Speech and Language Technology

Morphology & Transducers

Topics

- Intro to morphological analysis of languages
- Motivation for morphological analysis in NLP
- Morphological Recognition by FSAs
- Transducers
- Unsupervised Learning (2nd hour)

Source

- Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. Daniel Jurafsky & James H. Martin.

- Available online: http://www.cs.vassar.edu/~cs395/docs/3.pdf

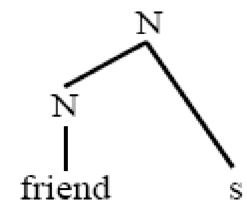
Intro to Morphological Analysis

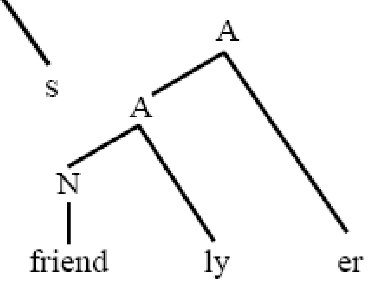
- Morphology is the study of the internal structure of words.
- Words structure is analyzed by composition of morphemes – the smallest units for grammatical analysis:
  - *Boys: boy-s*
  - *Friendlier*: friend-ly-er
  - *Ungrammaticality*: un-grammat-ic-al-ity
- Semitic languages, like Hebrew and Arabic, are based on templates and roots.
- We will concentrate on affixation-based languages, in which words are composed of stems and affixes.

Intro – Morphological Processes

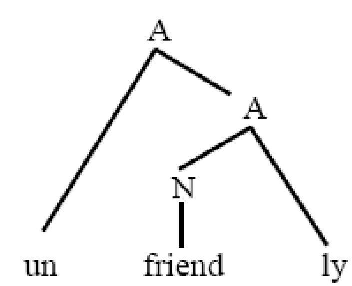
- Two types of morphological processes:
  - Inflectional (in-category; paradigmatic):
    - Nouns: *friend* → *friends*
    - Adjs*:* *friendly* → *friendlier*
    - Verbs*:* *do* → *does, doing, did, done*

    Stands for *gender, number, tense, etc.*





  - Derivational: (between-categories; non-paradigmatic)
    - Noun→ Adj: *friend* → *friendly*
    - Adj → Adj*:* *friendly* → *unfriendly*
    - Verb → Verb: *do* → r*edo*, *undo*



Regular vs. Irregular Inflection

- Regular Inflection – Rule-governed
  - The same morphemes are used to mark the same functions
  - The majority of verbs (although not the most frequent) are regular, for example:

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

  - Relevant also for nouns, e.g. –s for plural.

Regular vs. Irregular Inflection

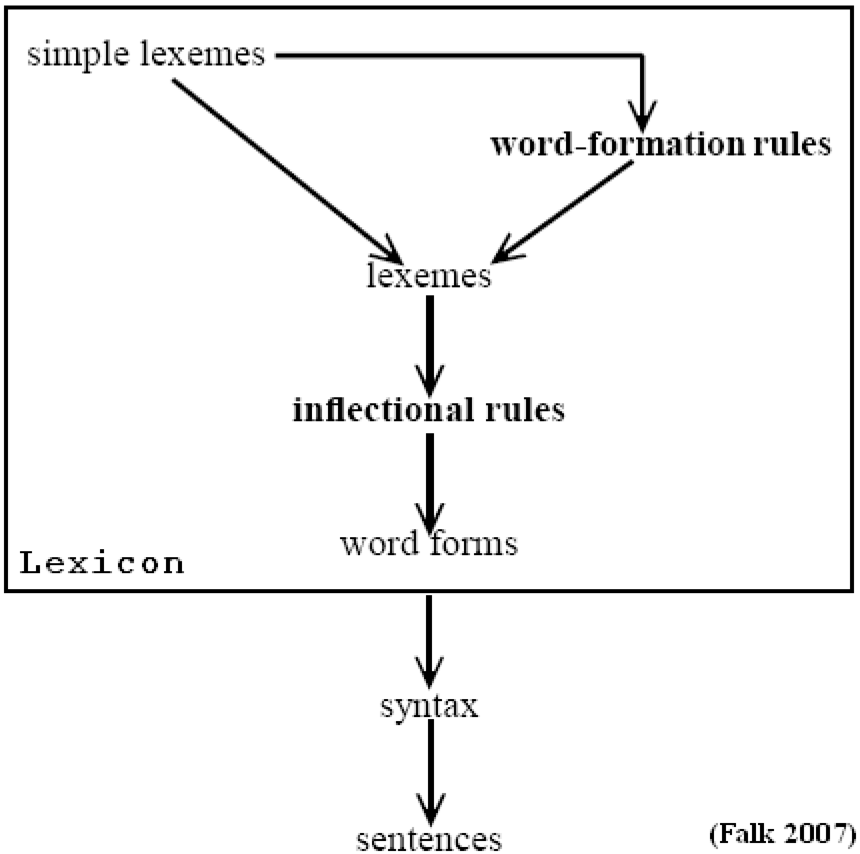
- Irregular Inflection – Idiosyncratic
  - Inflection according to several subclasses characterized morpho-phonologically (e.g. think → thought, bring → brought, etc.)

Morphological Form Classes	Irregularly Inflected Verbs		
stem	eat	catch	cut
-*s* form	eats	catches	cuts
-*ing* participle	eating	catching	cutting
Past form	ate	caught	cut
-*ed*/-*en* participle	eaten	caught	cut

  - Relevant also for nouns, e.g. *Analysis* (sg) → *Analyses* (pl)

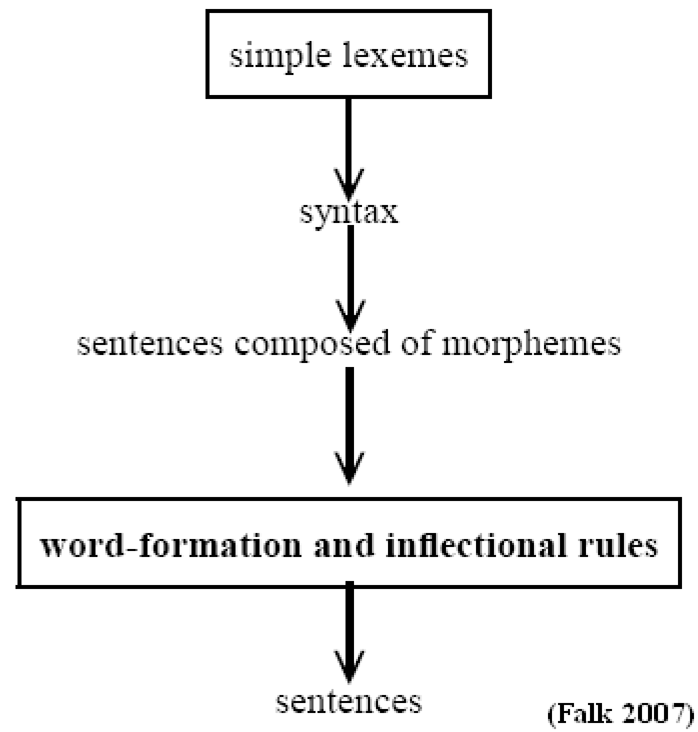
Intro – The Morpho-Syntax Interface

- Strong Lexicalism
  - The lexicon contains fully inflected/derived words.
  - Full separation between morphology and syntax (two engines)
  - Popular in NLP (e.g. LFG, HPSG)



Intro – The Morpho-Syntax Interface

- Non-Lexicalism
  - The lexicon contains only morphemes
  - The syntax creates both words **and** sentences (single engine of composition)
  - Popular in theoretical linguistics (e.g. Distributed Morphology)



Morphological Parsing

- The problem of recognizing that a word (like *foxes*) breaks down into component morphemes (*fox* and *–es*) and building a structured representation of this fact.
- So given the **surface** or **input form** *foxes*, we want to produce the parsed form VERB–want + PLURAL–es.

Issues We Will Not Address

- Analysis ambiguity: words with multiple analyses:
  - [un-lock]-able – something that can be unlocked.
  - un-[lock-able] – something that cannot be locked.
- Allomorphy: the <u>same</u> morpheme is spelled out as different allomorphs:
  - Ir-regular
  - Im-possible
  - In-sane
- Orthographic rules:
  - saving ← save + ing, flies ← fly + s.
  - Chomsky+an vs. Boston+**i**+an vs. disciplin+**ari**+an

Motivation for Morphology in NLP

- Search engines and information retrieval tasks (stemming)
- Machine Translation (stemming, applying morphological processes)
- Models for sentence analysis and construction (stemming, morphological processes, semantic features of morphemes)
- Speech recognition (the morpho-phonology interface, to be addressed later in this course)

The Conservative Approach

- Storing all possible breakdowns of all words in the lexicon.
- Problems:
  - Morphemes can be **productive**, e.g. *-ing* is a productive suffix that attaches to almost every verb.
    - It is inefficient to store all possible breakdowns while there a principle can be defined.
    - Productive suffixes even apply to new words; thus the new word *fax* can automatically be used in the *-ing* form: *faxing.*

The Conservative Approach

- Problems:
  - Morphologically complex languages, e.g. Finish:

arvo	n	lisä	vero	ttoma	sta
value	of	addition	tax	-less	from

Figure 1. Morpheme segmentation of the Finnish word 'arvonlisäverottomasta' ("from [something] exclusive of value added tax").

we cannot list all the morphological variants of every word in morphologically complex languages like Finish, Turkish, etc. (**agglutinative** languages)

FINITE-STATE MORPHOLOGICAL PARSING

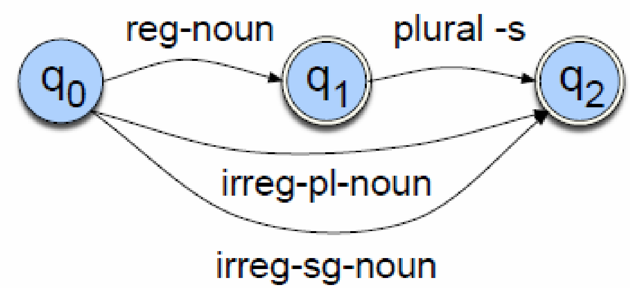
- Goal: to take input forms like those in the first column and produce output forms like those in the second.

English	
Input	Morphologically Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +Pl
geese	goose +N +Pl
goose	goose +N +Sg
goose	goose +V
gooses	goose +V +1P +Sg
merging	merge +V +PresPart
caught	catch +V +PastPart
caught	catch +V +Past

A FINITE-STATE LEXICON

- Computational lexicons are usually structured with a list of each of the stems and affixes of the language together with a representation of the morphotactics that tells us how they can fit together.
- For nouns inflection:

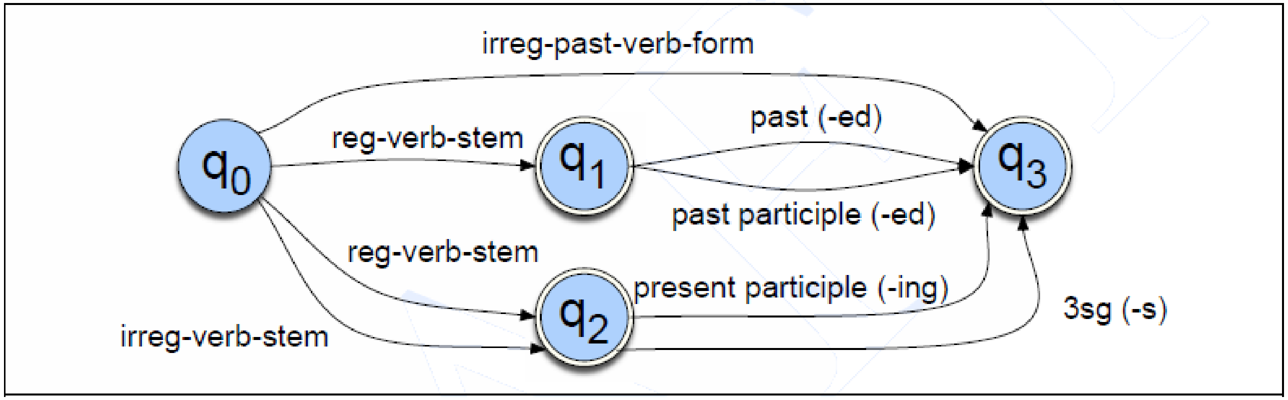
  (we assume that the bare nouns are given in advance)



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

A FINITE-STATE LEXICON

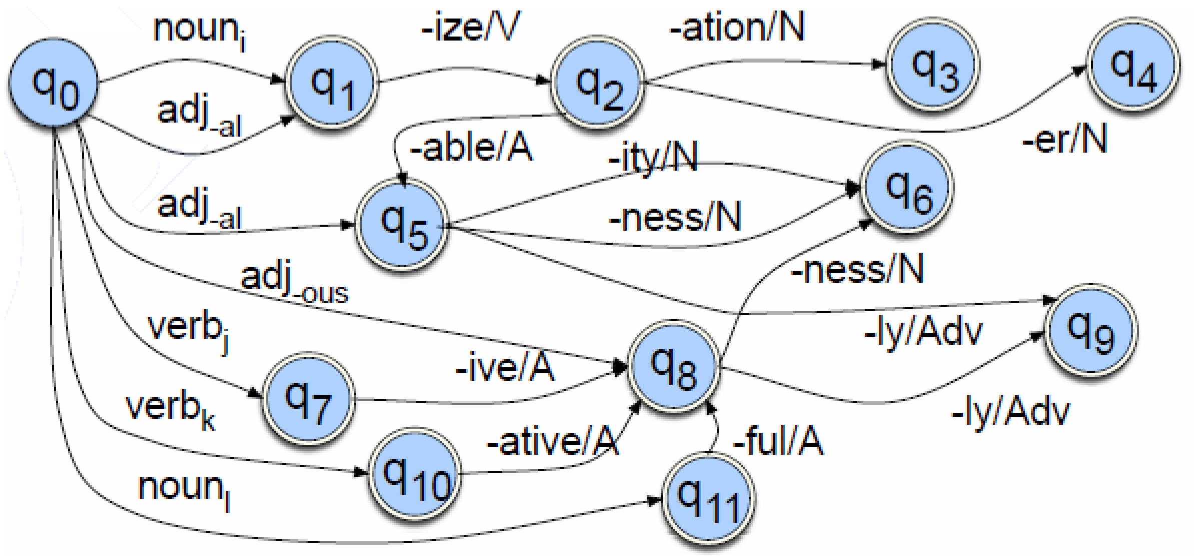
- For verbal inflection:



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

A FINITE-STATE LEXICON

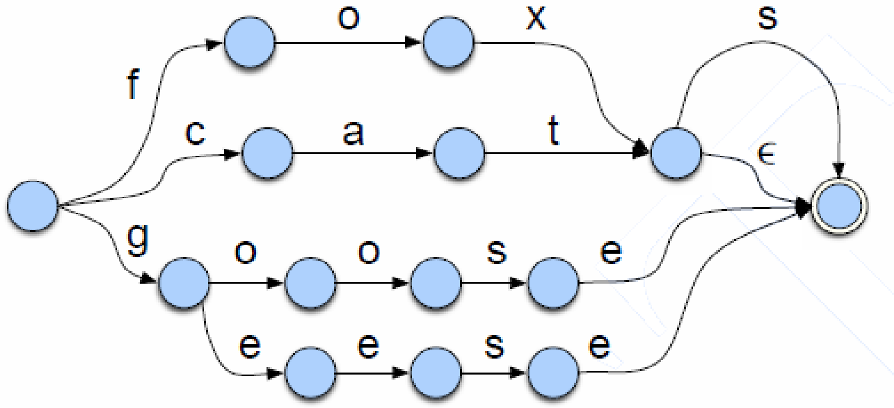
- The bigger picture:



- **morphotactics**: the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the English plural morpheme follows the noun.

Morphological Recognition by FSAs

- Determining whether an input string of letters makes up a legitimate English word or not.
- We do this by taking the FSAs and plugging in each "sub lexicon" into the FSA.
- That is, we expand each arc (e.g., the **reg-noun-stem** arc) with all the morphemes that make up the set of **reg-noun-stem**.
- The resulting FSA is defined at the level of the individual letter. (this diagram ignores orthographic rules like the addition of 'e' in 'fox<u>e</u>s'; it only shows the distinction between recognizing regular and irregular forms)

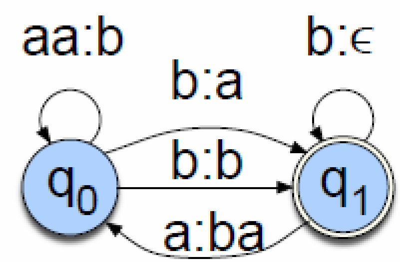


Finite-State Transducers

- A **finite-state transducer** or **FST** is a type of finite automaton which maps between two sets of symbols.
- We can visualize an FST as a two-tape automaton which recognizes or generates *pairs* of strings.
- This can be done by labeling each arc in the finite-state machine with two symbol strings, one from each tape.

Finite-State Transducers

- The FST has a more general function than an FSA; where an FSA defines a formal language by defining a set of strings, an FST defines a *relation* between sets of strings.
- Another way of looking at an FST is as a machine that reads one string and generates another.
- Example of FST as recognizer:

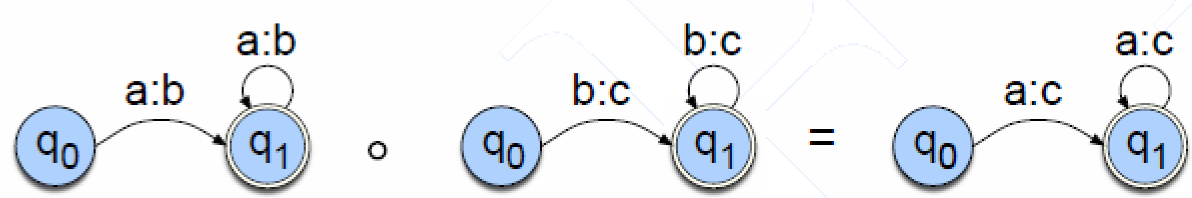


Finite-State Transducers

- Formally, an FST is defined as follows:
  - Q – 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
  - δ(q,w) – the transition function or transition matrix between states; Given a state $q \in Q$ and a string $w \in S_*$, d(q,w) returns a set of new states $Q' \in Q$.
  - σ(q,w) the output function giving the set of possible output strings for each state and input.

Operations on FSTs

- **Inversion**: The inversion of a transducer T (T^{-1}) 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 O_1 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2.
- The composition of [a:b] with [b:c] to produce [a:c]

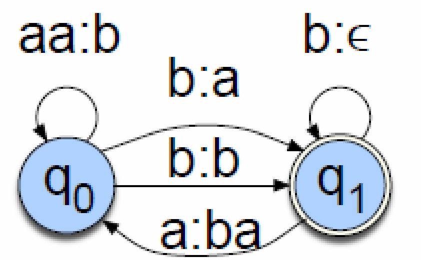


Sequential Transducers and Determinism

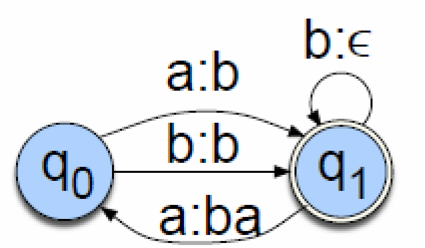
- Transducers can be non-deterministic: a given input can be translated to many possible output symbols.
- While every non-deterministic FSA is equivalent to some deterministic FSA, not all finite-state transducers can be determinized.
- **Sequential transducers**, by contrast, are a subtype of transducers that are deterministic on their input.
- At any state of a sequential transducer, each given symbol of the input alphabet Σ can label at most one transition out of that state.

Sequential Transducers and Determinism

- A non-deterministic transducer:



- A sequential transducer:

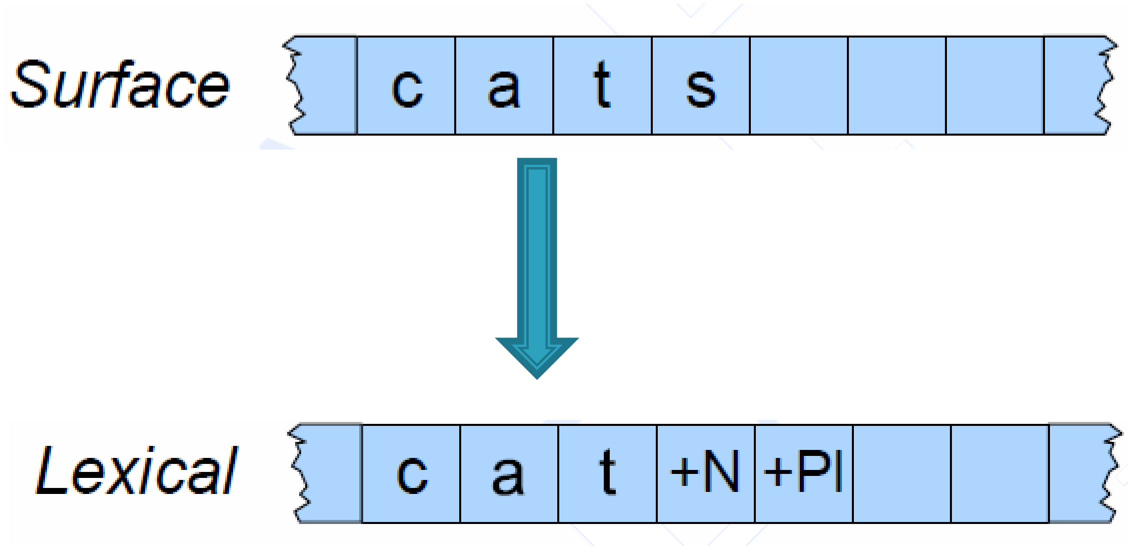


Sequential Transducers and Determinism

- **Subsequential transducer – a** generalization of sequential transducers is the which generates an additional output string at the final states, concatenating it onto the output produced so far.
- Sequential and subsequential transducers are important due to their efficiency; because they are deterministic on input, they can be processed in time proportional to the number of symbols in the input.
- Another advantage of subsequential transducers is that there exist efficient algorithms for their determinization (Mohri, 1997) and minimization (Mohri, 2000).
- However, While both sequential and subsequential transducers are deterministic and efficient, neither of them is able to handle ambiguity, since they transduce each input string to exactly one possible output string.
- Solution: see in the book.

Back to FINITE-STATE MORPHOLOGICAL PARSING

- We are interested in the transformation:

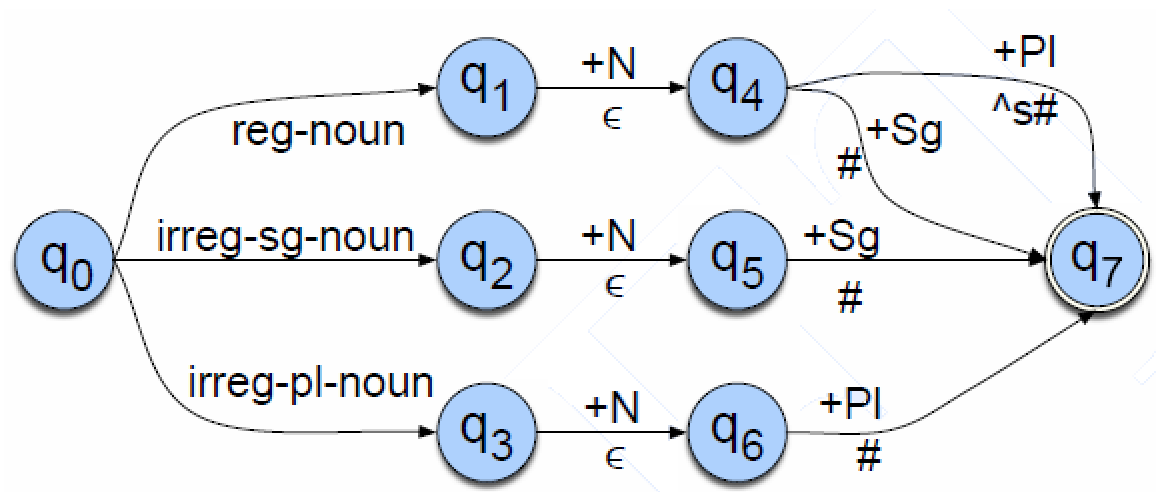


- The s**urface level** represents the concatenation of letters which make up the actual spelling of the word

- The **lexical level** represents a concatenation of morphemes making up a word

Back to FINITE-STATE MORPHOLOGICAL PARSING

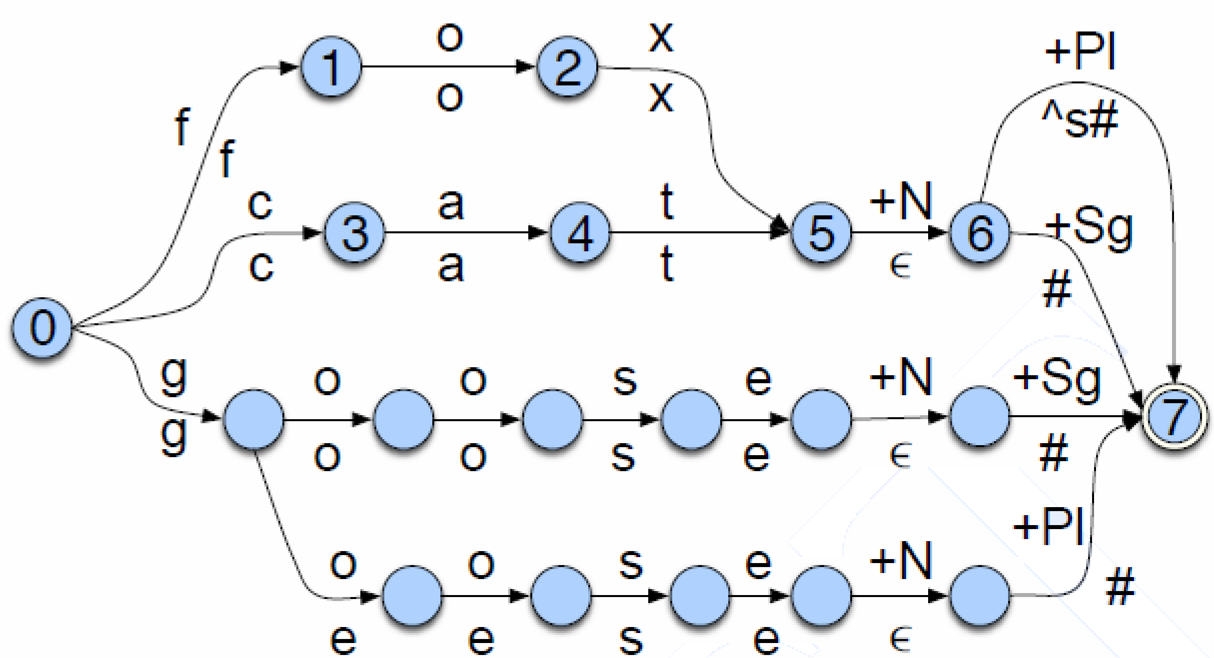
- A transducer that maps plural nouns into the stem plus the morphological marker +Pl, and singular nouns into the stem plus the morphological marker +Sg.
- Text below arrows: input; above: output.



Back to FINITE-STATE MORPHOLOGICAL PARSING

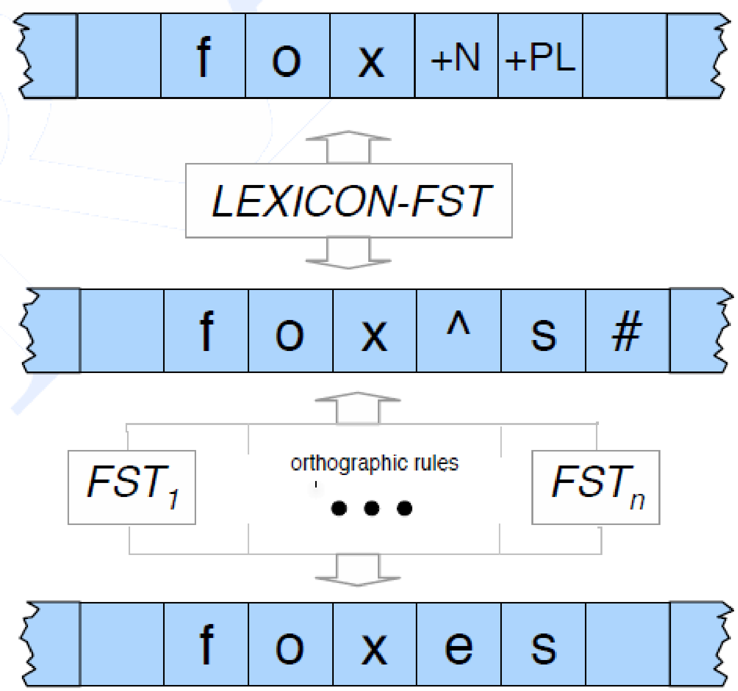
- Extracting the reg-noun, irreg-pl/sg-noun:

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



More topics covered in the book

- Taking into account orthographic rules (e.g. how to account for *fox<u>**e**</u>s*)
- Introducing an intermediate level of representation and composing FSTs:
- Allowing bi-directional transformation.



More topics covered in the book

- The Porter stemmer (‘unfriendly’→’friend’)
- Word and Sentence Tokenization (think of “said, ‘what’re you? Crazy?’ ’’ said Sadowsky. ‘‘I can’t afford to do that.’’
- Detecting and correcting spelling errors
- Minimum Edit Distance between strings (Dynamic Programming in brief)
- Some observations on human processing of morphology