

# Unit 2

## Language syntax and Semantics

# Contents

- **Morphological Analysis: What is Morphology?**
- **Types of Morphemes, Inflectional morphology & Derivational morphology,**
- Morphological parsing with Finite State Transducers (FST)
- Syntactic Analysis: Syntactic Representations of Natural Language,
- Parsing Algorithms,
- Probabilistic context-free grammars, and
- Statistical parsing Semantic Analysis: Lexical Semantic, Relations among lexemes & their senses  
Homonymy, Polysemy, Synonymy, Hyponymy, WordNet, Word Sense Disambiguation (WSD),
- Dictionary based approach, Latent Semantic Analysis

# Morphological Analysis: What is Morphology

- **Morphology** in the context of **Natural Language Processing (NLP)** refers to the **study of the structure and formation of words**.
- It focuses on understanding how words are constructed from **smaller units of meaning** called **morphemes**.
- **Key Terms:**
  - 1. Morpheme:** The smallest unit of meaning in a language.
    - 1. Free morphemes:** Can **stand alone as words** (e.g., *book*, *run*).
    - 2. Bound morphemes:** Cannot stand alone and need to be attached to other morphemes (e.g., *-ing*, *un-*).
  - 2. Morphological Analysis:** The **process of breaking down words into their morphemes** to understand their meaning, structure, and grammatical role.

# Types of Morphemes

## Morpheme

Free

Bound

Lexical

Functional

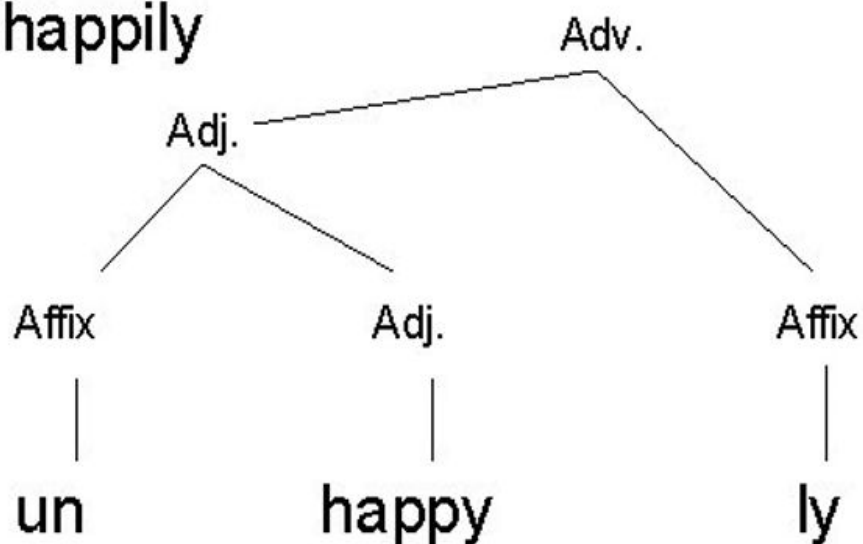
Derivational

Inflectional

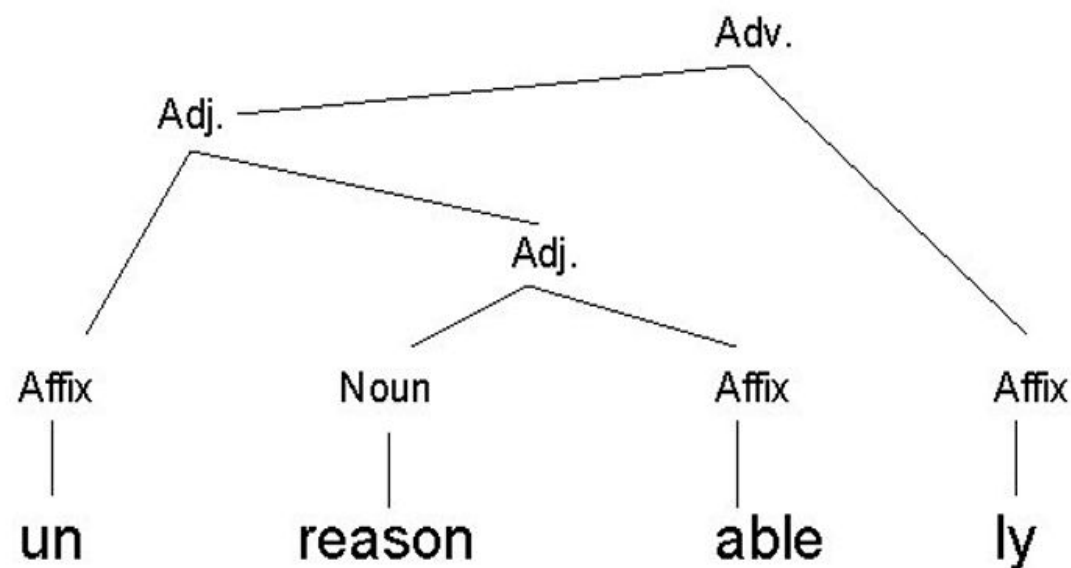
## SURVEY OF ENGLISH MORPHOLOGY

- It is often useful to distinguish two broad classes of **morphemes**: *stems* and *affixes*.
- The *stem* is the ‘**main**’ morpheme of the word, supplying the main meaning,
- The *affixes* add ‘**additional**’ meanings of various kinds.

### Unhappily



### Unreasonably



## SURVEY OF ENGLISH MORPHOLOGY

- *Affixes* are further divided into *prefixes*, *suffixes*, *infixes*, and *circumfixes*.
- *Prefixes* precede the stem(root).

Prefix	Meaning	Example
de-	undo	derail
ex-	non, out	ex-president, extend
in-	negate	incapable
anti-	negate	anti-social
pre-	before	predate
sub-	under, below	subway
un-	negate	undo
dis-	negate	disengage
mis-	wrongly	mistreat
non-	negate	nonsense
pro-	for	proclaim
re-	again, repeat	reread
trans-	across	transatlantic
bi-	two, twice	bilingual
co-	along with	co-author

➤ *Suffixes* follow the stem(root)

Suffix	Meaning	Example
-s	plural	horses
-s	third person singular verbal inflection	likes
-‘s	possession	Mary’s
-ed	past tense	walked
-en	past participle	eaten
-ing	progressive verbal inflection	reading
-er	comparative	brighter
-est	superlative	brightest



➤ **Circumfixes** do both *precede the stem* as well as *follow the stem*.

Root	Word class	Circumfixation	Gloss	Word class
Legal	Adjective	Il-legal-ity	Illegality	Noun
Liberal	Adjective	Il-liberal-ity	Illiberality	Noun

Roots	Word class	Circumfixation	Gloss	Word class
Imagine	verb	un-imagin-able	unimaginable	adjective
Accept	verb	<b>un-accept-able</b>	unacceptable	adjective
Question	verb/noun	<b>un-question-able</b>	unquestionable	adjective

Stem	Word class	Circumfixation	Gloss	Word class
Advisable	adjective	in-advisab-ly	inadvisably	adverb
Correct	adjective	in-correct-ly	incorrectly	adverb



➤ ***Infixes*** are inserted inside the stem.

- Infixes are relatively rare in English, but you can find them in the plural forms of some words.
- For example, **cupful**, **spoonful** and **passerby** can be pluralized as ***cupsful***, ***spoonsful***, and ***passersby***, using "s" as an infix.
- Another example is the insertion of an (often offensive) ***intensifier*** into a word, as in ***"fan-freakin'-tastic."*** Such whole-word insertions are sometimes called infixes, though this phenomenon is more traditionally known as tmesis.

## Inflection MORPHOLOGY

- English has a relatively simple inflectional system; only *nouns*, *verbs*, and sometimes *adjectives* can be *inflected*, and the number of possible inflectional affixes is quite small.
- English *nouns* have only two kinds of inflection: an affix that marks **plural** and an affix that marks **possessive**.

	Regular Nouns		Irregular Nouns	
Singular	cat	thrush	mouse	ox
Plural	cats	thrushes	mice	oxen

## Inflection MORPHOLOGY

- English verbal inflection is more complicated than nominal inflection.

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

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 participle	eaten	caught	cut

## Derivational MORPHOLOGY

- A very common kind of derivation in English is the formation of new nouns, often from verbs or adjectives. Process is called **NOMINALIZATION**.
- For example,
  - the suffix **-ation** produces nouns from verbs ending often in the
  - suffix **-ize** (computerize !computerization).

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



## Derivational MORPHOLOGY

- Adjectives can also be derived from nouns and verbs.

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

# ***SURVEY OF Hindi MORPHOLOGY***

## ➤ Prefixes In Hindi

### उपसर्ग

### **Prefix in Hindi**

Prefix	Meaning	Example
अति	अधिक/ज्यादा	अत्यधिक
अधि	ऊपर/श्रेष्ठ	अधिकार
अनु	पीछे	अनुचर
अप	बुरा/अभाव	अपयश
अभि	सामने/पास	अभियान
अव	हीन/नीचा	अवगुण
आ	तक/सहित	आजीवन

## ➤ Suffixes In Hindi

### प्रत्यय

मूल शब्द	नए शब्द	शब्द रचना
पूजा	पुजारी	(पूजा + आरी)
सोना	सुनार	(सोना + आर)
मानव	मानवता	(मानव + ता)
पुष्प	पुष्पित	(पुष्प + इत)
कलम	कलमदान	(कलम + दान)

## ➤ Circumfixes In Hindi

1. असभ्यता= अ+सभ्य+ता
2. असमर्थता= अ+समर्थ+ता
3. स्वतंत्रता= स्व+तंत्र+ता
4. परतंत्रता= पर+तंत्र+ता
5. बदनसीबी= बद+नसीब+ई



## -ई

- **Noun ---> Feminine Noun (Vocation)**  
नौकर (servant) ---> नौकरी (job, service)
- **Verb Stem ---> Abstract Feminine Noun**  
बोल(ना) (to speak)---> बोली (speech, language)
- **Masculine Noun ---> Diminutive Feminine Noun**  
डंडा (stick) ---> डंडी (small stick)
- **Adjective ---> Noun**  
अच्छा (good) ---> अच्छाई (goodness)
- **Noun ---> Noun / Adjective**  
हिन्दुस्तान ---> हिन्दुस्तानी

## -अ

- The prefix **अ** means “not”, “un-“, “im-“, “without”, “-less”, etc. It comes from Sanskrit/Hindi.
- **With adjectives:**  
अ + संभव (possible) = असंभव (impossible)  
अ + सफल (successful) =  
असफल (unsuccessful)
- **With nouns:**  
अ + हिंसा (violence) = अहिंसा (nonviolence)

## Relevance in NLP:

- Morphological analysis is crucial for many NLP tasks, such as:

**1. Text Normalization:** Converting words to their base form (e.g., stemming, lemmatization).

*e.g., running → run* (for better text comparison and search).

**2. Part-of-Speech Tagging:** Determining the grammatical role of a word in a sentence.

*e.g., "runs" as a verb vs. "runs" as a noun.*

**3. Machine Translation:** Breaking down words to understand their meaning in context.

**4. Information Retrieval:** Enhancing search results by understanding word variations.

*e.g., searching for "teach" should also find "teaching" and "teacher."*

## **Tools for Morphological Analysis in NLP:**

- 1. SpaCy:** For lemmatization and tokenization.
- 2. NLTK:** For stemming and morpheme-level analysis.
- 3. Morfessor:** Specialized for unsupervised morphological segmentation.

In essence, morphology **helps computers understand the structure and meaning of words, enabling better language processing and understanding.**

## ➤ *Finite-State Morphological Parsing*

In order to build a ***morphological parser***, we'll need at least the following:

- **A lexicon:** The list of stems and affixes, together with basic information about them (whether a stem is a Noun stem or a Verb stem, etc).
- **morphotactics:** the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the rule that the English plural morpheme follows the noun rather than preceding it.
- **orthographic rules:** these spelling rules are used to model the changes that occur in a word, usually when two morphemes combine (for example the **y** ----> **ie** spelling rule discussed above that changes city + -s to cities rather than citys).

# Finite-State Morphological Parsing




In order to build a morphological parser, we'll need at least the following:

1. **Lexicon:** the list of stems and affixes, together with basic information about them (whether a stem is a Noun stem or a Verb stem, etc.).




# Finite-State Morphological Parsing

 **2. Morphotactics:** the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word.

E.g. “-un” can be used as prefix but not as suffix

Eg. The English plural morpheme (-s or -es) follows the noun rather than preceding it is a morphotactic fact.

# Finite-State Morphological Parsing



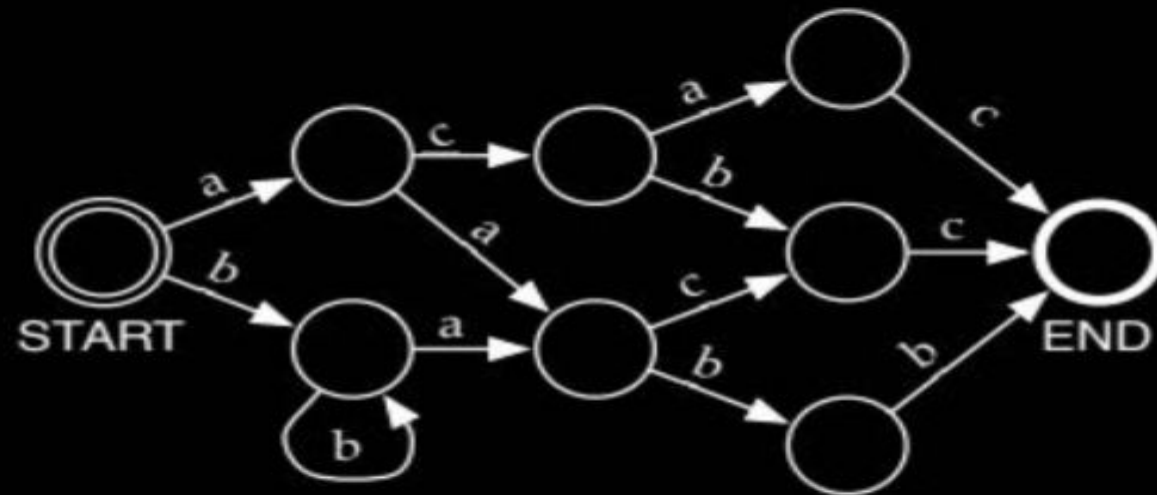
**3. Orthographic rules:** These spelling rules are used to model the changes that occur in a word, usually when two morphemes combine (e.g., the  $y \rightarrow ie$  spelling)

Eg. Cry – cried, Family - Families

Eg. घोडा : घोड्याचे , घोड्याला (आ कारान्त)

Eg. भाऊ : भावाला, भावास , भावास (ऊ कारांत )

# Finite State Automata



- ▶ States: start, end, intermediate
- ▶ Transitions between states
- ▶ Can be viewed as emitting or recognizing strings

# One Word



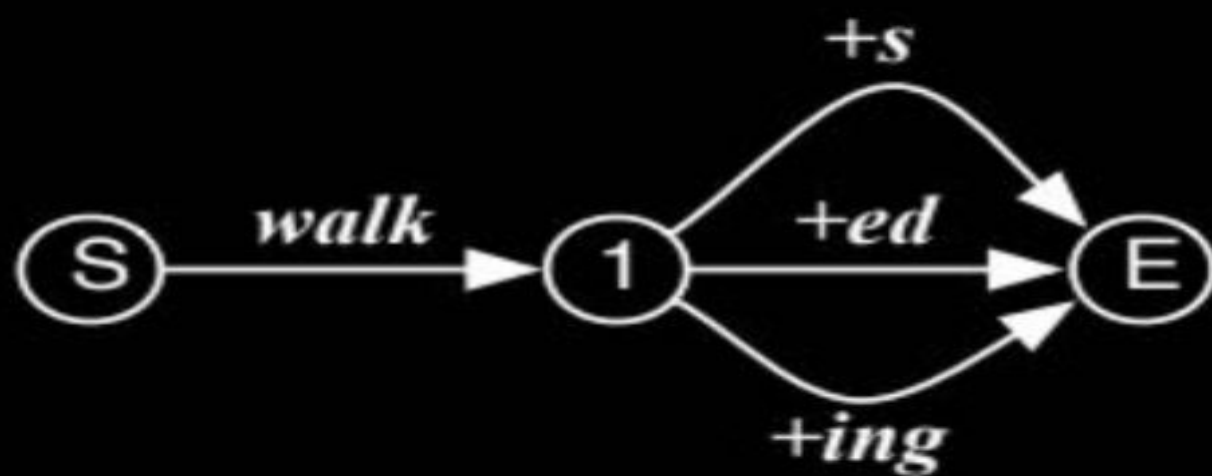
- ▶ Start state
- ▶ End state
- ▶ Transition that emits word/stem walk

## One Word and One Inflection



- ▶ Intermediate state
- ▶ First transition emits stem walk
- ▶ Second transition emits -ed

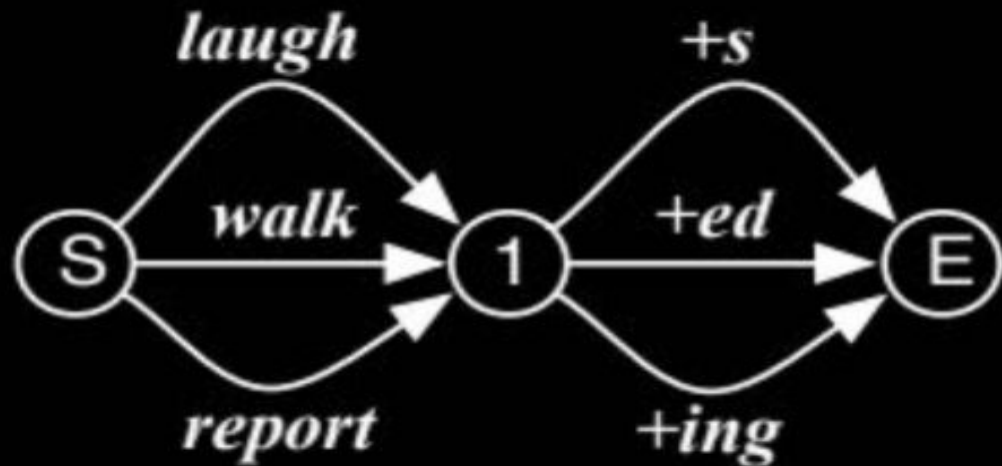
## One Word and Multiple Inflections



- ▶ Multiple affix transitions
- ▶ Three paths: walks, walked, walking



# Multiple Words and Multiple Inflections

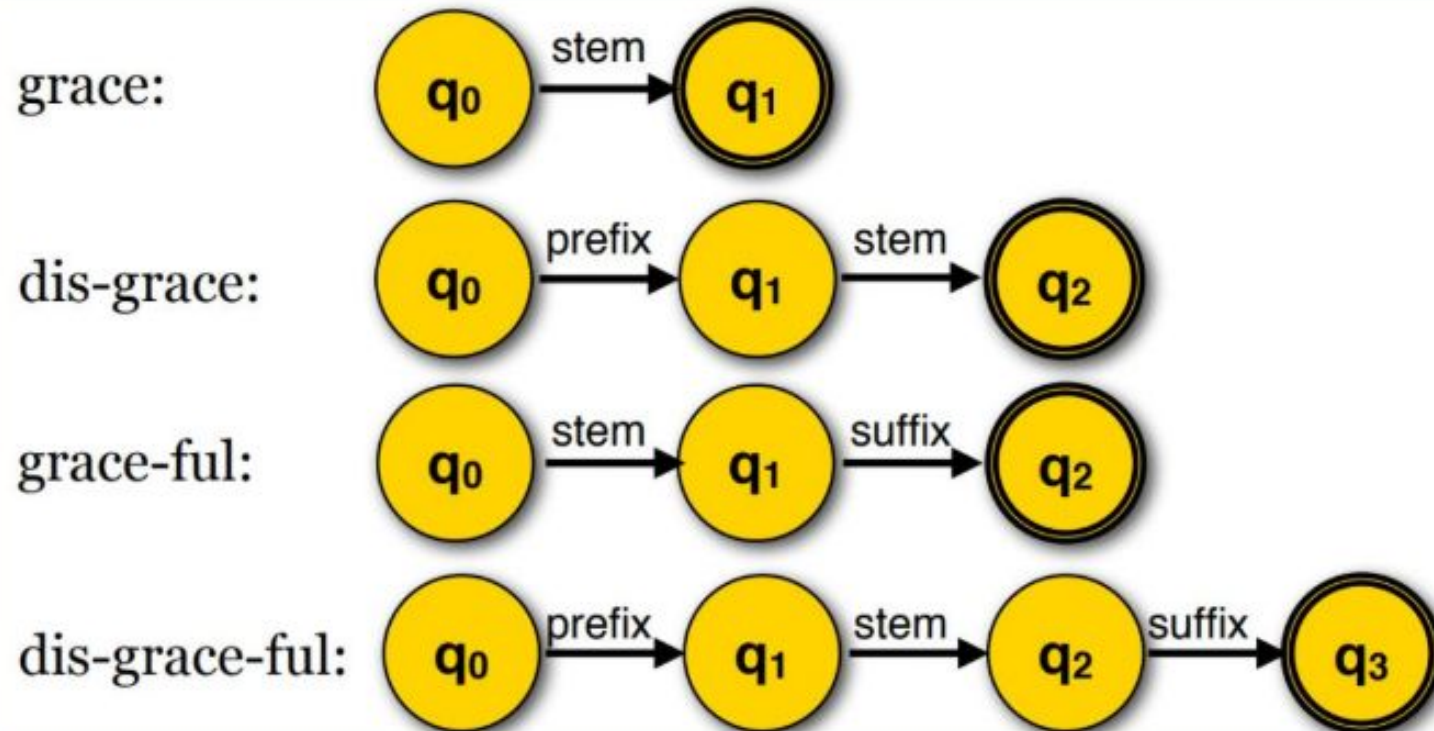


- ▶ Multiple stems
- ▶ Multiple paths: laughs, ..., walked, ..., reporting
- ▶ Implements regular verb morphology

# Finite State Transducers

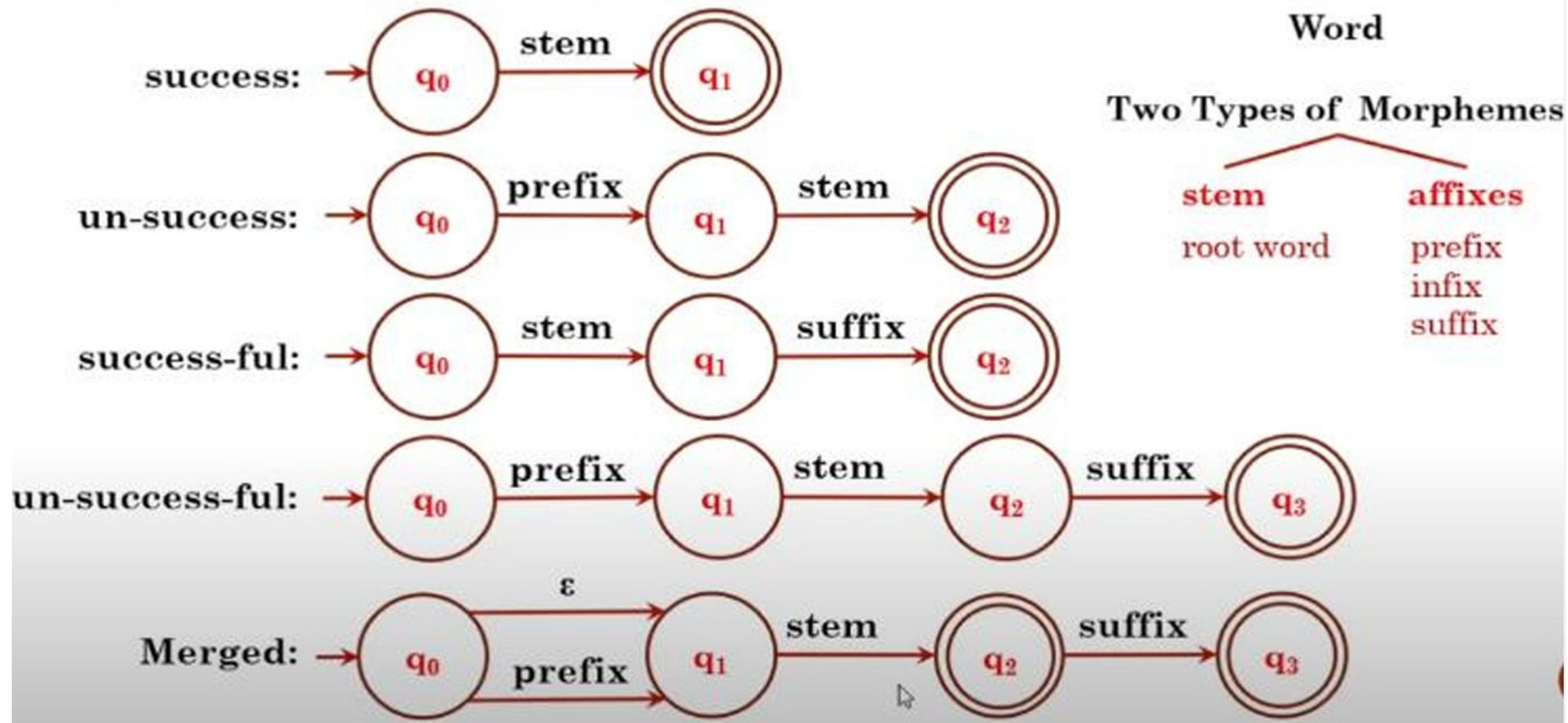


**FSAs can recognize (accept/reject) a string, but they don't tell us its internal structure.**



# FINITE STATE AUTOMATA FOR MORPHOLOGY

Morphology means study of word/making of word

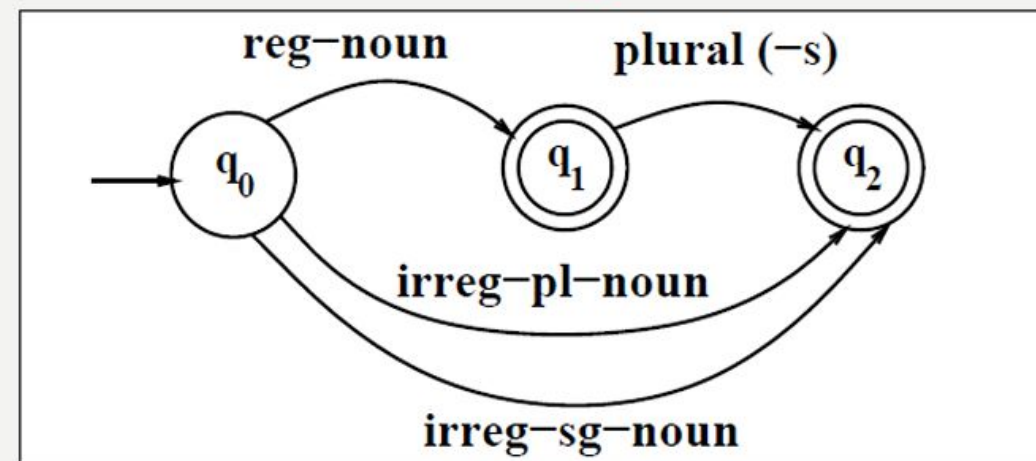




- lexicon includes *regular nouns (reg-noun)* that take the *regular -s plural* (e.g. cat, dog, fox).
- Plural of words like **fox** have an inserted **e**: foxes.
- The lexicon also includes **irregular noun** forms that don't take -s, both singular **irreg-sg-noun** (goose, mouse) and plural **irreg-pl-noun** (geese, mice).

A finite-state automaton for English nominal inflection

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

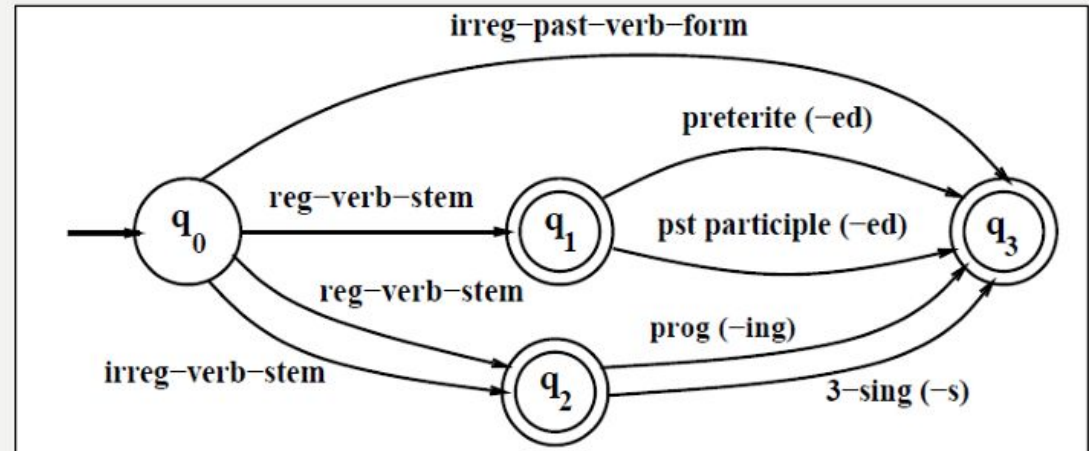


- The automaton **STATE** has **03** states, which are represented by nodes in the graph.
- **State 0** is the **START STATE** which we represent by the incoming arrow.
- **State 2** is the **final state** or accepting state, which we represent by the **double circle**.
- It also has four transitions, which we represent by arcs in the graph.

- This lexicon has three stem classes (**reg-verb-stem**, **irreg-verb-stem**, and **irreg-past-verb-form**),
- 4 more affix classes (**-ed past**, **-ed participle**, **-ing participle**, and **3rd singular -s**)

A finite-state automaton for English verbal inflection

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



- The automaton STATE has 04 states, which are represented by nodes in the graph.
- State 0 is the START STATE which we represent by the incoming arrow.
- State 4 is the final state or accepting state, which we represent by the double circle.
- It also has 08 transitions, which we represent by arcs in the graph.

➤ **Data on English adjectives**

**adj-root1** would include adjectives that can occur with ***un-*** and ***-ly*** (clear, happy, and real) while **adj-root2** will include adjectives that can't (big, cool, and red).

big, bigger, biggest

cool, cooler, coolest, coolly

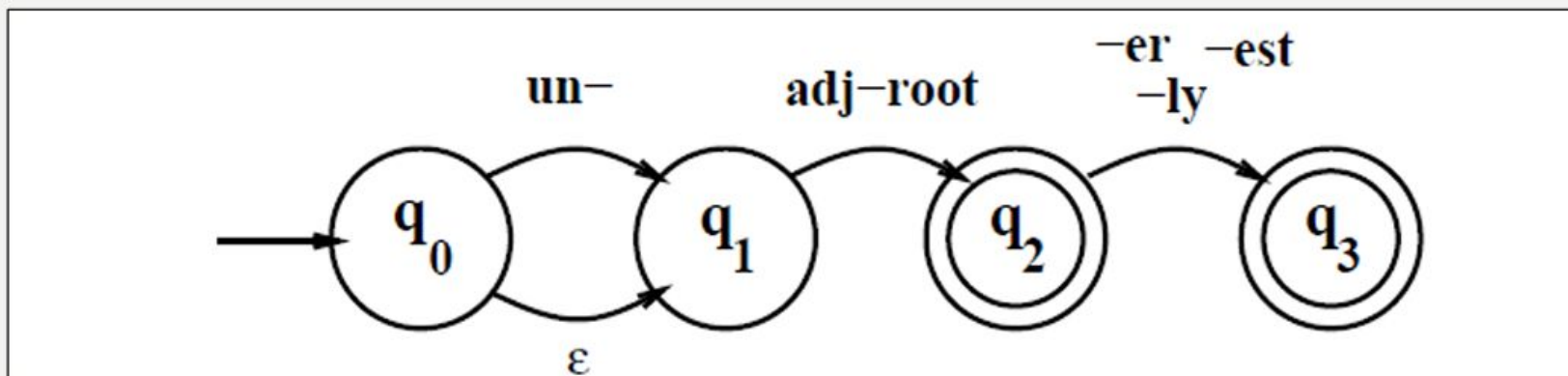
red, redder, reddest

clear, clearer, clearest, clearly, unclear, unclearly

happy, happier, happiest, happily

unhappy, unhappier, unhappiest, unhappily

real, unreal, really



it will also recognize ungrammatical forms like ***unbig, redly, and realest.***



big, bigger, biggest

cool, cooler, coolest, coolly

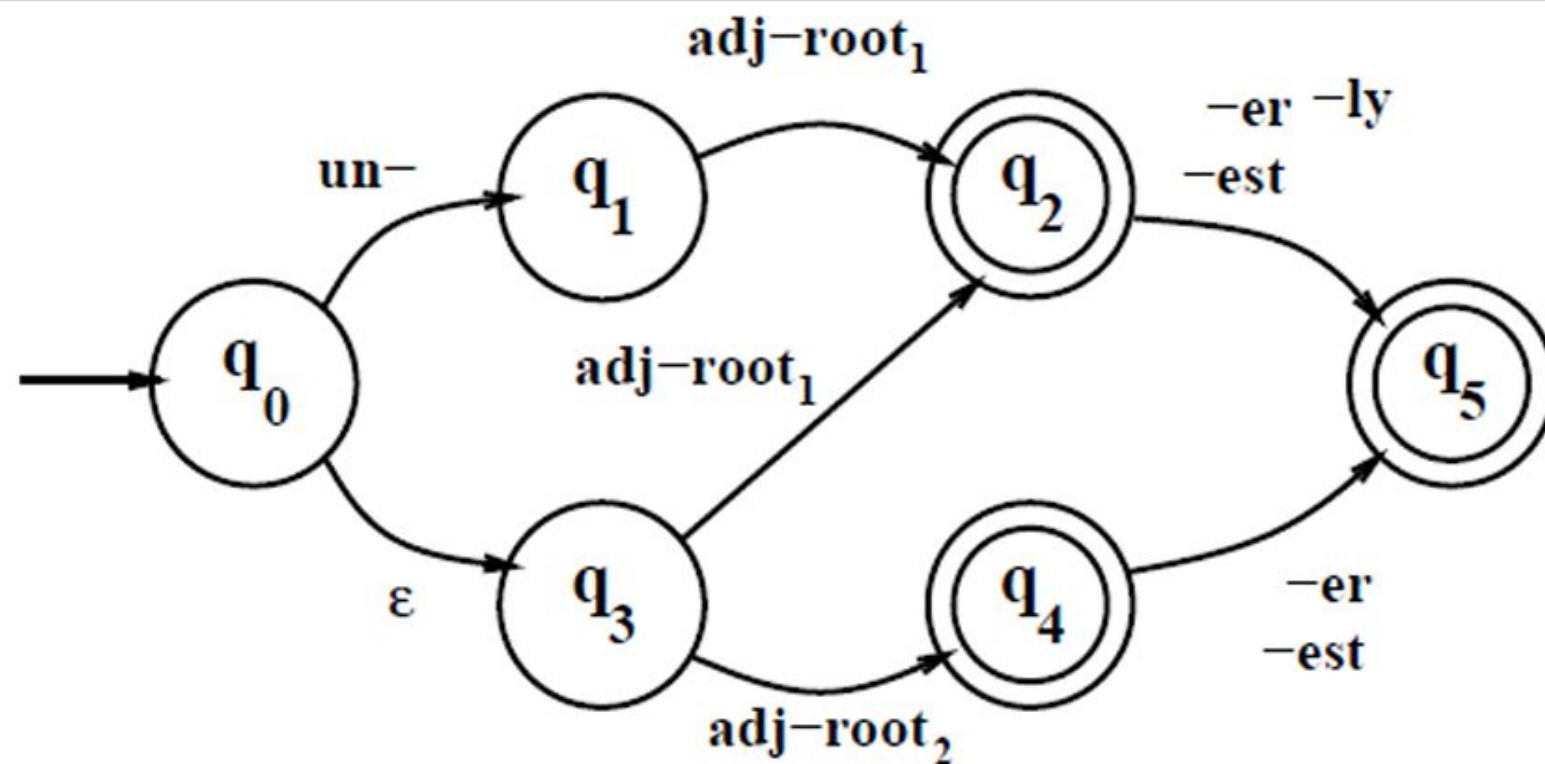
red, redder, reddest

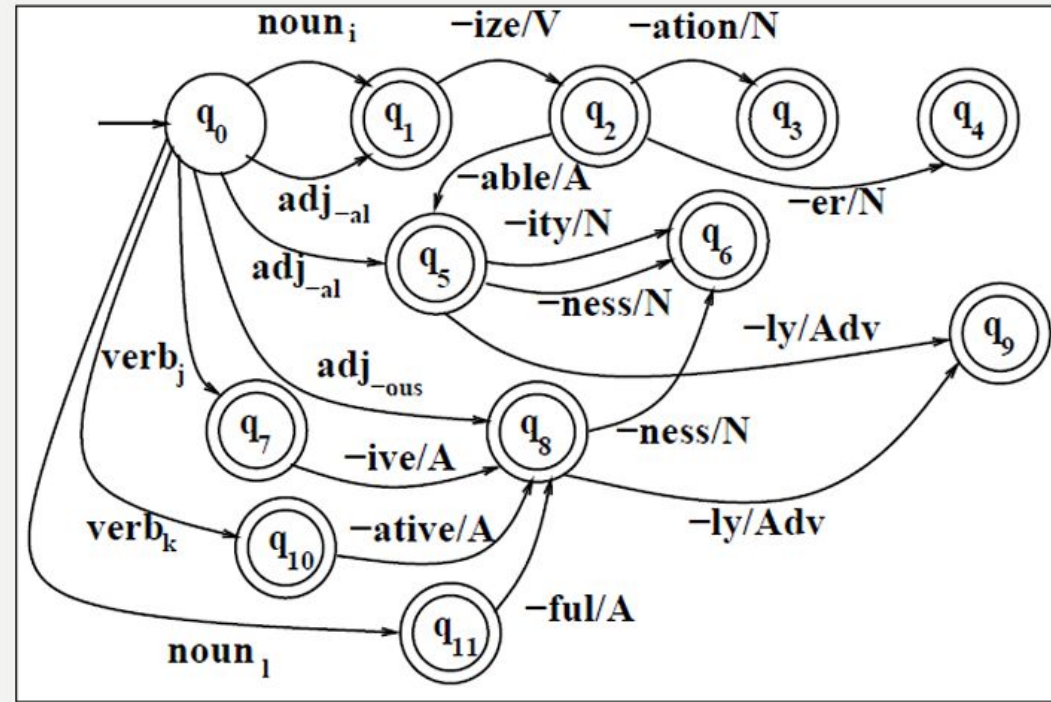
clear, clearer, clearest, clearly, unclear, unclearly

happy, happier, happiest, happily

unhappy, unhappier, unhappiest, unhappily

real, unreal, really





- Visual (Adjective) --> Visualize (verb) --> Visualization (noun)
- Author (Noun) --> Authorize (verb) --> Authorization (noun)
- Skill (Noun) --> Skillful (Adjective) --> Skillfulness (noun)
- --> Skillfully (Adverb)
- Create (verb) --> Creative (Adjective) --> Creativeness (noun)
- Normal(Noun) --> Normalize (Verb) --> Normalizer (noun)

{**q0** --> q1 --> q2--> **q3**}

{**q0** --> q1--> q2--> **q3**}

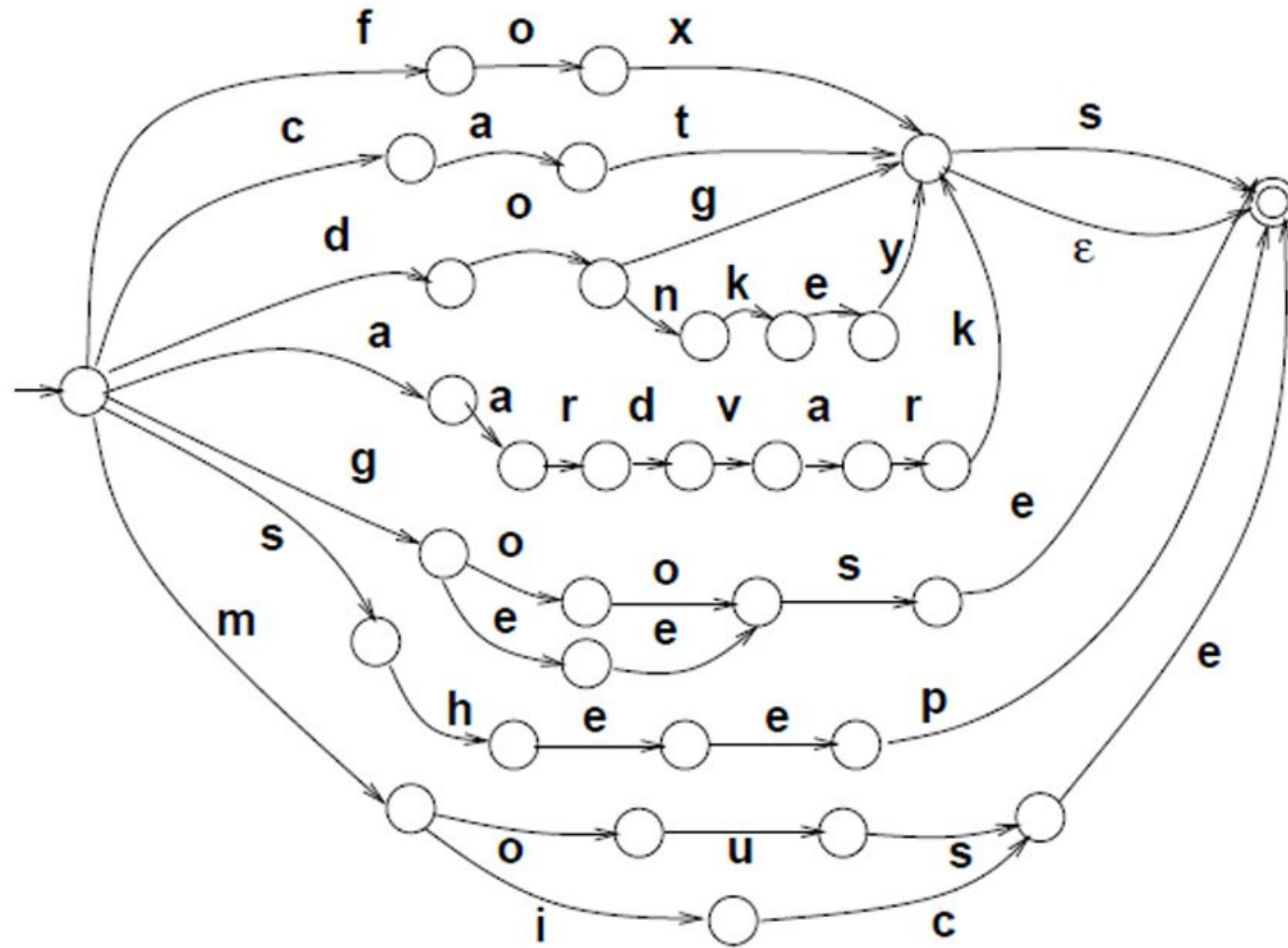
{**q0** --> q11 --> q8 --> **q6**}

{**q0** --> q11 --> q8 --> **q9**}

{**q0** --> q7--> q8 --> **q6**}

{**q0** --> q1 --> q2--> **q4**}

# Finite-State Morphological Parsing



- *fox*
- *geese*
- *goose*
- *cat*
- *sheep*
- *sheep*
- *dog*
- *mice*
- *mouse*
- *aardvark*

Compiled FSA for a few English nouns with their inflection

- **Morphological parsing** is the process of **analyzing the structure of words** to identify **their morphemes** (roots, prefixes, suffixes, etc.) and **their grammatical features**.
- **Finite State Transducers (FSTs)** are **commonly used for this purpose** in NLP because they provide an efficient, formal mechanism for handling morphological rules and transformations.

## What is a Finite State Transducer (FST)?

A Finite State Transducer is an **extension of a Finite State Automaton (FSA)**. While an FSA recognizes sequences of symbols, an FST maps input sequences to output sequences. It is **a directed graph** with:

1. **States:** Represent different stages in processing.
2. **Transitions:** Represent mappings between input and output symbols.
3. **Start and Accept States:** Define where processing begins and ends.

- FSTs **map** surface forms (the word as it appears) to their underlying lexical forms (**root morphemes and grammatical information**).

- **Tools for FST-based Morphological Parsing**

1. **XFST** (Xerox Finite-State Tools): A widely used toolkit for creating FSTs.
2. **HFST** (Helsinki Finite-State Technology): Open-source FST tools for NLP.
3. **Foma**: A fast and efficient tool for constructing FSTs.

Finite State Transducers (FSTs) are powerful tools for modeling linguistic processes, pattern recognition, and more. Key tools include:

**OpenFST:** A widely-used open-source library for constructing, optimizing, and applying FSTs. It supports various algorithms like composition, minimization, and determinization.

**Pynini:** A Python library built on OpenFST, ideal for linguistic applications like text normalization and phonological rule modeling.

**AT&T FSM Library:** A robust toolkit for creating and manipulating finite-state machines and transducers.

**SFST (Stuttgart FST):** Useful for computational linguistics, supporting linguistic rules and lexicons.

**Foma:** A tool for constructing and testing FSTs, often used in morphology and phonology.



Foma is a **free and open-source finite-state toolkit** developed by Måns Huldén.

It **includes a compiler, programming language, and C library** for constructing **finite-state automata and transducers**, commonly used in natural language processing tasks such as morphological analysis.

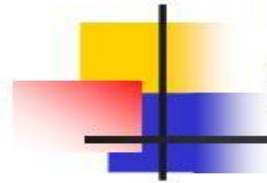
- <https://github.com/mhulden/foma/blob/master/foma/docs/simpleintro.md>
- You can explore Foma's source code, documentation, and examples on its GitHub repository, Link given above:
- Comprehensive introduction to Foma, including installation instructions and usage examples, refer to the [Simple Introduction to Foma](#) in the repository's documentation section.

Helsinki Finite-State Technology (HFST) is an **open-source toolkit for processing natural language morphologies using finite-state automata and transducers.**

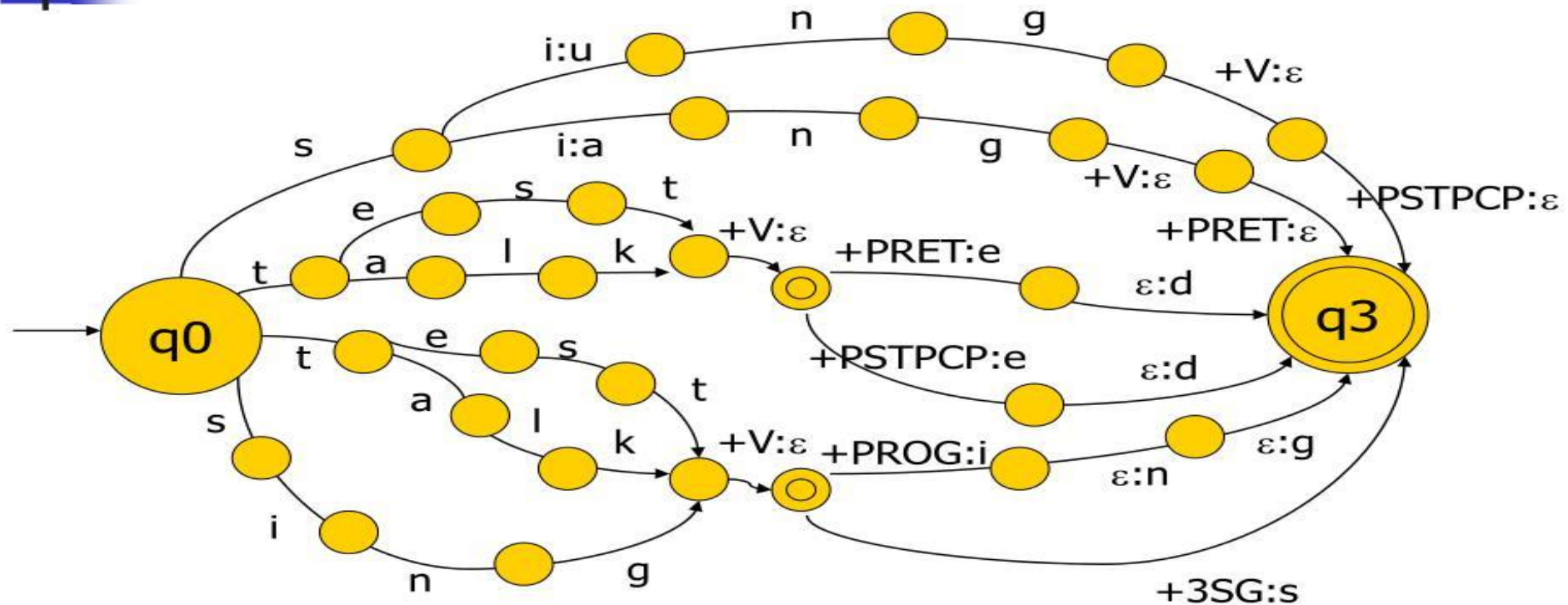
It serves as an **interface to multiple backends**, including **OpenFST, Foma, and SFST**, facilitating the creation and manipulation of linguistic tools such as **spell-checkers and morphological analyzers.**

<https://hfst.github.io/>

These platforms provide comprehensive information and tools to assist you in utilizing HFST for your computational linguistics projects.



## Finite-State Transducer for Inflection of the Verbs 'talk', 'test' and 'sing'



## Components of the FST

- **States (q0, q3):**
  - The yellow circles represent states in the FST.
  - q0 is the starting state.
  - q3 is the final state where the parsing or generation completes.
- **Transitions:**
  - The labeled arcs between states represent input-output mappings.
  - Example:
    - $s \rightarrow s$ : The input symbol s maps directly to output s (surface form remains the same).
    - $i:u$ : Maps the input i to the output u (e.g., vowel change for irregular verbs).
- **Special Notations:**
  - +V:ε: Represents a morphological marker for a verb (+V) that outputs nothing (ε means empty output).
  - +PRET:e: Marks the past tense with the suffix e.
  - +PROG:i: Marks the progressive tense with the suffix i.
  - +PSTPCP:e: Marks the past participle with the suffix e.

## Explanation of Transitions

- **Base Form:**
  - Starting at q0, the FST recognizes the base forms of the verbs talk, test, and sing by mapping their individual letters (e.g., t, a, l, k) to corresponding outputs.
- **Verb Features:**
  - The FST adds grammatical markers for:
    - **+V**: Identifies the word as a verb.
    - **+PRET**: Past tense (e.g., talked, tested, sang).
    - **+PROG**: Progressive tense (e.g., talking, testing, singing).
    - **+PSTPCP**: Past participle (e.g., talked, tested, sung).
- **Regular and Irregular Verbs:**
  - **Regular verbs** (e.g., talk, test):
    - Follow simple suffixation rules for past tense (+PRET:e) and progressive tense (+PROG:i).
  - **Irregular verbs** (e.g., sing):
    - Undergo vowel changes (e.g., i → u for sung) to indicate past participle.

## Examples of Verb Inflection

- **Regular Verb (talk):**

1. Input: talk
2. Transition: Add +PRET:e → talked
3. Transition: Add +PROG:i → talking

- **Irregular Verb (sing):**

1. Input: sing
  2. Transition: Apply vowel change i → u for +PSTPCP:e → sung
- Transition: Add +PROG:i → singing



## **Examples of Verb Inflection**

- **Regular Verb (talk):**

1. Input: talk
2. Transition: Add +PRET:e → talked
3. Transition: Add +PROG:i → talking

- **Irregular Verb (sing):**

1. Input: sing
2. Transition: Apply vowel change i → u for +PSTPCP:e → sung
3. Transition: Add +PROG:i → singing

## Applications of FSTs in NLP

### 1. Spell Checking:

Parse words to validate their morphological correctness.

### 2. Part-of-Speech Tagging:

Use morphological information to determine grammatical roles.

### 3. Machine Translation:

Map surface forms to their lexical forms for better translation accuracy.

### 4. Information Retrieval:

Normalize word forms (e.g., *run*, *running*, *ran* → *run*).

- **Morphological Parsing:** Decomposing surface forms like talked into talk + PRET.
- **Morphological Generation:** Generating surface forms like talked from talk + PRET.

parsing just the productive **nominal plural (-s)** and the **verbal progressive (-ing)**.

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

+N means that the word is a noun;  
+SG means it is singular,  
+PL that it is plural.

# Finite State Transducers



**Some irregular words require stem changes:**

**Past tense verbs: teach-taught, go-went, write-wrote**

**Plural nouns: mouse-mice, foot-feet, wife-wives**

# Finite State Transducers



**FSAs can recognize (accept/reject) a string, but they don't tell us its internal structure.**

**We need is a machine that maps (transduces) the input string into an output string that encodes its structure:**



# Finite State Transducers



**FST = FSA + output generating functionality**

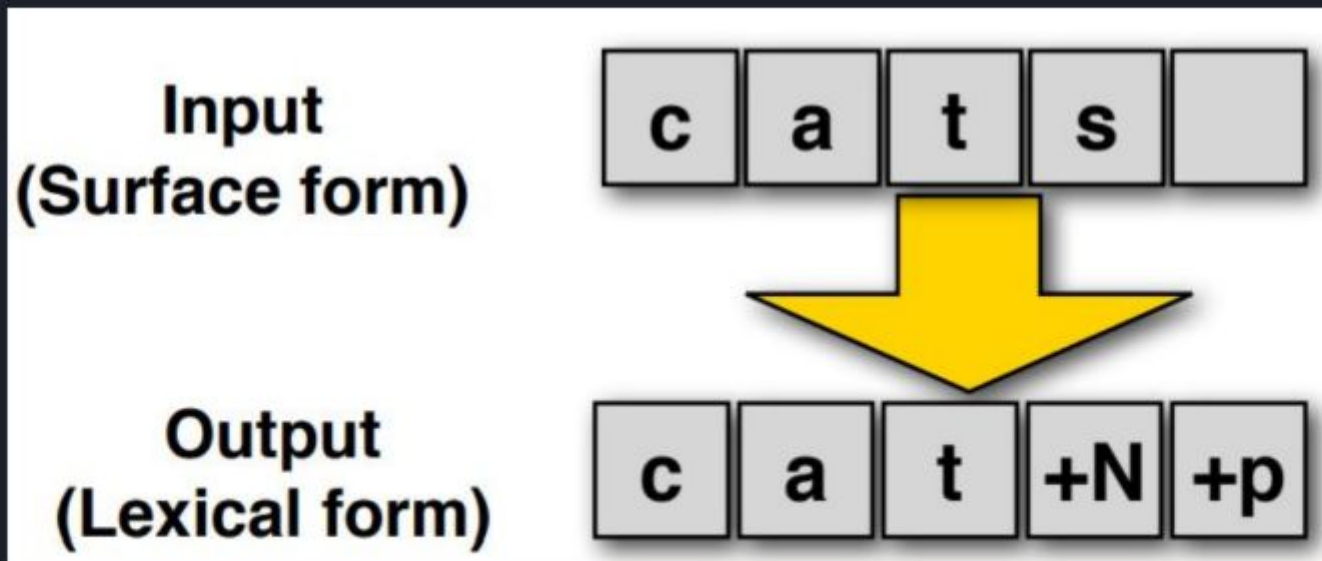
**A morphological analyzer should produce output in two ways....**

**walked => walk+V+past**  
**reporting => report+V+prog**

# Finite State Transducers



A morphological analyzer should produce output:



# Finite State Transducers



**A morphological analyzer should produce output in two ways....**

**2.**

**walk+V+past => walked**

**report+V+prog =>reporting**

# Syntactic Analysis: Syntactic Representations of Natural Language

- Syntactic analysis, or parsing, focuses on determining the grammatical structure of sentences in natural language.
- It identifies **how words combine to form phrases, clauses**, and complete sentences, adhering to the syntactic rules of a given language.
- Key syntactic representations include:

## Constituency-Based Representations

- **Parse Trees:** Hierarchical structures where nodes represent syntactic categories (e.g., NP, VP), and leaves correspond to words.
- **Phrase Structure Grammar:** Frameworks like Context-Free Grammars (CFG) underlie these representations.

## **Dependency-Based Representations**

- **Dependency Trees:** Represent relationships between words using directed edges, emphasizing head-dependent structures.
- **Dependency Parsing:** Efficient for languages with flexible word order.

## **Hybrid Representations**

- Combine constituency and dependency representations for richer syntactic insights.



# Parse Trees

- A **parse tree** (or **syntax tree**) is a tree structure that represents the syntactic structure of a sentence according to a given grammar, typically a **context-free grammar (CFG)**.
- It visualizes how words in a sentence are grouped into phrases and how these phrases relate to one another hierarchically.

## **Components of a Parse Tree**

- 1.Root Node:** Represents the start symbol of the grammar, often denoted as S (Sentence).
- 2.Non-Terminal Nodes:** Represent intermediate syntactic categories like  
3.noun phrases (NP), verb phrases (VP), or prepositional phrases (PP).
- 3.Terminal Nodes:** Represent the actual words (or tokens) in the sentence.