
Advanced Natural Language Processing

Lecture 3

Morphology and Finite State Machines

Philipp Koehn

28 September 2010



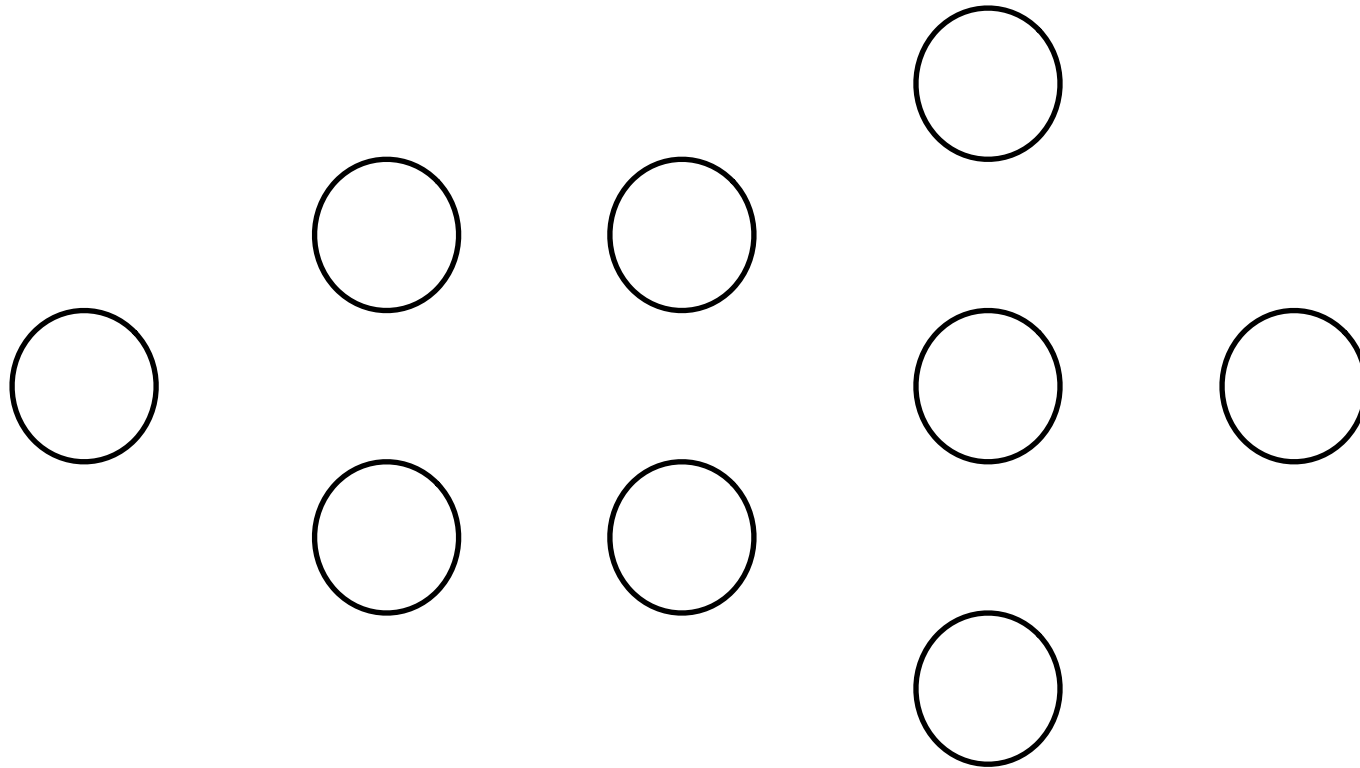
Tasks

- Generation
 - given: lemma and morphological properties
 - wanted: surface form
- Recognition
 - given: surface form
 - wanted: yes/no decision if it is in the language
- Analysis
 - given: surface form
 - wanted: lemma and morphological properties

Word Lists

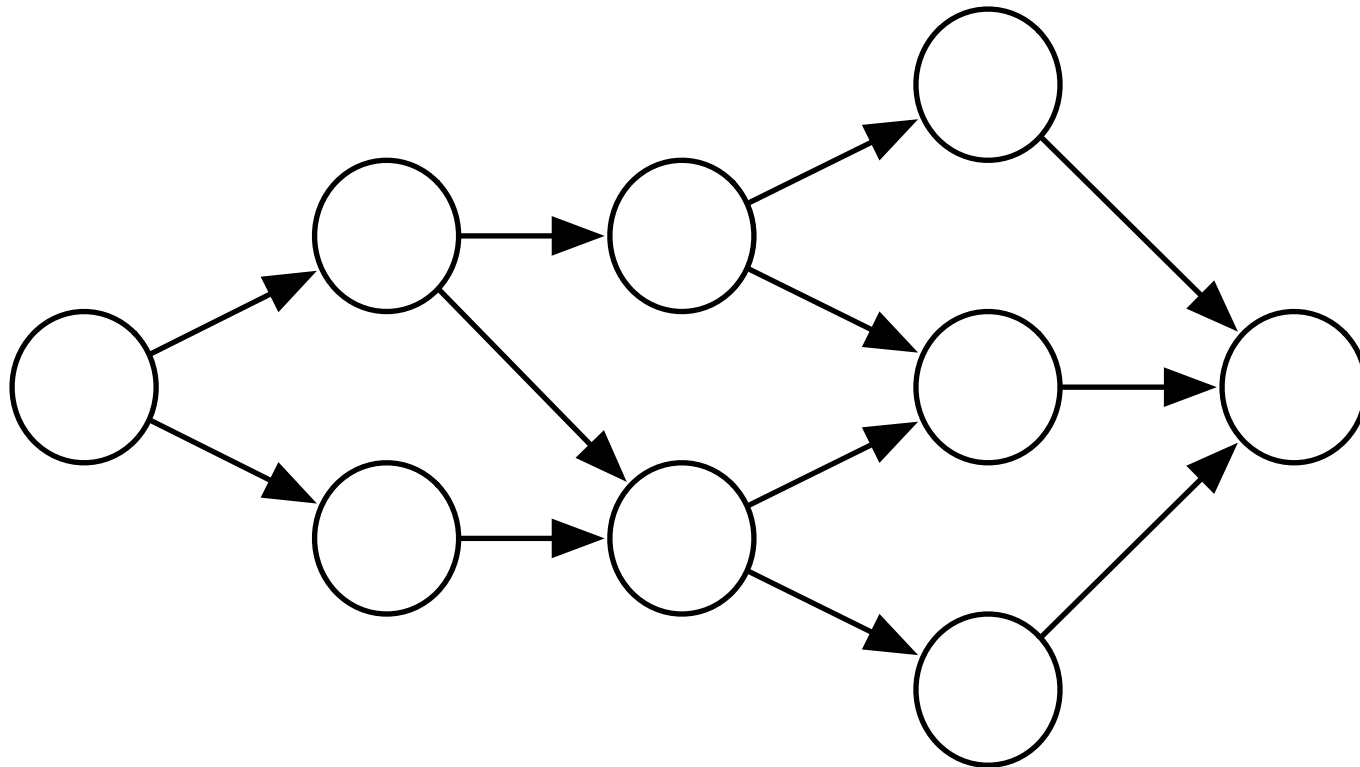
- Simple Solution
 - create a list of all surface forms and their morphological properties
 - solve tasks by checking against list
- But...
 - list can become very long
 - list fails to generalize for productive morphology
- Instead: use finite state machines
(also called finite state automatons)

Finite State Machines: States



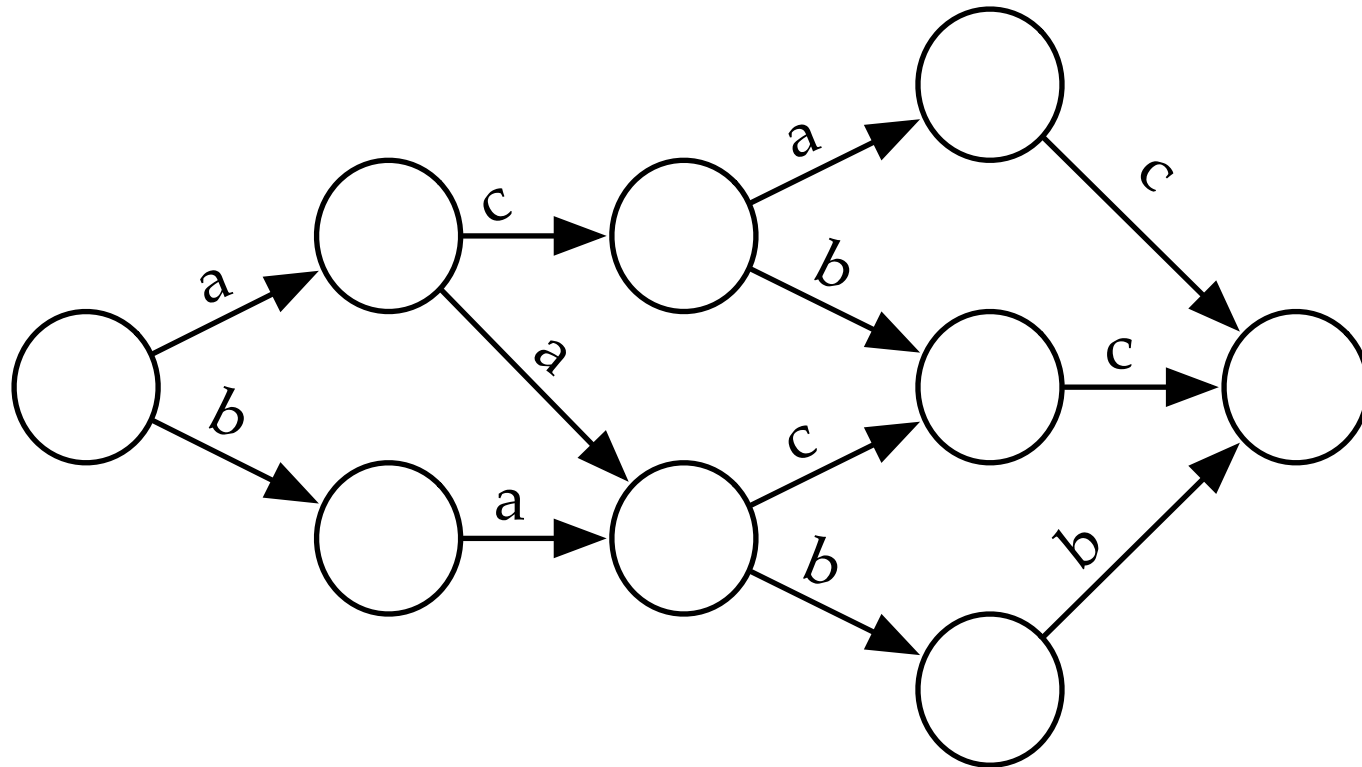
places we may find ourselves in

Finite State Machines: Transitions



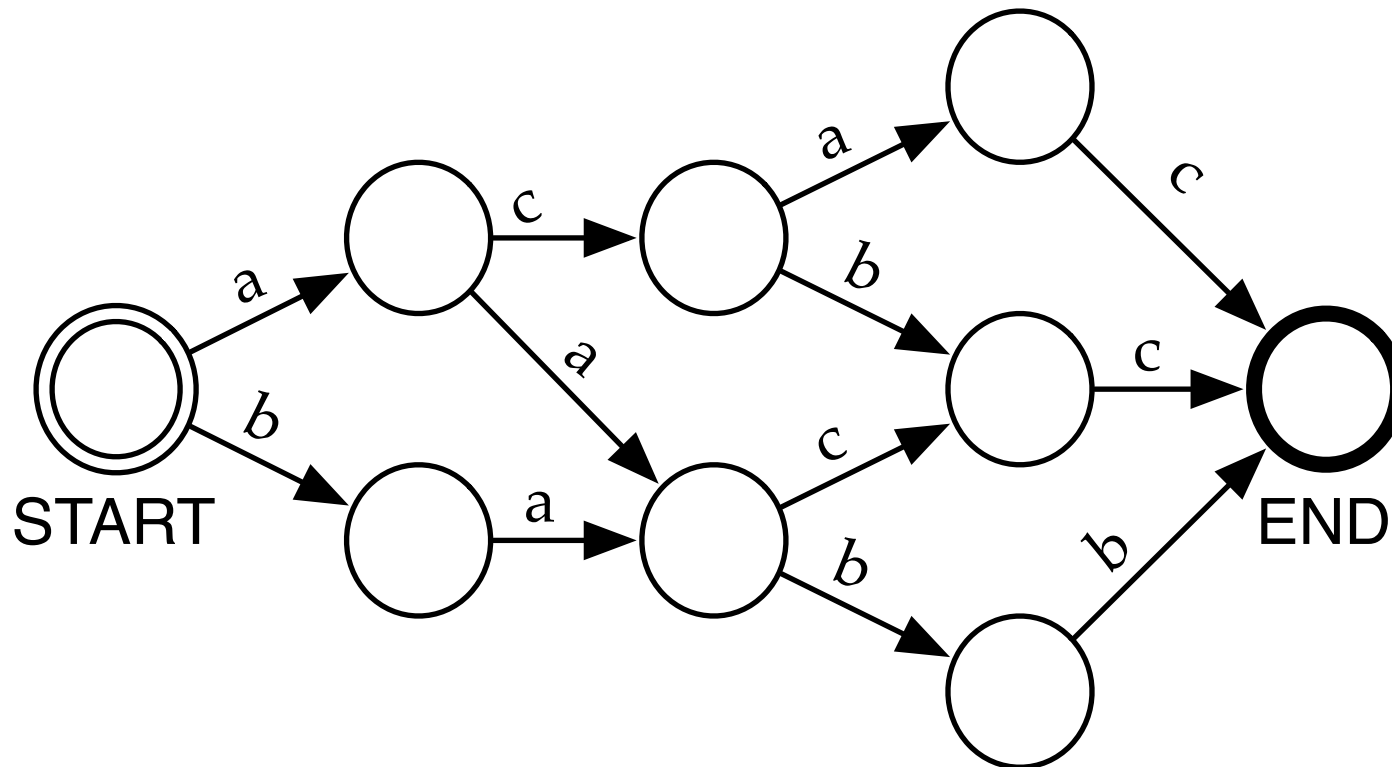
moving between the states

Finite State Machines: Emissions



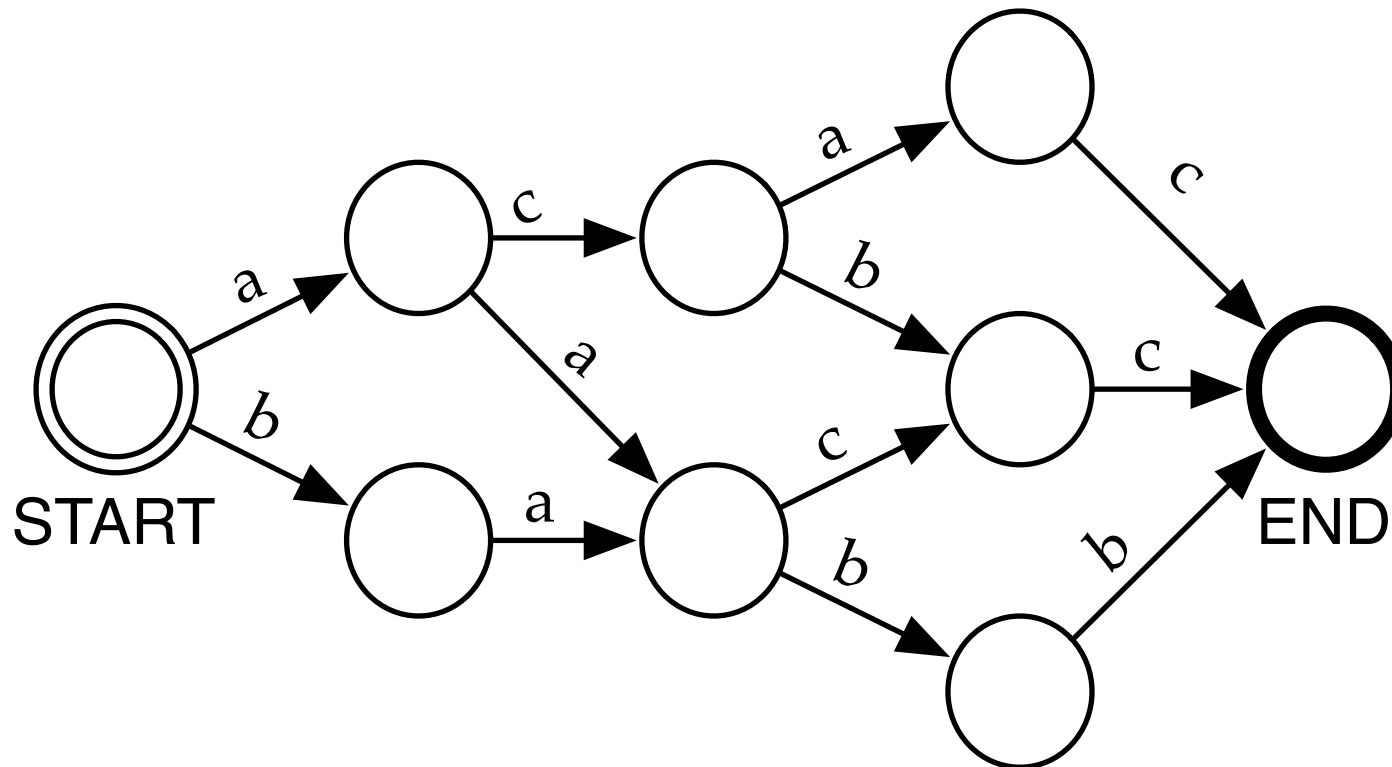
emissions: letters produced at each transition

Finite State Machines: Start and End



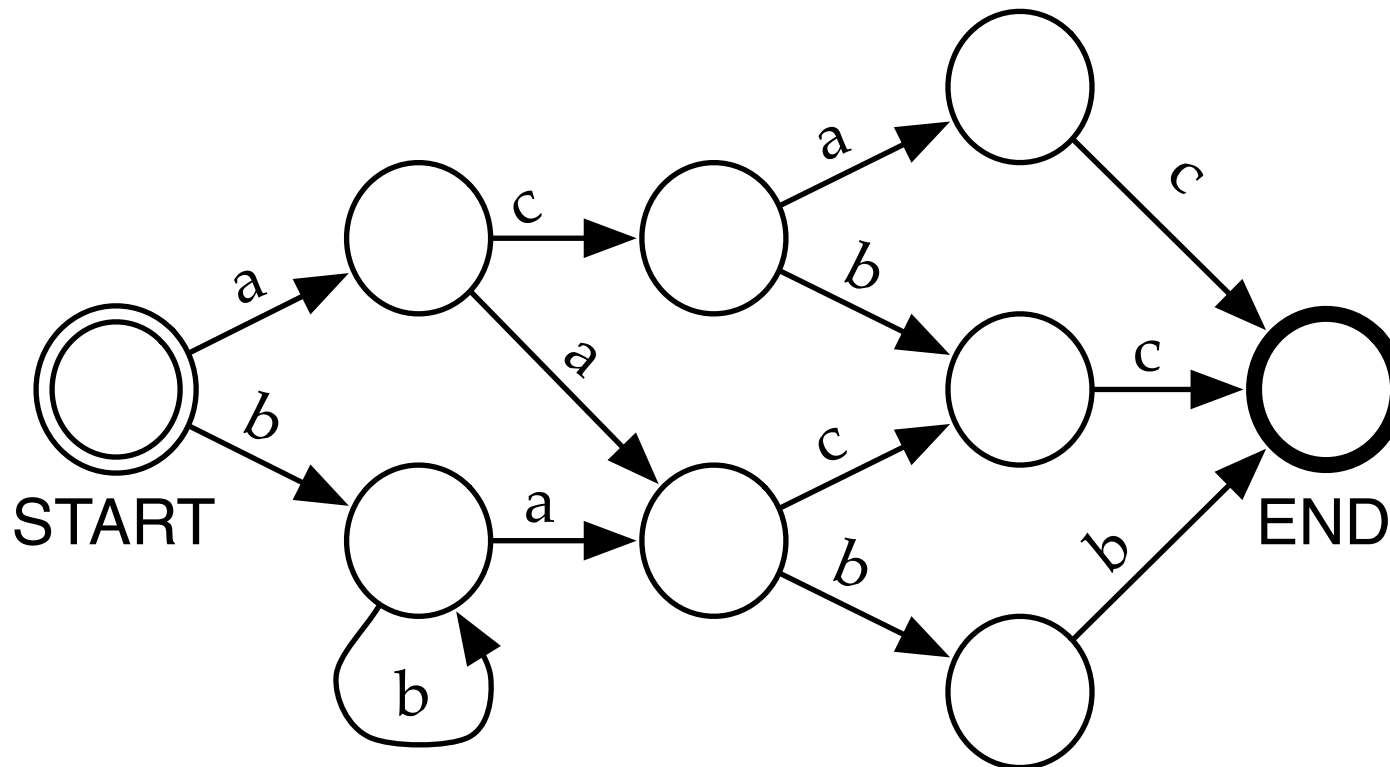
begin at start state, finish at end state

Finite State Machines: Start and End



generated language: { $acac$, $acbc$, $aacc$, $aabb$, $bacc$, $babb$ }

Finite State Machines: Loops



loop \rightarrow infinite language: $\{ \text{babb}, \text{bbabb}, \text{bbbabb}, \text{bbbbabb}, \text{bbbbbabb}, \dots \}$

Regular Languages

- Finite state machines produce regular languages
- Easy to deal with:
determining if a word can be generated by the language straight-forward
- Not all languages can be generated by a finite state machine
example: $a^n b^n = \{ ab, aabb, aaabbb, aaaabbbb, \dots \}$
(would require an infinite number of states)

Regular Expressions

- Common feature of programming languages
- Examples
 - `ls *.jpg`
 - `if ($word = /^[A-Z].*/) { $name = 1; }`
 - `if ($name = /^[WB]ill/) { print "Will or Bill" }`
- Regular expressions have same power as finite state machines
 - define regular languages
 - can be encoded as finite state machines

Chomsky Hierarchy

- Four major types of formal languages
 3. **regular** (generated by finite state machines)
 2. **context-free** (will be covered in later lectures on syntax)
 1. **context-sensitive**
 0. **recursive enumerable** (anything a computer program can produce)
- Languages higher up in the Chomsky hierarchy increasingly complex
 - can describe more languages
 - harder to computefor instance: for type-0 it is not generally possible to determine if a specified word can be generated by the language

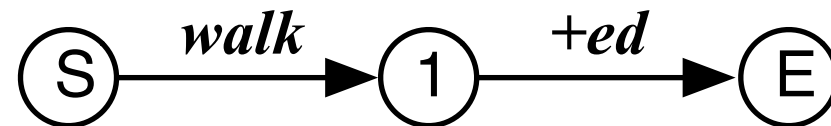
One Word



Basic finite state machine:

- start state
- transition that emits the word *walk*
- end state

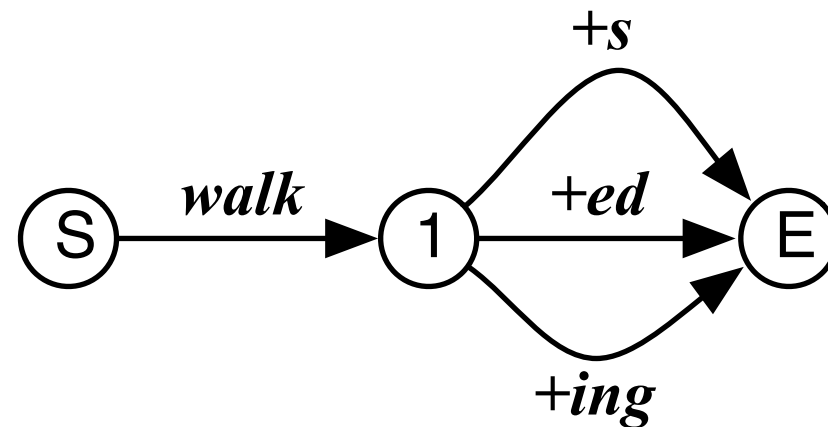
One Word and One Inflection



Two transitions and intermediate state

- first transition emits *walk*
 - second transition emits *+ed*
- *walked*

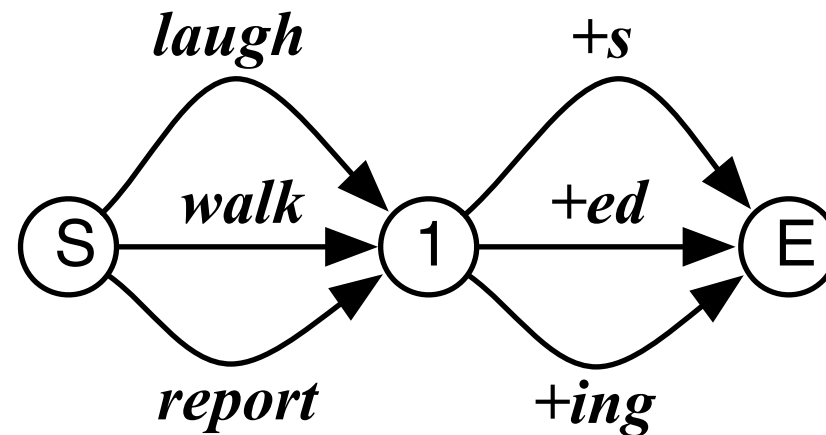
One Word and Multiple Inflections



Multiple transitions between states

- three different paths
- walks, walked, walking

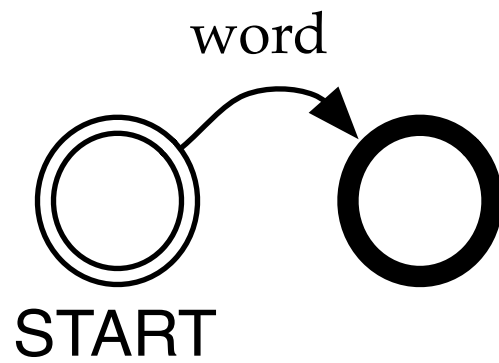
Multiple Words and Multiple Inflections



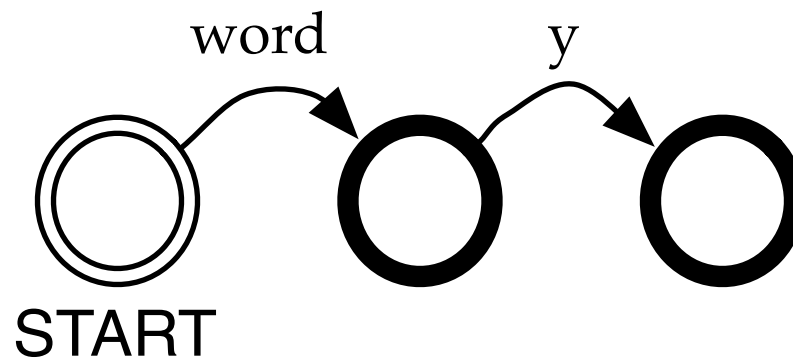
Multiple stems

- implements regular verb morphology
- laughs, laughed, laughing
walks, walked, walking
reports, reported, reporting

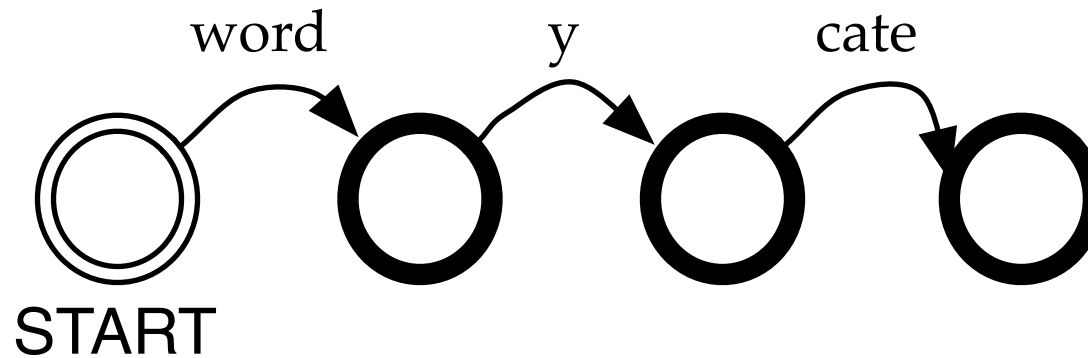
Derivational Morphology



Derivational Morphology

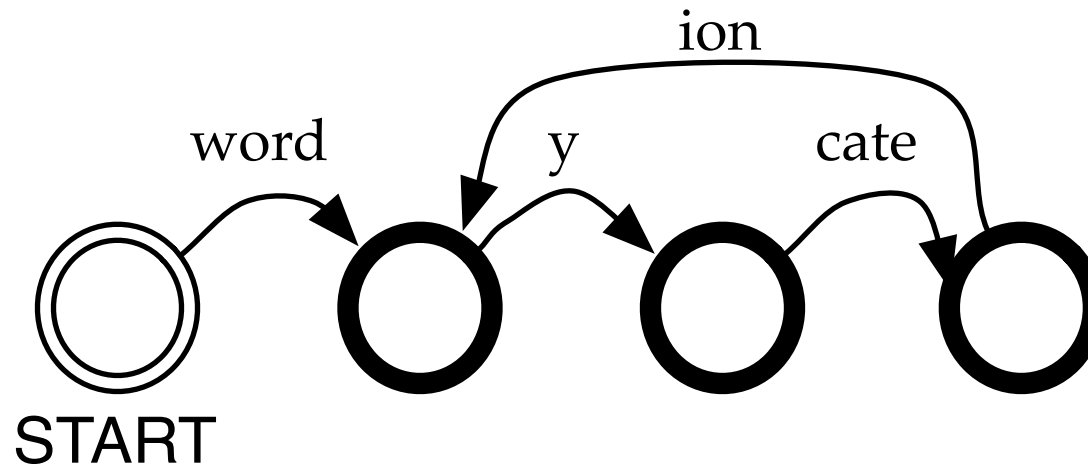


Derivational Morphology

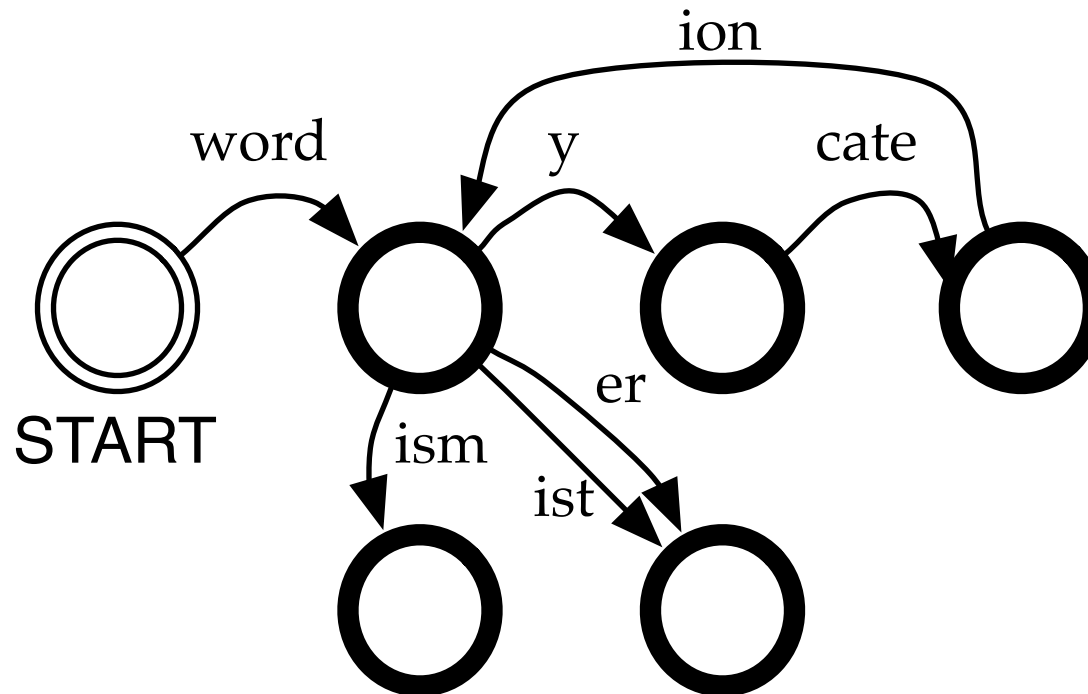


but: *wordicate* not *wordycate*! we'll come back to that later...

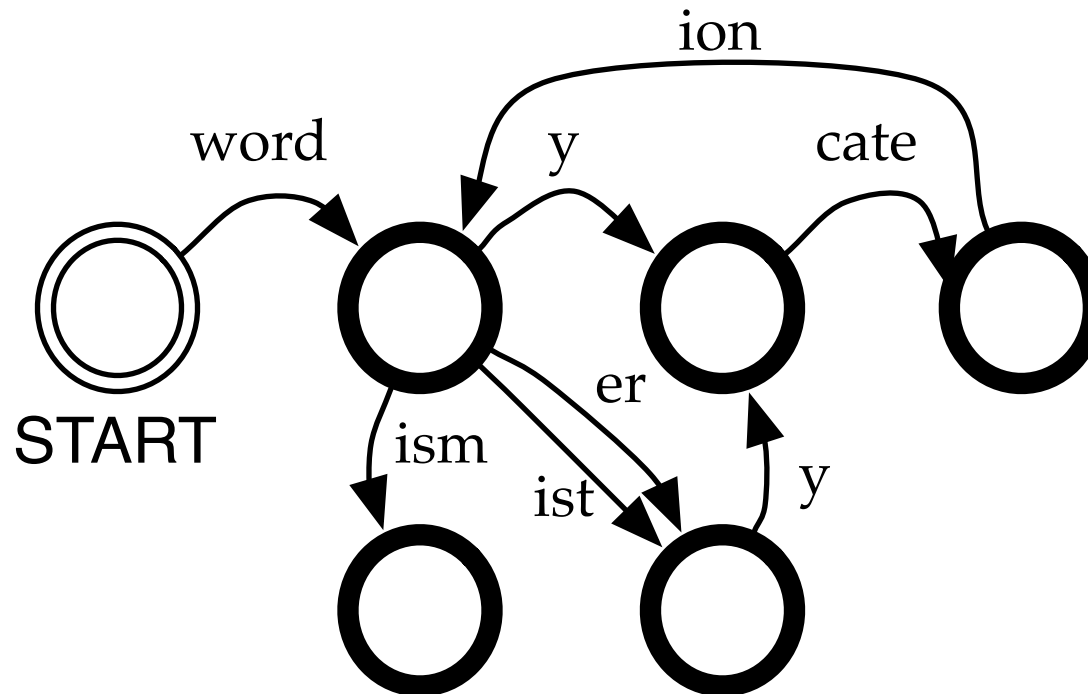
Derivational Morphology



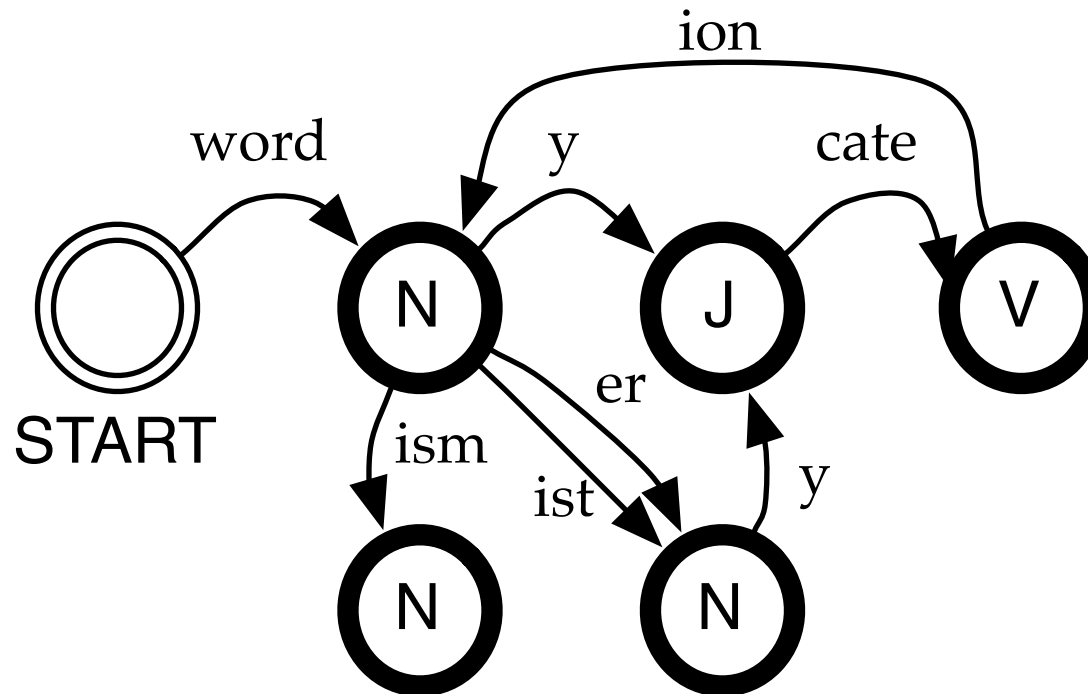
Derivational Morphology



Derivational Morphology



Marking Part of Speech



Concatenation

- Constructing an FSM gets very complicated
- Build components
 - **L**: lexicon
 - **D**: derivational morphology
 - **I**: Inflectional morphology
- Concatenate **L + D + I**

What Data is Required?

- Lexicon of lemmas
 - very large
 - needs to be collected by hand
 - ... or automatically learned from corpora?
- Inflection and derivation rules
 - not large
 - but requires understanding of the language
 - ... can be learned automatically?

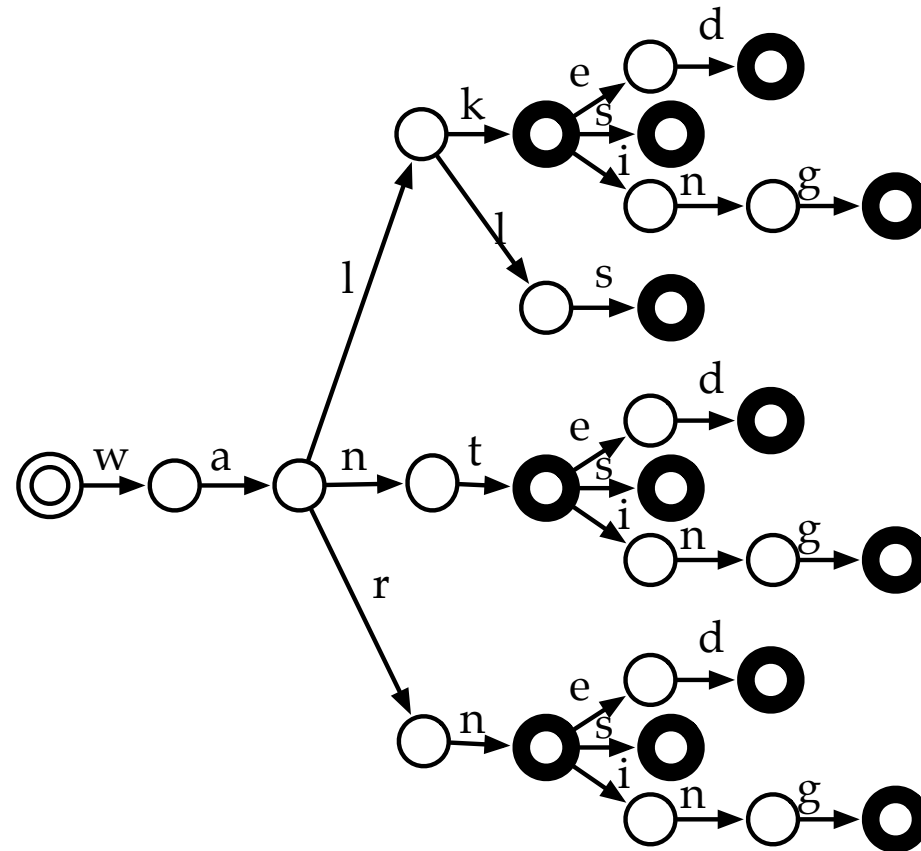
Automatic Learning of Morphology

- Step 1: Collect words in a large corpus

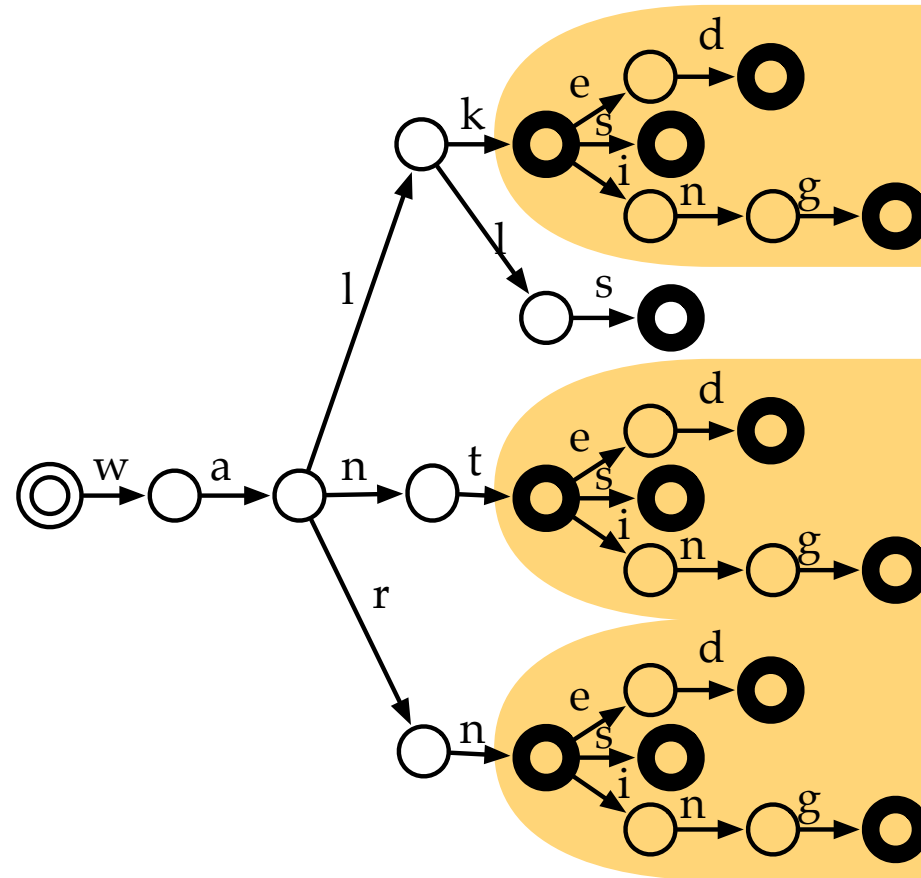
→ Word list

... walk walked walking walks wall walls want wanted wanting wants
warn warned warning warns ...

Compile into a Trie



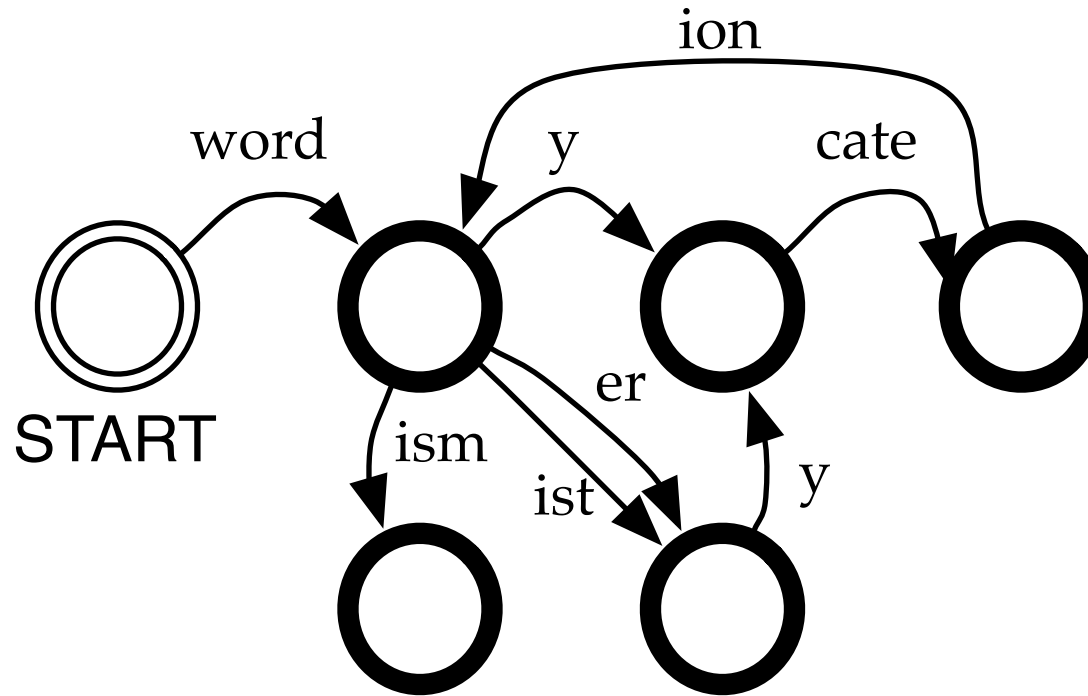
Identify Frequent Suffix Trees



Discovered Morphology

- Stems — with common suffix tree
 - walk
 - want
 - wish
- Morphemes — frequent suffix tree
 - ϵ
 - ed
 - s
 - ing

One Last Problem



wordy_{cate}, wordy_{cate}ion, ...

Two Level Morphology

- Morphological Recognizer
(as described here)
- Morphographemic Transducer
 - rewrite rules, such as $y:i$ or $e:\epsilon$
 - apply rules in certain contexts
 - implemented as a finite state transducer
(finite state machine that consumes and emits letters)
- More information: Oflazer (2009): Computational Morphology
http://fsmnlp2009.fastar.org/Program_files/Oflazer_slides.pdf

Tools Available

- AT&T FSM Library and Lextools

<http://www2.research.att.com/~fsmtools/fsm/>

- OpenFST (Google and NYU)

<http://www.openfst.org/>

- Carmel Toolkit

<http://www.isi.edu/licensed-sw/carmel/>

- FSA Toolkit

<http://www-i6.informatik.rwth-aachen.de/~kanthak/fsa.html>