MENOUFIA UNIVERSITY
FACULTY OF COMPUTERS AND INFORMATION

Fourth Year (Second Semester)
CS Dept., (CS 436)

Natural Language Processing NLP

Lecture Five

Dr. Hamdy M. Mousa

MORPHOLOGY AND FINITE-STATE TRANSDUCERS

Singulars & Plurals

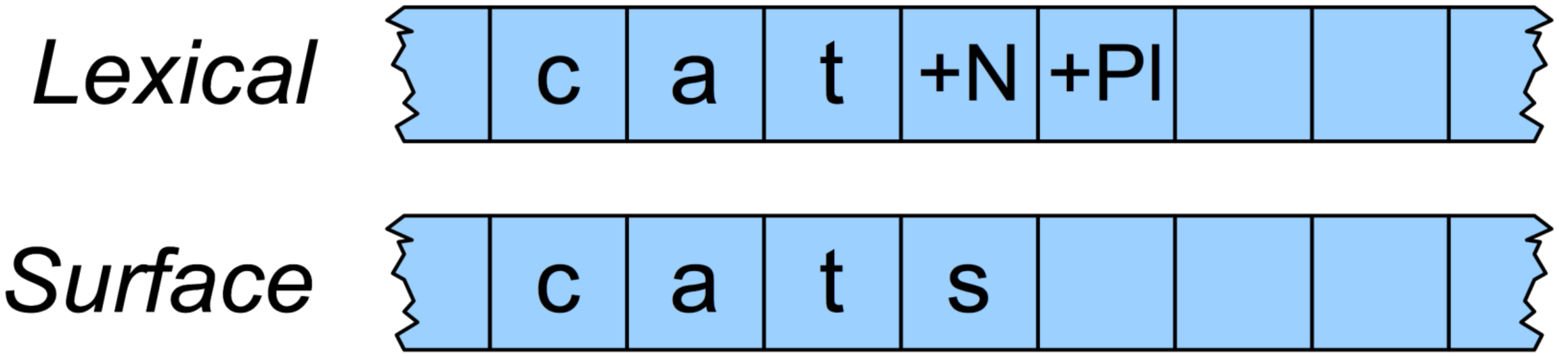
- Hunting for the plurals of these animals takes more than just tacking on an ***s***.
- *The plural of*
  - *Fox is foxes*
  - *Goose is geese.*
  - *Fish don't usually* change their form when they are plural *one fish, two fish, red fish).*

Two level morphology

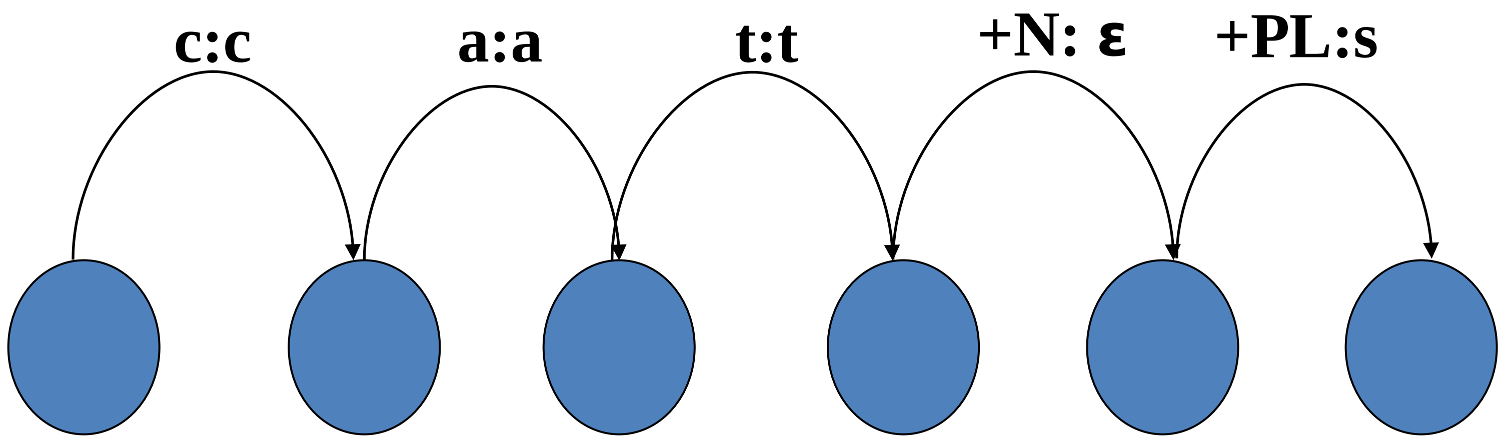
- We will do this via a version of two-level morphology, first proposed by Koskenniemi (1983).
- Two level morphology represents a word as a correspondence between
  - ***lexical level***, which represents a simple concatenation of morphemes making up a word,
  - ***surface level***, which represents the actual spelling of the final word.

Morphological Parsing

- Morphological parsing is implemented by building mapping rules that map letter sequences
- ***surface level*** →actual spelling like cats
- ***lexical level*** → stem for a word **+** *morphological information* (+N +PL)



Transitions



- **c:c** means read a c on one tape and write a c on the other
- **+N:ε** means read a +N symbol on one tape and write nothing on the other
- **+PL:s** means read +PL and write an s
- *Note the conventions:* x:y represents an input symbol x and the output symbol y.

Finite-State Transducers

- The automaton that we use for performing the mapping between these two levels is the finite-state transducer or FST.
- A transducer maps between one set of symbols and another;
- FST as a two-tape automaton which recognizes or generates pairs of strings

Finite-State Transducers

- The FST has a more general function than an FSA;
  - FSA defines a formal language by defining a set of strings.
- FST defines a *relation* between sets of strings.
  - This relates to another view of an FST; as a machine that *reads one string and generates* another.

Finite-State Transducers

- **FST as recognizer**: a transducer that takes a pair of strings as input and outputs *accept* if the string-pair is in the string-pair language, and reject if it is not.
- **FST as generator**: a machine that outputs pairs of strings of the language.
  - Thus the output is a yes or no, and a pair of output strings.
- **FST as translator**: a machine that reads a string and outputs another string.
- **FST as set relater**: a machine that computes relations between sets.

Morphological parsing

- Morphological analysis or parsing can either be
  - An important stand-alone component of many applications (spelling correction, information retrieval)
  - link in a chain of further linguistic analysis
- FSTs are bidirectional, i.e. can be used for parsing and generation

Mealy machine extension

- A formal definition of FST (based on the **Mealy machine** extension to a simple FSA):
- FST is a 5-tuple consisting of:
  - Q: set of states $\{q_0,q_1,q_2,q_3,q_4\}$
  - Σ: an alphabet of complex symbols, each is an i/o pair such that $i \in I$ (an input alphabet) and $o \in O$ (an output alphabet) and Σ is in I x O
  - q_0: a start state
  - F: a set of final states in Q $\{q_4\}$
  - $\delta(q,i:o)$: a transition function mapping Q x Σ to Q

Mealy machine extension

- Given a state $q \in Q$ and complex symbol $i : o \in \Sigma$, $\delta(q,i:o)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;
- Where an FSA accepts a language stated over a finite alphabet of single symbols, such as the alphabet of our sheep language:

  $\Sigma = \{b, a, !\}$
- an FST accepts a language stated over pairs of symbols, as in:

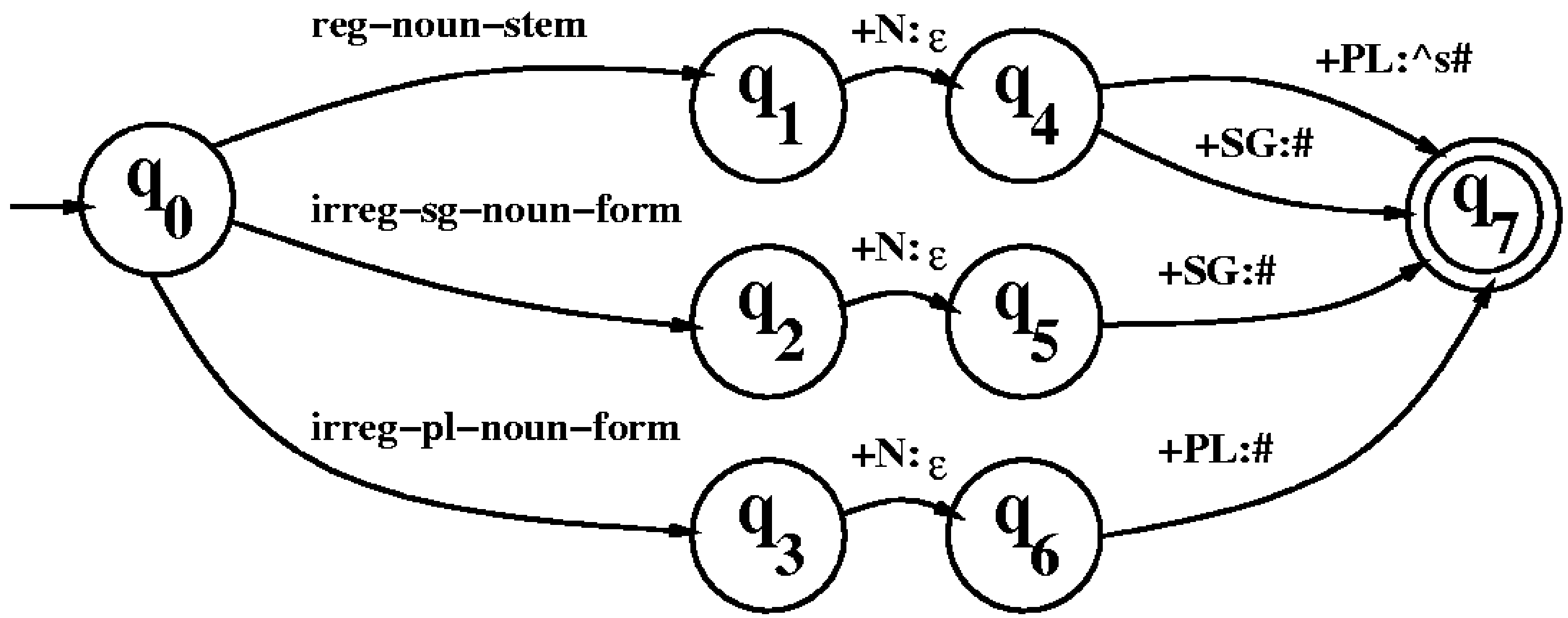
  $\Sigma = \{a : a, b : b, ! : !, a : !, a : \varepsilon, \varepsilon : !\}$

Closure Properties

- Besides union, FSTs have two additional closure properties:
  - **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.
  - **composition:** if T_1 is a transducer from I_1 to O_1 and T_2 a transducer from I_2 to O_2, then $\boldsymbol{T_1 \circ T_2}$ maps from I_1 to O_2.

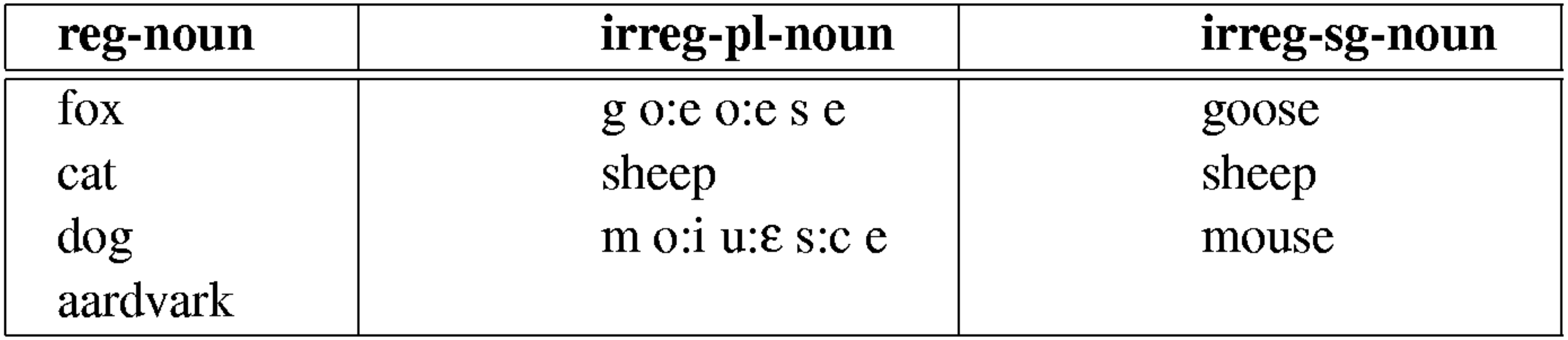
Morphological noun parser

- morphological features (+SG and +PL) that correspond to each morpheme.
  - Note
    - ε The empty string
    - # The word boundary symbol
    - ^ The morpheme-boundary symbol.



Orthographic convention

- Since surface geese maps to underlying goose, the new lexical entry will be
  - 'g: g o: e o: e s: s e: e'.
- Regular forms are simpler; the two-level entry for fox will now be
  - ' f: f o: o x: x'

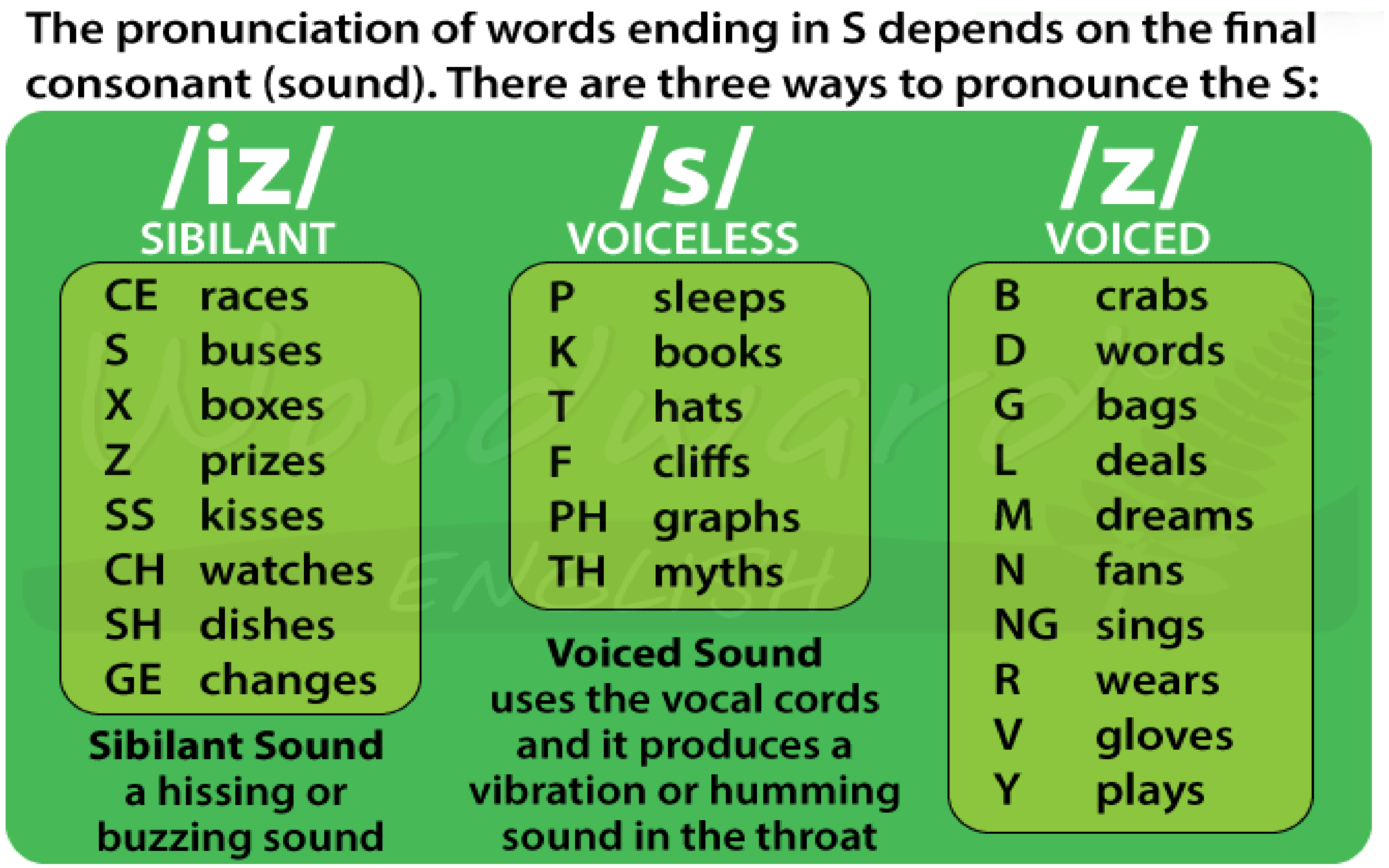


reg-noun	**irreg-pl-noun**	**irreg-sg-noun**
fox	g o:e o:e s e	goose
cat	sheep	sheep
dog	m o:i u:ε s:c e	mouse
aardvark		

Morphological parser

- morphological parser needs to map from **surface forms**
- Of course, its not as easy as
  - **“cat +N +PL” ↔ “cats”**
- As we saw earlier there are geese, mice and oxen
- spelling/pronunciation changes that go along with inflectional changes
  - Cats Vs Dogs
  - Fox and Foxes

Pronunciation of S



The pronunciation of words ending in S depends on the final consonant (sound). There are three ways to pronounce the S:

/iz/ SIBILANT		/s/ VOICELESS		/z/ VOICED	
CE	races	P	sleeps	B	crabs
S	buses	K	books	D	words
X	boxes	T	hats	G	bags
Z	prizes	F	cliffs	L	deals
SS	kisses	PH	graphs	M	dreams
CH	watches	TH	myths	N	fans
SH	dishes			NG	sings
GE	changes			R	wears
				V	gloves
				Y	plays

Sibilant Sound a hissing or buzzing sound

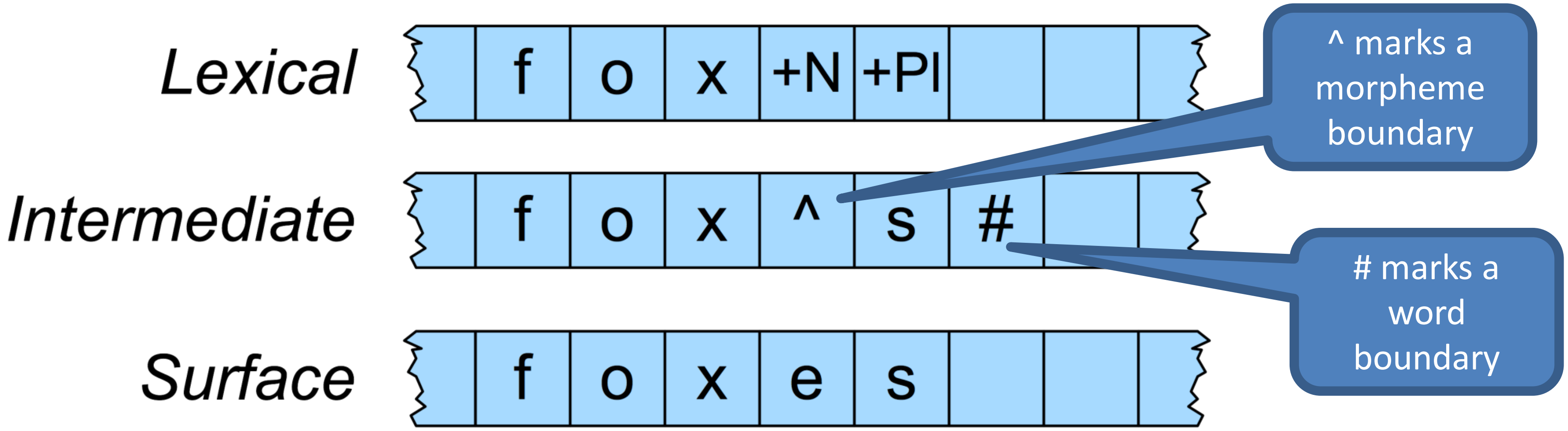
Voiced Sound uses the vocal cords and it produces a vibration or humming sound in the throat

Multi-Tape Machines

- To deal with these complications,
  - Add more tapes
  - use the output of one tape machine as the input to the next (cascade of FSTs)
- So to handle irregular spelling changes we'll add intermediate tapes with intermediate symbols

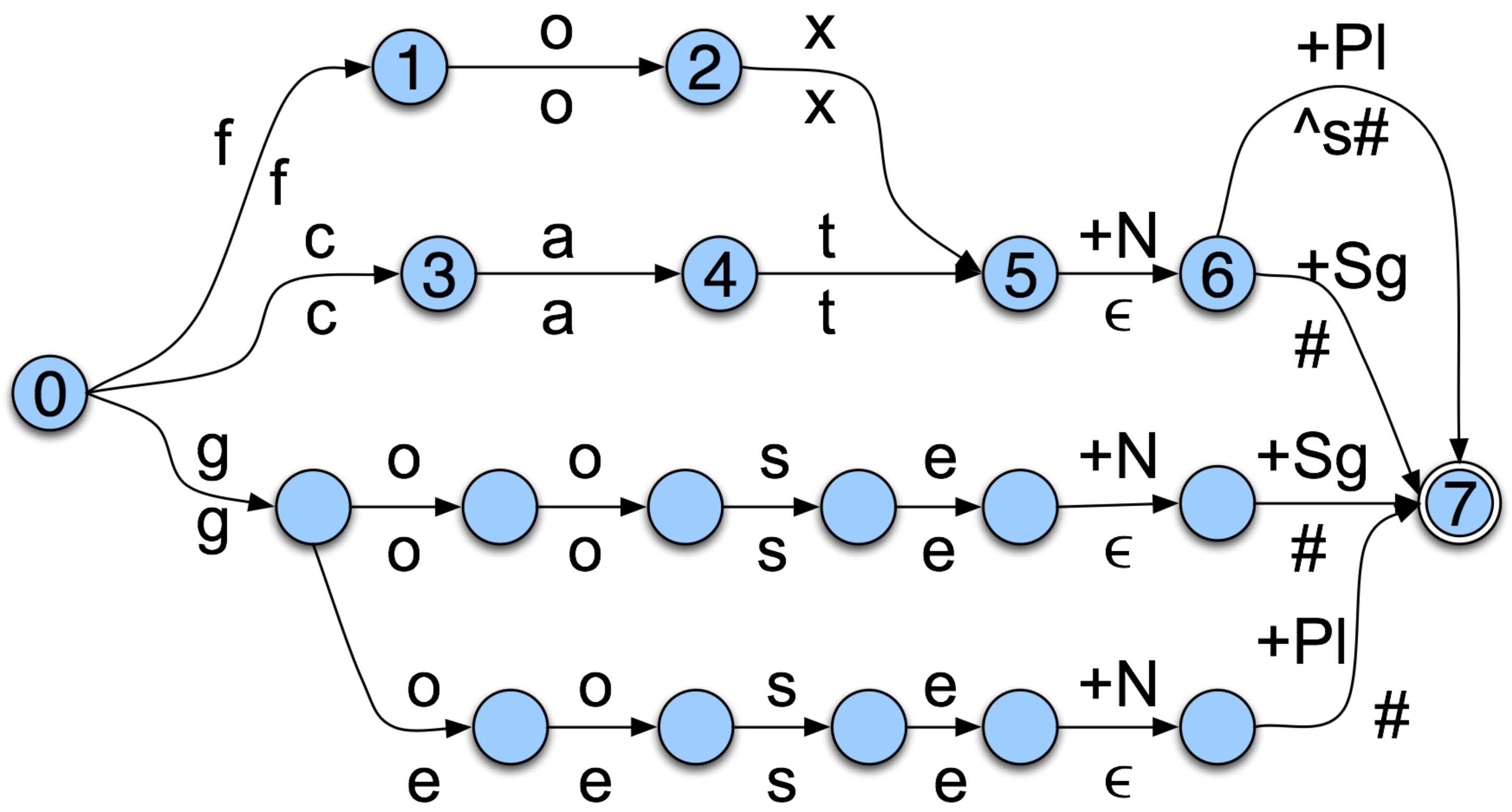
- **The three levels:** lexical, intermediate, and surface.

Multi-Level Tape Machines



- We use one machine to transduce between the lexical and the intermediate level, and another to handle the spelling changes to the surface tape

Multi-Level Tape Machines



Orthographic (Spelling) Rules

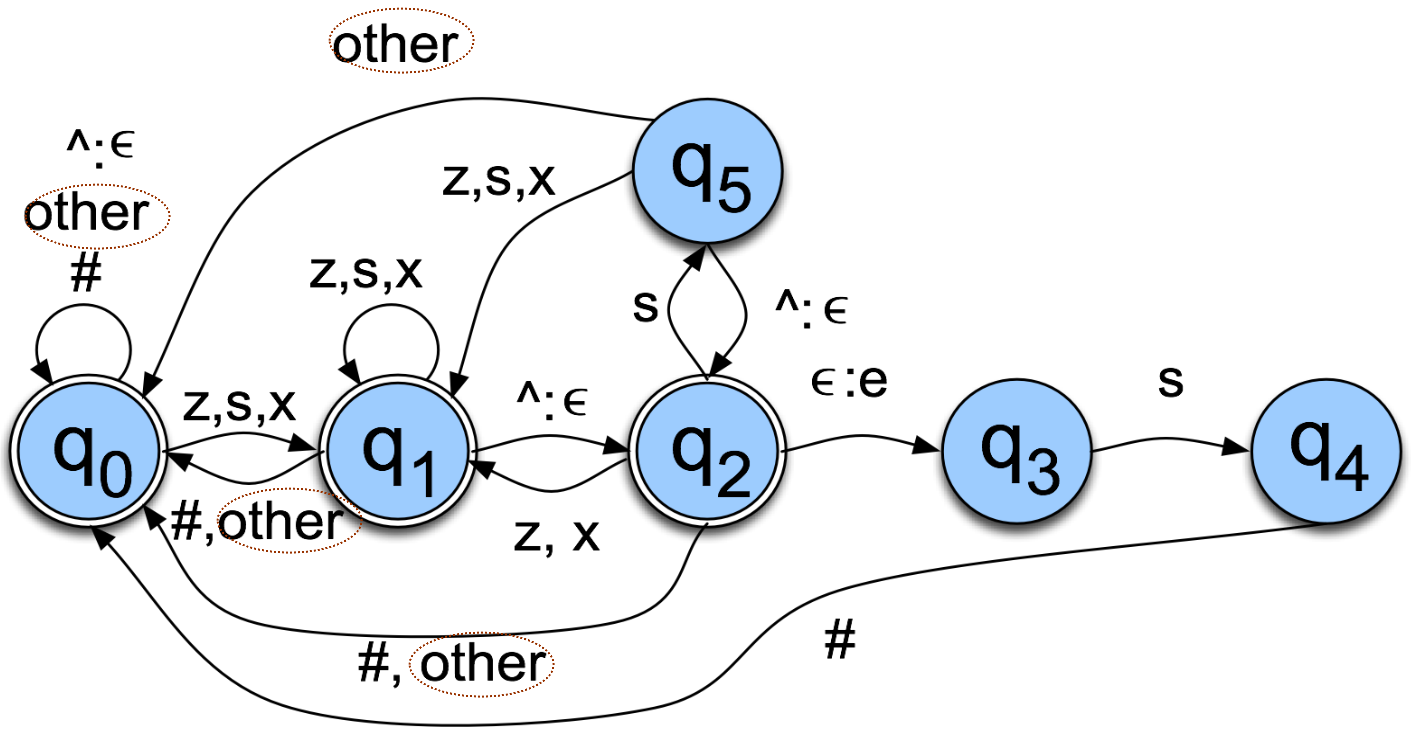
- **Rule:** "insert an **e** on the surface tape just when the lexical tape has a morpheme ending in x (or z, etc) and the next morpheme is -s.
- Here's a formalization of the rule:

$$\varepsilon \rightarrow e\,/\left\{\begin{matrix} x \\ s \\ z \end{matrix}\right\} \hat{}\ \underline{\quad\quad} s\#$$

means 'insert an e after a morpheme-final x, s, or z, and before the morpheme s'.

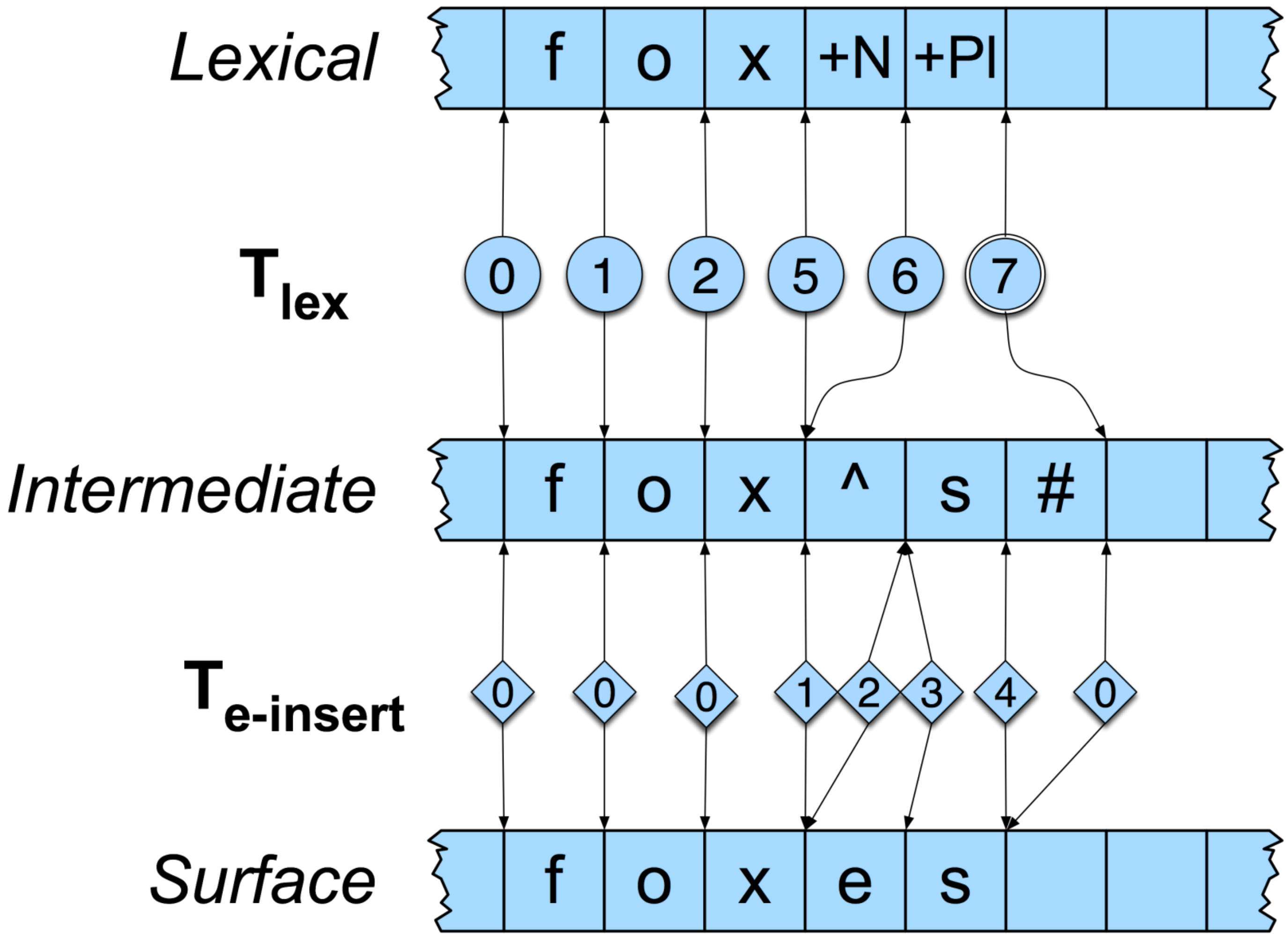
E - insertion Rule

- The add an “e” rule as in fox^s# ↔ foxes#

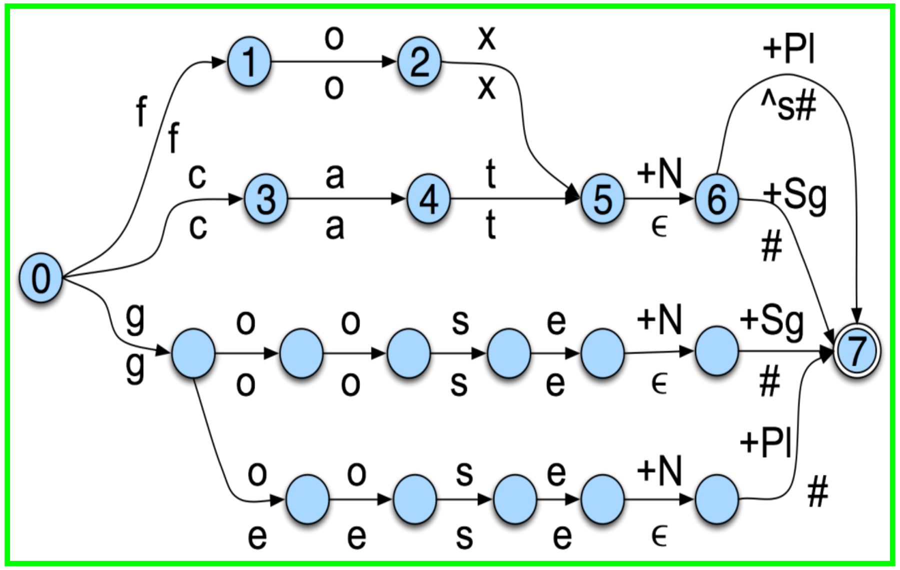


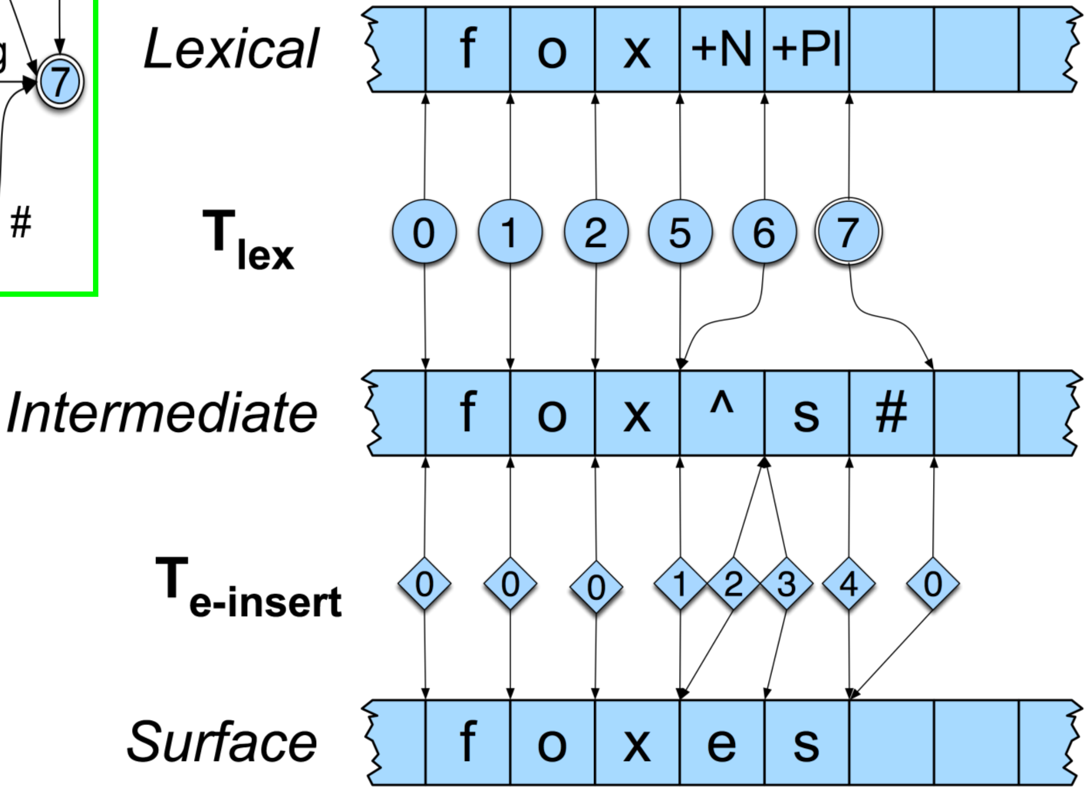
- The “**other**” symbol is used as shown to safely pass through any parts of words that don't play a role in the e-insertion rule.

E - insertion Rule

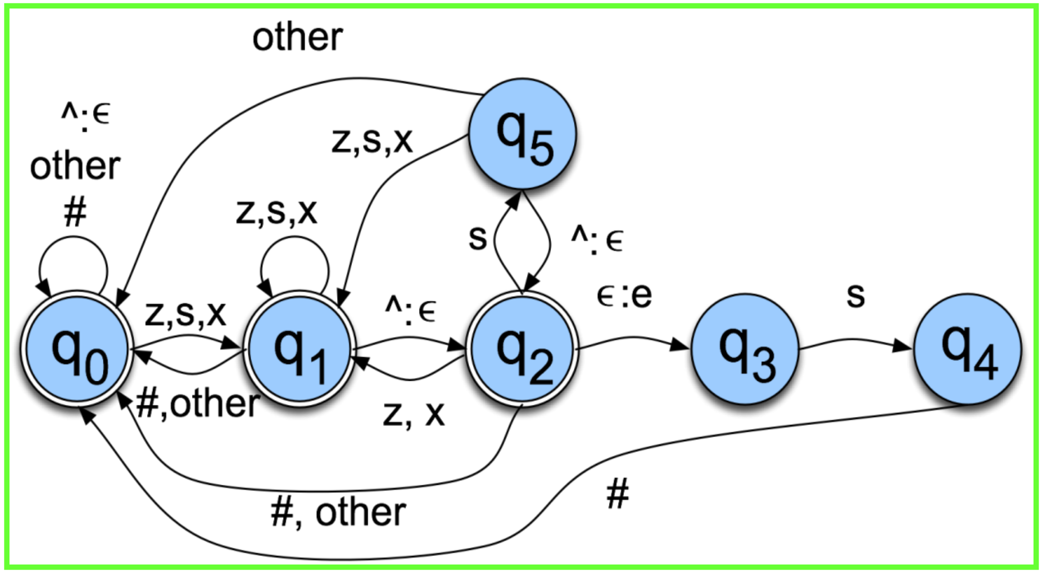


Lexical Rule



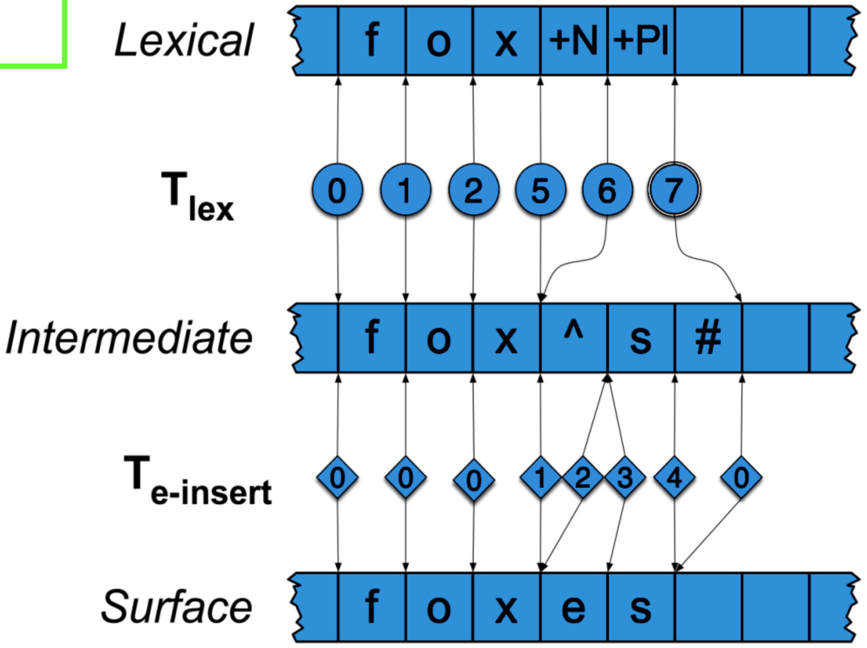


E - insertion Rule



NOTE:

- this machine is that it has to do the right thing for inputs to which it doesn't really apply. So...
  - Fox → foxes but bird → birds



Cont., E - insertion Rule

- State q0: models having seen only default pairs unrelated to the rule, is an accepting state.
- State q1: models having seen a z, s, or x.
- State q2 : models having seen the morpheme boundary after the z, s, or x, and again is an accepting state.
- State q3 : models having just seen the e-insertion; it is not an accepting state, since the insertion is only allowed if it is followed by the s morpheme and then the end-of-word symbol #.
- State q5 is used to insure that the e is always inserted whenever the environment is appropriate; the transducer reaches q5 only when it has seen an s after an appropriate morpheme boundary. If the machine is in state q5 and the next symbol is #, the machine rejects the string (because there is no legal transition on # from q5).

Orthographic (Spelling) Rules

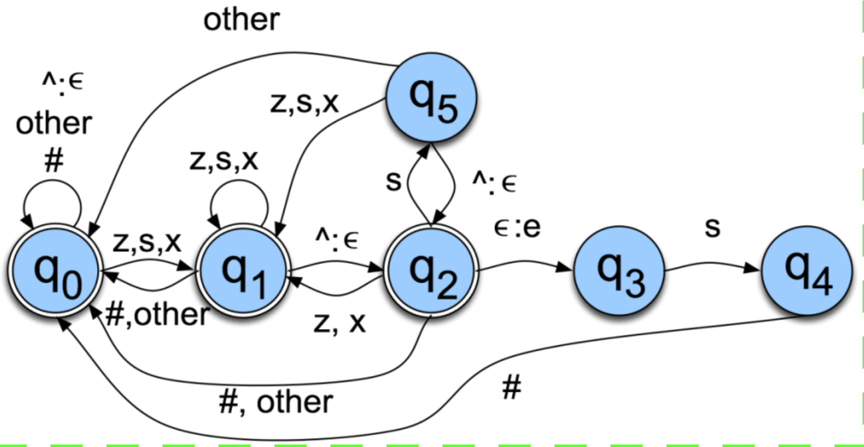
Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before -*ing*/-*ed*	beg/begging
E deletion	Silent e dropped before -*ing* and -*ed*	make/making
E insertion	e added after -*s*, -*z*, -*x*, -*ch*, -*sh*, before –*s*	watch/watches
Y replacement	-*y* changes to -*ie* before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

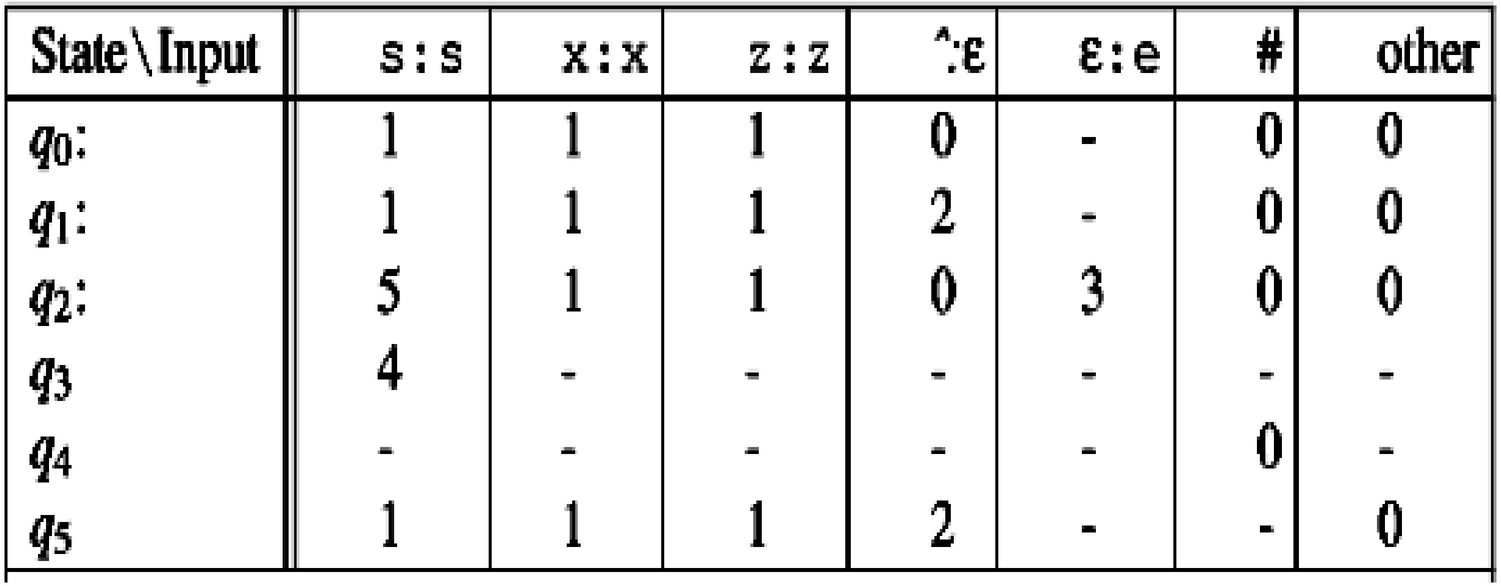
Orthographic (or Spelling) Rules and FSTs

- The idea in building a transducer for a particular rule is to express only the constraints necessary for that rule, allowing any other string of symbols to pass through unchanged.
- This rule is used to insure that we can only see the ε:e pair if we are in the proper context.

Orthographic (Spelling) Rules

- The transition table for the rule (E-insertion) which makes the **illegal transitions** explicit with the '-' symbol.



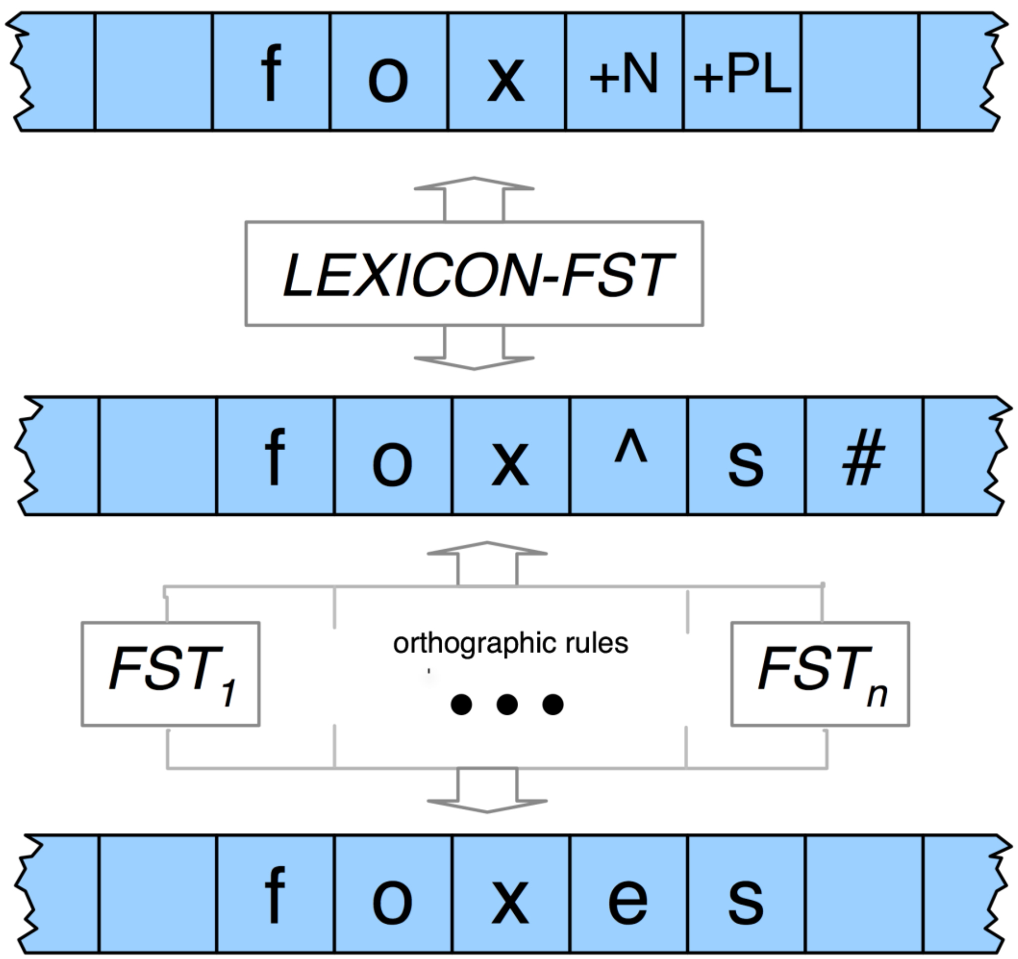


State\Input	s:s	x:x	z:z	^:ε	ε:e	#	other
q_0:	1	1	1	0	-	0	0
q_1:	1	1	1	2	-	0	0
q_2:	5	1	1	0	3	0	0
q_3	4	-	-	-	-	-	-
q_4	-	-	-	-	-	0	-
q_5	1	1	1	2	-	-	0

COMBINING FST LEXICON AND RULES

- **One FST** that has explicit information about the lexicon (actual words, their spelling, facts about word classes and regularity).
  - – Lexical level to intermediate forms
- **Larger set** of machines that capture orthographic/spelling rules.
  - – Intermediate forms to surface forms

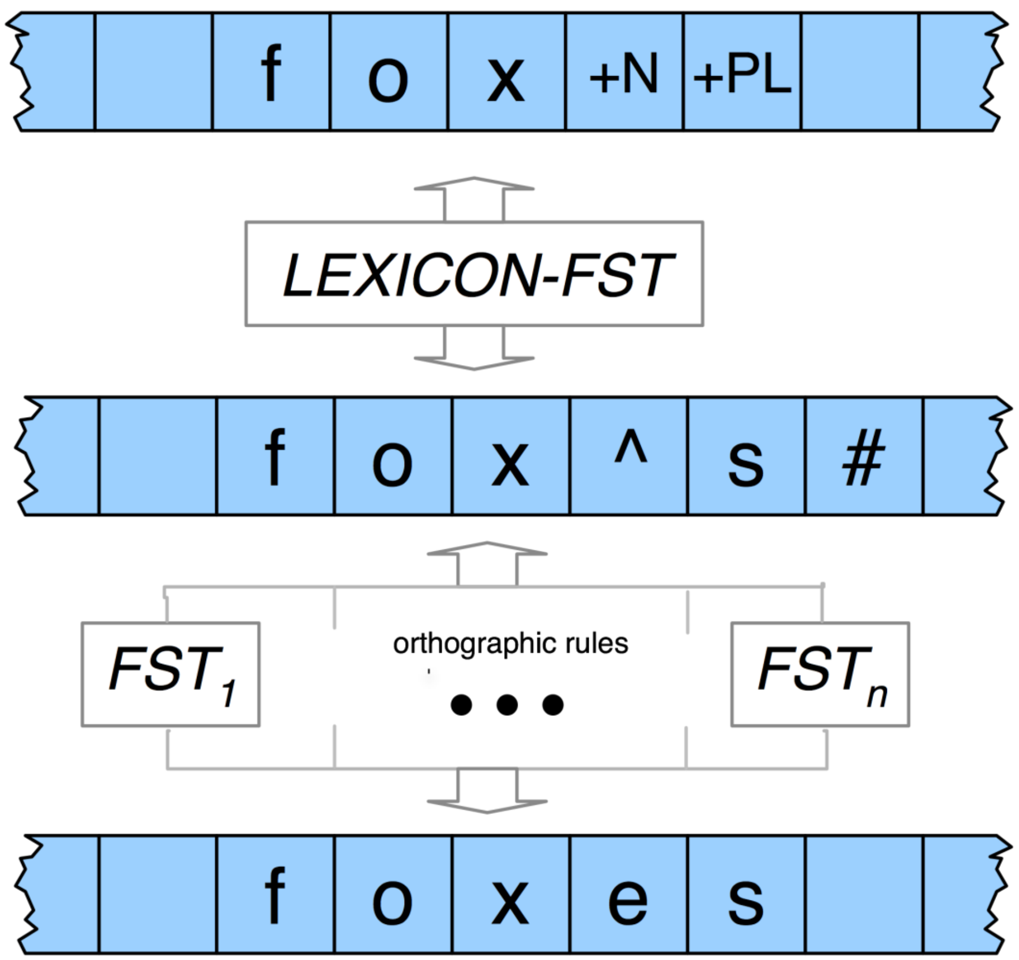
COMBINING FST LEXICON AND RULES



Cascades

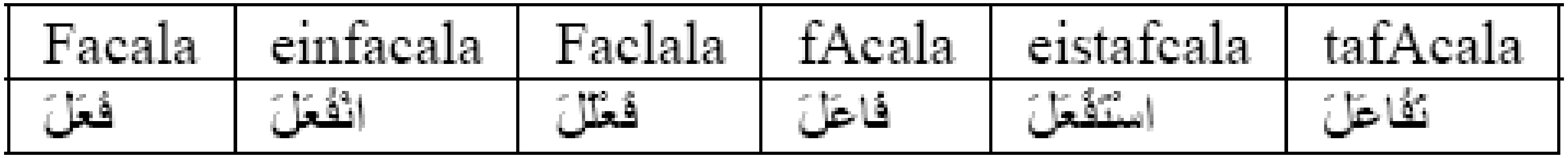
- This is a common architecture
    - Overall processing is divided up into distinct rewrite steps
    - The output of one layer serves as the input to the next
    - The intermediate tapes may or may not wind up being useful in their own right

COMBINING FST LEXICON AND RULES



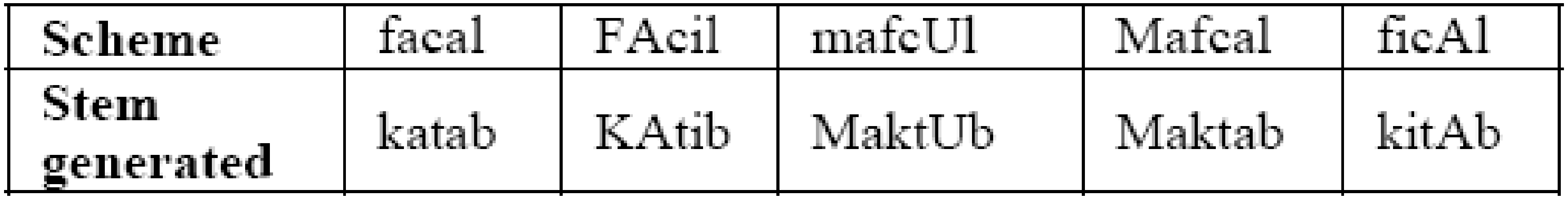
Arabic

- Root “ktb” / ك ت ب/
- Scheme “tafAcala” / تَفَاعَلَ/
- Stem “takAtaba” / تَكَاتَبَ/



Facala	einfacala	Faclala	fAcala	eistafcala	tafAcala
فَعَلَ	اِنْفَعَلَ	فَعْلَلَ	فَاعَلَ	اِسْتَفْعَلَ	تَفَاعَلَ

Some examples of schemes to generate stems from the root “ktb” (كتب)



Scheme	facal	FAcil	mafcUl	Mafcal	ficAl
Stem generated	katab	KAtib	MaktUb	Maktab	kitAb

Arabic

- For the affixes morphemes, they can be added before or after a root or a stem as a prefix or suffix. For prefixes, they can be added before the stem like the prefix “sa” /س /which used to express the future while suffixes can be added after the stem like the suffix “Una” / ون /

Arabic

By definition, Morphology, simply called صرف in Arabic is the study of the structural formation of words and the cases that affect the formation. The literal meaning of صرف is indicative of what Morphology is all about. Literally, صرف means to change i.e. to change from one form to another form. Technically, صرف means a structural change affecting a word called كلمة from one form to another form or to several forms. For instance, you will observe in the above text how the word كَتَبَ changed from one form to several forms. The word كَتَبَ (to write) has structurally changed severally e.g.

كَتَبَ   يَكْتُبُ   كِتَابَةً   كَاتِبٌ   مَكْتُوبٌ   أَكْتُبْ   لاَ تَكْتُبْ   مَكْتَبٌ   مَكْتَبَةً

Arabic

In the same text, the word changed from (كَتَبَ) to (كَاتَبَ) i.e. to correspond with e.g.

كَاتَبَ يُكَاتِبُ مُكَاتَبَةً مُكَاتِبٌ مُكَاتَبٌ كَاتِبْ لا تُكَاتِبْ

It can further change to كَتَّبَ (meaning to make one write)

كَتَّبَ يُكَتِّبُ تَكْتِيبًا مُكَتِّبٌ مُكَتَّبٌ كَتِّبْ لا تُكَتِّبْ

It can change to إكْتَتَبَ (meaning to subscribe to a paper or a journal)

إكْتَتَبَ يَكْتَتِبُ إكْتِتَابًا مُكْتَتِبٌ إكْتَتِبْ لا تَكْتَتِبْ

It can also change to إسْتَكْتَبَ **i.e**. to ask one to write.

إسْتَكْتَبَ يَسْتَكْتِبُ إسْتِكْتَابًا مُسْتَكْتِبٌ مُسْتَكْتَبٌ إسْتَكْتِبْ لا تَسْتَكْتِبْ

It can change to تَكَاتَبَ

تَكَاتَبَ يَتَكَاتَبُ تَكَاتُبًا مُتَكَاتِبٌ مُتَكَاتَبٌ تَكَاتَبْ لا تَتَكَاتَبْ

REPORT

- **Choose a Domain for your Corpora:**
  - Technology
  - Computers
  - Engineering,………..
  - Medical, ………
  - Management
  - Weather
  - Sport
  - Economics
  - Politics
  - Education
  - Health care
  - Religion
  - History
  - Traditional Poems
  - New Poems
  - …………

Cont., REPORT

- **Put your choice on the discussion list named 'My Corpora'.**
- **Collect text files and keep them in one directory**
- ***Suggested total size (sum of sizes of all text files)***
  - ***larger than 10Mbyte of Arabic text***
- **Write a program that allows the user to browse and select a directory, then the program will list the names of the text files in that directory.**
- **You can use any language you are mastering. However, Python might be a good choice**
- **After building your corpora, you need to find the most used 100 words in your corpora. You might do that by writing a program that let the user choose the directory of the corpora where the text files are located and find the most use n words. Where n could be 100.**