Lecture 10
Morphology and Finite State Transducers

Intro to NLP, CS585, Fall 2014

http://people.cs.umass.edu/~brenocon/inlp2014/

Brendan O'Connor (http://brenocon.com)

- PS2 out tonight: due next Friday at midnight
- David will post OH for early next week
- I might have to change mine (next Thurs)

Word-internal structure

- What's inside a word?
  - *walk, walked, walks:* Our models think these are different. Should they be?
  - Motivations: less sparsity, deeper understanding
- Orthography: spelling conventions
  - *"blacke as the night"* vs *"black as the night"*
    Not really linguistically meaningful
  - *yesss* vs *yesssssssss*
    (Linguistically meaningful?...)
- Morphology: linguistically productive
  - Rule-based approaches are very common here

Morphology: Phenomena

- Inflection
- Derivation
- Compounding
- Cliticization

- Inflectional morphology: modify root to a word of the same class, due to grammatical constraints like agreement
- e.g. regular verbs. (Exceptions?)

Morphological Form Classes	Regularly Inflected Verbs			
stem	walk	merge	try	map
-s form	walks	merges	tries	maps
-ing participle	walking	merging	trying	mapping
Past form or *-ed* participle	walked	merged	tried	mapped

- English is relatively simple

- Derivational morphology: modify root to a word of a different class
  - *derivational*
    derive -ation -al
- Can be tricky
  - *universe --> uni- verse ?*
  - *universal --> uni- verse -al ???*

- Compounding
  - *baseball desktop*

- Cliticization

Full Form	Clitic	Full Form	Clitic
am	’m	have	’ve
are	’re	has	’s
is	’s	had	’d
will	’ll	would	’d

Full morph. parsing

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	‘ducks’
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	‘duck’
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	‘I drink’
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	‘I sing’
goose	goose +N +Sg	canto	canto +N +Masc +Sg	‘song’
goose	goose +V	puse	poner +V +Perf +1P +Sg	‘I was able’
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	‘he/she came’
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	‘wine’
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	‘place’
caught	catch +V +Past			

Figure 3.2 Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

- Need
  - 1. Lexicon of stems and affixes (maybe guess stems...)
  - 2. Morphotactics: morpheme ordering model
  - 3. Orthographic rules: spelling changes upon combination (city + -s --> citys -> cities)

Lexicons

- Noun inflection: need three word lists for nouns
  - regular nouns
  - irregular plural nouns
  - irregular singular nouns
- FSA represents all inflected nouns: don't have to store plural forms.
- Abbreviated form below (What does the full FSA look like?)

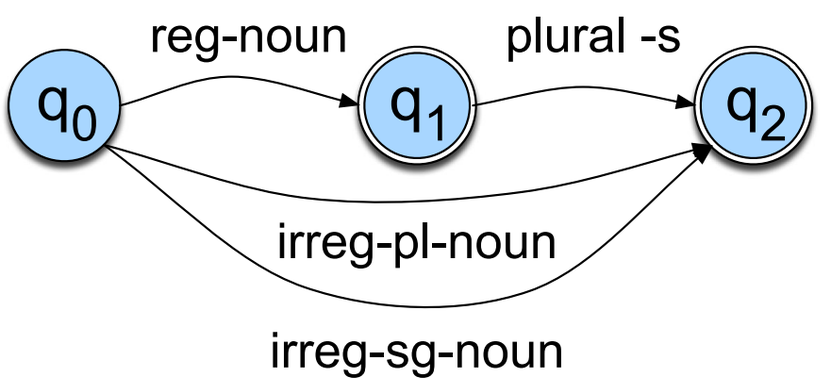

Figure 3.3 A finite-state automaton for English nominal inflection.

Derivaional morph. example

- Derivational data we want to model:
  adjectives become opposites, comparatives, adverbs

big, bigger, biggest,	cool, cooler, coolest, coolly
happy, happier, happiest, happily	red, redder, reddest
unhappy, unhappier, unhappiest, unhappily	real, unreal, really
clear, clearer, clearest, clearly, unclear, unclearly	

Derivaional morph. example

- Derivational data we want to model:
  adjectives become opposites, comparatives, adverbs

big, bigger, biggest,	cool, cooler, coolest, coolly
happy, happier, happiest, happily	red, redder, reddest
unhappy, unhappier, unhappiest, unhappily	real, unreal, really
clear, clearer, clearest, clearly, unclear, unclearly	

- Task: recognition. Proposed model

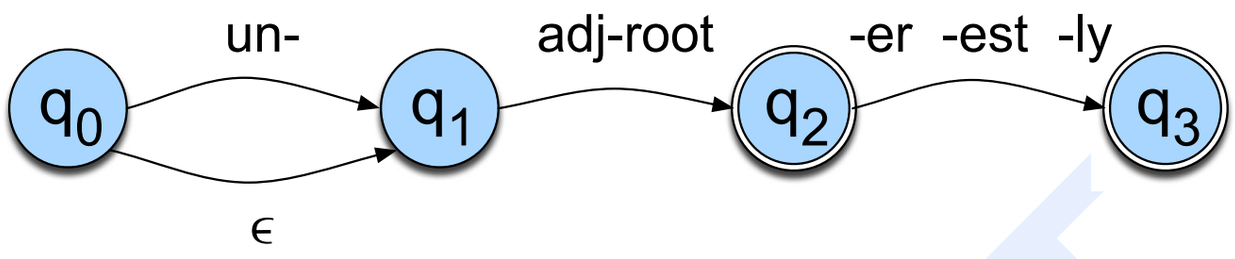

Figure 3.5 An FSA for a fragment of English adjective morphology: Antworth's Proposal #1.

- Any false positives? (Compare: Child language learning)

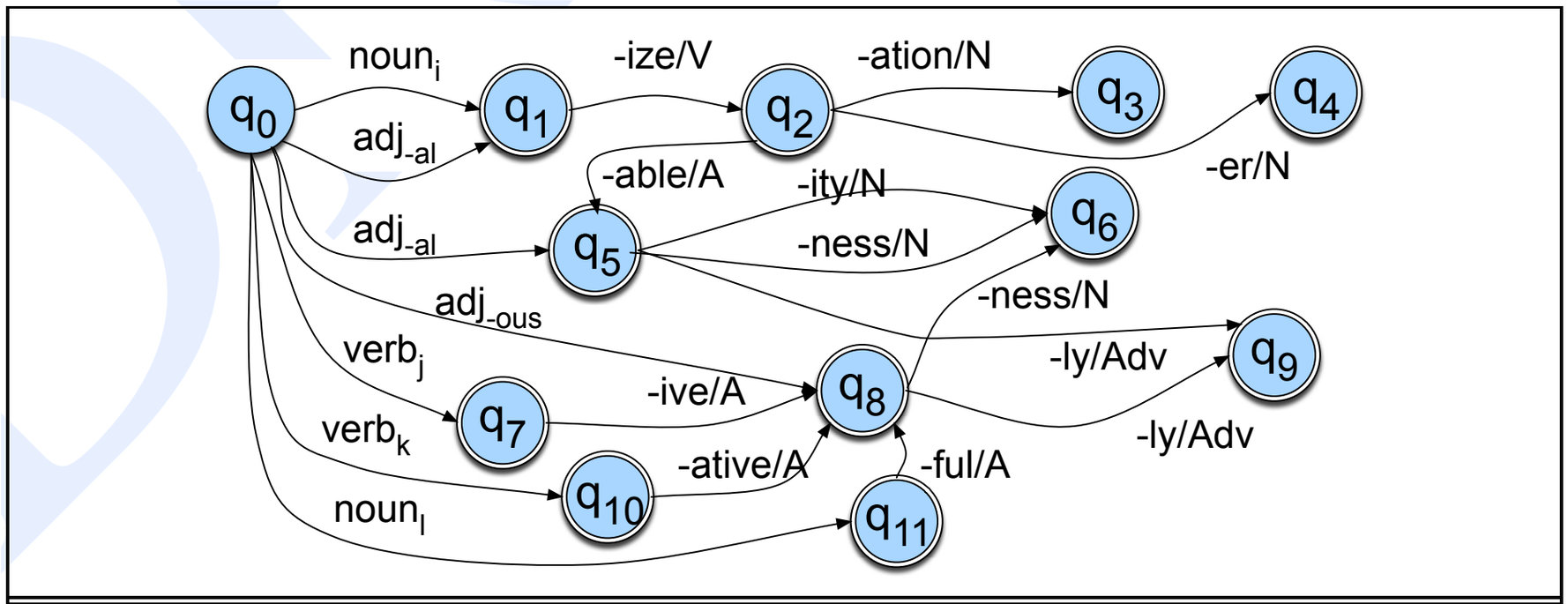

Figure 3.6 An FSA for another fragment of English derivational morphology.

Recognition vs Parsing

- Morphological recognition
  - is_pasttense_verb(*loved*) --> TRUE
  - Finite State Automata can do this
- Morphological parsing: what is its breakdown?
  - parse(*loved*) --> *take*/VERB *-n*/PAST-TENSE
  - Finite State Transducers can do this

Finite State Transducers

- FSAutomata
  - An FSA represents a set of strings. e.g.
    {*walk, walks, walked, love loves, loved*}
    - *Regular language.*
  - A *recognizer* function.
    recognize(str) -> true or false
- FSTransducers
  - An FST represents a set of *pairs of strings* (think of as input,output pairs)
    { *(walk, walk+V+PL), (walk, walk+N+SG), (walked, walk+V+PAST)* ...}
    - *Regular relation.* (Not a function!)
  - A *transducer* function: maps input to zero or more outputs.
    transduce(*walk*) --> {*walk+V+PL, walk+N+SG*}
    Can return multiple answers if ambiguity: e.g. if you don't have POS-tagged input, "walk" could be the verb "They walk to the store" versus the noun "I took a walk".
  - Generic *inversion* and *composition* operations.

Finite State Transducers

- FSAutomata have *input* labels.
  - One input tape
- FSTransducers have *input:output* pairs on labels.
  - Two tapes: input and output.

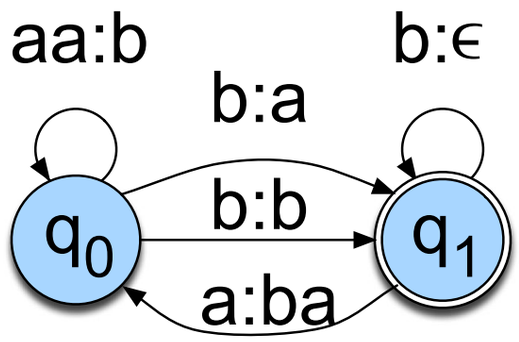

Figure 3.8 A finite-state transducer, modified from Mohri (1997).

Finite State Transducers

- FSAutomata have *input* labels.
- FSTransducers have *input:output* pairs on labels.

Q	a finite set of N states $q_0, q_1, \ldots, q_{N-1}$
Σ	a finite set corresponding to the input alphabet
$q_0 \in Q$	the start state
$F \subseteq Q$	the set of final states
$\delta(q, w)$	the transition function or transition matrix between states; Given a state $q \in Q$ and a string $w \in \Sigma^*$, $\delta(q, w)$ returns a set of new states $Q' \in Q$. δ is thus a function from $Q \times \Sigma^*$ to 2^Q (because there are 2^Q possible subsets of Q). δ returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.

Finite State Transducers

- FSAutomata have *input* labels.
- FSTransducers have *input:output* pairs on labels.

New →

Q	a finite set of N states $q_0, q_1, \ldots, q_{N-1}$
Σ	a finite set corresponding to the input alphabet
Δ	a finite set corresponding to the output alphabet
$q_0 \in Q$	the start state
$F \subseteq Q$	the set of final states
$\delta(q,w)$	the transition function or transition matrix between states; Given a state $q \in Q$ and a string $w \in \Sigma^*$, $\delta(q,w)$ returns a set of new states $Q' \in Q$. δ is thus a function from $Q \times \Sigma^*$ to 2^Q (because there are 2^Q possible subsets of Q). δ returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.

New → $\sigma(q,w)$ the output function giving the set of possible output strings for each state and input. Given a state $q \in Q$ and a string $w \in \Sigma^*$, $\sigma(q,w)$ gives a set of output strings, each a string $o \in \Delta^*$. σ is thus a function from $Q \times \Sigma^*$ to 2^{Δ^*}

Inversion

- **inversion**: The inversion of a transducer T (T^{-1}) simply switches the input and output labels. Thus if T maps from the input alphabet I to the output alphabet O, T^{-1} maps from O to I.

$$T = \{ (a, a1), (a, a2), (b, b1), (c, c1), (c, a) \}$$

$$T^{-1} = \{ (a1, a), (a2, a), (b1, b), (c1, c), (a, c) \}$$

Composition

- **composition**: If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from O_1 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2.

- As transducer functions,
  (T1 o T2)(x) = T1(T2(x))

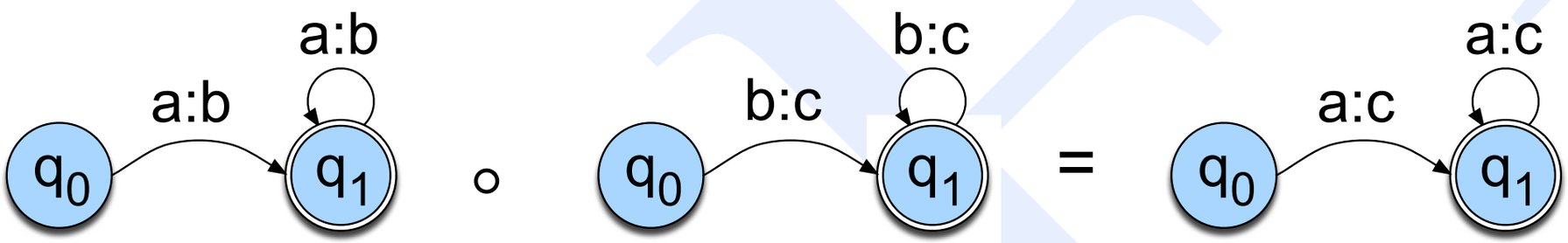

Figure 3.9 The composition of `[a:b]+` with `[b:c]+` to produce `[a:c]+`.

Generic FST operations

- There exist generic algorithms for inversion and composition (and minimization...)
- Chain together many transducers
- e.g. OpenFST open-source library

FSTs for morph parsing

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	‘ducks’
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	‘duck’
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	‘I drink’
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	‘I sing’
goose	goose +N +Sg	canto	canto +N +Masc +Sg	‘song’
goose	goose +V	puse	poner +V +Perf +1P +Sg	‘I was able’
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	‘he/she came’
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	‘wine’
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	‘place’
caught	catch +V +Past			

Figure 3.2 Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

FSTs for morph parsing

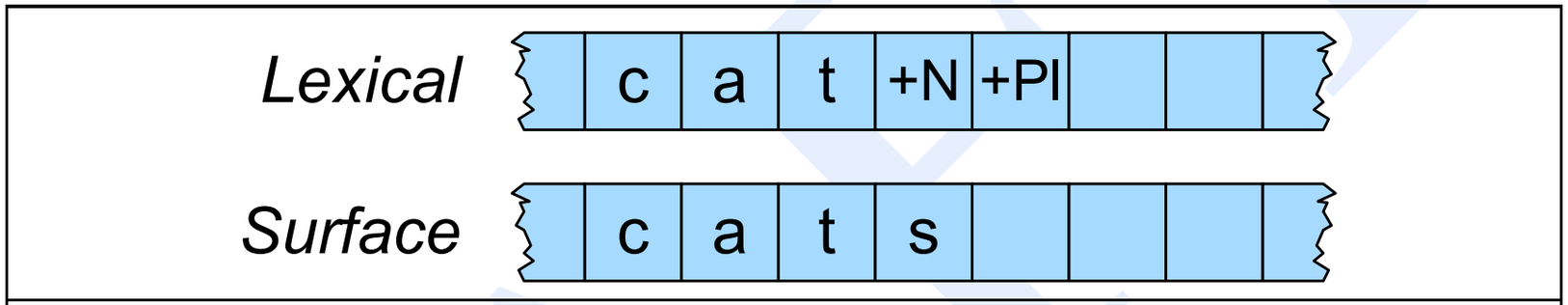

Figure 3.12 Schematic examples of the lexical and surface tapes; the actual transducers will involve intermediate tapes as well.

reg-noun	irreg-pl-noun	irreg-sg-noun
fox	g o:e o:e s e	goose
cat	sheep	sheep
aardvark	m o:i u:ϵ s:c e	mouse

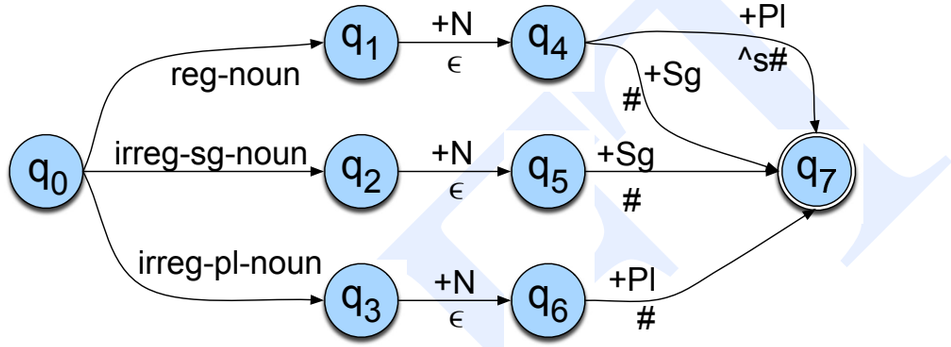

Figure 3.13 A schematic transducer for English nominal number inflection T_{num}. The symbols above each arc represent elements of the morphological parse in the lexical tape; the symbols below each arc represent the surface tape (or the intermediate tape, to be described later), using the morpheme-boundary symbol ^ and word-boundary marker #. The labels on the arcs leaving q_0 are schematic, and need to be expanded by individual words in the lexicon.

Compose

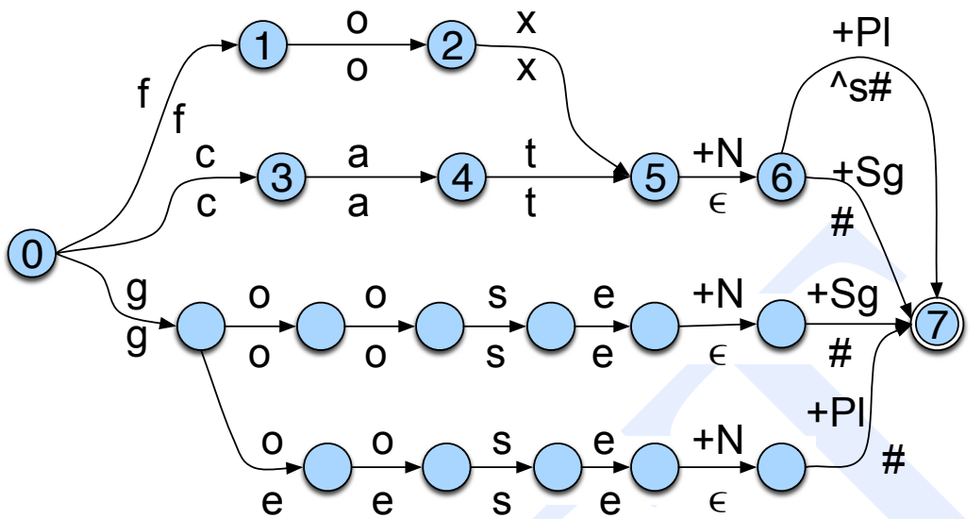

Figure 3.14 A fleshed-out English nominal inflection FST T_{lex}, expanded from T_{num} by replacing the three arcs with individual word stems (only a few sample word stems are shown).

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing*/*-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*,*-z*,*-x*,*-ch*, *-sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

Other applications of FSTs

- Spelling correction: make an FST that can capture common mistakes
  - Transpositions: *teh* -> *the*
- Speech recognition: phonemes, pronunciation dictionaries, words...
  - *OpenFST* library: originally developed for speech applications
- Machine translation: Model 1 can be thought of as an FST!
- Usually use *weighted* FSTs: probabilities of spelling errors, word/phrase translations, etc.

Stemming: simple morph. parse

- Porter stemmer: cascade of rewrite rules.
  Output of one stage is the input for the next

ATIONAL → ATE (e.g., relational → relate)
ING → ϵ if stem contains vowel (e.g., motoring → motor)

- Can be thought of as a lexicon-free FST
- Is stemming always good? Depends on language, amount of morph. productiveness, and data size

Errors of Commission		Errors of Omission	
organization	organ	European	Europe
doing	doe	analysis	analyzes
generalization	generic	matrices	matrix
numerical	numerous	noise	noisy
policy	police	sparse	sparsity