

# Syntax Done Right

LING 106

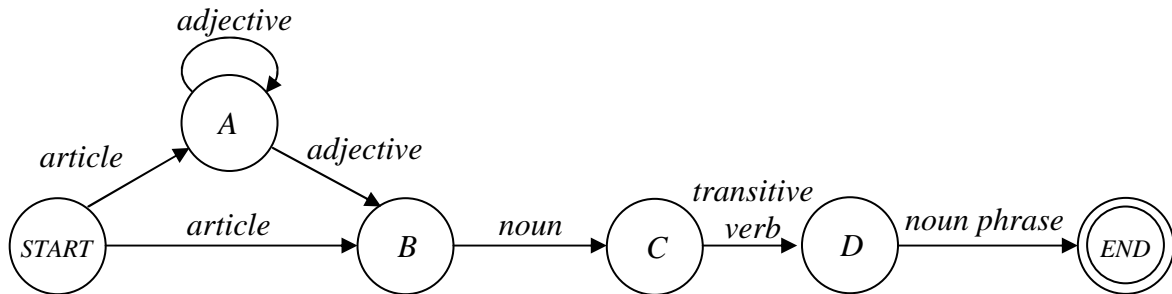
March 25, 2009

## 1. MODELING SYNTAX

So there are things that we now know don't work:

- Listing every string in the language.
- Using very local contexts (i.e., the things a FSA can do)
  - Long distance dependencies
  - Can **is** follow **my parents** in a string? (You can't tell without more context.)

But could this FSA be on the right track?



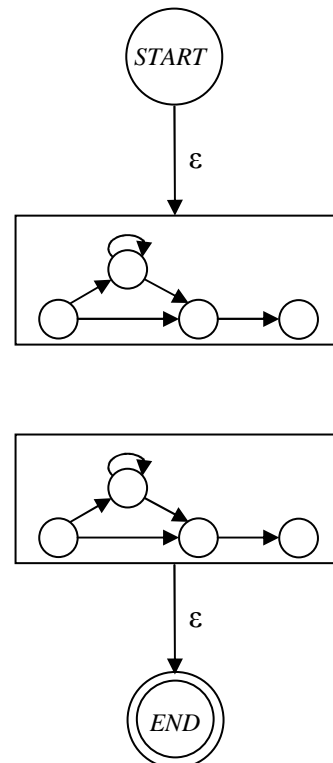
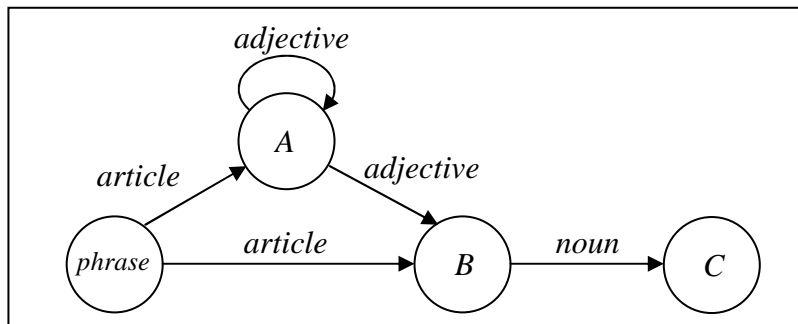
Well, we have...

- **The dog saw a cat.**
- **A cat saw the dog.**
- **The happy dog saw every sad cat.**
- **Every sad cat saw the happy dog.**

That “noun phrase” at the end isn't quite working.

### 1.1. Boxing up the FSA

We want to box up the “noun phrase” part and use it repeatedly:



...except that the box can appear in even more places, e.g.

- **The dog** {slept/barked/walked/jumped/...}
- **The cat** {saw/liked/chased/...} **the dog**.
- **The dog** {saw/liked/chased/...} **the cat**.
- **The cat** slept {on/under/near/...} **the dog**.
- **John** {gave/sold/handed/...} **his cousin the dog**.
- **John** {gave/handed/...} **the dog a biscuit**.

That means repeating the “noun phrase box” in a lot of different places, which is irritatingly hard for a FSA, or right linear grammar...

$$S \rightarrow \textit{Art Adj}^* N \textit{IV}$$
$$S \rightarrow \textit{Art Adj}^* N \textit{TV Art Adj}^* N$$
$$S \rightarrow \textit{Art Adj}^* N \textit{DV Art Adj}^* N \textit{Art Adj}^* N$$

(S = sentence, Art = article, Adj = adjective, N = noun, IV = intransitive verb, TV = transitive verb, DV = ditransitive verb)

...but not context-free grammars!

## 1.2. Turning the box into a CFG

We want is “an article, followed by any number of adjectives, followed by a noun”. Which means we want a rewrite rule:

$$\textit{noun-phrase} \rightarrow \textit{article adjective}^* \textit{noun}$$

or, abbreviating the nonterminals:

$$\textit{NP} \rightarrow \textit{Art Adj}^* N$$

[Note: if you don't like “Adjective\*”, you can use “NP → Art AdjP N”, and then have rules for adjective phrases, i.e. “AdjP → Adj AdjP” and “AdjP → Adj”.]

That means the above rules can be condensed

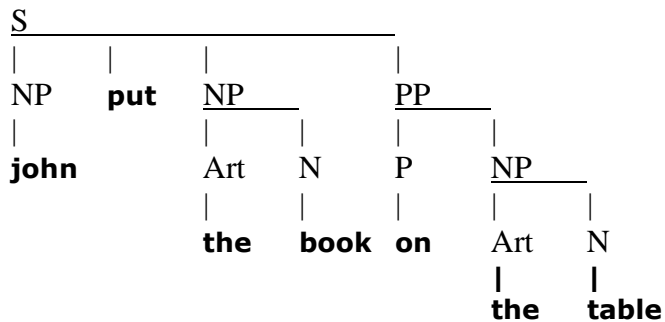
$$S \rightarrow \textit{NP IV}$$
$$S \rightarrow \textit{NP TV NP}$$
$$S \rightarrow \textit{NP DV NP NP}$$

And we can go beyond the NP box—so given both the “cat slept” sentence above, and **John put the book{on/under/near/...} the dog**, we may want:

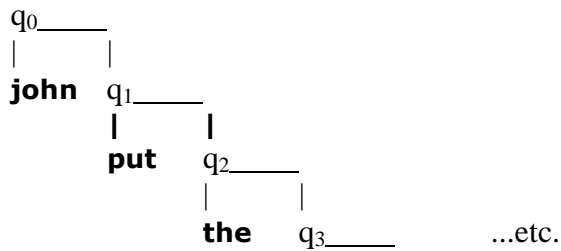
$PP \rightarrow Prep NP$   
 $S \rightarrow S PP$   
 $S \rightarrow NP \text{ put } NP PP$

## 2. USING THE RESULTING STRUCTURES

This gives us a degree of structure we didn't get from FSAs:



vs.



Why have structure? It allows us to talk about...

- Conjunction
  - “John” can be replaced with “John and Mary”
  - table ~ table and shelf
  - the table ~ the table and a nearby shelf
  - on ~ on or near

but not

- \*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 on the table.

but not

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

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

(i.e., it's not just the fact that "Mary" comes between "John" and the pronoun that refers to him; there's something more, concerning the structure)

- Yes/No Questions

Yes/No Questions

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

So to form a YNQ out of

- The girl who is eating ice cream **is** happy.

and assuming we don't even consider "move the third word to the start of the sentence"...which "is" gets moved? The first one? The last one? All of them?

- 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?

# Finite State Transducers

## 3. PHONOLOGY AND MORPHOLOGY

### 3.1. Phonology

Phonology 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<sup>h</sup>ot], [k<sup>h</sup>od], [k<sup>h</sup>otʃ]

...by going through the word and applying rules.

- /k/ becomes [k<sup>h</sup>] at the start of a stressed syllable.

Aside: why have these rules at all? Why not just store **coat**’s pronunciation as [k<sup>h</sup>ot]?

- Because the patterns are completely predictable.
- Because the patterns are productive—that is, given a new word, e.g. a loan word from another language, the rule is applied.
- Because of interaction with morphology (see below).

### 3.2. Morphology

Morphology takes pieces of words (“morphemes”)...

- **-s**: </s/, noun suffix, ‘plural’>

...and combines them with words to make new words, perhaps multiple times.

- **coat + s = coats**
- **industry + -al = industrial**  
**industrial + -ize = industrialize**  
**industrialize + -ation = industrialization**

- Turkish **uygarlaştıramadıklarımızdanmışsınız** =

<b>uygar</b>	<b>laş</b>	<b>tır</b>	<b>ama</b>	<b>dık</b>	<b>lar</b>	<b>ımız</b>	<b>dan</b>	<b>miş</b>	<b>sınız</b>
civilize	become	cause	not able	past partic.	plur.	1 <sup>st</sup> pl. possess	ablative	past	2 <sup>nd</sup> pl.

= “you are from the ones we could not civilize”

Note that phonology operates on the results of the morphology!

- **coats:** /kot/ + /-s/ = [k<sup>h</sup>ots]
- **codes:** /kod/ + /-s/ = [k<sup>h</sup>odz]
- **coaches:** /kotʃ/ + /-s/ = [k<sup>h</sup>otʃəz]
- /-s/ becomes [z] after voiced sounds (/d, b, g, .../; vowels)
- Insert a /ə/ between two sibilants (z, s, ʃ, a few others)

### 3.3. *Successive application of phonological rules*

Phonological rules often follow in a progression—taking the two rules above, for **coaches**:

	Input	/kotʃ/ + /s/
Rule 1: Insert a /ə/...		kotʃəs
Rule 2: /s/ becomes /z/...		kotʃəz
Aspiration Rule		k <sup>h</sup> otʃəz
Output:		<b>k<sup>h</sup>otʃəz</b>

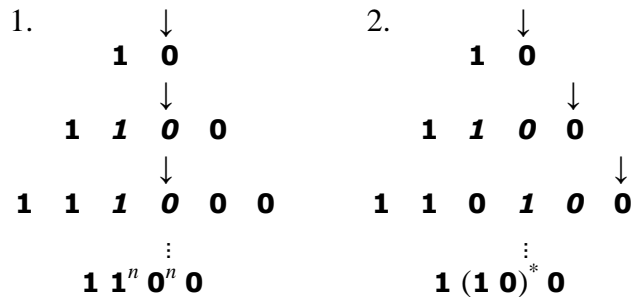
Order matters! If Rules 1 and 2 were switched:

	Input	/kotʃ/ + /s/
Rule 2: /s/ becomes /z/...		kotʃs
Rule 1: Insert a /ə/...		kotʃəs
Aspiration Rule		k <sup>h</sup> otʃəs
Output:		<b>k<sup>h</sup>otʃəs</b> ← wrong!

## 4. JOHNSON'S DISCOVERY

C. Douglas Johnson (1972): a phonological rule proceeds left-to-right through a word, and never goes back and reapplies to the result of an earlier change.

- Consider the optional rule  $\varepsilon \rightarrow \mathbf{10} / \_ \mathbf{0}$ , i.e. “optionally insert a **10** before a **0**”.



In (1), the **10** is inserted before a **0** that resulted from an earlier application of the rule. In (2), the **10** is always inserted before an already-existing **0**.

*Result:* (1) does not give a regular language; (2) does. Because phonological rules are like (2): a finite state machine is sufficiently powerful to model phonology!

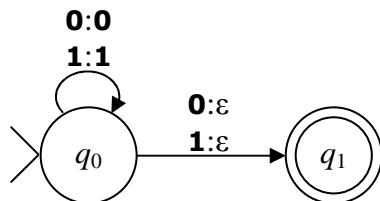
## 5. FINITE STATE TRANSDUCERS

Instead of taking an input string, a finite state 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).
- Or: given an output string, it interprets it as an input string (or strings).

### 5.1. A few examples

Finite state transducer  $T_1$ :



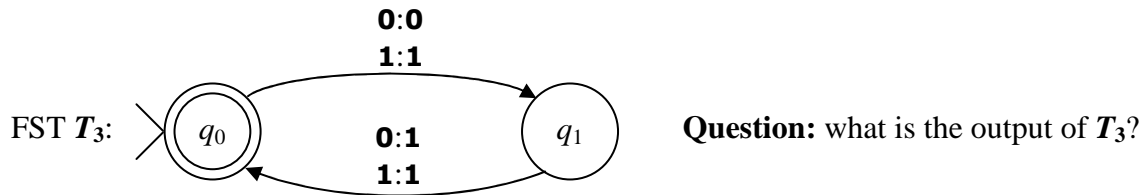
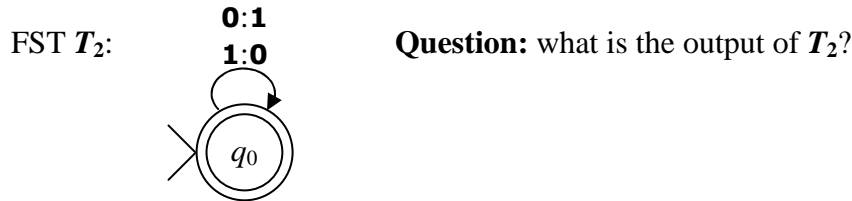
Does this machine accept the pair  $\langle \mathbf{1010}, \mathbf{101} \rangle$ ? Yes, because:

Current State	Input Symbol	Output Symbol	New State
$q_0 (= s)$	<b>1</b>	<b>1</b>	$q_0$
$q_0$	<b>0</b>	<b>0</b>	$q_0$
$q_0$	<b>1</b>	<b>1</b>	$q_0$
$q_0$	<b>0</b>	$\varepsilon$	$q_1$
$q_1$	(end)		ACCEPT

Another way to represent this:

	<b>1</b>		<b>0</b>		<b>1</b>		<b>0</b>	← Input string	
$q_0$	↓	$q_0$	↓	$q_0$	↓	$q_0$	↓	$q_1$	← Current state
	<b>1</b>		<b>0</b>		<b>1</b>		$\epsilon$	← Output string	

Generally:  $T_1$  accepts a pair where the output string is the input string without its final symbol (or: it takes a string and returns it without its last symbol).



## 5.2. Technical 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.  $\Sigma$  is the alphabet;
- c.  $\delta$  is the transition function;
- d.  $s \in Q$  is the start state;
- e.  $F \subseteq Q$  is the set of accept states.

**Definition:** A TRANSITION FUNCTION  $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$  is a mapping from state/input-symbol/output-symbol triplets to sets of states.

**Definition:** A REGULAR RELATION is a relation—i.e. a set of pairs of strings—that can be modeled by a FST. (Johnson's discovery: natural language phonology is a regular relation.)

### 5.3. Example and application

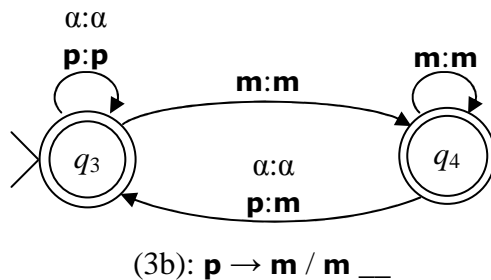
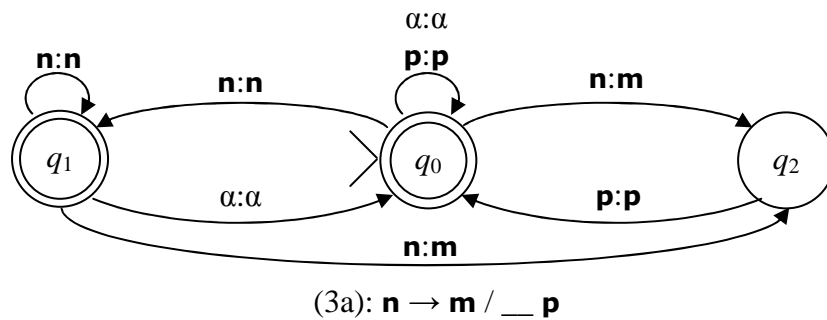
Take the following rules (not from an actual language, but from an example by Lauri Karttunen), which apply in the given order:<sup>1</sup>

- (3) a.  $n \rightarrow m / \_ p$  (i.e., “n becomes m before a p”)  
 b.  $p \rightarrow m / m \_$  (i.e., “p becomes m after an m”)

A phonological derivation for the input string /**kanpan**/:

Input: **k a n p a n**  
 Apply (3a): **k a m p a n** ← Intermediate, unpronounced stage  
 Apply (3b): **k a m m a n** ...which is the output, i.e. what gets pronounced.

So instead, let’s write the rules in (3) as FSTs. Here,  $\alpha$  represents anything other than **p** or **n**.



We could then follow the two transducers in succession. <**kanpan**, **kamman**> is accepted:

	<b>k</b>	<b>a</b>	<b>n</b>	<b>p</b>	<b>a</b>	<b>n</b>			
$q_0$	$\downarrow$	$q_0$	$\downarrow$	$q_2$	$\downarrow$	$q_0$	$\downarrow$	$q_0$	<i>Input string</i>
	<b>k</b>	<b>a</b>	<b>m</b>	<b>p</b>	<b>a</b>	<b>n</b>			<i>FST (3a)</i>
									<i>Intermediate string</i>
$q_3$	$\downarrow$	$q_3$	$\downarrow$	$q_4$	$\downarrow$	$q_3$	$\downarrow$	$q_3$	<i>FST (3b)</i>
	<b>k</b>	<b>a</b>	<b>m</b>	<b>m</b>	<b>a</b>	<b>n</b>			<i>Output string</i>

<sup>1</sup> The example here is slightly more complex than the example in class—the machine in (3a) is more precise.

But  $\langle \text{kanpan, kampan} \rangle$  is not:

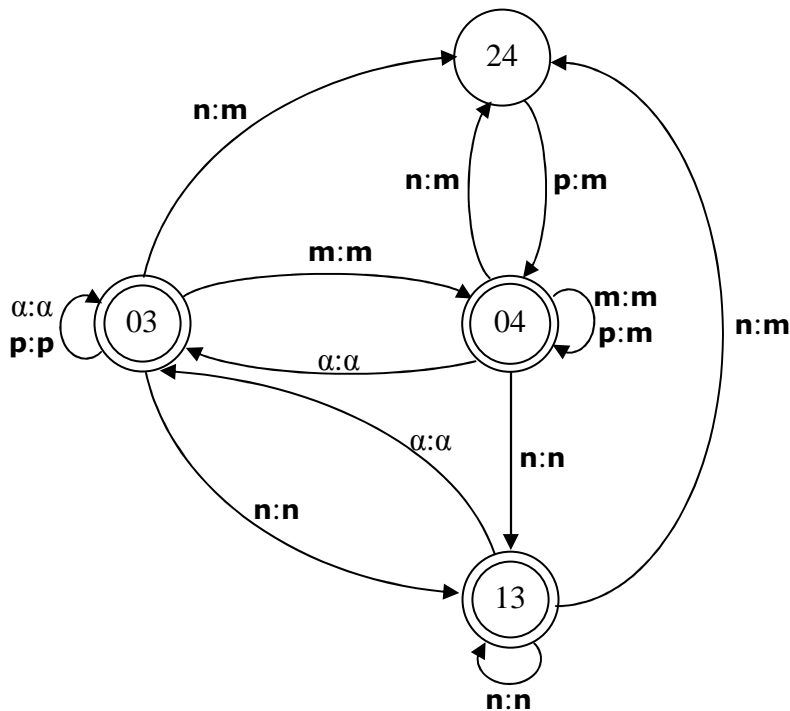
	<b>k</b>	<b>a</b>	<b>n</b>	<b>p</b>	<b>a</b>	<b>n</b>	<i>Input string</i>
$q_0$	$\downarrow$	$q_0$	$\downarrow$	$q_2$	$\downarrow$	$q_0$	FST (3a)
	<b>k</b>	<b>a</b>	<b>m</b>	<b>p</b>	<b>a</b>	<b>n</b>	<i>Intermediate string</i>
$q_3$	$\downarrow$	$q_3$	$\downarrow$	$q_4$	$\downarrow$		FST (3b)
	<b>k</b>	<b>a</b>	<b>m</b>	<b>p</b>			<i>Output string</i>
					BLOCKED!		
					STRING REJECTED!		

#### 5.4. So: why use a FST?

Answer: because it allows us to easily combine the two FSTs above into a single FST, thereby setting up a single input/output correspondence.

*Sidebar:* at this point, we technically need a proof that operations are closed on FSTs the way they are for FSAs, preferably with the same kind of proof-by-construction.

We're going to skip that, because it's a little tedious. But the same basic principles apply—each state in the new machine corresponds to a pair from the old machines, and each transition  $x:z$  corresponds to transitions  $x:y$  and  $y:z$  from the old machines.



A FST that recognizes both rules in (3) simultaneously

Result of running <**kanpan, kamman**> through the above machine:

	<b>k</b>		<b>a</b>		<b>n</b>		<b>p</b>		<b>a</b>		<b>n</b>		<i>Input string</i>
03	↑	03	↑	03	↑	24	↑	04	↑	03	↑	13	FST
	<b>k</b>		<b>a</b>		<b>m</b>		<b>m</b>		<b>a</b>		<b>n</b>		<i>Output string</i>

Result of running <**kanpan, kampan**> through the above machine:

	<b>k</b>		<b>a</b>		<b>n</b>		<b>p</b>		<b>a</b>		<b>n</b>		<i>Input string</i>
03	↑	03	↑	03	↑	24	↑	BLOCKED!					FST
	<b>k</b>		<b>a</b>		<b>m</b>		<b>p</b>	NO PATH!					<i>Output string</i>

Results:

- Computationally effective parsing. Note that the output string **kamman** corresponds to input strings **kanpan, kampan, kamman**.

Suppose you have a computer that has **kan-** and **-pan** in its lexicon; if it hears **kamman**, you want it to recognize it as **kan+pan**, and to make sure that no other alternative is valid. That means undoing the rules—and undoing a single FST is computationally much easier than holding onto possibilities while undoing multiple rules.

- New ways of looking at phonology. Intermediate stages—e.g., **kampan**, or in the English plural example from earlier, **kotʃəs**—which are neither part of the lexicon (input) nor part of the spoken language (output) are no longer needed as part of the theory.

This *two-level phonology* led the way to a wholly new approach in the 1990s...but that's another topic for another course.