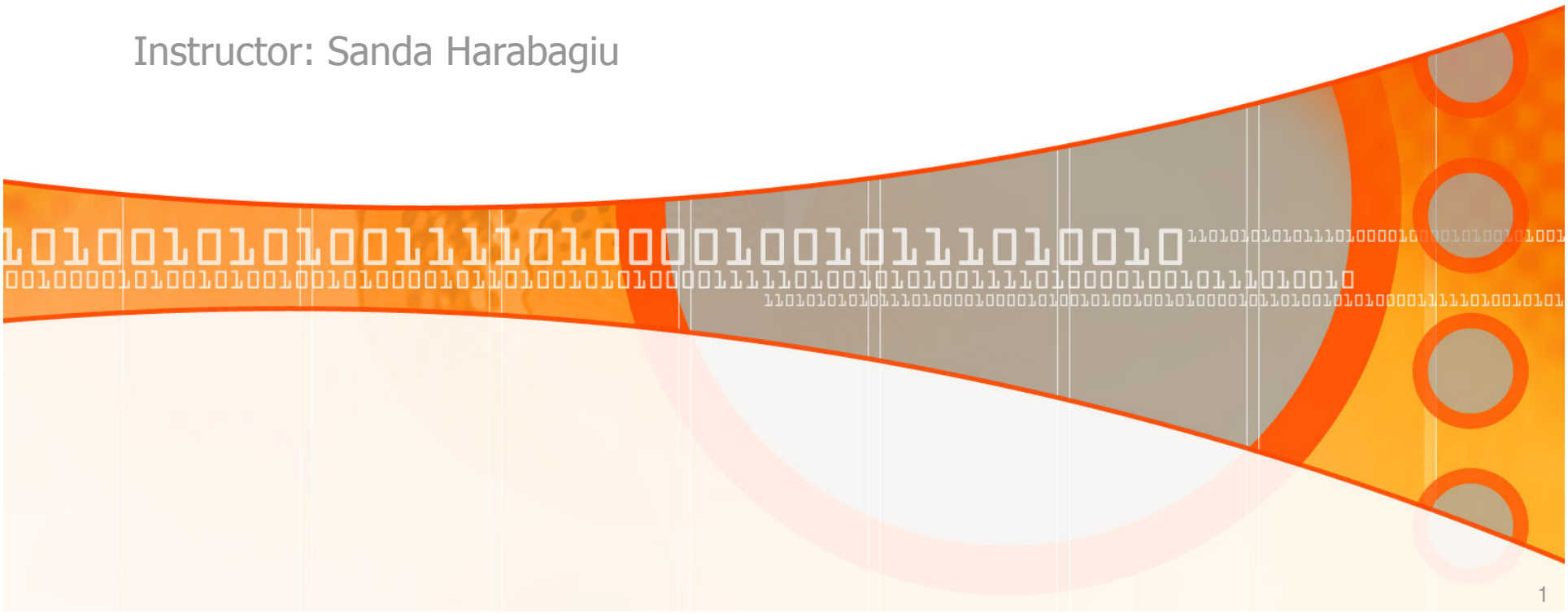
Natural Language Processing
CS 6320
Lecture 5
Words and Transducers

Instructor: Sanda Harabagiu



Morphology and Finite-State Transducers 1/2

- ***Morphology*** *is the study of the way words are built from smaller units called morphemes*
  - *A morpheme is a minimal meaning-bearing unit in language.*
  - *Example: the word DOG has a single morpheme; the word CATS has two: (1) CATand (2) –S*
- *There are two classes of morphemes:* stems and affixes.
- *Stems -- are the main morphemes of words.*
- *Affixes – add additional meaning to the stems to modify their meanings and grammatical functions*

Morphology and Finite-State Transducers 2/2

- *There are four forms of affixes:*
  1. *Prefixes – precede the stem*
  2. *Suffixes – follow the stem*
  3. *Infixes – inserted inside the stem*
  4. *Circumfixes – both precede and follow the stem.*
- *Examples:*
  - *Prefixes - "un", "a"*
  - *Suffixes - plurals, "ing"*
  - *Infixes - not common in English*
  - *Circumfixes : unbelievably*

  un + believe + able + ly

Kinds of Morphology

- *The usage of prefixes and suffixes concatenated to the stem creates a concatenative morphology.*
- *When morphemes are combines in more complex ways, we have a non-concatenative morphology.*

- *Inflectional morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of the same class*
  - *Special case: Templatic morphology, also known as root-and-pattern morphology:*
    - *Used in semitic languages*, e.g. Arabic, Hebrew

Example of Templatic Morphology

- In Hebrew, a verb is constructed using two components:
  - A root (consisting usually of 3 consonants CCC) – carrying the main meaning
  - A template which gives the ordering of consonants and vowels and specifies more semantic information about the resulting verb, e,g, the voice (active, passive)
- *Example:* the tri-consonant root lmd (meaning: learn, study)
- can be combined with a template CaCaC for active voice to produce the word lamad = "he studied"
- can be combined with the template CiCeC for intensive to produce the word limed = "he tought"
- can be combined with a template CuCaC for active voice to produce the word lumad = "he was taught"

Producing words from morphemes

- *By inflection – Inflectional morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of the same class*
- *By derivation – Derivational Morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of a different class.*
- *By compounding – by combining multiple words together, e.g. "doghouse"*
- *By cliticization – by combining a word stem with a clitic. A clitic is a morpheme that acts syntactically like a word but it is reduced in forms and it is attached to another word. Eg.g I've.*

Clitization

- *A clitic is a unit whose status lies between that of an affix and a word!*
  - *The phonological behavior of clitics is like affixes: they then to be short and unaccented*
  - *The syntactic behavior is more like words: they often act as pronouns, articles, conjunctions or verbs.*

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

Inflectional Morphology 1/2

- *Nouns: have an affix for plural and an affix for possessive*
  - *Plural – the suffix –s and the alternative spelling –es for regular nouns*

Singular	Plural
dog	dogs
farm	farms
school	schools
car	cars

Singular	Plural
ibis	ibises
waltz	waltzes
box	boxes
butterfly	butterflies

Irregular nouns:

Singular	Plural
mouse	mice
ox	oxen

  - *Possessive - for words not ending in "s" the affix is "'s" (children/ children's) and for words ending in "s" the affix is "'" (llama/llama's; llamas/llamas') (Euripides/Euripides' comedies)*

Inflectional Morphology 2/2

- *Verbal inflection is more complex than nominal inflection*
- *There are three classes of verbs in English:*
  - *Main verbs (eat, sleep, walk)*
  - *Modal verbs (can will, should)*
  - *Primary verbs (be, have, do)*

Regular/ Irregular Verbs

Morphological Class	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

Morphological Class	Irregularly Inflected Verbs		
stem	eat	catch	cut
-s form	eats	catches	cuts
-ing participle	eating	catching	cutting
preterite	ate	caught	cut
past participle	eaten	caught	cut

Spanish Verb System

	Present Indicative	Imperfect Indicative	Future	Preterite	Present Subjunctive	Conditional	Imperfect Subjunctive	Future Subjunctive
1SG	amo	amaba	amaré	amé	ame	amaría	amara	amare
2SG	amas	amabas	amarás	amaste	ames	amarías	amaras	amares
3SG	ama	amaba	amará	amó	ame	amaría	amara	amáreme
1PL	amamos	amábamos	amaremos	amamos	amemos	amaríamos	amáramos	amáremos
2PL	amáis	amabais	amaréis	amasteis	améis	amaríais	amarais	amareis
3PL	aman	amaban	amarán	amaron	amen	amarían	amaran	amaren

To love in Spanish.
Some of the inflected forms of the verb “amar” in European Spanish
50 distinct verb forms for each regular verb.
Example the verb “amar” = “to love”

Derivational Morphology

- *Changes of word class*
  - *Nominalization: the formation of new nouns from verbs or adjectives.*

Suffix	Base Verb/Adjective	Derived Noun
-ation	computerize (V)	computerization
-ee	appoint (V)	appointee
-er	kill (V)	killer
-ness	fuzzy (A)	fuzziness

  - *Adjectives can also be derived from verbs.*

Suffix	**Base Noun/Verb**	**Derived Adjective**
-al	computation (N)	computational
-able	embrace (V)	embraceable
-less	clue (N)	clueless

Morphological Parsing

- *In general parsing means taking an input and producing a structure for it.*
- *Morphological parsing takes as an input words and produces a structure that reveals its morphological features.*

English		Spanish		
Input	**Morphological Parse**	**Input**	**Morphological Parse**	**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 +3P +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			

Building a Morphological Parser

We need:

1. *Lexicon - is the list of stems and affixes and basic information about them*
2. *Morphotactics – the model of morpheme ordering that expains which classes of morphemes can follow other classes of morphemes inside a word.*
3. *Ortographic rules - or spelling rules model the changes that occur in a word when morphemes are combined.*

Morpholgy and FSAs

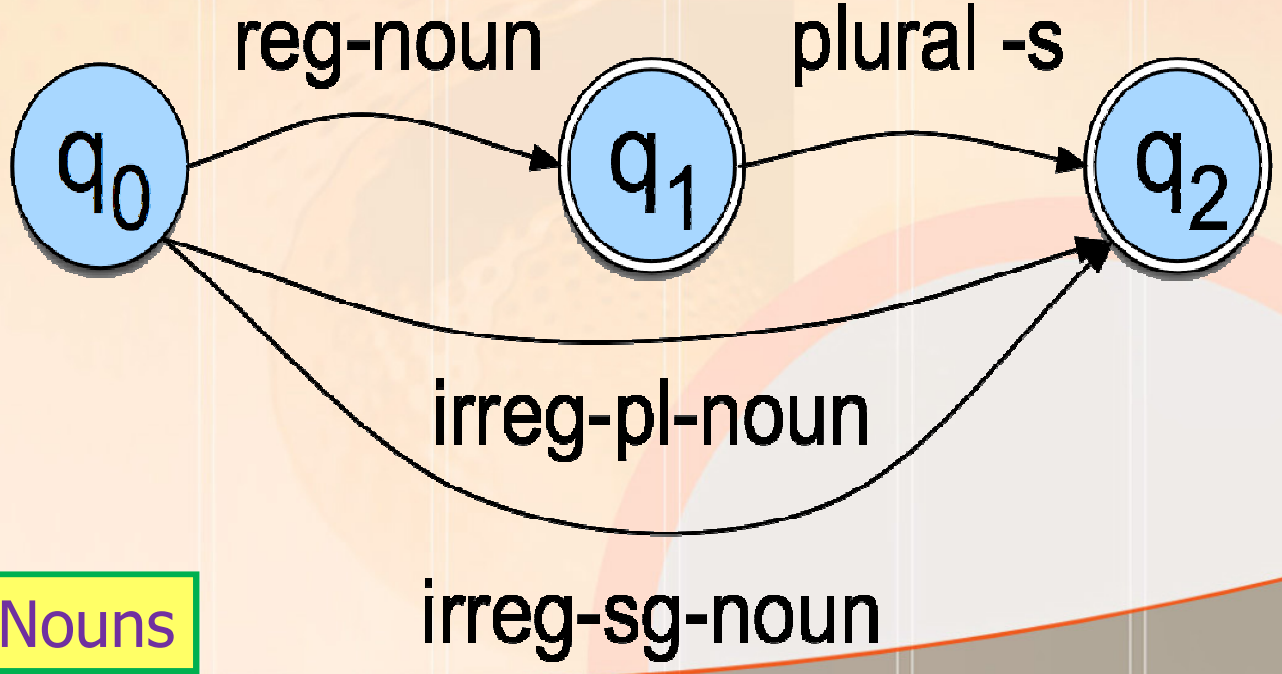
- *We'd like to use the machinery provided by FSAs to capture these facts about morphology*
  - *Accept strings that are in the language*
  - *Reject strings that are not*
  - *And do so in a way that doesn't require us to in effect list all the words in the language*

Start Simple

- *Regular singular nouns are ok*
- *Regular plural nouns have an -s on the end*
- *Irregulars are ok as is*

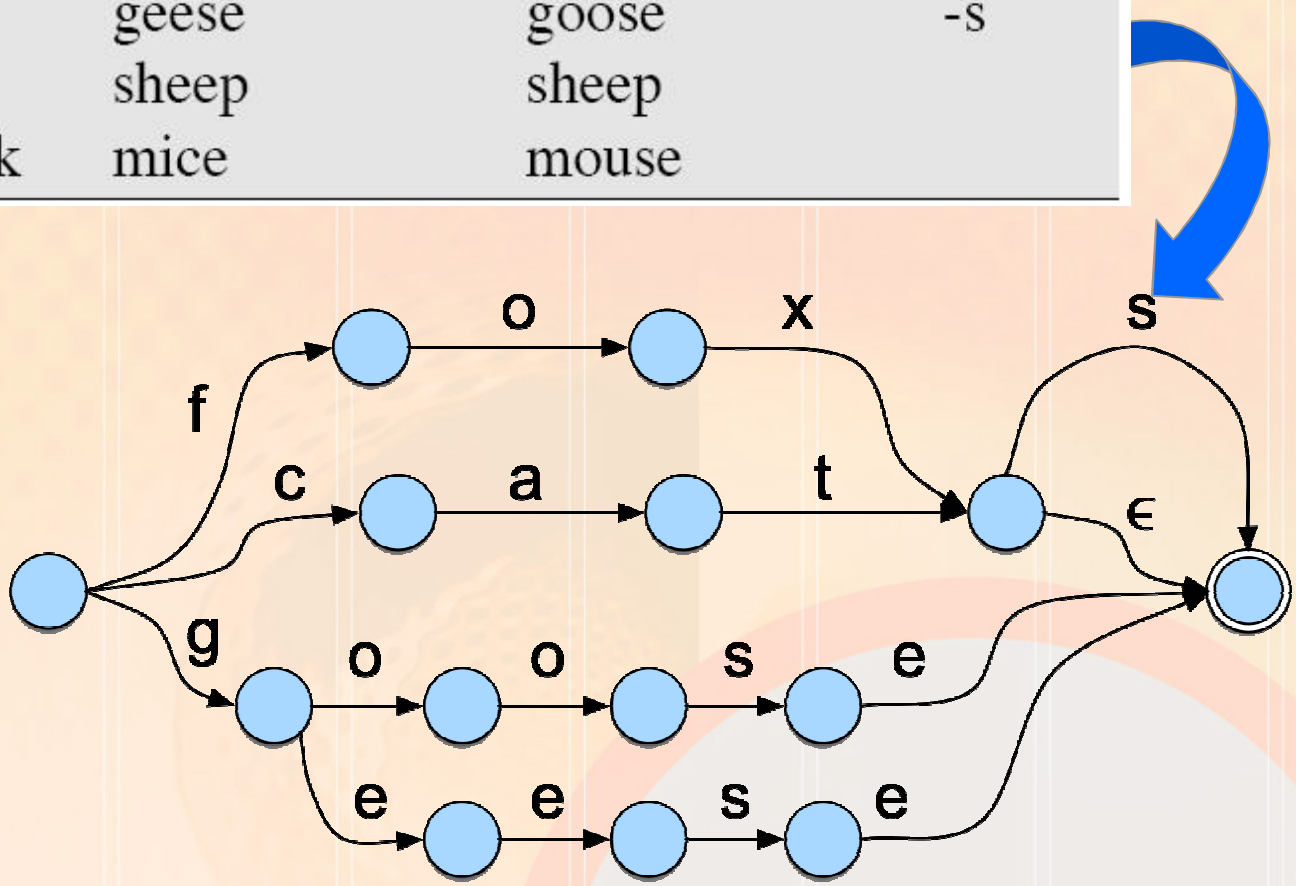
Building a finite-state lexicon: simple rules

- *A lexicon is a repository of words.*
- *Possibilities:*
  - *List all words in the language*
  - *Computational lexicons: list all stems and affixes of a language + representation of the morphotactics that tells us how they fit together.*
    - *A example of morphotactics: the finite-state automaton (FSA) for English nominal inflection*



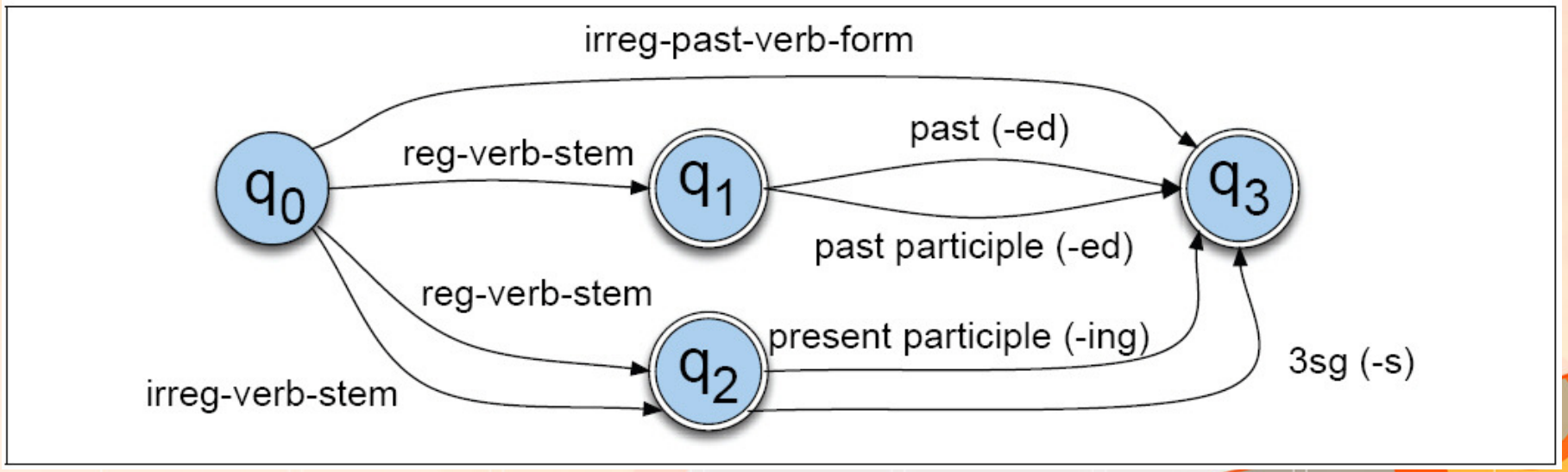
Now Plug in the Words

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s
cat	sheep	sheep	
aardvark	mice	mouse	



FSA for English Verb Inflection

reg-verb-stem	irreg-verb-stem	irreg-past-stem	past	past-part	pres-part	3sg
walk	cut	caught	-ed	-ed	-ing	-s
fry	speak	ate				
talk	sing	eaten				
impeach		sang				



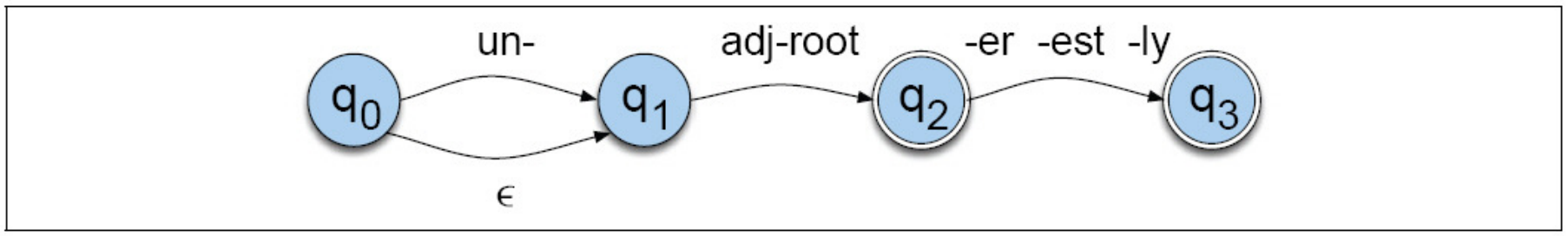
Models for derivational morphology

More complex than inflectional morphology. FSA tend to be more complex.

Simple case: morphotactics of English adjectives

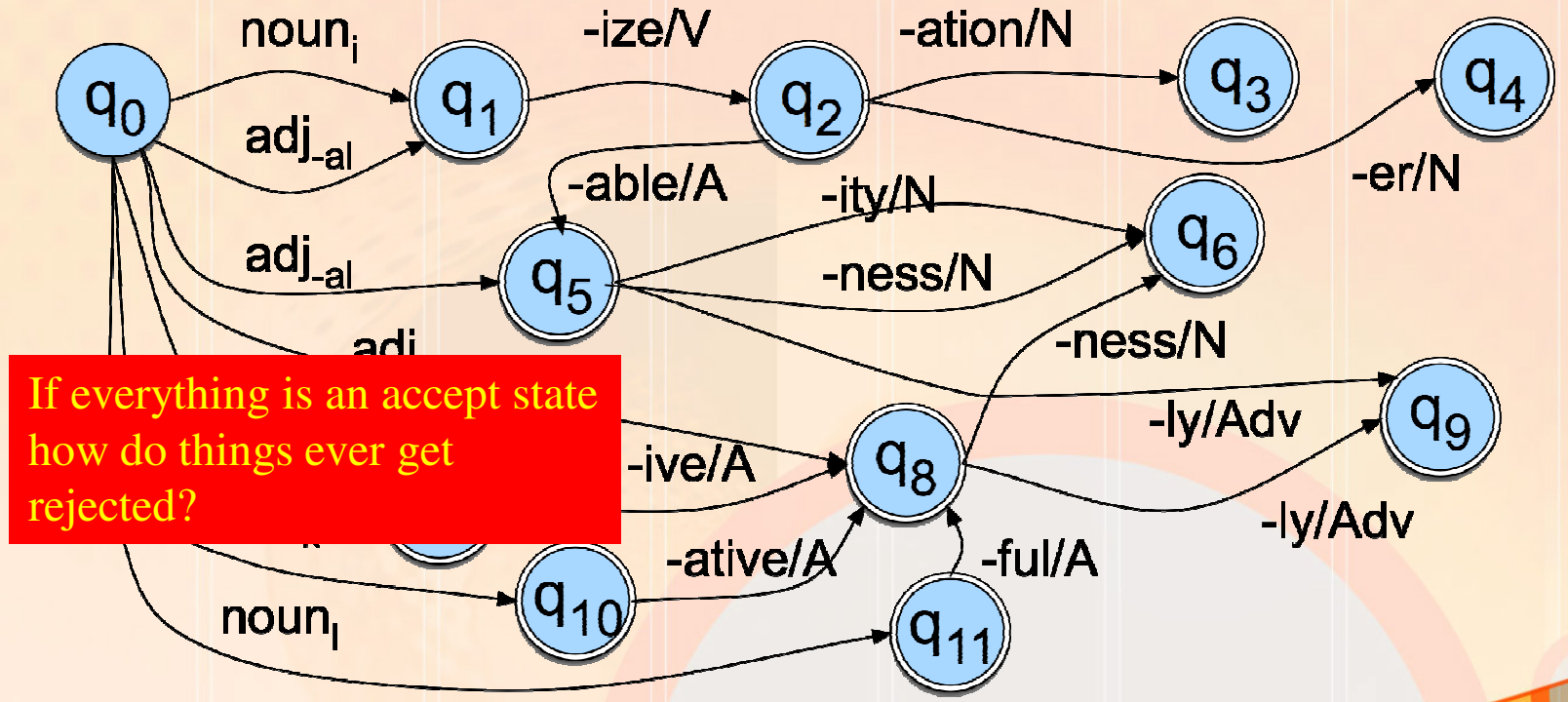
- *Examples from Antworth (1990)*

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	



- While this will recognize most of the adjectives, it will also recognize ungrammatical forms like: *unbig, redly, realest*.

Derivational Rules



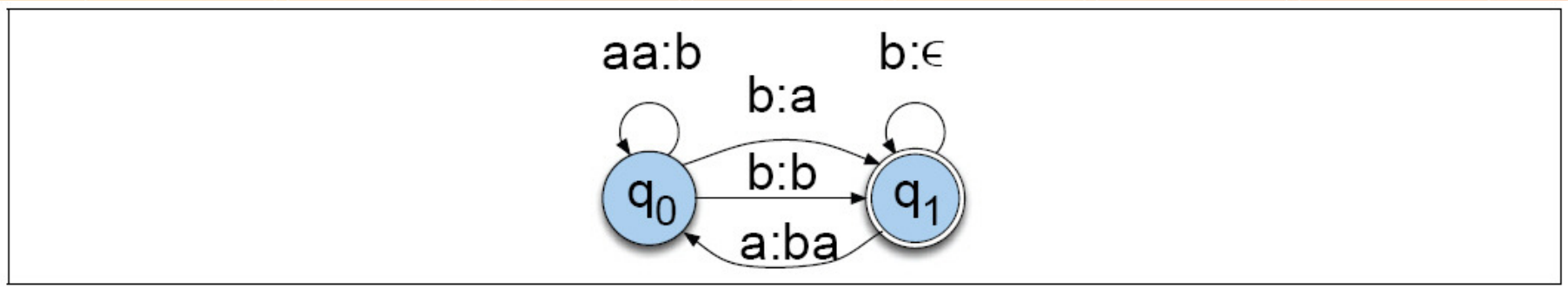
If everything is an accept state how do things ever get rejected?

Parsing/Generation vs. Recognition

- *We can now run strings through these machines to recognize strings in the language*
- *But recognition is usually not quite what we need*
  - *Often if we find some string in the language we might like to assign a structure to it (parsing)*
  - *Or we might have some structure and we want to produce a surface form for it (production/generation)*
- *Example*
  - *From "cats" to "cat +N +PL"*

Finite-State Transducers

- A transducer maps between one representation and another.
- A finite-state transducer (FST) is a type of finite automaton which maps between two sets of symbols.
  - *An FST is a two-tape automaton which recognizes or generates pairs of strings*. Thus we can label each arc in the finite-state machine with two symbols, one for each tape.

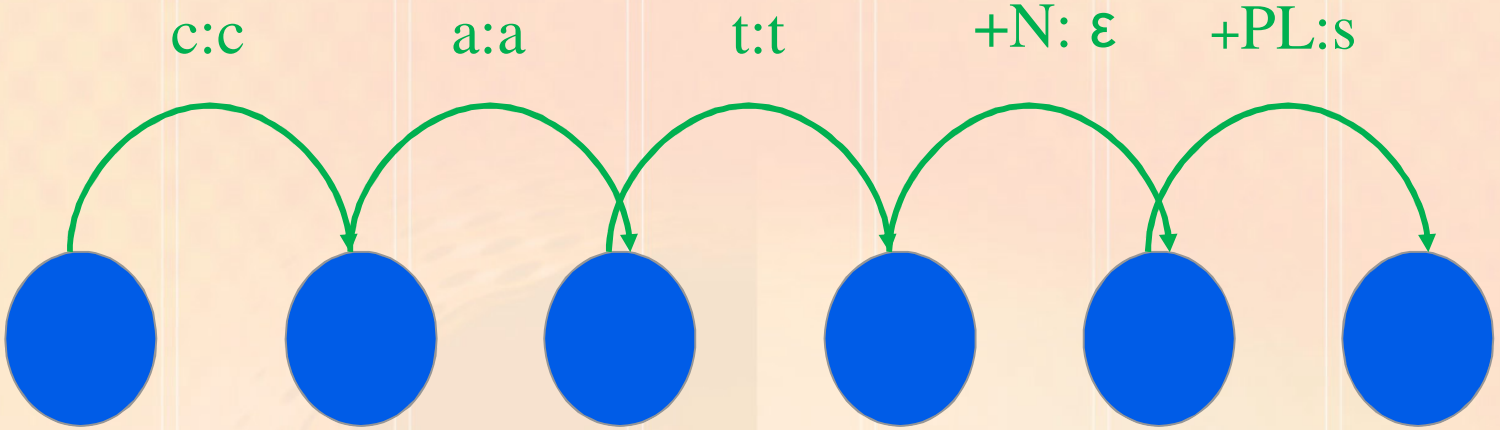


Finite State Transducers

- *The simple story*
  - *Add another tape*
  - *Add extra symbols to the transitions*

  - *On one tape we read "cats", on the other we write "cat +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*

Typical Uses

- *Typically, we'll read from one tape using the first symbol on the machine transitions (just as in a simple FSA).*
- *And we'll write to the second tape using the other symbols on the transitions.*

Ambiguity

- *Recall that in non-deterministic recognition multiple paths through a machine may lead to an accept state.*
  - *Didn't matter which path was actually traversed*
- *In FSTs the path to an accept state does matter since different paths represent different parses and different outputs will result*

FSTs and FSAs

- FSTs have a more general function than FSAs:
  - An FSA defines a formal language by defining a set of strings
  - An FST defines a relation between sets of strings

- Another view: an FST is a machine that reads one string and generates another one.

A four-fold way of thinking

- 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. The output is a *yes* or a *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.

Parsing is done with the FST.
A transducer maps one set of symbols into another.
An FST uses a finite automata.
Two level morphology : for each word is the correspondence between lexical level and surface level.

Formal definition of an FST

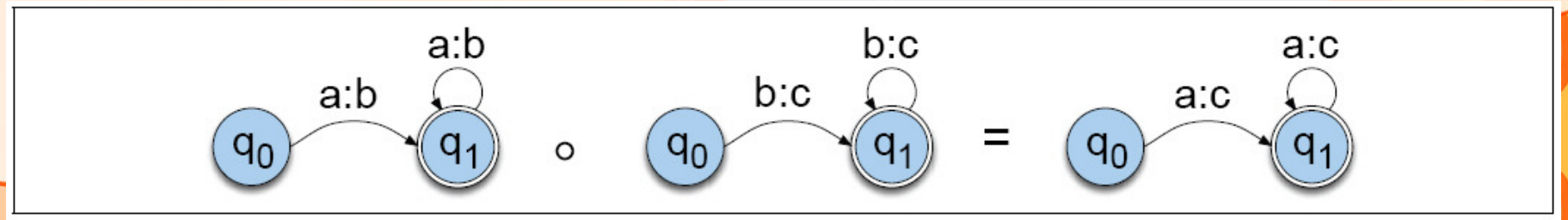
- *Defined by 7 parameters:*

1. *Q: a finite state of N states q_0, q_1, … , q_n*
2. *Σ: a finite set corresponding to the input alphabet*
3. *Δ: a finite set corresponding to the output alphabet*
4. *$q_o \in Q$: the start state*
5. *$F \subseteq Q$: the set of final states*
6. *$\delta(q,w)$: the transition function (or transition matrix between states)*
7. *$\sigma(q,w)$: the output function, giving the set of all possible output strings for each state and input.*

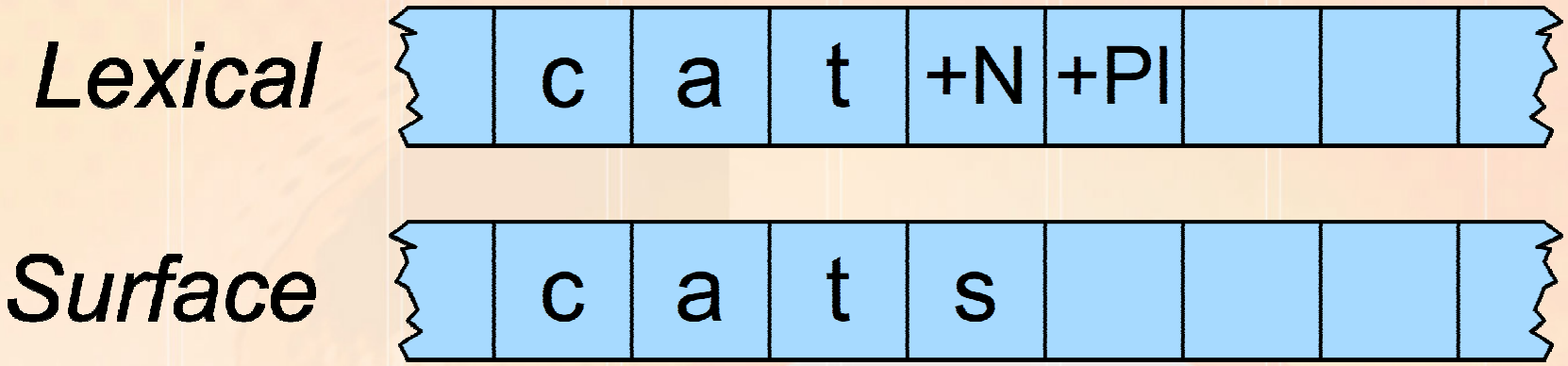
Regular Relations

- *Remember: FSAs are isomorphic to regular languages.*
- *FSTs are isomorphic to regular relations.*

- *Definition: Regular relations are sets of pairs of strings!!! (extension to regular languages, which are sets of strings)*
- *Operations:*
  - *Inversion: the inversion of a Transducer T (T^{-1}) switches the input with the output labels*
  - *Composition: If T_1 is a transducer from I_1 to O_1 and T_2 is a transducer from O_1 to O_2 then $T_1 \circ T_2$ is a transducer from I_1 to O_2*



FSTs for Morphological Parsing

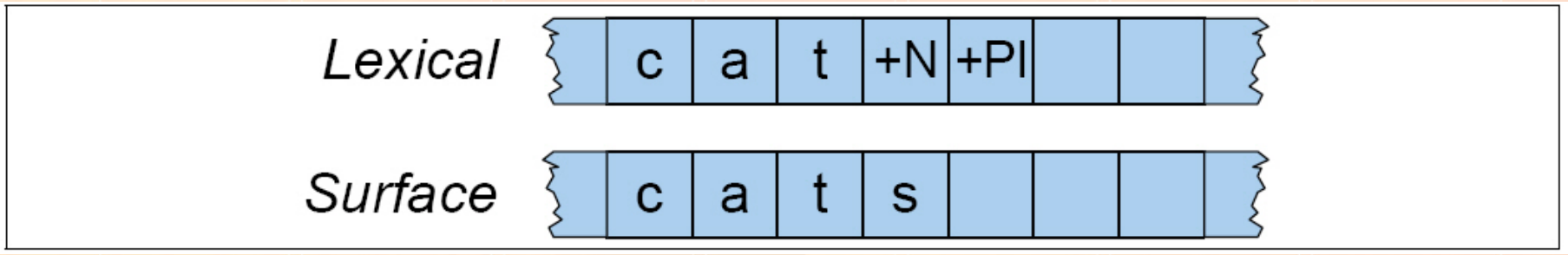


Applications

- *The kind of parsing we're talking about is normally called morphological analysis*
- *It can either be*
  - *An important stand-alone component of many applications (spelling correction, information retrieval)*
  - *Or simply a link in a chain of further linguistic analysis*

Morphological Parsing with Finite-State Transducers

- In finite-state morphology, we represent a word as a correspondence between a lexical level (concatenations of morphemes) and a surface level (concatenation of letters which make up the spelling of the word).



- For finite-state morphology, it is convenient to view an FSt as having two tapes. The lexical tape is composed from characters from one alphabet Σ. The surface tape is composed from characters from another alphabet Δ.

The two alphabets can be combined in a new alphabet of complex symbols Σ'. Each complex symbol is composed of an input-output pair: (i,o) with $i \in \Sigma$ and $o \in \Delta$, thus $\Sigma' \subseteq \Sigma \times \Delta$. Note, Σ and Δ may include the epsilon symbol ε

Feasible pairs

- The pairs of symbols from Σ' are called feasible pairs.
- Meaning: each feasible pair symbol *a:b* from Σ' expresses how the symbol *a* from one tape is mapped in the symbol *b* on the other tape.
  - Example: *a:ε* means that a on the upper tape will correspond to nothing on the lower tape.
  - Pairs *a:a* are called default pairs and we refer to them by the single letter *a*.

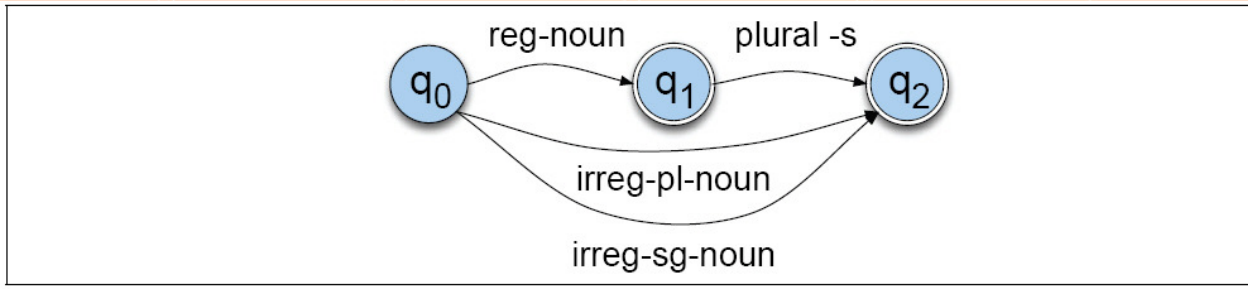
An FST mophological parser can be built from a morphotactic FSA by adding an extr "lexical" tape and the appropriate morphological features.

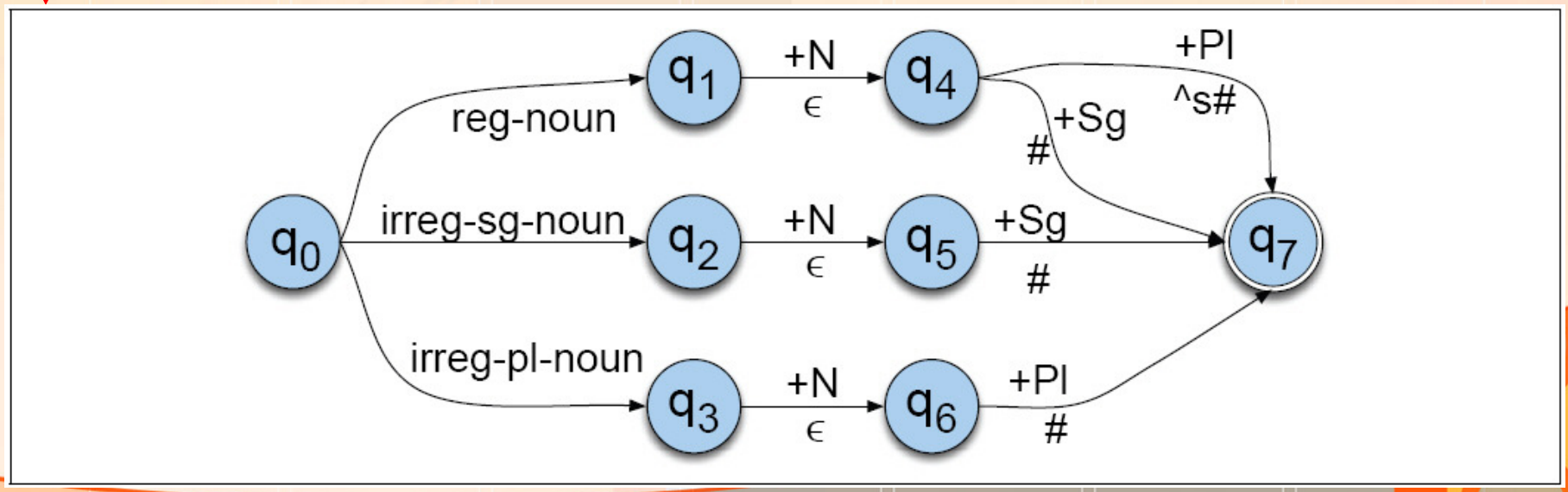
How??

How??

- Use nominal *morphological features* (+Sg and +Pl) to augment the FSA for nominal inflection:

The symbol ^ indicates morpheme boundary
The symbol # indicates word boundary



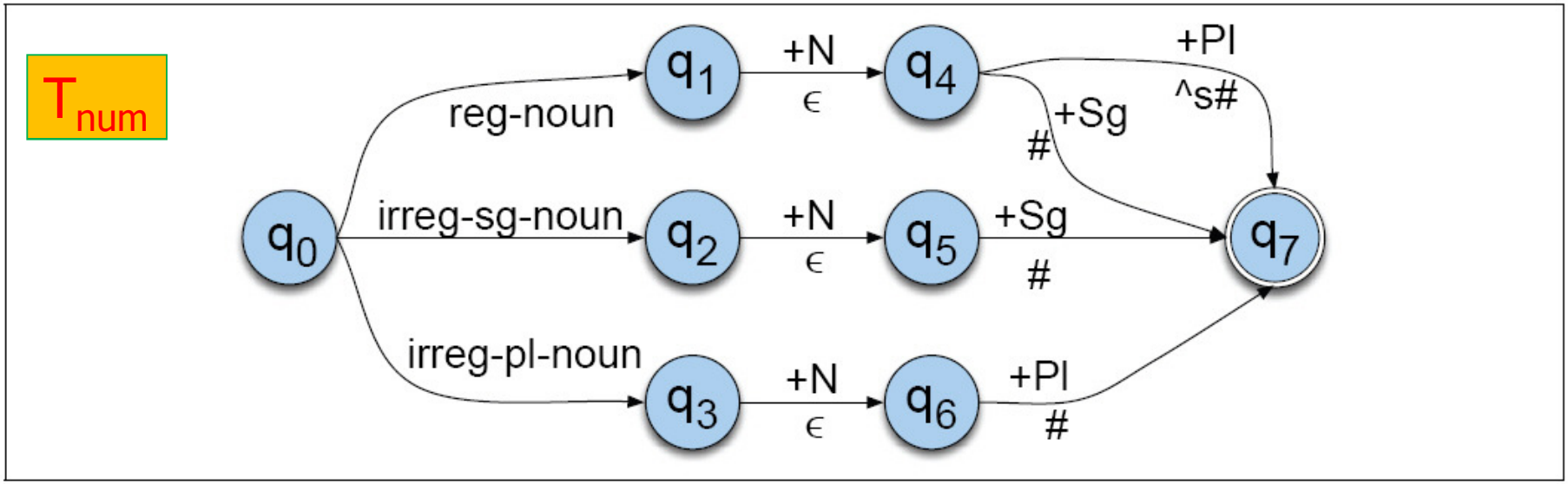


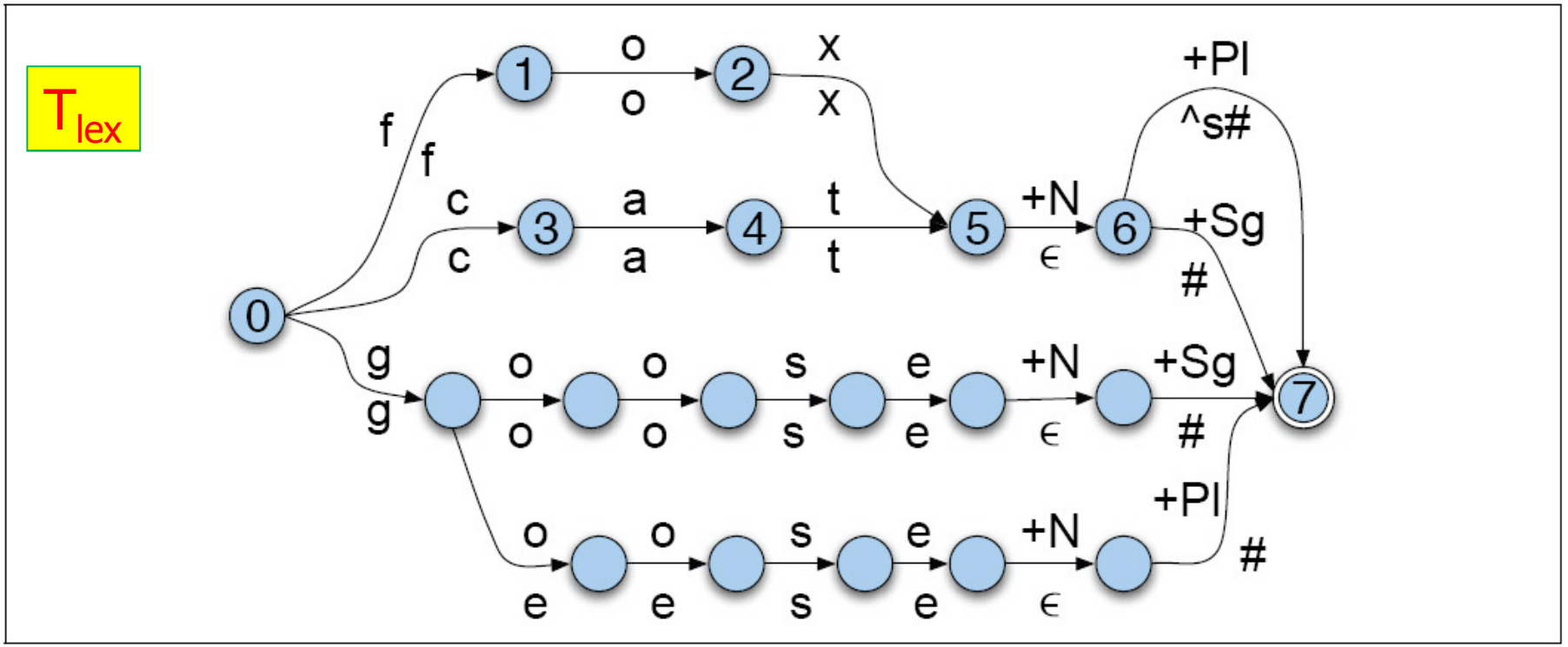
Expanding FSTs with lexicons

- Update the lexicon such that irregular plurals such as geese will parse into the correct stem: goose +N +Pl
- The lexicon can have also two levels
  - Example: "g:g o:e o:e s:s e:e" or "g o:e o:e s e"
- The lexicon will be more complex:

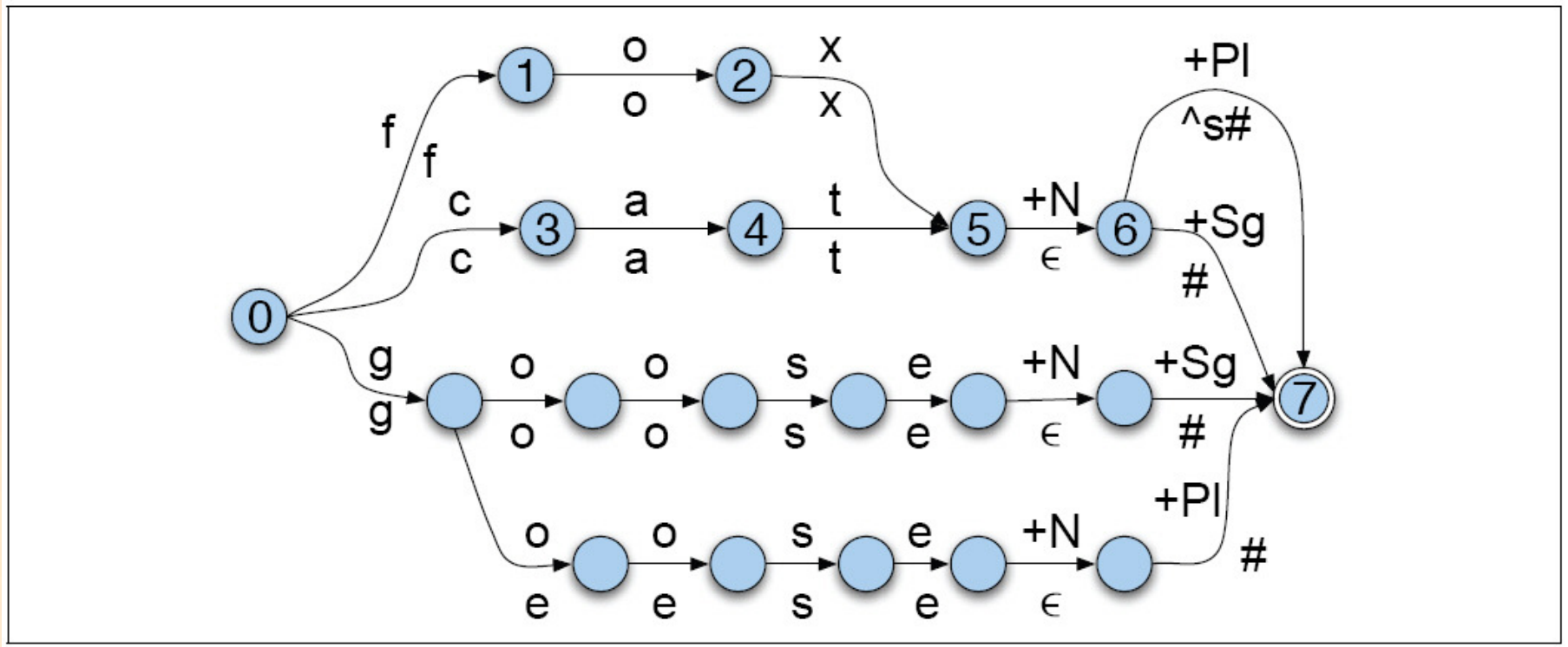
reg-noun	irreg-pl-noun	irreg-sg-noun
fox	g o:e o:e s e	goose
cat	sheep	sheep
aardvark	m o:i u:ϵ s:c e	mouse

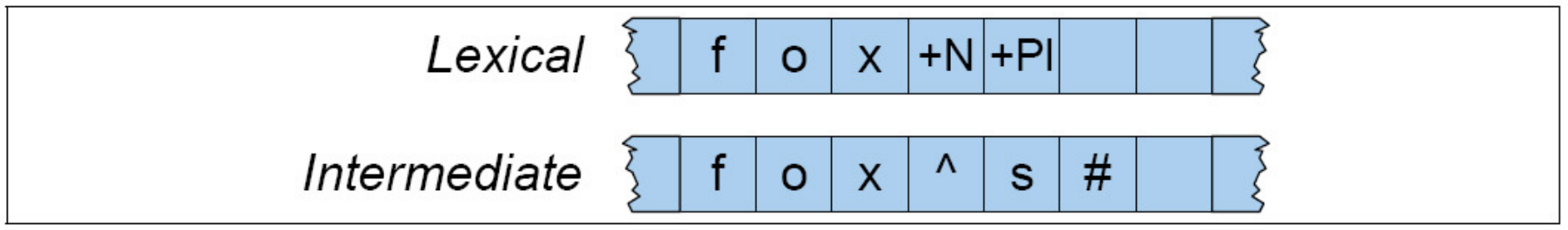
Expanding a nominal inflection FST





Intermediary Tapes



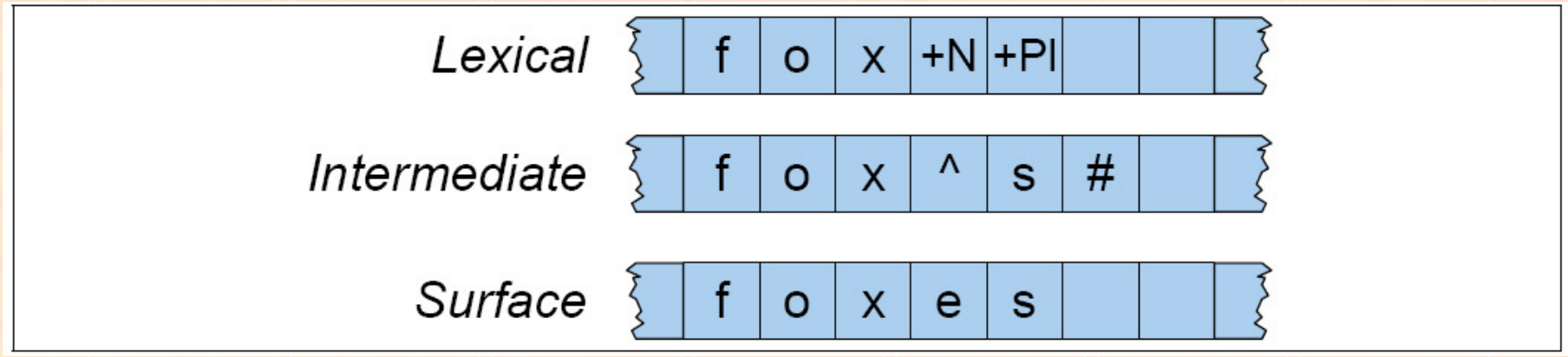


Transducers and Orthographic Rules

- Problem: English often requires spelling changes at morpheme boundaries
- Solution: introduce spelling rules (orthographic 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	verbs ending with *vowel + -c* add *-k*	panic/panicked

How to handle context-specific spelling rules.
cat + N + PL -> cats (this is OK)
fox + N + PL -> foxs (this is not OK)

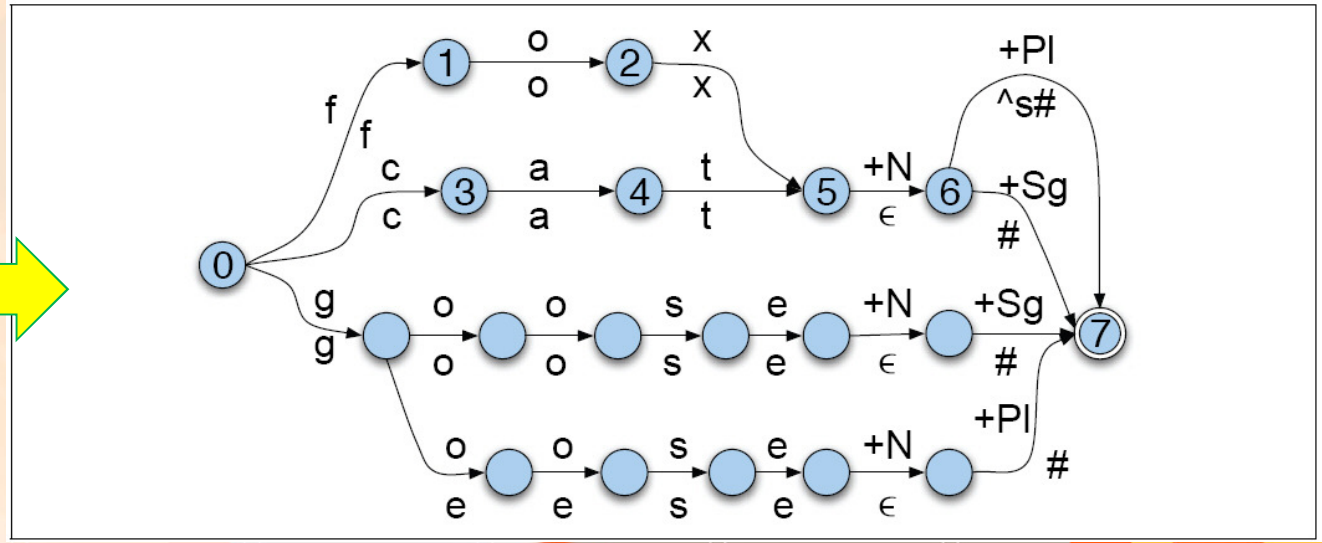


Example of lexical, intermediate and surface tapes.

Between each tape there is a two-level transducer!

FST between the lexical and intermediate levels

The E-insertion rule between the intermediate Land the surface level



E-insertion rule

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

How do we formalize it?

Chomsky and Halle's Notation

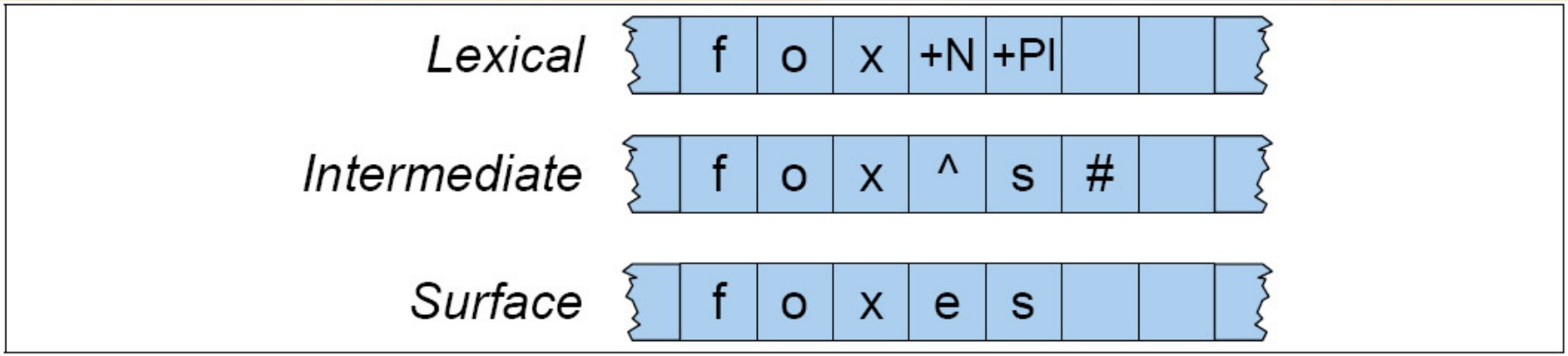
- *a -> b / c____d*

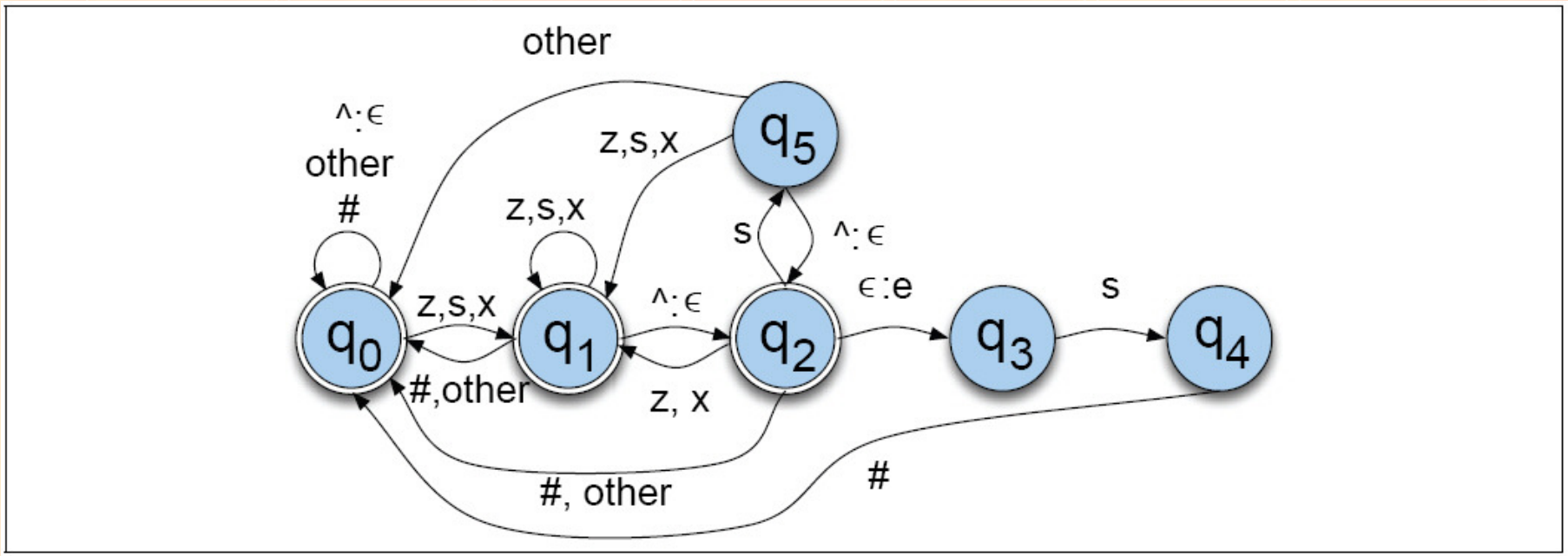
  rewrite a as b when it occurs between c and d.

- *ε -> e / {x, s, z} ^____s#*

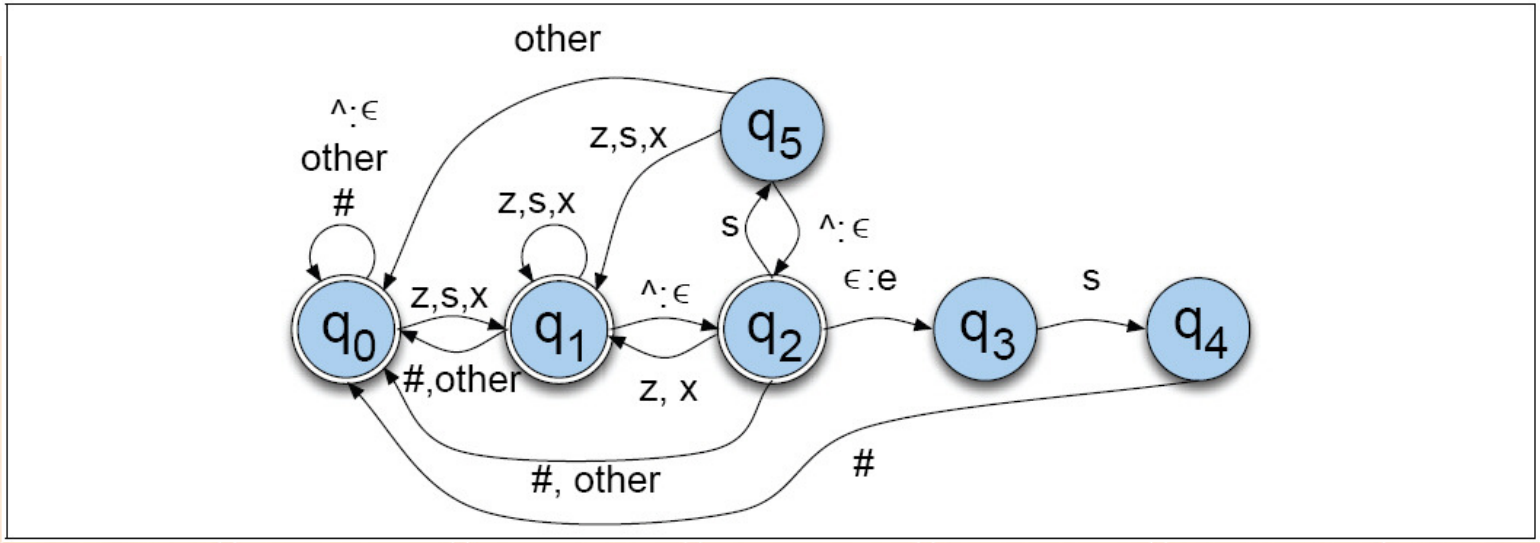
Insert e on the surface tape when the lexical tape has a morpheme ending in x, s or z and the next morpheme is –s.

The E-insertion rule



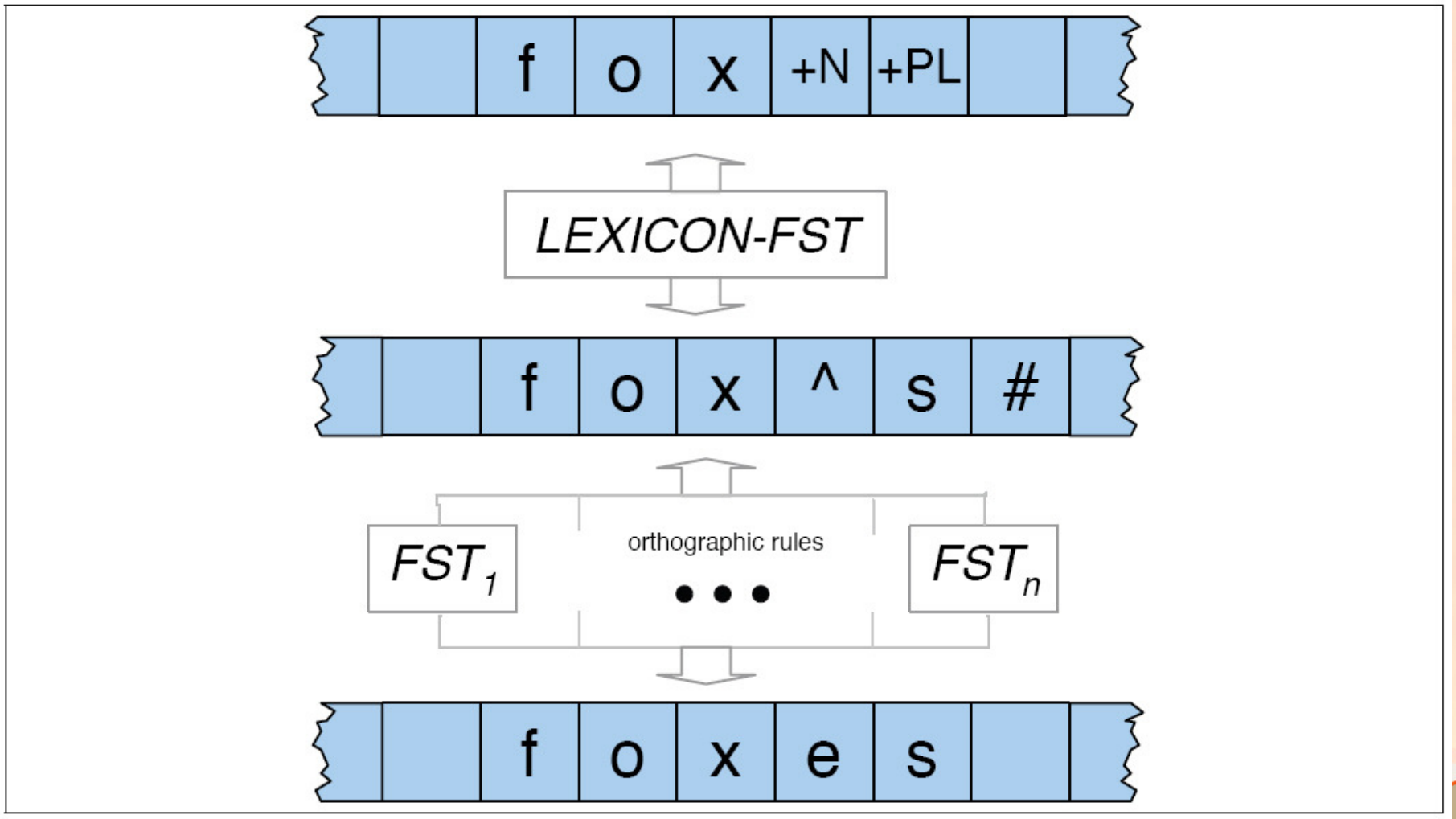


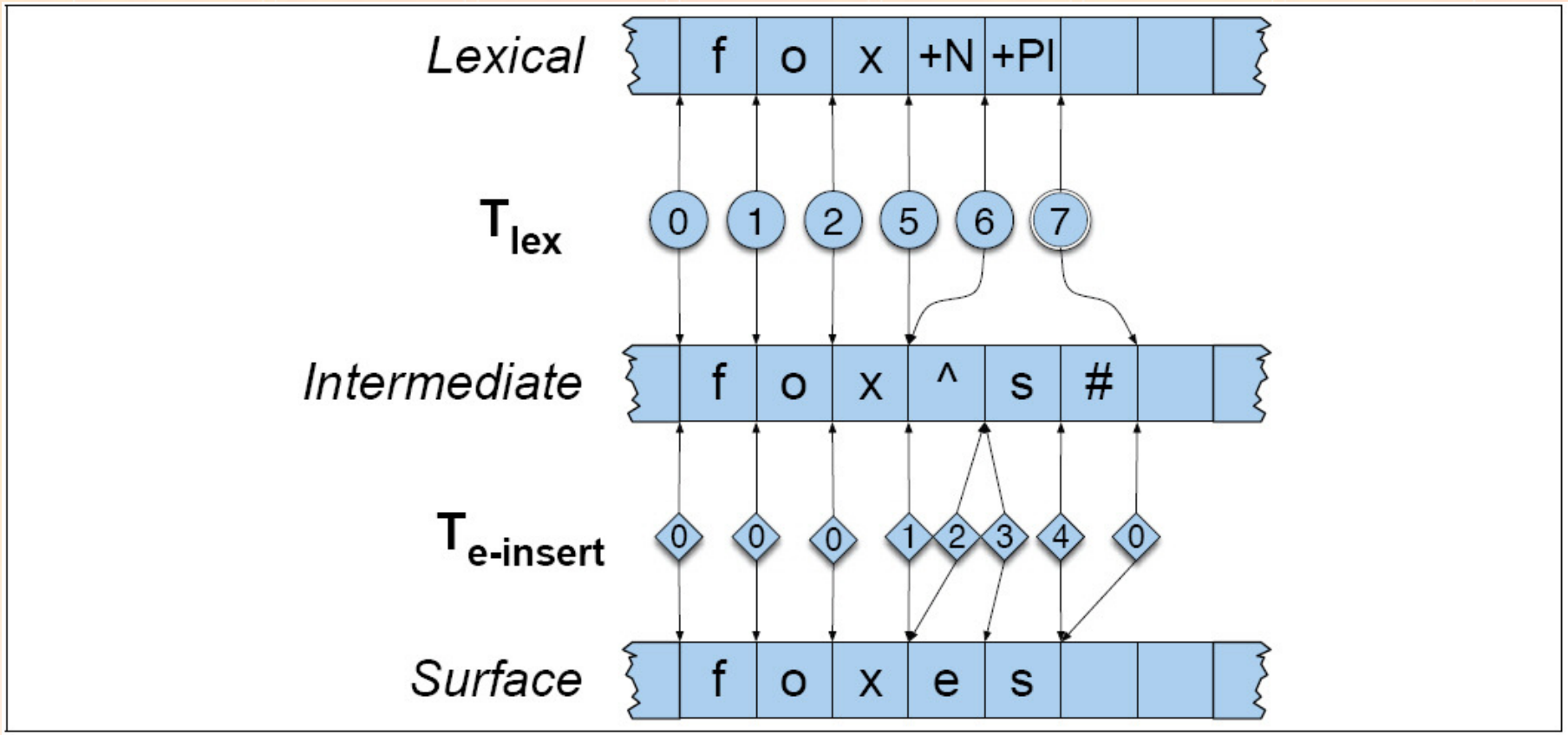
Q_o models having seen only default pairs, unrelated to the rule
Q_1 models having seen z,s or x
Q_2 models having seen the morpheme boundary after the z,s,x
Q_3 models having just seen the E-insertion – not an accepting state
Q_5 is there to insure that *e* is always inserted when needed



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

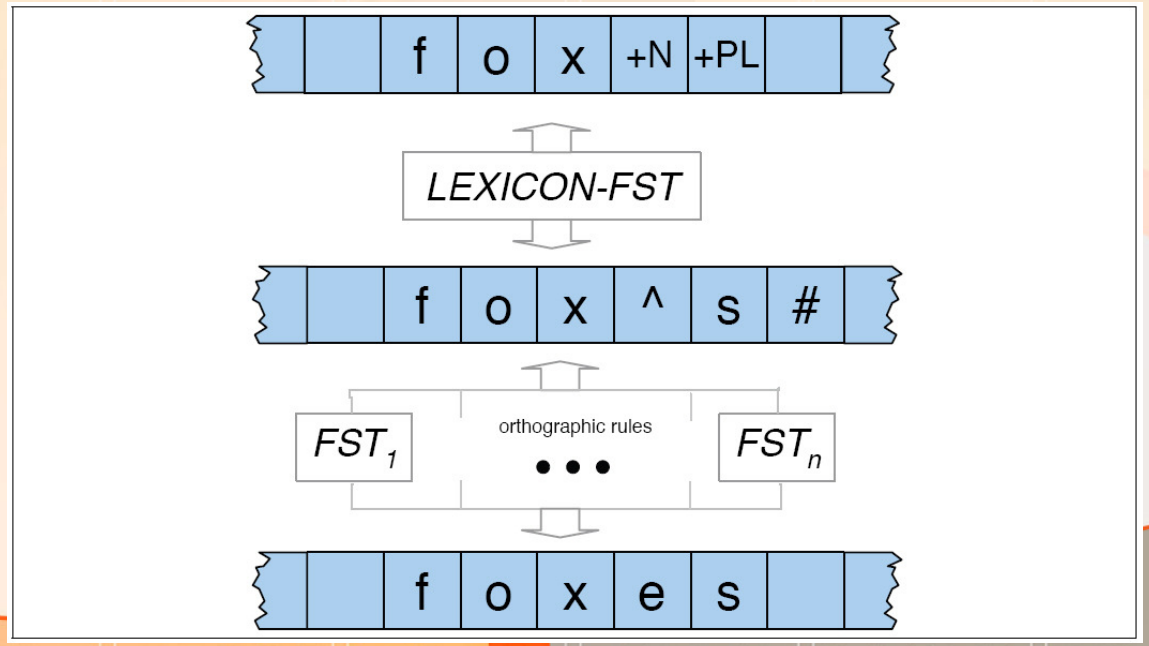
Combining FST Lexicon and Rules




Lexical
f o x +N +Pl
T lex
0 1 2 5 6 7
Intermediate
f o x ^ s #
T e-insert
0 0 0 1 2 3 4 0
Surface
f o x e s

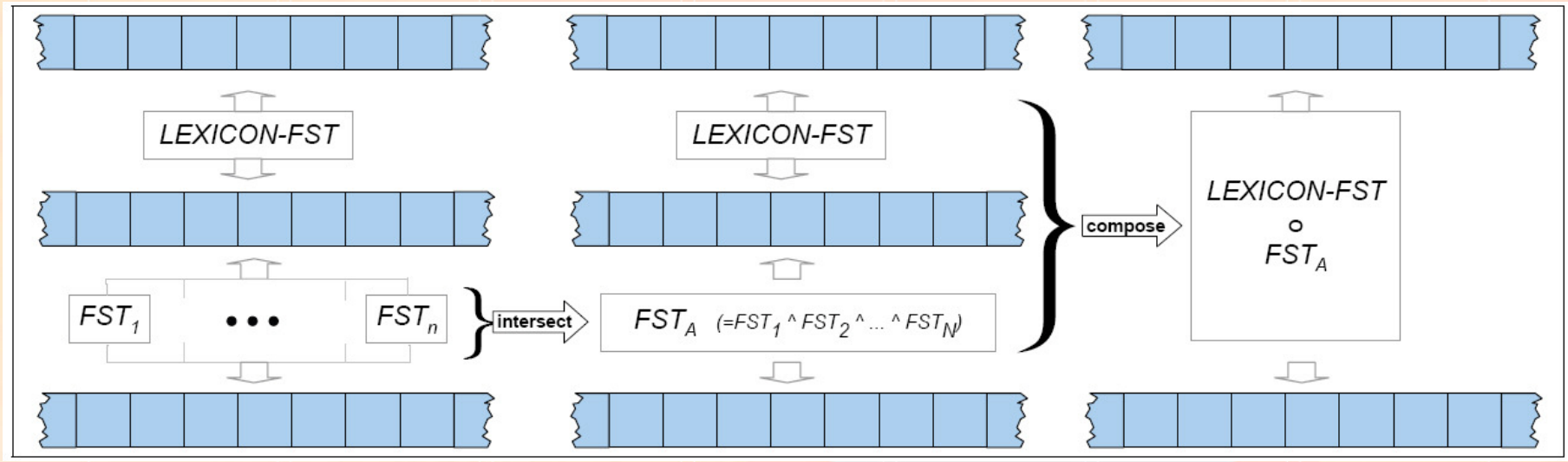
Some difficulties

- Parsing the lexical tape from the surface tape
- Generating the surface tape from the lexical tape.
- Parsing has to deal with ambiguity
- Disambiguation requires some external evidence:
  - Example: "I saw two foxes yesterday" Fox is a noun !
    "That trickster foxes me everytime!" Fox is a verb!



Automaton Intersection

Transducers in parallel can be combined by **automaton intersection**:
Take the Cartesian product of the states and create a new set of output states



Composition

1. *Create a set of new states that correspond to each pair of states from the original machines (New states are called (x,y), where x is a state from M1, and y is a state from M2)*
2. *Create a new FST transition table for the new machine according to the following intuition…*

Composition

- *There should be a transition between two states in the new machine if it's the case that the output for a transition from a state from M1, is the same as the input to a transition from M2 or…*

Composition

- $\delta_3((x_a,y_a), i{:}o) = (x_b,y_b)$ *iff*
  - *There exists c such that*
  - $\delta_1(x_a, i{:}c) = x_b$ *AND*
  - $\delta_2(y_a, c{:}o) = y_b$