Lemmatization, Automata and Transducers

Chapters 2 – 3 J&M'09

Lemmatization

Lemmatization: deep decompositional analysis of word structure.

Basics:

- List all simple words (lemmas) in a database (vocabulary)
- Decompose complex words into a lemma and affixes

Finite-State Morphological Parsing

- Automata
- Transducers

Morphology

There are four kinds of affixes:

(1) a. prefixes (*un-*, *re-*, *anti-*, ...)

b. suffixes (*-ed*, *-ing*, *-s*, ...)

c. circumfixes (*em-...-en* in '**em**bold**en**' & '**em**bigg**en**')

d. infixes (affix inside root, similar to 'abso-fucking-lutely')

Morphology

Derivational affixes radically change the root meaning

Affix	**Category change**	**Examples**
-able	verb ⇝ adjective	fixable, doable, understandable
-ive	verb ⇝ adjective	assertive, impressive, restrictive
-er	verb ⇝ noun	teacher, worker
-ful	noun ⇝ adjective	faithful, hopeful, dreadful
-en	adjective ⇝ verb	deaden, blacken, harden
-ize	adjective ⇝ verb	modernize, nationalize
-ly	adjective ⇝ adverb	quietly, slowly, carefully
-ness	adjective ⇝ noun	sadness, badness
...	...	...
de-	verb ⇝ verb	deactivate, demystify
dis-	verb ⇝ verb	discontinue, disobey
re-	verb ⇝ verb	rethink, redo, restate
un-	verb ⇝ verb	untie, unlock
un-	adjective ⇝ adjective	unhappy, unfair
...	...	...

Derivation is usually fairly restricted

- *-ant* combines with roots of Latin origin:

  (2) a. assist+ant
  b. combat+ant

  not with roots of Germanic origin:

  (3) a. *help+ant
  b. *fight+ant

- *-en* combines with a monosyllabic root ending in a stop:

  (4) a. soft+en, white+en, mad+en
  b. *blue+en, *angry+en, *slow+en

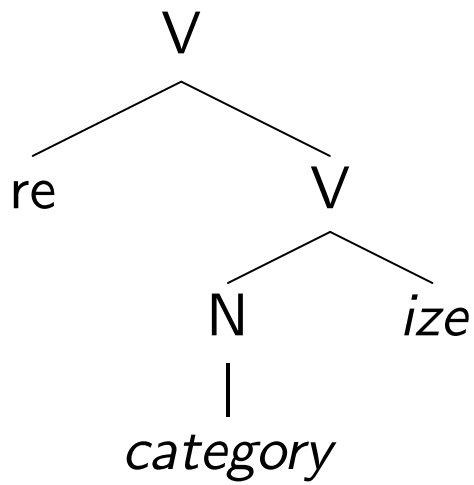
- degree *-er* and *-est* combine with monosyllabic adjectives:

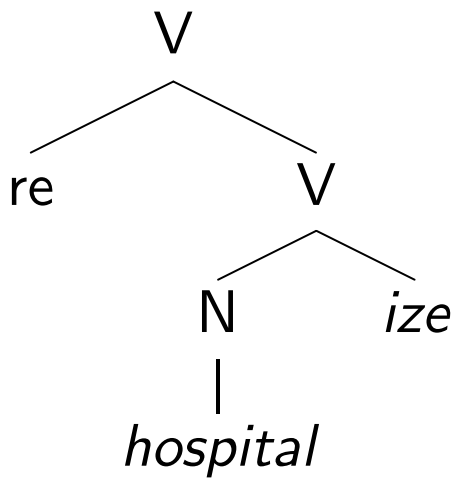
  (5) a. quick+er, quick+est
  b. fast+er, fast+est
  c. *rapid+er, *rapid+est
  d. *interesting+er, *interesting+est

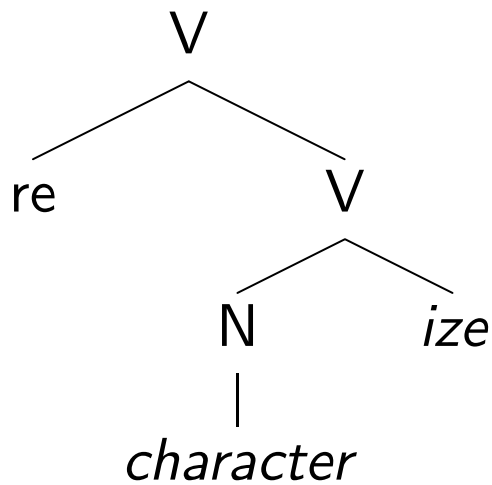
Inflectional affixes do not radically change the root

Affix	Example
Plural *-s*	the cats
3rd person singular present *-s*	he jumps
Progressive *-ing*	he is jumping
Past tense *-ed*	he jumped
Past participle *-en/-ed*	He has eaten / jumped
Comparative *-er*	the smaller one
Superlative *-est*	the smallest one
Possessive *-'s*	the cat's tail

Morphological analysis (V = verb; N = noun):







We can tell that -*ize* attaches before -*re* :

**recategory*, **rehospital*, **recharacter*

Derivational rules:

re + V = V

N + ize = V

Morphology

Sometimes the orthography changes:

(6) a. run + ing = running

b. leave + ing = leaving

c. Un + imagine + ative + ly = unimaginatively

And sometimes it may look like there are affixes but there are not:

(7) a. deceive ¬ = de + ceive

b. receive ¬ = re + ceive

c. permit ¬ = per + mit

d. tenable ¬ = ten + able

Morphology

Allomorph: alternative phonetic realization of a morpheme.

- Plural 's' has three allomorphs:

  [s] in *cat***s**, *dart***s**, *cop***s**, *cuff***s**
  [z] in *shoe***s**, *dog***s**, *law***s**, *car***s**
  [əz] in *hors***es**, *dish***es**, *bush***es**

- Past tense suffix has three allomorphs:

  [d] in *love***d**, *answer***ed**, *pull***ed**, *plann***ed**
  [t] in *kiss***ed**, *jump***ed**, *laugh***ed**
  [əd] in *state***d**, *rate***d**, *treat***ed**

- Negation prefix has five allomorphs:

  [im] in **im***possible*, **im***balance*
  [iŋ] in **in***complete*
  [iɹ] in **ir***responsible*
  [il] in **il***legible*
  [in] in **in***excusable*

Morphology

At the level of characters, English spelling is particularly unruly:

(8) a. **f**our, a**ff**ection, **ph**ase, enou**gh**
b. t**o**, t**oo**, t**wo**, thr**ough**, thr**ew**, cl**ue**, sh**oe**
c. alth**ough**, d**ough**, gr**ow**, b**oa**t, n**o**
d. d**a**me, d**a**d, f**a**ther, c**a**ll, vill**a**ge, m**a**ny
e. h**ea**d, h**ea**t, h**ea**rd,
f. S**u**zy, b**u**sy, **u**nion, f**u**dge

(9) a. ret**ain**, Britt**ain**
b. bl**ea**k, br**ea**k
c. h**ow**, l**ow**
d. d**augh**ter, l**augh**ter
e. w**ind**, m**ind**
f. bl**ood**, f**ood**
g. m**ould**, w**ould**
h. t**oward**, f**orward**

Morphology

(10) a. r**ou**nded, w**ou**nded
b. s**o**me, h**o**me
c. sh**o**es, g**o**es
d. fr**ie**nd, f**ie**nd
e. i**vy**, pri**vy**
f. r**i**ver, r**i**val, s**i**t
g. t**om**b, b**om**b, c**om**b
h. str**a**nger, **a**nger
i. fin**g**er, gin**g**er
j. f**u**ry, b**u**ry
k. m**i**nt, p**i**nt
l. h**ea**ve, h**ea**ven
m. t**ou**r, t**ou**r, **ou**r
n. crev**i**ce, dev**i**ce
o. **ea**r, **ea**rn, **u**rn
p. s**e**ven, **e**ven

Morphology

Type	**# Languages**
Little or no inflectional morphology	141
Predominantly suffixing	406
Moderate preference for suffixing	123
Approximately equal amounts of suffixing and prefixing	147
Moderate preference for prefixing	94
Predominantly prefixing	58
Total (source: WALS)	969

Morphology

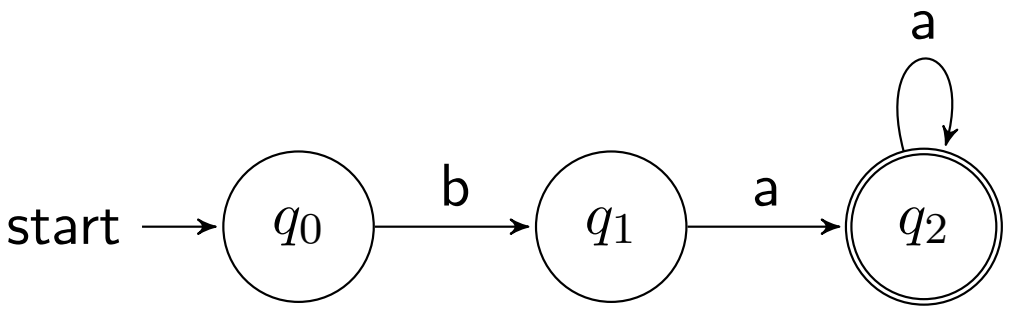
Language	**average # of morphemes per word**
Vietnamese	1.06
Yoruba	1.09
English	1.68
Old English	2.12
Swahili	2.55
Turkish	2.86
Russian	3.33
Inuit (Eskimo)	3.72

Finite-State Automata

A FSA has five parts:

- A finite set of states (e.g. $\{q_0, q_1, q_2\}$)
- A finite input set of symbols (e.g. $\{a, b\}$)
- A start state q_0
- A set of final states (e.g. $\{q_2\}$)
- A set of transitions (e.g. $\{\delta(q_0, b, q_1), \delta(q_1, a, q_2), \delta(q_2, a, q_2)\}$)

Graphically:



This FSA accepts the following strings:
ba, *baa*, *baaa*, *baaaa*, *baaaaaa*, *baaaaaaa*, etc..

Finite-State Automata

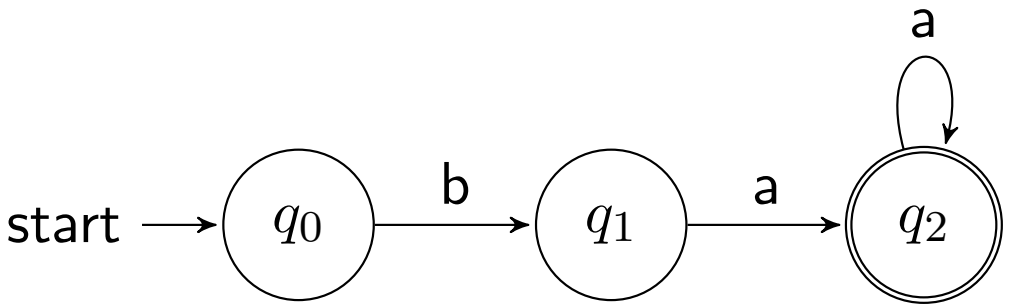
Prolog allows a direct implementation of FSA:

```
1 q0([b|L]):- q1(L).
2 q1([a|L]):- q2(L).
3 q2([a|L]):- q2(L).
4 q2([]).
```

Execution:

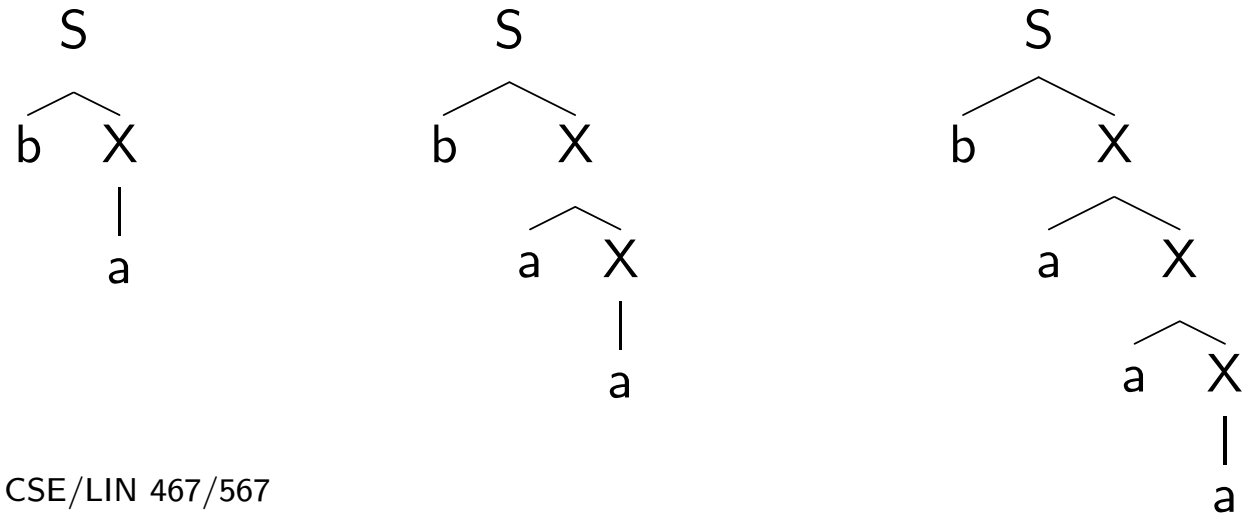
```
1 ?- q0([b,a,a,a,a,a,a,a,a,a]).
2 yes
3
4 ?- q0([b,a,a,a,a,a,a,a,a,a,b]).
5 no
6
7 ?- q0([baaa]).
8 no
```

Finite-State Automata



Every FST is equivalent to some Type-3 grammar in the ChomskySchützenberger hierarchy:

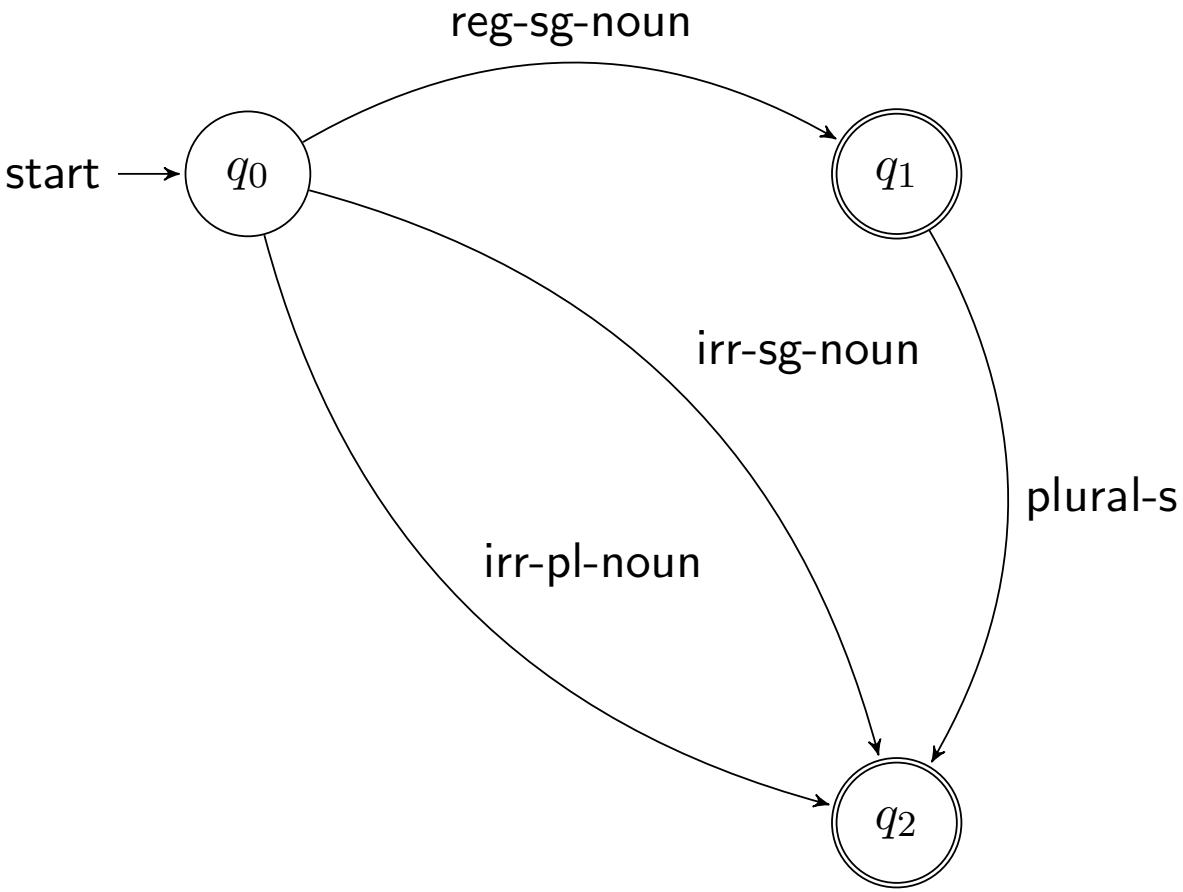
S → b X
X → a
X → a X



Excursion: the **Chomsky-Schützenberger hierarchy**

Language	**Grammar**	**Automaton**
Regular	$A \rightarrow a$, $A \rightarrow aB$	Finite state machine
Context-free	$A \rightarrow \gamma$	Non-deterministic pushdown automaton
Context-sensitive	$\gamma A \beta \rightarrow \alpha \gamma \beta$	Linear-bounded non-deterministic Turing machine
Recursively enumerable	$\alpha \rightarrow \beta$	Turing machine

FSA with word class labels for recognition of nouns



reg-sg-noun = Ns that inflect regularly: *cat*, *fox*, *boat*, *idea*, ...
irr-sg-noun = Ns that inflect irregularly: *mouse*, *goose*, *sheep*, *moose*, *die*
plural-s = plural suffixes: *s*, *es*
irr-pl-noun = irregularly inflected Ns *mice*, *geese*, *sheep*, *moose*, *dice*, ...

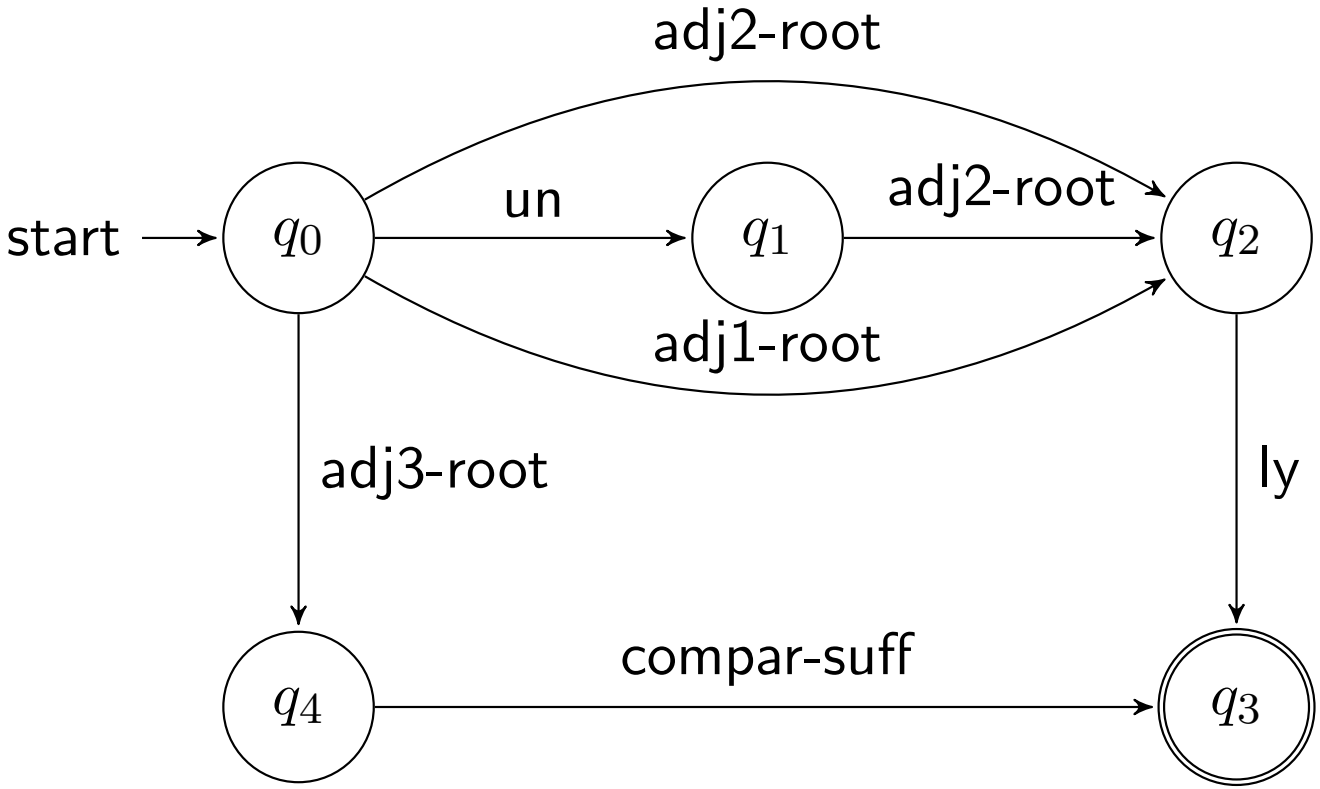
Finite-State Automata

```
1  q0(X):- irr_pl_noun(X).
2  q0(X):- irr_sg_noun(X).
3  q0(X):- atom_concat(A,B,X), reg_noun(A), q1(B).
4  q1('').
5  q1(X):- plural_s(X).
6
7  reg_noun(dog).
8  reg_noun(bush).
9  plural_s(s).
10 plural_s(es).
11 irr_sg_noun(mouse).
12 irr_pl_noun(mice).
```

```
1  ?- q0(dogs).
2  yes
3  ?- q0(mouses).
4  no
```

Finite-State Automata

FSA for English adjective morphology:

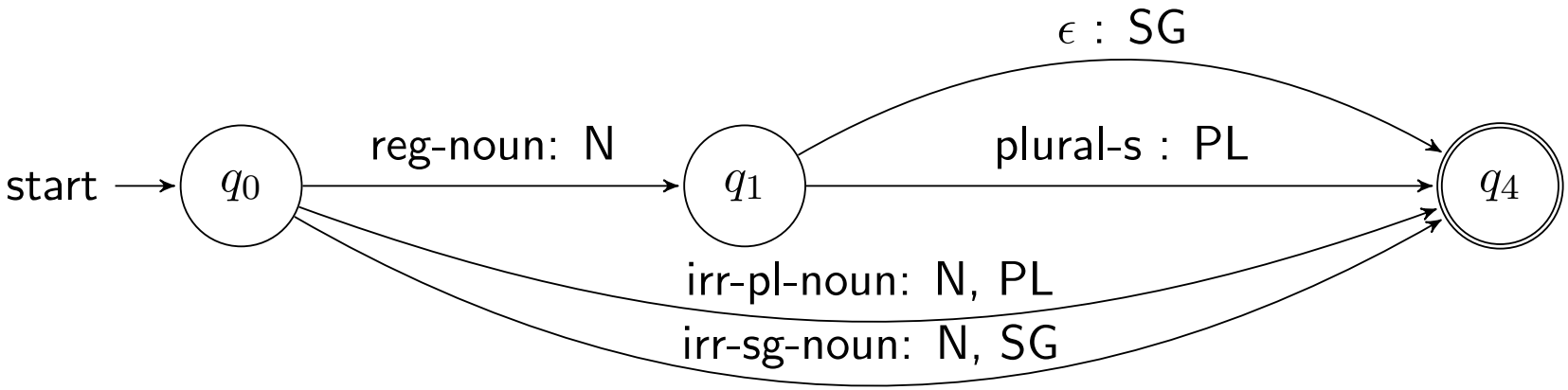


adj1-root: *dead*, *sad*, *sweet*, ... **adj2-root:** *clear*, *happy*, *healthy*, ...
compar-suff: *er*, *est*
adj3-root: *big*, *small*, *ill*, ..., **adj1-root**, **adj2-root**

Finite-State Transducers

= FSA that writes output.

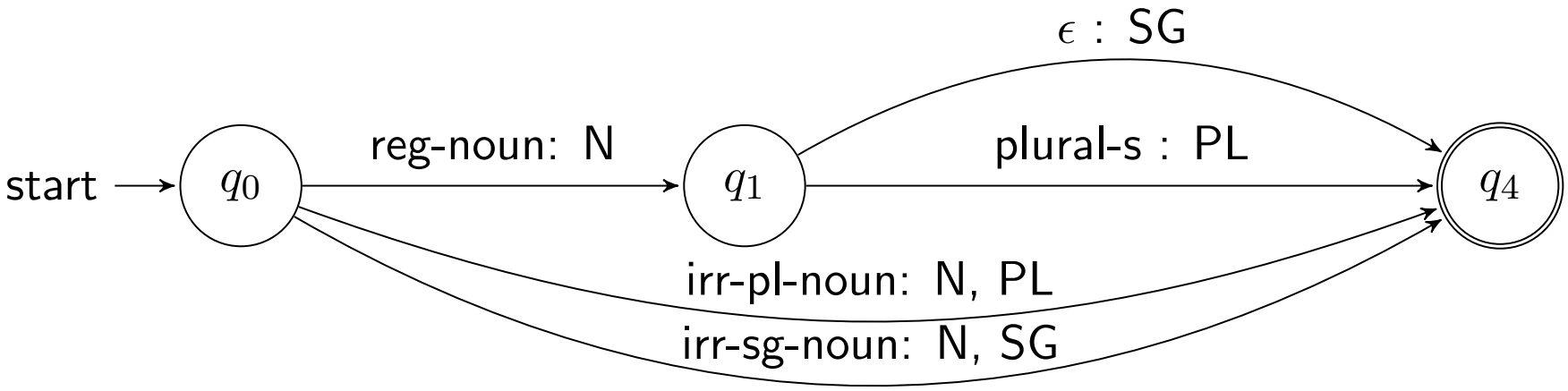
Each transition in a FST is labeled *Input:Output*



Examples:

- from *cat* we obtain *N-SG*
- from *cats* we obtain *N-PL*
- from *mouse* we obtain *N-SG*
- from *mice* we obtain *N-PL*

Finite-State Transducers




```
1 q0(X,[n,sg]):-  irr_sg_noun(X).
2 q0(X,[n,pl]):- irr_pl_noun(X).
3 q0(X,[n|L]):- atom_concat(A,B,X), reg_noun(A), q1(B,L).
4 q1(X, [pl]):- plural_s(X).
5 q1('', [sg]).
```

Finite-State Transducers

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

Finite-State Transducers

```
1  lemmatize([],[]).
2  lemmatize(L1,[Token|L2]):-
3          fsa(L1,Token,L3),
4          lemmatize(L3,L2).
5
6  fsa(L1,X,L2):- noun_fsa(L1,X,L2).
7  fsa(L1,X,L2):- verb_fsa(L1,X,L2).
8
9  noun_fsa([N|L1],[N,n,sg],L1):-
10         reg_noun(N).
11
12 noun_fsa([N|L1],[X,n,pl],L1):-
13         atom_concat(X,s,N),
14         reg_noun(X).
```

```
1  ?- lemmatize([the,cat,sneezed], X).
2  X = [[the,d],[cat,n,sg],[sneeze,v,past]]
```

Finite-State Transducers

Problem: how to deal with ambiguity?

```
?- lemmatize([[he], [saw], [her], [duck]], X).
X = [[he,pr,sg,masc],[see,v,past],[she,pr,sg,fem],[duck,v,bse]] ;
X = [[he,pr,sg,masc],[see,v,past],[she,d,sg,fem],[duck,n,sg]] ;
X = [[he,pr,sg,masc],[see,v,past],[she,pr,sg,fem],[duck,n,bse]] ;
X = [[he,pr,sg,masc],[see,v,past],[she,d,sg,fem],[duck,v,bse]]
X = [[he,pr,sg,masc],[saw,n,sg],[she,pr,sg,fem],[duck,v,bse]] ;
X = [[he,pr,sg,masc],[saw,n,sg],[she,d,sg,fem],[duck,n,sg]] ;
X = [[he,pr,sg,masc],[saw,n,sg],[she,pr,sg,fem],[duck,n,sg]] ;
X = [[he,pr,sg,masc],[saw,n,sg],[she,d,sg,fem],[duck,v,sg]] ;
X = [[he,pr,sg,masc],[saw,v,pres],[she,pr,sg,fem],[duck,v,bse]] ;
X = [[he,pr,sg,masc],[saw,v,pres],[she,d,sg,fem],[duck,n,sg]] ;
X = [[he,pr,sg,masc],[saw,v,pres],[she,pr,sg,fem],[duck,n,bse]] ;
...
```

Two solutions:

- Enrich the system with the word order rules of English
- Look up in a large text what is the most common order