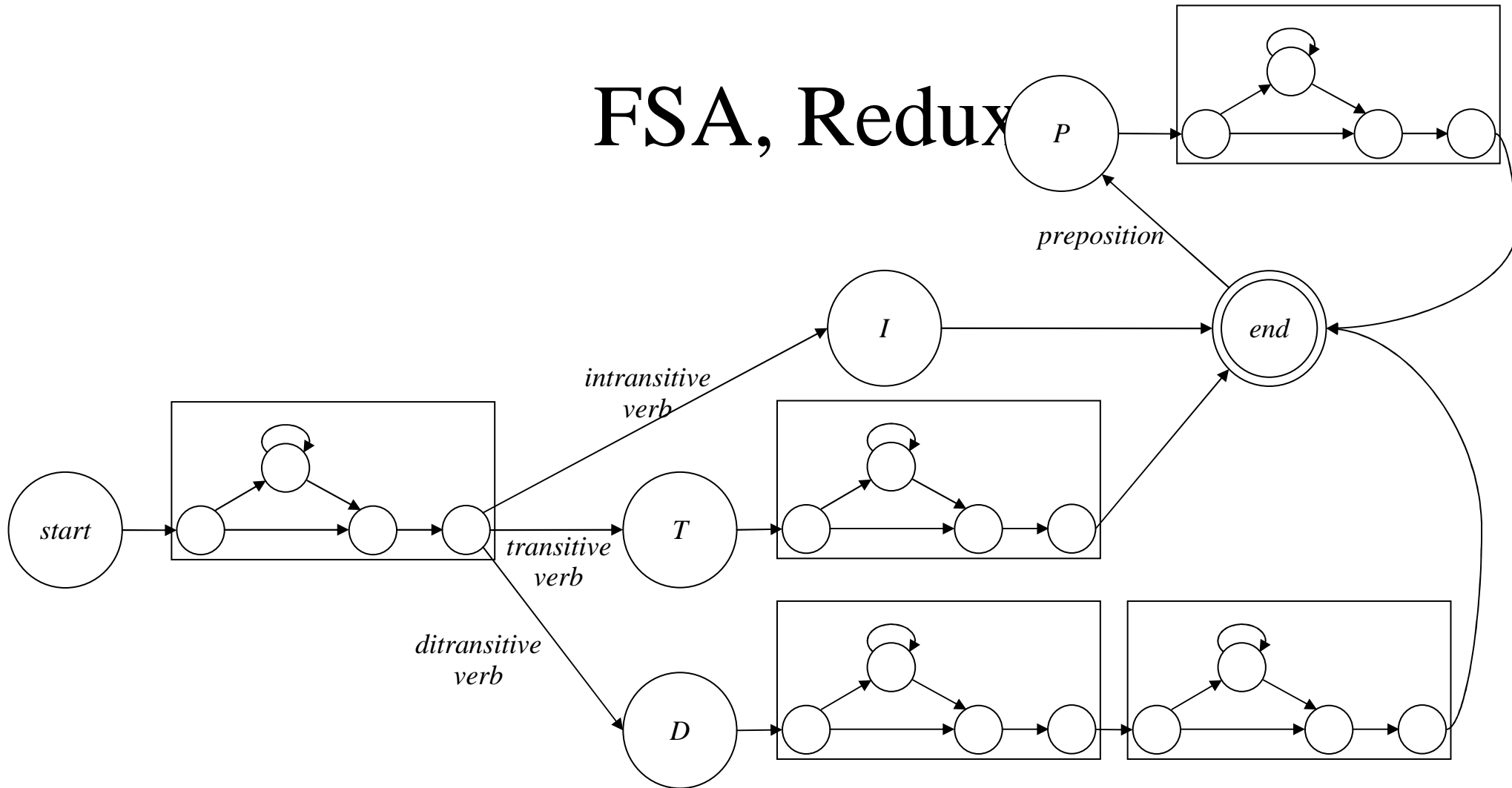
- This works. Well, except that...

FSA, Redux

- Some more data:
 - **the dog** { **slept/barked/walked/jumped/...** }
 - **the cat** { **saw/liked/chased/...** } **the dog**
 - **the dog** { **saw/liked/chased/...** } **the cat**
 - **John** { **gave/sold/...** } **his cousin the dog**
 - **John** { **gave/...** } **the dog** a biscuit
 - etc.
- Our “noun phrase” box has to appear in a lot of different places in the FSA....



FSA, Redux



- That's irritating, from the point of view of FSAs.
- ...but not context-free grammars!

Turning the FSA “Box” into a CFG

- If what we want is “an article, followed by any number of adjectives, followed by a noun”:
- *noun-phrase* \rightarrow *article adjective* noun*
NP \rightarrow Art Adj* N
 - ...where *noun-phrase* (NP), *article* (Art), *adjective* (Adj), *noun* (N) are all non-terminals;
 - and *adjective** means “zero or more adjectives”
- Note: if you don’t like *adjective**, you can use...
 - NP \rightarrow Art AdjP N
 - AdjP \rightarrow Adj AdjP
 - AdjP \rightarrow Adj

Using CFGs

- So for our data:
 - **the dog** {slept/barked/walked/jumped/...}
 - **the cat** {saw/liked/chased/...} **the dog**
 - **the dog** {saw/liked/chased/...} **the cat**
 - **John** {gave/sold/...} **his cousin the dog**
 - **John** {gave/...} **the dog** a biscuit
- Instead of:
 - Sentence \rightarrow **Article Adjective* Noun** Intransitive-Verb
 - S \rightarrow **Art Adj* N** Transitive-Verb **Art Adj* N**
 - S \rightarrow **Art Adj* N** Ditransitive-Verb **Art Adj* N Art Adj* N**
- We have:
 - **NP** \rightarrow Art Adj* N
 - S \rightarrow **NP** Intransitive-Verb
 - S \rightarrow **NP** Transitive-Verb **NP**
 - S \rightarrow **NP** Ditransitive-Verb **NP NP**

Using CFGs

- In fact:
 - We drew a box for noun phrases, because a variety of things could be interchanged.
 - We could do the same thing for other phrases.
- e.g., **John put the book...**
 - ...**on** { **the table/a tall shelf/his foot** }
 - ...**near** { **the table/a tall shelf/his foot** }
 - ...**under** { **the table/a tall shelf/his foot** }
 - generally: “Preposition NP”.
- From there:
 - $S \rightarrow NP \textbf{put} NP PP$
 - $PP \rightarrow \text{Prep} NP$

In A Tree

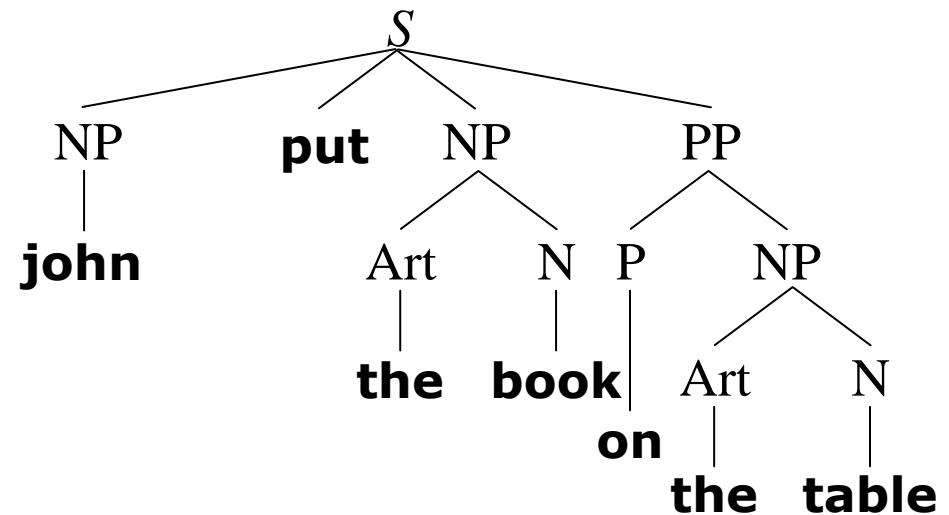
- Rules:

- $S \rightarrow NP \text{ put } NP PP$
- $PP \rightarrow P NP$
- $NP \rightarrow \text{Art Adj}^* N$

- More rules:

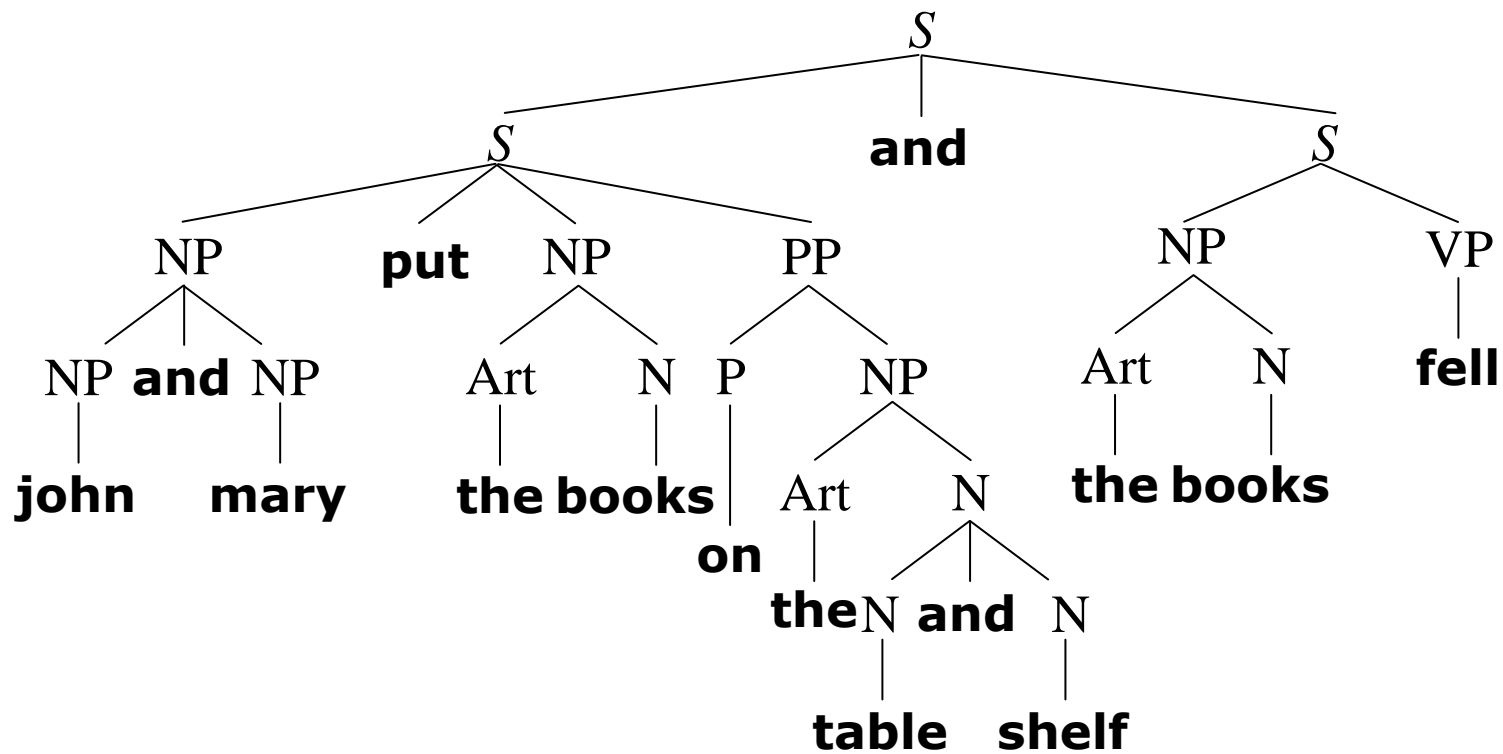
- $NP \rightarrow \text{john}$
- $N \rightarrow \text{table} \mid \text{shelf} \mid \text{foot}$
- $\text{Art} \rightarrow \text{the} \mid \text{a} \mid \text{an} \mid \text{his}$
- $\text{Adj} \rightarrow \text{tall}$
- $\text{Prep} \rightarrow \text{on} \mid \text{near} \mid \text{under}$

$S \Rightarrow$
 $NP \text{ put } NP PP \Rightarrow$
 $[\text{john}] \text{ put } [\text{Art } N] [P NP] \Rightarrow$
 $[\text{john}] \text{ put } [[\text{the}] [\text{book}]] [P [\text{Art } N]] \Rightarrow$
 $[\text{john}] \text{ put } [[\text{the}] [\text{book}]] [[\text{on}] [\text{the}] [\text{table}]]$



Things We Can Do With Trees

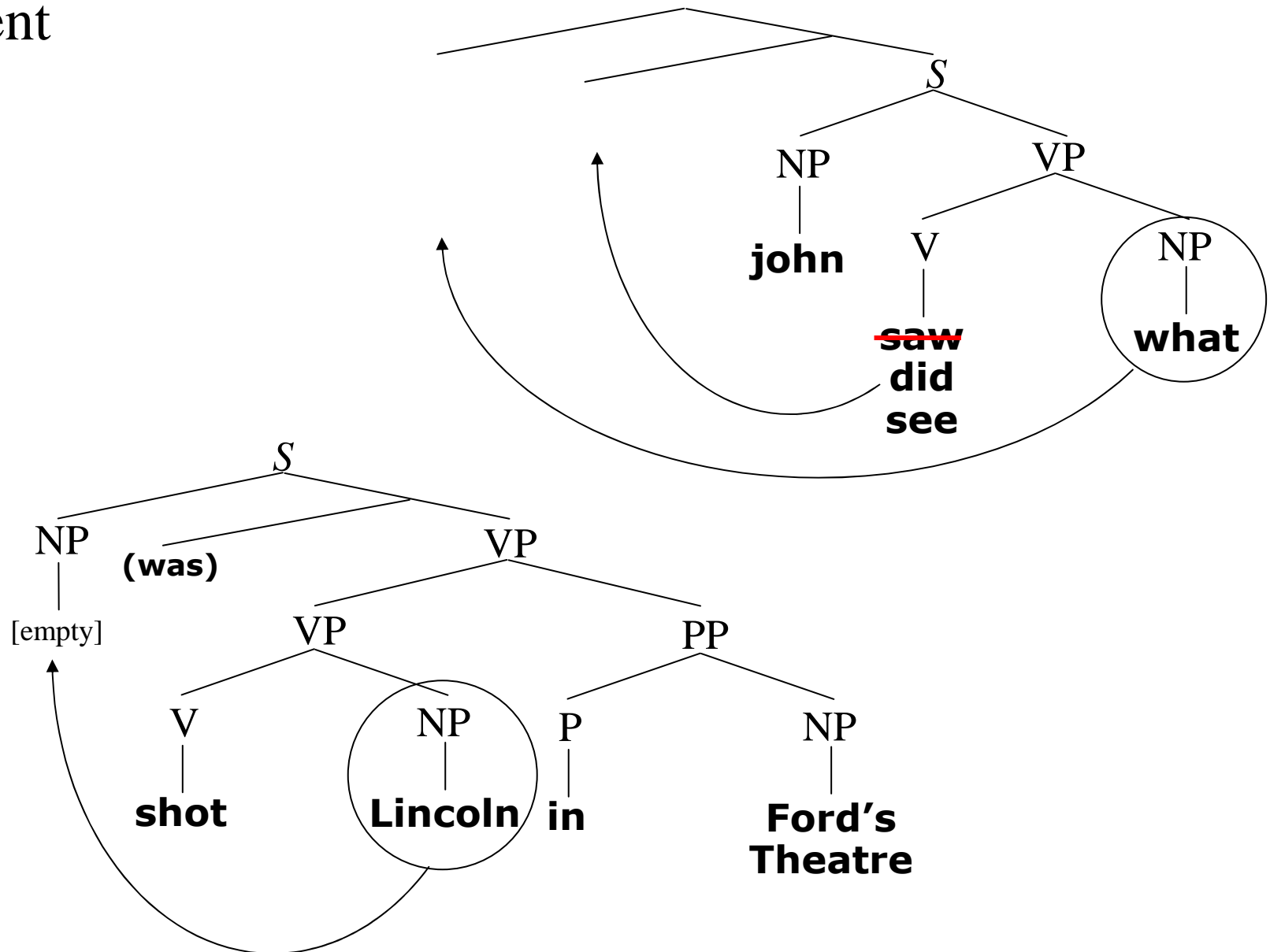
- Conjunction



- etc.

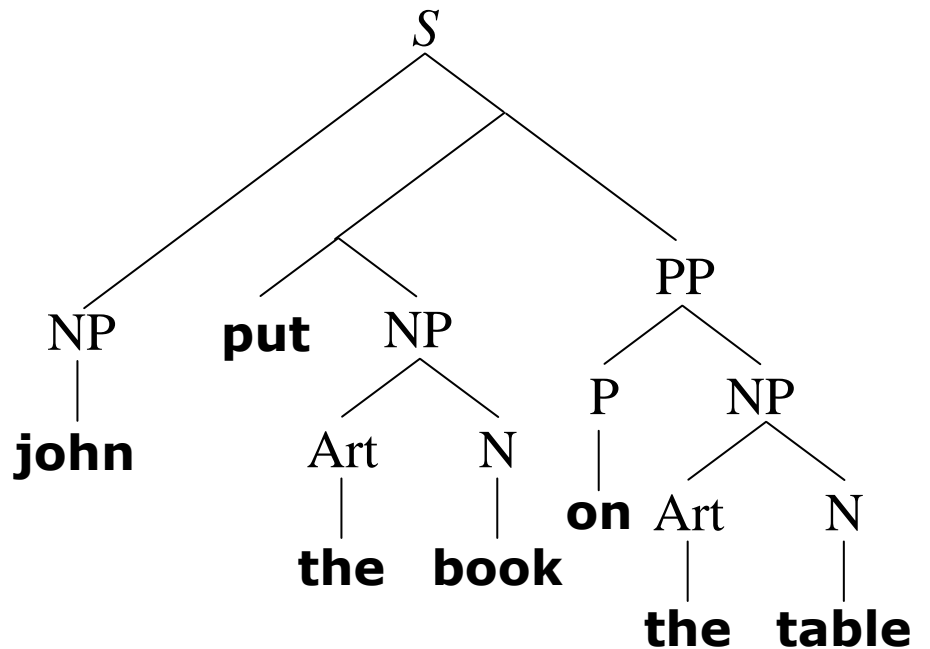
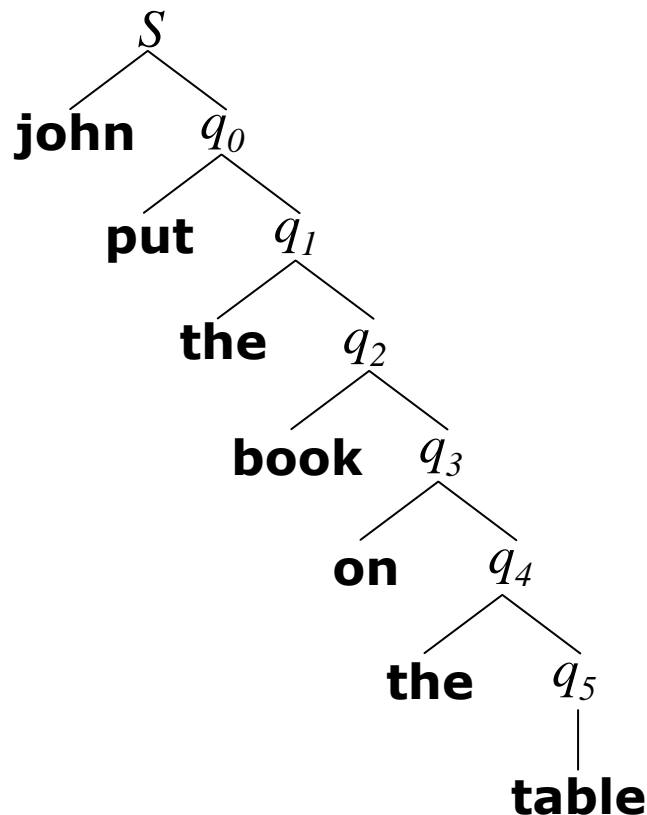
Things We Can Do With Trees

- Movement



Things We Can Do With Trees

- More generally: we can refer to structure in a way that we couldn't with a Right Linear Grammar (aka a regular language).



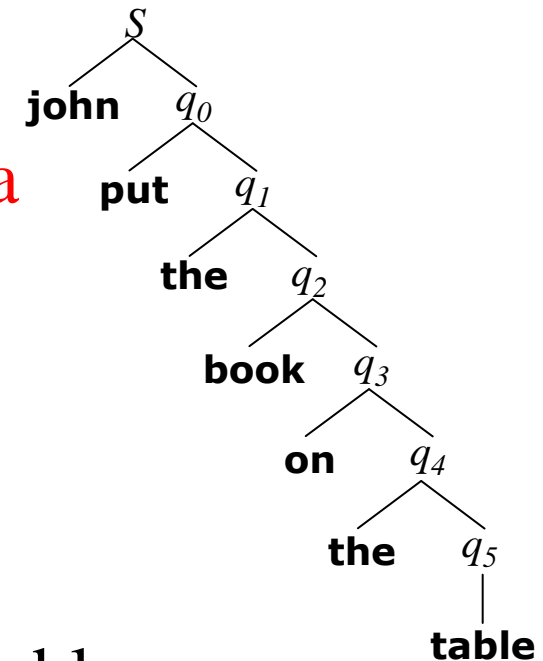
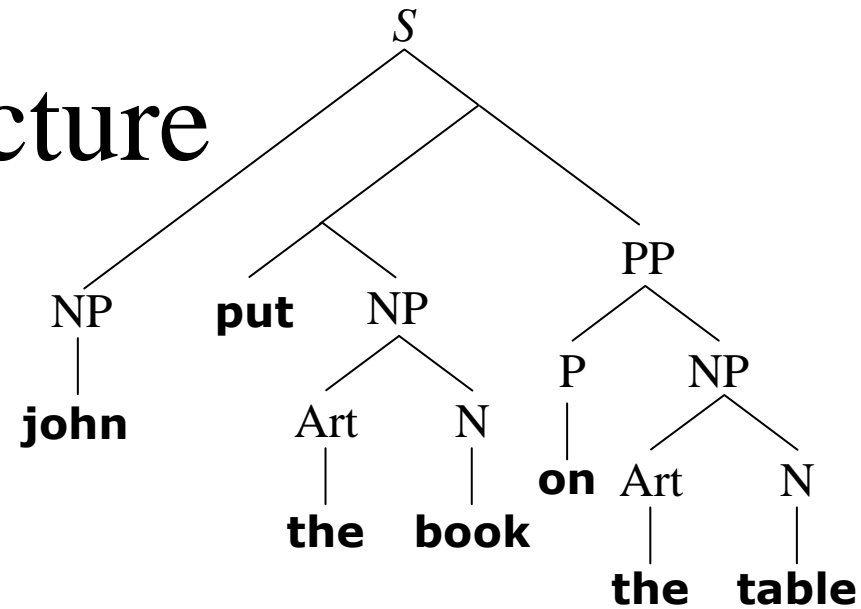
Why We Need Structure

- Conjunction

- john ~ john and mary
- table ~ table and shelf
- the book ~ the book and a magazine
- on ~ on or near
- *book on the ~ book on the and paper on a

- Cleft sentences

- It was **John** who put the book on the table
- It was **the book** that John put on the table
- It was **on the table** that John put the book
- *It was **put the** that John did book on the table



Why We Need Structure

- Pronouns

- John admires himself.
- *John wants Mary to admire himself.
- John seems to Mary to admire himself.

- Yes/No Questions

- The girl **is** eating ice cream. →
 - **Is** the girl eating ice cream?
- The girl **is** happy. →
 - **Is** the girl happy?
- The girl who **is** eating ice cream **is** happy. →
 - ~~Who the girl **is** eating ice cream **is** happy?~~
 - **Is** the girl who eating ice cream **is** happy?
 - **Is** the girl who **is** eating ice cream happy?
 - **Is is** the girl who eating ice cream happy?

Phonology,
Morphology, and
Finite State Transducers

Phonology and Morphology

- What phonology does:
 - Takes forms stored in the lexicon...
 - **coat**: </**kot**/, *noun*, ‘heavy overgarment used for warmth’>
 - **code**: </**kod**/, *noun*, ‘secret writing system’>
 - **coach**: </**kotʃ**/, *noun*, ‘person who trains a sports team’>
 - ...and turns them into pronounced strings.
 - [**k^hot**], [**k^hod**], [**k^hotʃ**]
 - What morphology does:
 - Takes pieces of words (“morphemes”)...
 - **-s**: </**s**/, *noun suffix*, ‘plural’>
 - ...and combines them with words to make new words.
 - **coats** ([**k^hots**]), **codes** ([**k^hodz**]), **coaches** ([**k^hotʃəz**])
- note:
phonology
happens
here too!

How They Do It

- Phonology: go through the word and apply rules
 - e.g., /**k**/ becomes [**k^h**] at the start of a word*
 - /**s**/ becomes [**z**] after certain consonants at the end of the word**
 - Insert a /**ə**/ between two sibilants (**z**, **s**, **ʃ**, a few others)
- Morphology: add morphemes in succession
 - [[**coat**] + **s**]
 - [[[[**industri**] + **al**] + **iz**] + **ation**]
 - **uygarlaştıramadıklarımızdanmışsınız**
“you are from the ones we could not civilize”

uygar	laş	tır	ama	dık	lar	ımız	dan	mış	sınız
civilize	<i>become</i>	<i>cause</i>	<i>not</i>	<i>past</i>	<i>plur.</i>	<i>1stpl</i>	<i>ablative</i>	<i>past</i>	<i>2ndpl.</i>
			<i>able</i>	<i>partic.</i>		<i>poss.</i>	<i>case</i>		

*actually, at the start of any stressed syllable; and not when preceded by an “s”, so neither /**k**/ in **skunk** is aspirated, nor either /**k**/ in **collect**. For more information, I recommend LING 330.

**Specifically, voiced consonants; and the change might actually go in the other direction. For more information, etc.

An Aside on Stored Forms

- We might ask, why have this...
 - </**kot**/, *noun*, ‘heavy overgarment used for warmth’>
 - /**k**/ becomes [**k^h**] at the start of a word
- ...instead of this?
 - </**k^hot**/, *noun*, ‘heavy overgarment used for warmth’>
- Answer:
 - “/k/ becomes [k^h]” is completely predictable, so why store it for each word individually?
 - When we borrow words from other languages, or pronounce foreign names, we apply these rules.
 - In some cases we see alternation depending on the morphology.
 - e.g. the **t** in **write+r** and the **d** in **ride+r** are pronounced the same; but they’re still stored differently on **write** and **ride**.

Successive Rule Application

- So we have rules, e.g.,
 - Rule 1: insert /ə/ between two sibilants (**z**, **s**, **ʃ**, ...)
 - Rule 2: /-**s**/ becomes [**z**] after a vowel
 - e.g., **plays** = /**plei** + **s**/ = [**pleiz**], not [**pleis**], “place”
- And these rules apply one at a time, in order:

Input: /**k o t ʃ s**/
Apply Rule 1: **k o t ʃ ə s**
Apply Rule 2: **k o t ʃ ə z**
Apply ^h Rule: **k^h o t ʃ ə z**
Output: [**k^h o t ʃ ə z**]

Input: /**k o t ʃ s**/
Apply Rule 2: **k o t ʃ s**
Apply Rule 1: **k o t ʃ ə s**
Apply ^h Rule: **k^h o t ʃ ə s**
Output: [**k^h o t ʃ ə s**]

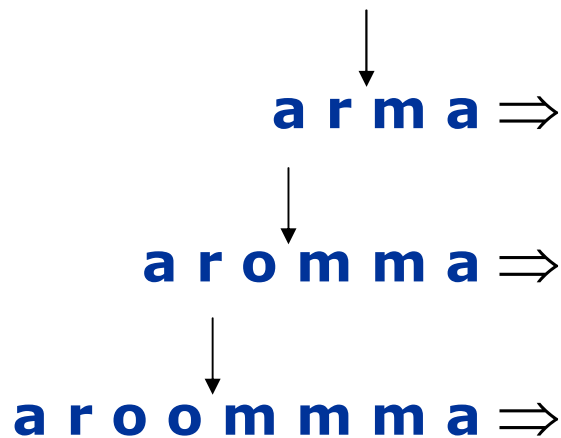
wrong!

- ...and the order matters.

Johnson's Discovery

- C. Douglas Johnson (1972): a phonological rule proceeds left-to-right through a word, and never goes back to reapply to an earlier change.
 - Suppose we have a language with the rule:
optionally insert /**om**/ before /**m**/
 - Then the word /**arma**/ in that language:

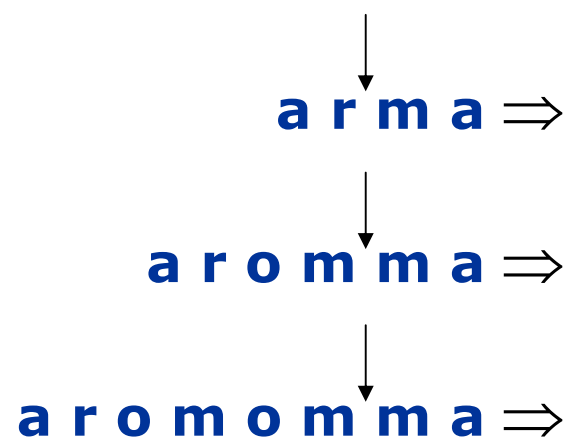
Able to move backwards:



ar (oⁿ mⁿ) ma

Not a regular language / is not allowed

Only moving left-to-right:



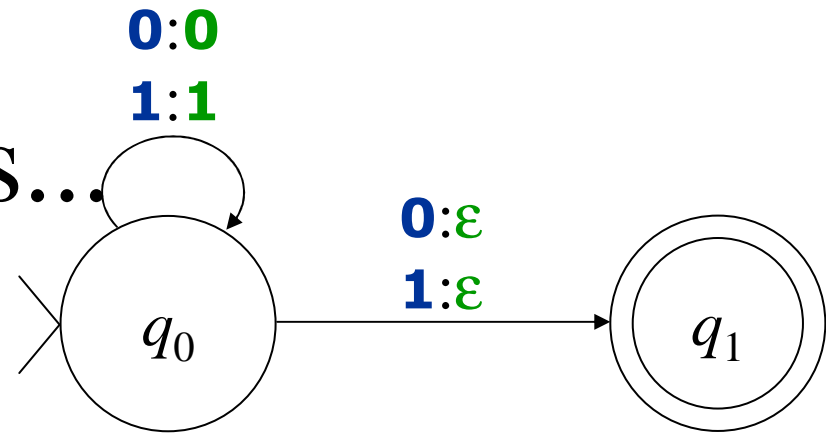
ar (om)^{*} ma

Regular language / is allowed

Finite State Transducers

- If phonology is a regular process, it can be modeled with a finite state machine!
- But instead of just an automaton, we need a **transducer**. Instead of taking an input string, a transducer takes an **input/output pair** and either accepts or rejects it.
 - Another way to read this: given an input string, it returns an output string (or strings).
 - If the machine is nondeterministic, you may get more than one output string.
 - Or: given an output string, it interprets it as an input string (or strings).
 - Even if the machine is deterministic, the mapping from input to output can be many-to-one, i.e. the machine may recognize both $\langle x, z \rangle$ and $\langle y, z \rangle$.
- Transducers are much like other machines, but the transitions are labeled with mappings.

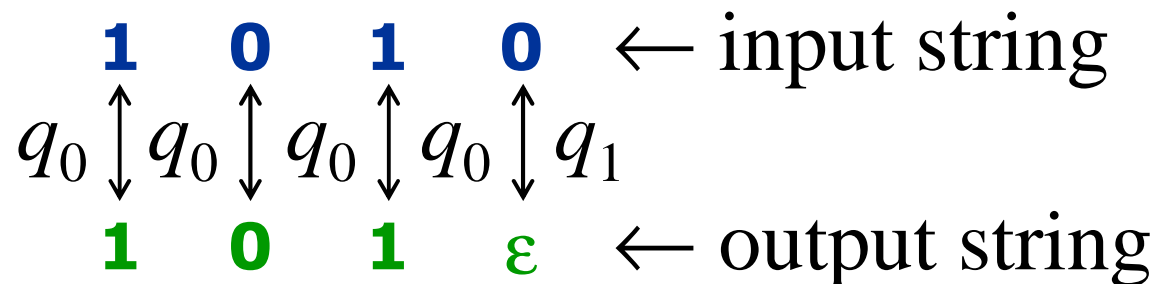
A Few Examples...



- Does this machine accept the pair $\langle \mathbf{1010}, \mathbf{101} \rangle$? *yes*

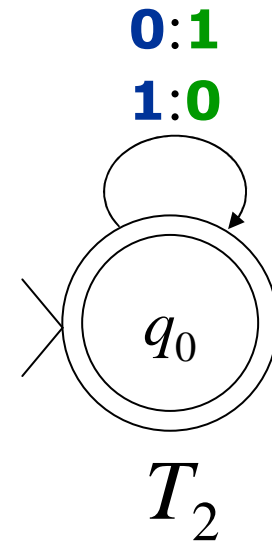
<i>Current State</i>	<i>Input Symbol</i>	<i>Output Symbol</i>	<i>New State</i>
$q_0 (= s)$	1	1	q_0
q_0	0	0	q_0
q_0	1	1	q_0
q_0	0	ϵ	q_1
q_1	(end)		

- Another way to represent this:

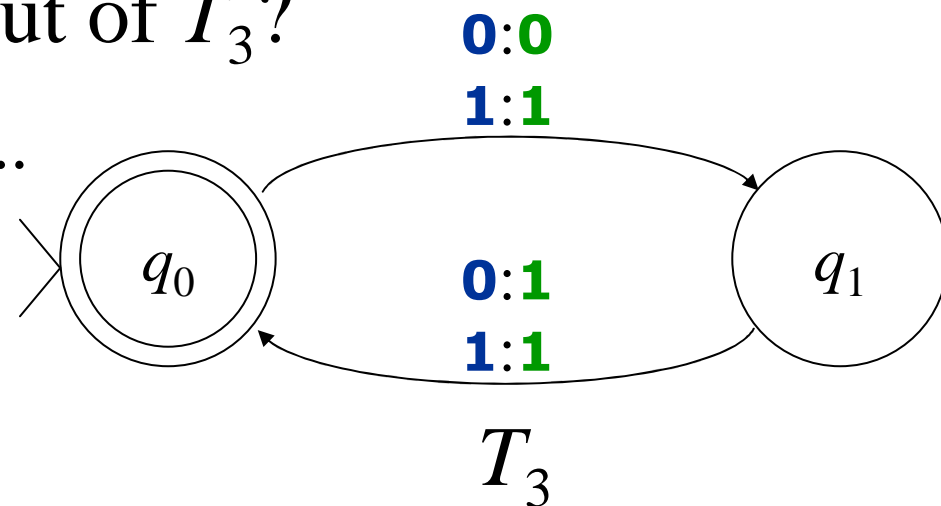


More examples

- **Question:** What is the output of T_2 ?
 - For instance, for what string x is $\langle \mathbf{1010}, x \rangle$ a pair accepted by the machine? (i.e., what is the machine's output for the input $\mathbf{1010}$?) **0101**



- **Question:** What is the output of T_3 ?
 - What is the output of T_3 for...
 - $\mathbf{1010}$? **1111**
 - $\mathbf{0000}$? **0101**
 - $\mathbf{0101}$? **0101**
 - $\mathbf{10101}$? **reject**



Formal Definition

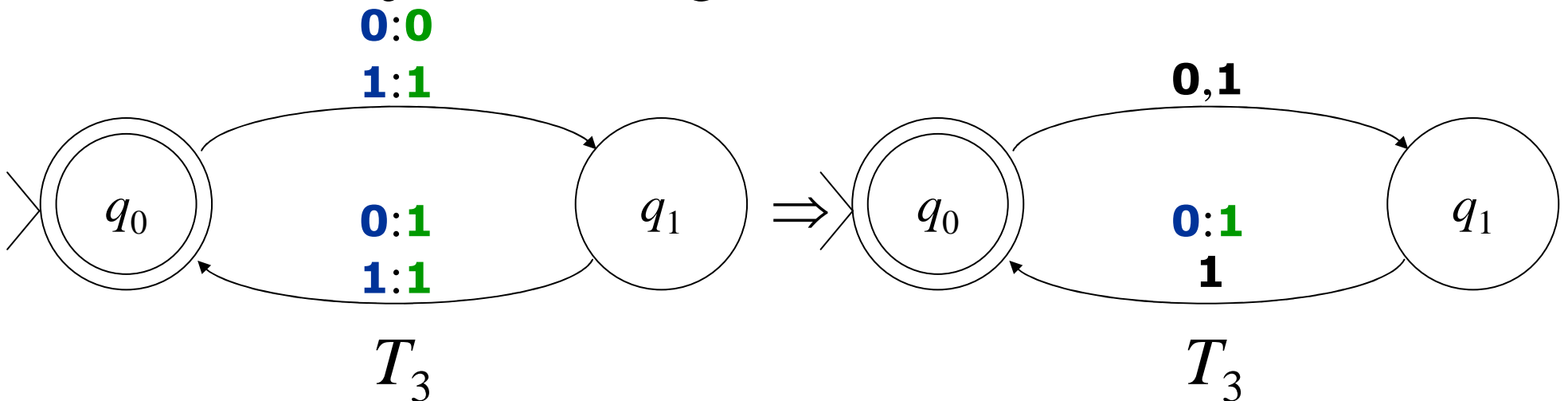
Definition: A **finite state transducer** is a 5-tuple*,
 $\langle Q, \Sigma, \delta, s, F \rangle$, in which:

- a. Q is a finite set of states
- b. Σ is the alphabet
- c. δ is the transition function:
$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q)$$
- d. $s \in Q$ is the start state
- e. $F \subseteq Q$ is the set of accept states (or “end states”)

*There are other ways to define this, e.g. as a 7-tuple, which separately specifies (a) an input alphabet Σ and an output alphabet Δ , and (b) a transition function $\delta : Q \times \Sigma \rightarrow Q$ and an output function $\sigma : Q \times \Sigma \rightarrow \Delta$. Here, I'm assuming the input and output alphabets are the same, and incorporating the output function into the transition function. Now stop reading the footnote and pay attention.

A Helpful Aside

- Finite State Automata are actually a kind of Finite State Transducer, in which each transition x can be read as $x:x$.
- That is, the input is the same as the output.
- In fact, to underscore this fact, we can label any $x:x$ transition just as x , e.g.



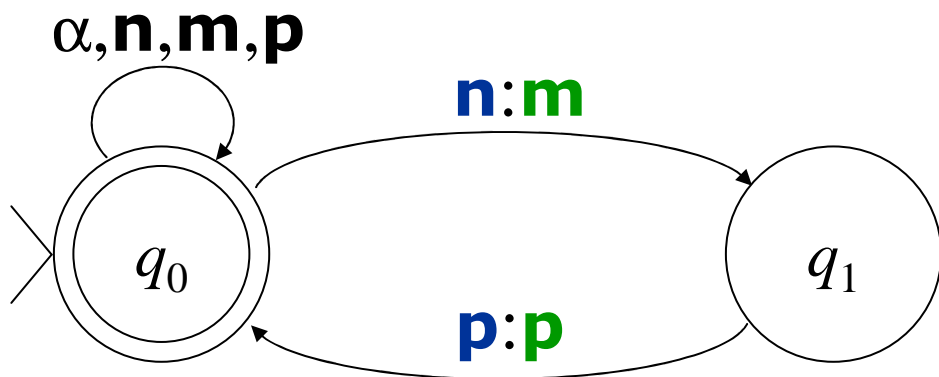
Using Transducers in Phonology

- Let's use a somewhat simpler set of rules than the English plural—the following rules are based on an example by Lauri Karttunen (not a real language):
 - Rule 1: /**n**/ becomes [**m**] before **p**
 - Rule 2: /**p**/ becomes [**m**] after **m**
- Once again, if we treat these as rules and go through them in order:

Input:	/ k a n p a n /
Apply Rule 1:	k a m p a n
Apply Rule 2:	k a m m a n
Output:	[k a m m a n]

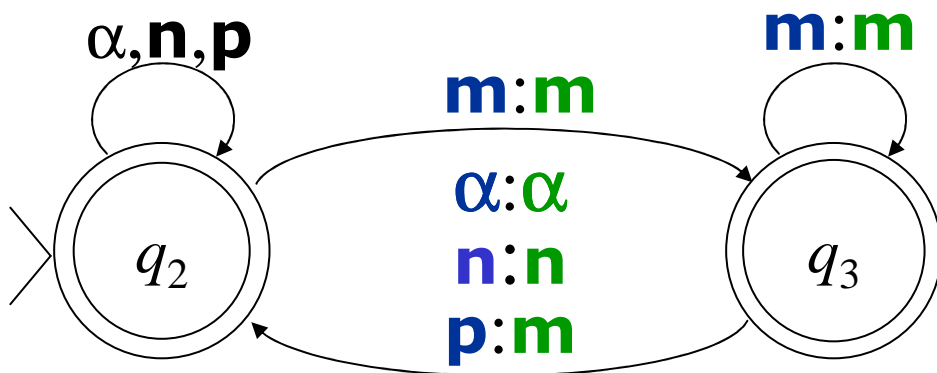
Using Transducers in Phonology

- Rule 1: /**n**/ becomes [**m**] before **p**



Note 1: α means “anything in the alphabet that isn’t otherwise mentioned in this FST”

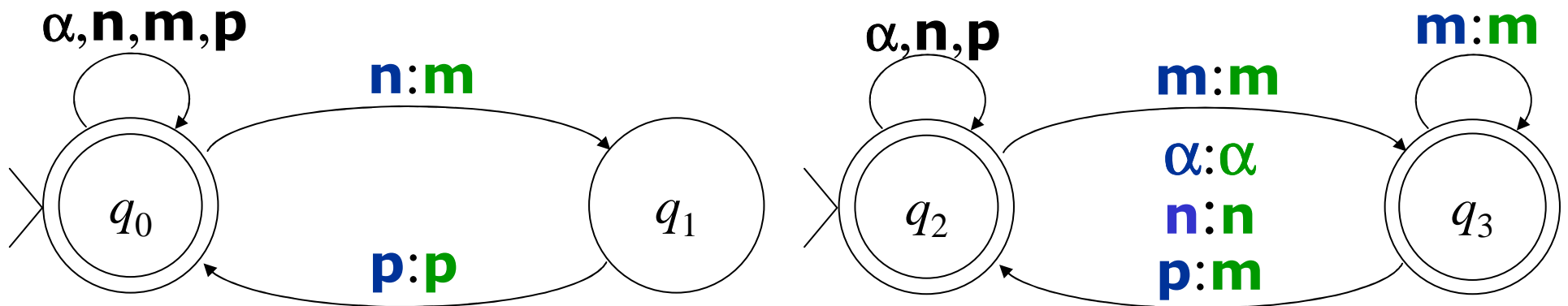
- Rule 2: /**p**/ becomes [**m**] after **m**



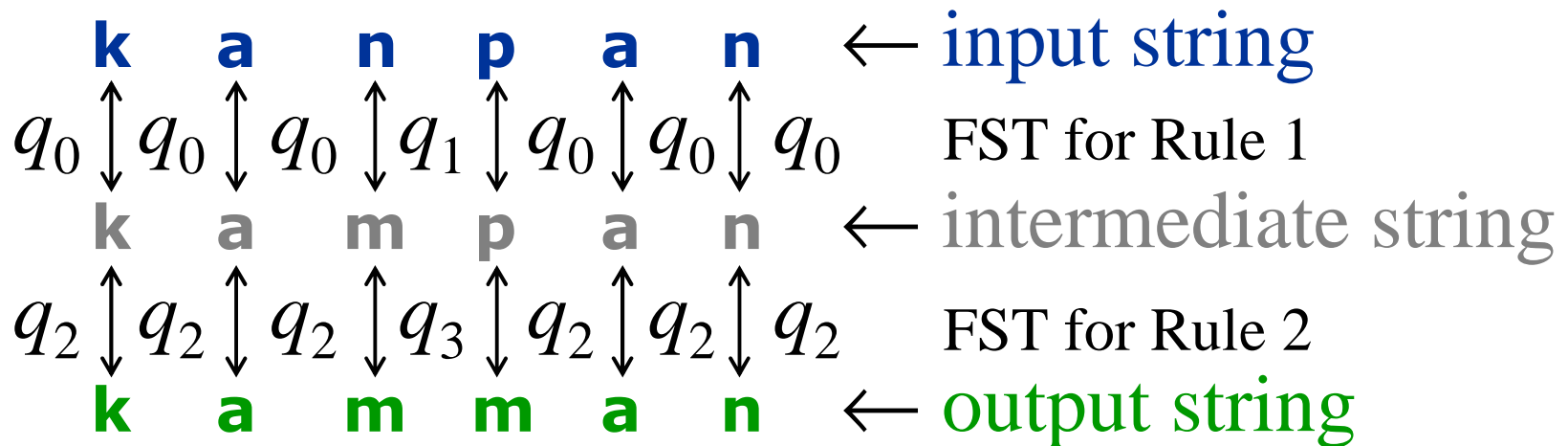
Note 2: this’ll ensure that **n** will become **m** *only* before **p**, but not that it *always* will.

But for simplicity’s sake, we’ll leave that part out. (See the class notes for full details.)

Using Transducers in Phonology

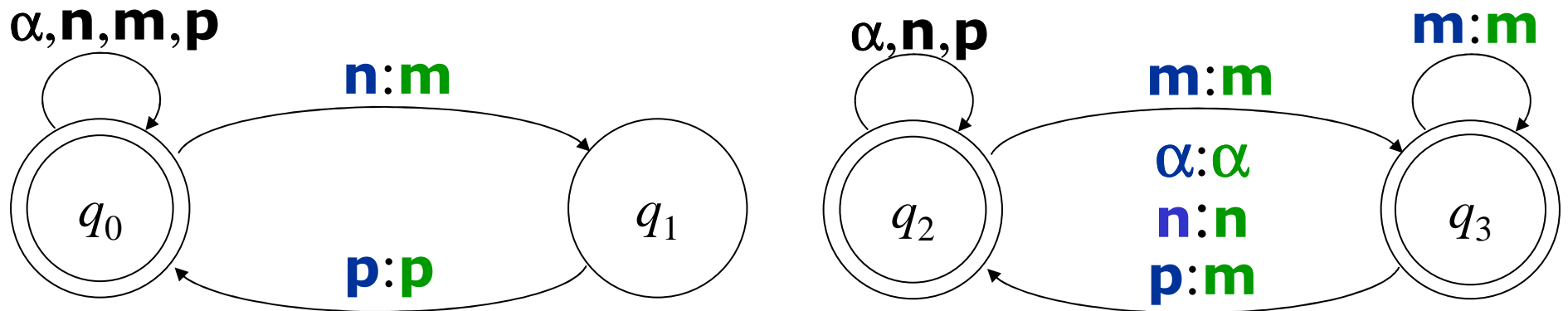


Does Rule 1 followed by Rule 2 accept the pair <**kanpan**, **kamman**>?

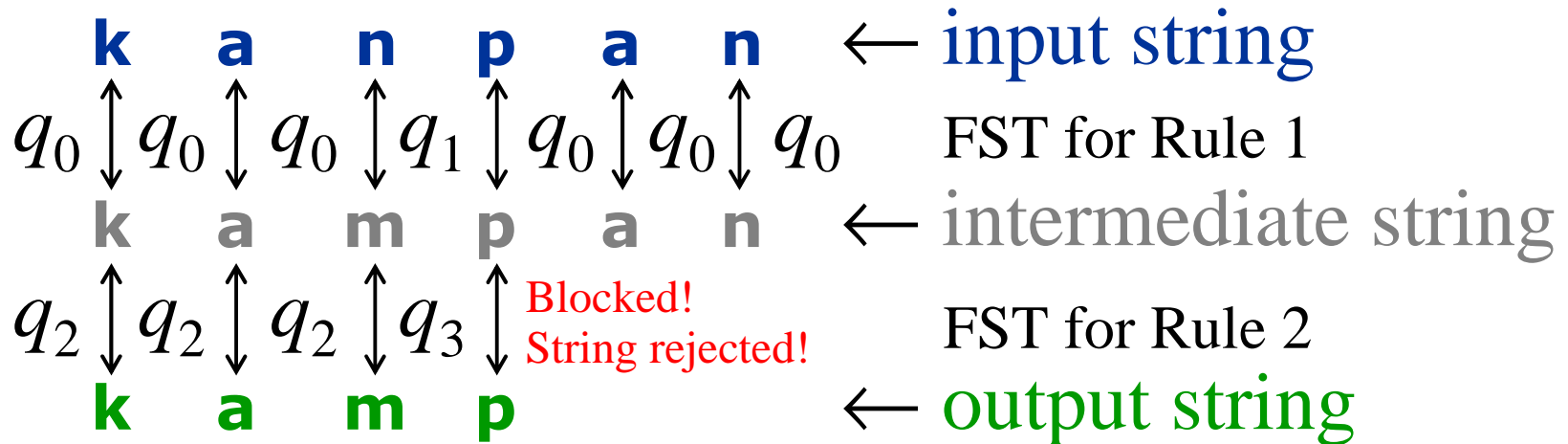


Answer: yes. So the word /**kanpan**/ would be pronounced [**kamman**].

Using Transducers in Phonology



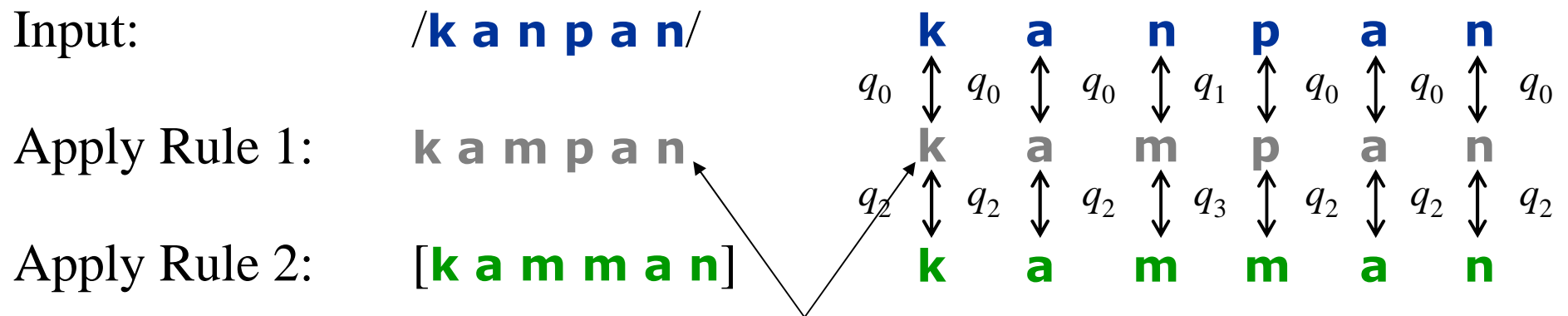
Does Rule 1 followed by Rule 2 accept the pair $\langle \mathbf{kanpan}, \mathbf{kampan} \rangle$?



Answer: no.

The Real Use of Transducers

- Because phonology is regular, transducers are a useful model (unlike in syntax).
- But there is something a little odd...



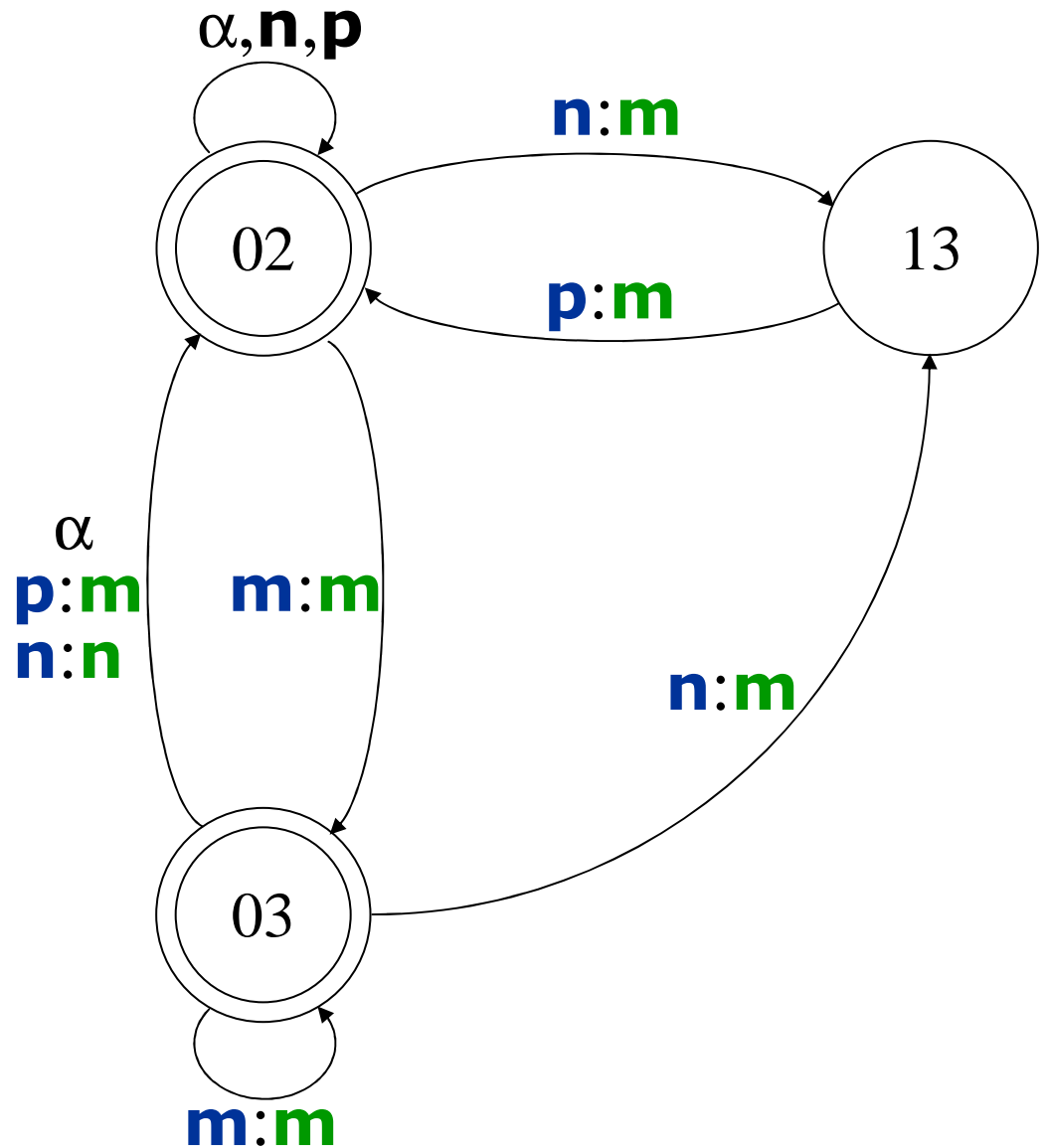
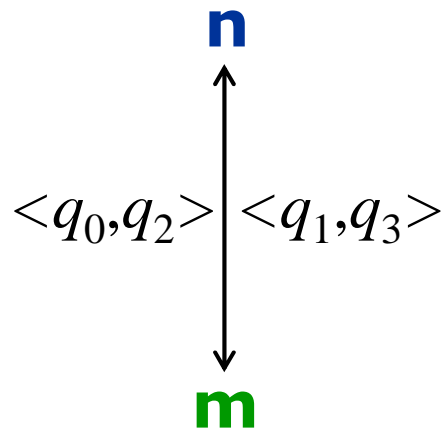
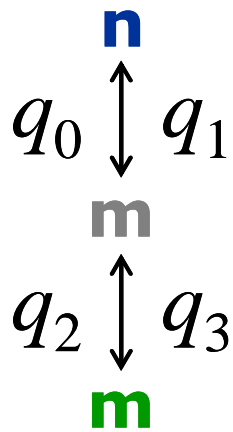
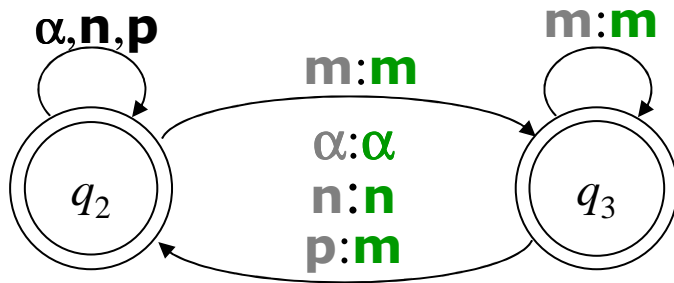
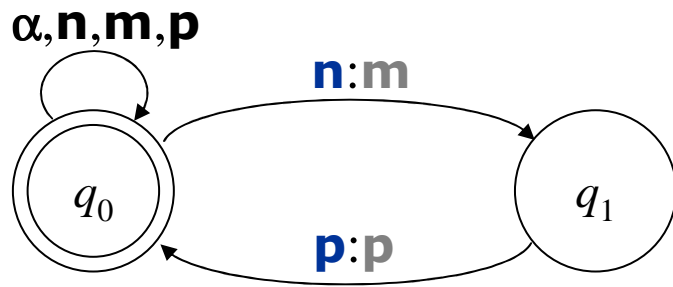
kanpan is the form in the lexicon and **kamman** is the form that gets pronounced.

But what's **kampan**? Why is it part of the process?

The Real Use of Transducers

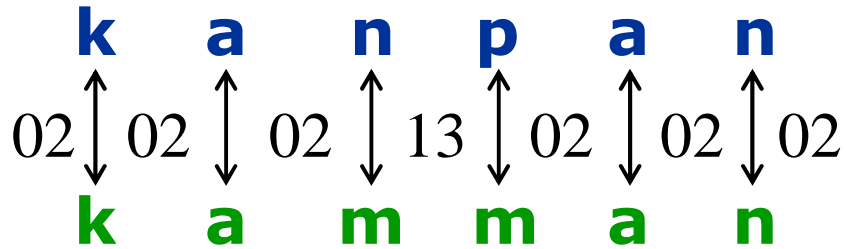
- Combining rules is hard.
 - Rule 1: /**n**/ becomes [**m**] before **p**
 - Rule 2: /**p**/ becomes [**m**] after **m**
 - Rule 1+2: /**np**/ becomes [**mm**] and /**mp**/ becomes [**mm**]
 - OK, that wasn't so hard—but what happens when there are three rules? Ten? More?
 - And of course, 1+2 isn't any easier to compute than 1 followed by 2. You end up having to check each conjunct separately...
 - And it's not really quite as local.
- But combining finite state machines is easy!
 - Actually, finite-state transducers are a little harder to combine than automata. Don't worry about the algorithm. (Unless you really, really like that sort of thing, in which case, come ask me later.) But still easy.
 - And following a single combined machine is much easier than reading a rule with a conjunction in it. (Especially for a computer.)

Combining Transducers

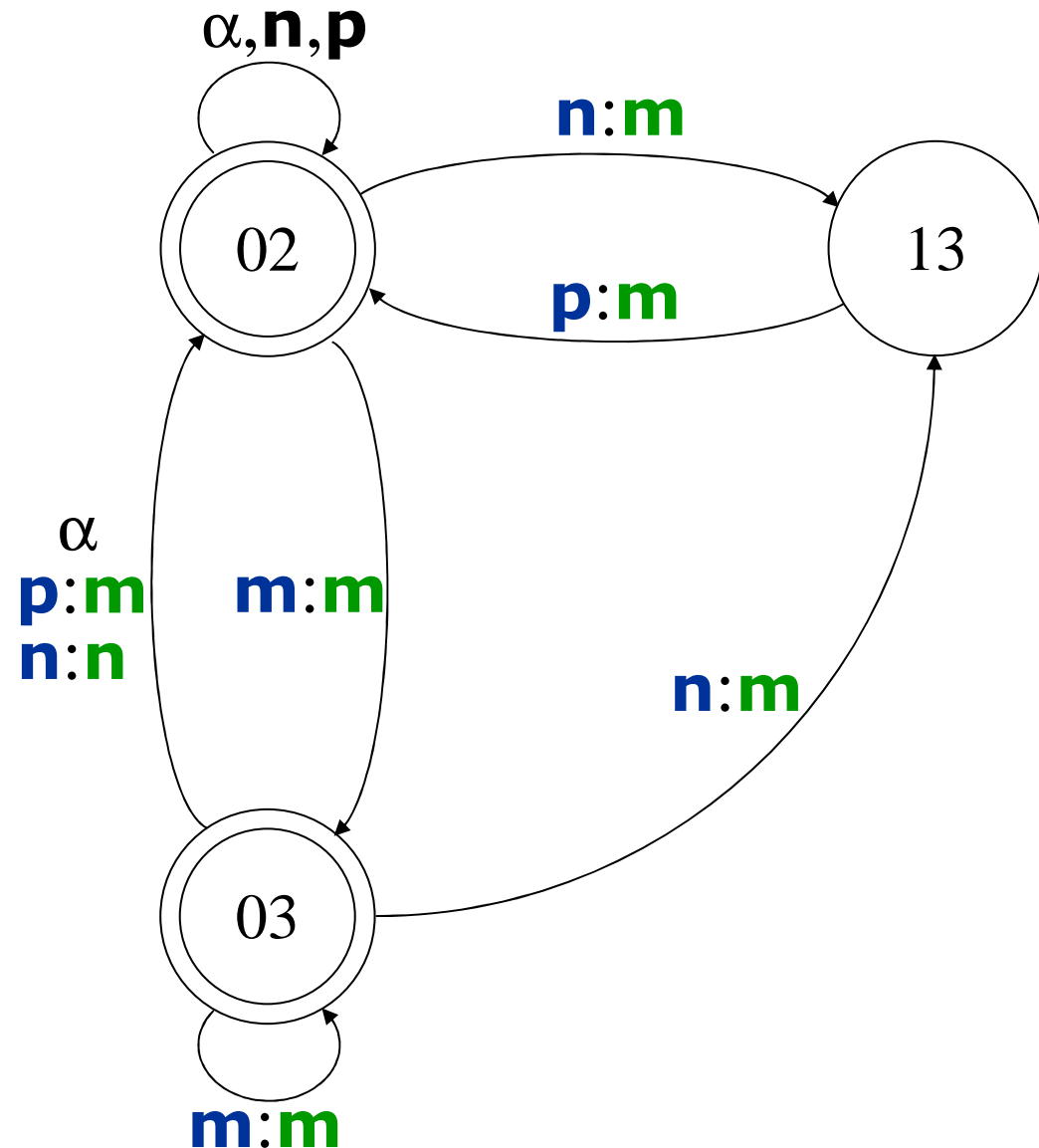
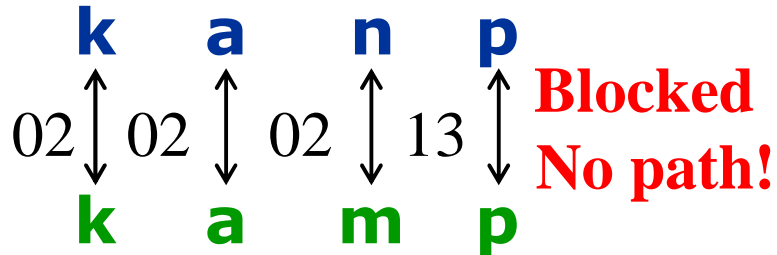


“Two-Level Phonology”

- The pair <kanpan, kamman>:



- The pair <kanpan, kampan>:



Results

- Computationally effective parsing.
 - For a computer to recognize the word [**kamman**], it has to undo the rules and check the result against the stored lexicon. (Is /**kamman**/ a word? Is /**kampan**/, or /**kanpan**/, or /**kan-**/ and /**-pan**/?)
 - Undoing a single FST is much easier than undoing multiple rules.
- New ways of looking at phonology.
 - Intermediate stages (**kampan**) which aren't lexical words (input) and aren't valid spoken words (output) are no longer part of the theory.
 - Sparked a new, non-rule-based way of thinking about phonology.