Chapter 2
Words and Morphology

Er. Shiva Ram Dam

Assistant Professor

Gandaki University

Content:

1. Principle of word construction (Suffix, Prefix, Stem, Affixes)
2. Morphemes. Morphology and morphotactic
3. Morphological Process
4. Morphotactic and orthographic rules
5. Morphological representation and FSM
6. Introduction to FSM and FST
7. Morphological parsing and FST
8. FST operations
9. Application of Morphology in NLP

2.1 Principle of Word Construction

Key Components of Word Construction

- **Morpheme** – the smallest unit of meaning in a language.
  - Example:
    - "cats" → **cat** (root morpheme) + **-s** (plural morpheme)
    - "unhappiness" → **un-** (prefix) + **happy** (root) + **-ness** (suffix)
- **Root / Stem** – the base form of a word that carries core meaning.
  - Example: "play" in "playing," "played," "player."
- **Affix** – a morpheme attached to the root to modify meaning.
  - **Prefix:** added before the root → "un–happy"
  - **Suffix:** added after the root → "kind–ness"
  - **Infix** (rare in English): inserted inside a word (common in other languages; e.g., Tagalog *s-um-ulat*)
  - **Circumfix**: prefix + suffix together (common in German: *ge-spiel-t*)

Stem / Root

- The **main part** of a word.
- It carries the **core meaning**.
- Prefixes and suffixes attach to the stem.
- A stem can be:
  - **Free morpheme** (can stand alone):
    play, work, help, teach
  - **Bound morpheme** (cannot stand alone):
    struct (as in *construct, structure*),
    bio (life),
    tele (distance)
- **Examples:**
  - *help* → helper, helpful, helped
  - *struct* → construct, restructure, structure

Prefix

- A prefix is added **before** the stem.
- It changes **meaning**, not usually the word class (verb → verb, adj → adj).
- **Common prefixes:**
  - **un-** = not → *unhappy* (not happy)
  - **re-** = again → *redo* (do again)
  - **pre-** = before → *preview* (view before)
  - **dis-** = opposite / not → *disconnect*
- **More examples:**
  - *mis* + understand → **misunderstand**
  - *inter* + national → **international**

Suffix

- A suffix is added **after** the stem.
- A suffix may change:
  - **Meaning**
  - **Part of speech** (e.g., verb → noun, noun → adjective)
- **Types of suffixes:**
  - **(A) Inflectional suffixes**
    - Do not create new words.
    - Only change **form** (tense, plural, comparison).
    - Examples:
      - -s → *books*
      - -ed → *worked*
      - -ing → *running*
      - -er, -est → *taller, tallest*

- **(B) Derivational suffixes**
  - Create **new words**.
  - Often change **word class**.
  - Examples:
    - -ness: happy → happiness
    - -ment: achieve → achievement
    - -er: teach → teacher
    - -ion: act → action
    - -able: read → readable
    - -ful: beauty → beautiful

Affixes (Prefix + Suffix)

- **Affix** is a general term that includes:
  - **Prefixes** (before the word)
  - **Suffixes** (after the word)
- So, All prefixes and suffixes are affixes.
- Example:
  - **un** + help + **ful**
    = unhelpful
  - **re** + write + **ing**
    = rewriting
- Affixes can:
  - Change meaning
  - Change word class
  - Change grammatical form

Most common prefixes and suffixes in English Language

Prefix	Meaning	Key Word
anti-	against	antifreeze
de-	opposite	defrost
dis-*	not, opposite of	disagree
en-, em-	cause to	encode, embrace
fore-	before	forecast
in-, im-	in	infield
in-, im-, il-, ir-*	not	injustice, impossible
inter-	between	interact
mid-	middle	midway
mis-	wrongly	misfire
non-	not	nonsense
over-	over	overlook
pre-	before	prefix
re-*	again	return
semi-	half	semicircle
sub-	under	submarine
super-	above	superstar
trans-	across	transport
un-*	not	unfriendly
under-	under	undersea

Suffix	Meaning	Key Word
-able, -ible	can be done	comfortable
-al, -ial	having characteristics of	personal
-ed*	past-tense verbs	hopped
-en	made of	wooden
-er	comparative	higher
-er,	one who	worker, actor
-est	comparative	biggest
-ful	full of	careful
-ic	having characteristics of	linguistic
-ing*	verb form/ present participle	running
-ion, -tion, -ation, ition	act, process	occasion, attraction
-ity, -ty	state of	infinity
-ive, -ative, -itive	adjective form of a noun	plaintive
-less	without	fearless
-ly*	characteristic of	quickly
-ment	action or process	enjoyment
-ness	state of, condition of	kindness
-ous, -eous, -ious	possessing the qualities of	joyous
-s, -es*	more than one	books, boxes
-y	characterized by	happy

In Nepali Language

- **उपसर्ग (Prefix) — शब्दको अगाडि जोडिने भाग**
- उपसर्गहरू मूल शब्दको **पहिले** बसेर नयाँ अर्थ वा जोर दिन्छन्।
- **धेरै प्रयोग हुने नेपाली उपसर्गहरू:**

उपसर्ग	अर्थ	उदाहरण
अ-	नकारात्मक / अभाव	अशिक्षा, अयोग्य
अति-	धेरै, अत्यधिक	अतिथकान, अतिसूक्ष्म
सु-	राम्रो, सद	सुसंस्कार, सुबुद्धि
दु- / दुः -	खराब, बिग्रिएको	दुर्व्यवहार, दुःख, दुःशासन
प्र-	आरम्भ, अगाडि	प्रारम्भ, प्रवास, प्रगति
वि-	विभाजन/अनेक	विवरण, विकल्प
अनु-	पछाडि, अनुसरण	अनुशासन, अनुकरण

प्रत्यय (Suffix) — शब्दको पछाडि जोडिने भाग

- प्रत्ययहरू मूल शब्दको **पछि** बसेर:
  - शब्दको **वर्ग बदल्छन्** (जस्तै, क्रिया → संज्ञा)
  - अर्थ **विस्तार** गर्छन्
  - नयाँ शब्द **निर्माण** गर्छन्
- **नेपालीमा प्रयोग हुने सामान्य प्रत्ययहरू:**

प्रत्यय	कार्य	उदाहरण
-पन	गुण, अवस्था	राम्रो → राम्रोपन, सजिलो → सजिलोपन
-कार	कार्य गर्ने व्यक्ति	खेती → किसानकार? (but 'कार' as suffix: शिक्षक → शिक्षकार हुदैन) but correct: लेख+कार = लेखक
-कर्ता / -क	कार्य गर्ने व्यक्ति	लेख → लेखक, पढ → पढक (पढ्ने)
-दार	सम्बन्धित व्यक्ति	काम → कामदार
-इ / -ई	अवस्था	ठूलो → ठूली (gender), रामो → रामई
-नी	स्त्री जाति	गायक → गायिका / गायनी
-वादी	सोच/धारणा मान्ने	समाज → समाजवादी, धर्म → धार्मिकवादी
-मुखी	तिर उन्मुख	पहाड → पहाडीमुखी, उत्तर → उत्तरमुखी

अफिक्स (Affixes) : उपसर्ग + प्रत्यय दुवैको सामूहिक नाम

- अफिक्स भनेको उपसर्ग र प्रत्यय दुवैलाई जनाउने ठूलो शब्द हो।
- **उदाहरण:**
- **उपसर्ग + मूल शब्द + प्रत्यय**
  - अ + धर्म + क → **अधर्मक**
  - प्र + जय + य → **प्रजयय** (प्रायोगिक → प्रयोगको)
  - सु + मन + य → **सुमन्य** (संस्कृत मूल)
- **सरल उदाहरण:**
  - अ + योग् + य = **अयोग्य**
  - अत् + सुख् + त = **अतिसुख्त / असुख्त** (rare in modern Nepali)

2.2 Morphemes, Morphology and Morphotactic

Morphology

- Morphology is the study of **word structure** and the rules governing word formation.
- It explains:
  - How smaller units called **morphemes** combine to form words
  - How words change to express **grammatical functions**
  - How new words are created
- In NLP, morphology is essential for understanding and generating natural language.

Two Main Types of Morphology

- 1. Derivational Morphology
- 2. Inflectional Morphology

1. Derivational Morphology

- Creates **new words** or changes the **part of speech** (POS).
- Examples:
  - *happy → happiness*
  - *read → readable*
  - *nation → national → nationalize*
- Characteristics:
  - Often changes meaning
  - May change POS (adj → noun, noun → verb)

2. Inflectional Morphology

- Adds grammatical information but **does not change the word class**.
- Examples:
  - Plural: *cat* → *cats*
  - Past tense: *walk* → *walked*
  - Comparative: *big* → *bigger*
- Characteristics:
  - Same meaning category
  - Same POS
  - Required by grammar (tense, number, case)

Morphemes and Morphotactic

Morpheme:

- A **morpheme** is the **smallest meaning-bearing unit** in a language.
  It is the minimal linguistic unit that carries **semantic** (meaning) or **grammatical** information.
- A morpheme cannot be divided into smaller parts **without losing or changing its meaning**.
- **Types of Morphemes**

  1. Free Morphemes

  - These morphemes **can stand alone** as independent words.
  - Examples: *book*, *cat*, r*un*, *big*
  - These words carry meaning by themselves.

  2. Bound Morphemes

  - These morphemes **cannot stand alone**.. They must attach to a free morpheme.
  - Examples:
    - *-s* (plural marker)
    - *-ed* (past tense)
    - *un-* (negation)
    - *re-* (repeat)
  - You cannot say "*-ed*" or "*un-*" by themselves—they need a root.

Morphemes

Morpheme:

- A **morpheme** is the **smallest meaning-bearing unit** in a language.
  It is the minimal linguistic unit that carries **semantic** (meaning) or **grammatical** information.
- A morpheme cannot be divided into smaller parts **without losing or changing its meaning**.
- **Types of Morphemes**

  1. Free Morphemes

  - These morphemes **can stand alone** as independent words.
  - Examples: *book*, *cat*, *run*, *big*
  - These words carry meaning by themselves.

  2. Bound Morphemes

  - These morphemes **cannot stand alone**.. They must attach to a free morpheme.
  - Examples:
    - *-s* (plural marker)
    - *-ed* (past tense)
    - *un-* (negation)
    - *re-* (repeat)
  - You cannot say "*-ed*" or "*un-*" by themselves—they need a root.

Parts of A Morphological Processor

- For a morphological processor, we need at least followings:
  1. **Lexicon** : It is the **list of stems and affixes together** with basic information about them such as their main categories (noun, verb, adjective, …) and their sub-categories (regular noun, irregular noun, …).
  2. **Morphotactic** : The **model of morpheme ordering** that explains which classes of morphemes can follow other classes of morphemes inside a word.
  3. **Orthographic Rules (Spelling Rules)** : These spelling rules **are used to model changes that occur in a word** (normally when two morphemes combine).

1. Lexicon

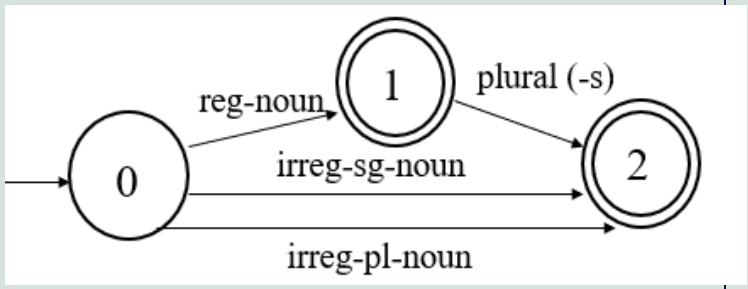
- A lexicon is a **repository for words** (stems).
- They are **grouped according to their main categories**.
  - noun, verb, adjective, adverb, …
- They may also be divided into sub-categories.
  - regular-nouns, irregular-singular nouns, irregular-plural nouns, …
- The simplest way to create a morphological parser: put all possible words (together with its inflections) into a lexicon.

2. Morphotactic

- **The rules that govern the order of morphemes in a word, Which morphemes can follow which morphemes**.
- **Morphotactic** is the **syntax of morphemes: what order they come in**, what kind of units they make.

Lexicon:

regular-noun	irregular-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s
cat	sheep	sheep	
dog	mice	mouse	



- Simple English Nominal Inflection (Morphotactic Rules)

Example:

(q0, fox)→ q1

(q1, s) →q2

Here, **cat** and **cats** are accepted by the FSM

3. Orthographic Rules

- We need FSTs to map intermediate level to surface level.
- For each spelling rule we will have a FST, and these FSTs run parallel.

- Some of English Spelling Rules:
  - **a) consonant doubling:**
    - 1-letter consonant doubled before ing/ed
    - Example: beg/begging
  - **b) E deletion:**
    - Silent e dropped before ing and ed
    - Example: make/making

- **c) E insertion:**
  - e added after s, z, x, ch, sh before s
  - Example: watch/watches
- **d) Y replacement:**
  - y changes to ie before s, and to i before ed
  - Example: try/tries try/tried
- **e) K insertion:**
  - verbs ending with vowel+c we add k
  - Example: panic/panicked

Morphological Processes

- Morphological processes describe **how new words are formed** or **how existing words change their form** while expressing grammatical or semantic information.
- It deals with:
  - **How words are formed from smaller meaningful units** (morphemes)
  - **The internal structure of words**
  - **Rules governing word formation** (morphotactics)

Major morphological processes:

Process	Purpose	Example	NLP Challenge
Affixation	Add grammatical/semantic info	un+happy	Affix segmentation
Derivation	Create new word	happy → happiness	POS prediction
Inflection	Grammatical variation	run → running	Tense/number features
Compounding	New concept	black + board	Tokenization
Reduplication	Plural/intensity	Tweeny-weeny	Language-specific rules
Alternation	Irregular change	sing → sang	Irregular lexicon
Suppletion	Completely irregular	go → went	Lexical exceptions
Cliticization	Attach functional words	I am → I'm	Tokenization
Zero derivation	Change POS	run (N/V)	Context sensitivity

1. Affixation

- Affixation is the **most common** morphological process in many languages, especially English.
- **Types:**
  - **Prefixation** – affix added before the stem
    - *un- + do* → **undo**
  - **Suffixation** – affix added after the stem
    - *teach + -er* → **teacher**
  - **Infixation** – affix inserted inside a stem
    - Mostly seen in languages like Tagalog: *sulat* (write) → *sumulat* (wrote)
  - **Circumfixation** – affixes placed around a root
    - German: *ge- + lieb + t* → **geliebt** (loved)
- **Importance in NLP**
  - Crucial for **morphological segmentation** and **affix detection**
  - Used in FST-based morphological analyzers

2. Derivation

- Derivation **creates new words** or **changes the part of speech**.
- **Examples:**
  - Noun → Adjective: *danger* → *dangerous*
  - Adjective → Noun: *happy* → *happiness*
  - Verb → Noun: *run* → *runner*
- **NLP Relevance:**
  - Helps in **lemmatization** to understand base concepts
  - Useful in **word embeddings** where related words share roots

3. Inflection

- Inflection creates **grammatical variants** of the same word, without changing the word class or meaning category.
- **English inflectional processes:**
  - Pluralization: *cat → cats*
  - Tense marking: *walk → walked*
  - Progressive: *eat → eating*
  - Comparative/Superlative: *big → bigger → biggest*
- **NLP Relevance:**
  - Essential for **POS tagging**, **parsing**, **MT**, and **speech recognition**
  - Inflection-heavy languages (Finnish, Turkish, Nepali) require robust analyzers

4. Compounding

- Combining two or more words to create a new word.
- **Types:**
  - **Closed compounds:** *blackboard, football*
  - **Hyphenated:** *mother-in-law*
  - **Open compounds:** *credit card*
- **NLP Relevance:**
  - Tokenization challenges (e.g., German **Donaudampfschifffahrtsgesellschaft**)
  - Important in information retrieval and search engines

- **5. Reduplication**
  - Repeating a part or whole of the root to change meaning.
  - **Examples:**
    - Indonesian: *buku* (book) → *buku-buku* (books)
    - Hindi: धीरे-धीरे (slowly-slowly → gradually)
  - **NLP Relevance:**
    - Requires special handling in morphological segmentation

- **6. Alternation (Ablaut/Internal Change)**
  - Internal vowel changes create different forms.
  - **Examples:**
    - *sing → sang → sung*
    - *foot → feet*
    - *goose → geese*
  - **NLP Relevance:**
    - FSTs must include **lexical lookup rules** for irregular patterns
    - Common in morphological analyzers

- **7. Suppletion**
  - A completely irregular morphological change; the root changes entirely.
  - **Examples:**
    - *go → went*
    - *good → better → best*
    - *person → people*
  - **NLP Relevance:**
    - Cannot be handled by rule-based morphology alone
    - Requires **lexicon lists** or exception dictionaries

- **8. Cliticization**
  - Clitics are morphemes that behave like words but attach phonologically to another word.
  - **Examples:**
    - *I'm = I am*
    - *she'll = she will*
    - French: *je t'aime* (I you-love)
  - **NLP Relevance:**
    - Needs careful handling during tokenization and POS tagging
    - Many languages (Nepali, French, Spanish) have abundant clitics

- **9. Zero Derivation (Conversion)**
  - A word changes its class **without** changing its form.
  - **Examples:**
    - *to Google* (noun → verb)
    - *to run → a run* (verb → noun)
  - **NLP Relevance:**
    - POS tagging must rely on context
    - Example: *"The run was long" vs "I run daily"*
- **10. Blending**
  - Parts of two words combined.
  - **Examples:**
    - *smoke + fog → smog*
    - *breakfast + lunch → brunch*
  - **NLP Relevance:**
    - Challenging for morphological parsing
    - Often seen in creative language (social media)

- **11. Back-formation**
  - Removing an affix-like part to form a new word.
  - **Examples:**
    - *editor → edit*
    - *babysitter → babysit*
  - **NLP Relevance:**
    - Adds ambiguity in lemmatization

Application of Morphology in NLP

Morphology is foundational to many NLP tasks:

1. Lemmatization & Stemming

Removing morphological variations.
NLTK & spaCy use FST-based or rule-based approaches.

2. Part-of-Speech Tagging

Morphological features help disambiguate words.

3. Information Retrieval/Search

Stemming improves search accuracy.
running, ran → run

4. Machine Translation

Needed for generating grammatically correct target-language forms.

5. Speech Recognition

Morphological cues improve pronunciation modeling.

6. Text-to-Speech

Morphology influences stress, pronunciation, inflection.

7. Spell Checking & Correction

Uses morphological rules to suggest correct forms.

8. Named Entity Recognition

Morphological cues help identify entities (e.g., *-wala, -dhar*).

9. Language Modeling

Morphology-aware subword models (BPE, WordPiece) outperform word models for rich-morphology languages.

How to remove prefix, suffix and affix in English using python?

- You can do it *manually*, with *word lists*,
- or using **NLTK's PorterStemmer**.

1. Remove Prefix Manually (simple string rules)

- You can define a list of common prefixes and remove them if the word starts with them.

```
def remove_prefix(word, prefixes):
    for p in prefixes:
        if word.startswith(p):
            return word[len(p):]
    return word

prefixes = ["un", "re", "pre", "dis",
"mis", "non", "in", "im"]
print(remove_prefix("unhappy", prefixes))
print(remove_prefix("rewrite", prefixes))
```

Output:

happy
write

2. Remove Suffix Manually (common suffix list)

```
def remove_suffix(word, suffixes):
    for s in suffixes:
        if word.endswith(s):
            return word[:-len(s)]
    return word

suffixes = ["ing", "ed", "er", "tion",
"ness", "able", "ful"]
print(remove_suffix("happiness", suffixes))
print(remove_suffix("teacher", suffixes))
print(remove_suffix("running", suffixes))
```

Output:

happi
teach
runn

3. Remove Affixes (Both Prefix + Suffix)

```
def remove_affixes(word, prefixes, suffixes):
    # remove prefix if matches
    for p in prefixes:
        if word.startswith(p):
            word = word[len(p):]
            break
    # remove suffix if matches
    for s in suffixes:
        if word.endswith(s):
            word = word[:-len(s)]
            break
    return word
prefixes = ["un", "re", "dis", "mis", "in"]
suffixes = ["ing", "ed", "er", "ness", "ment", "able", "ful"]
print(remove_affixes("unhappiness", prefixes, suffixes))
print(remove_affixes("rewriting", prefixes, suffixes))
print(remove_affixes("disagreement", prefixes, suffixes))
```

Output

happi
writ
agree:

4. Using NLTK Stemming (Automatically removes affixes)

- This is the *best simple method*—handles many English morphological rules.

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

words = ["running", "happiness", "unhappy", "rewriting", "agreement"]

for w in words:
    print(w, "→", stemmer.stem(w))
```

Output:
running → run
happiness → happi
unhappy → unhappi
rewriting → rewrite
agreement → agre

What is Lemmatization?

- **Lemmatization** is the process of reducing a word to its **base or dictionary form**, called a **lemma**.
  - The lemma is the *real* word you would find in a dictionary.
  - Unlike stemming, lemmatization uses **grammar + vocabulary** to produce a correct root word.
- **Why do we need lemmatization?**
  - To normalize words for NLP (search engines, text mining, chatbots, etc.)
  - To make different word forms comparable
  - To understand meaning more accurately

Why lemmatization is powerful

- Recognizes irregular forms
  - e.g., “went” becomes **go**, “better” becomes **good**
- Understands part of speech
  - “better” (adjective) → good
  - “better” (verb: *to better something*) → better
- Produces valid English words
  - stemming: *studies* → *studi*
  - lemmatization: *studies* → *study*

Lemmatization vs Stemming (Quick difference)

Feature	Stemming	Lemmatization
Output	Cut-off version, may not be a real word	Real meaningful word
Uses grammar?	No	Yes
Example	*better* → *bett*	*better* → *good* (correct lemma)

Examples of Lemmatization

Verbs

Word	Lemma
running	run
ate	eat
gone	go
writing	write

Nouns

Word	Lemma
children	child
mice	mouse
boxes	box
cars	car

Adjectives

Word	Lemma
better	good
worse	bad
happier	happy

Code Example

```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

lemmatizer = WordNetLemmatizer()

words = ["running", "mice", "went", "better", "cars", "studies"]

for w in words:
    print(w, "→", lemmatizer.lemmatize(w))
```

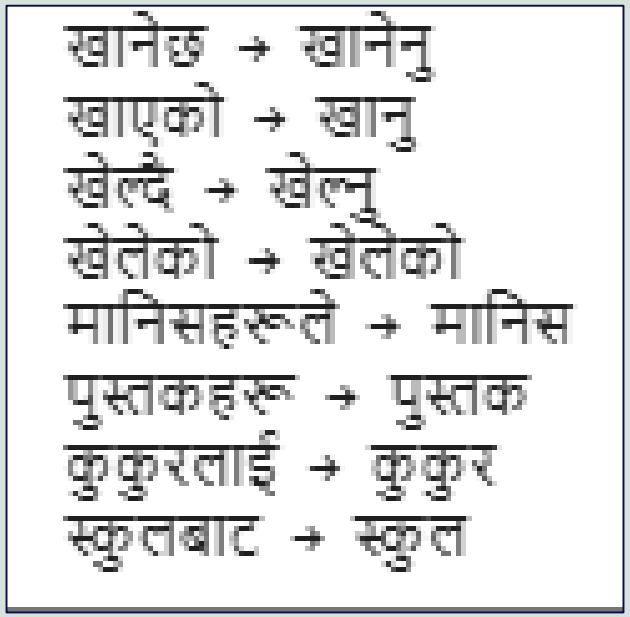
Output :

running → running
mice → mouse
went → went
better → better
cars → car
studies → study

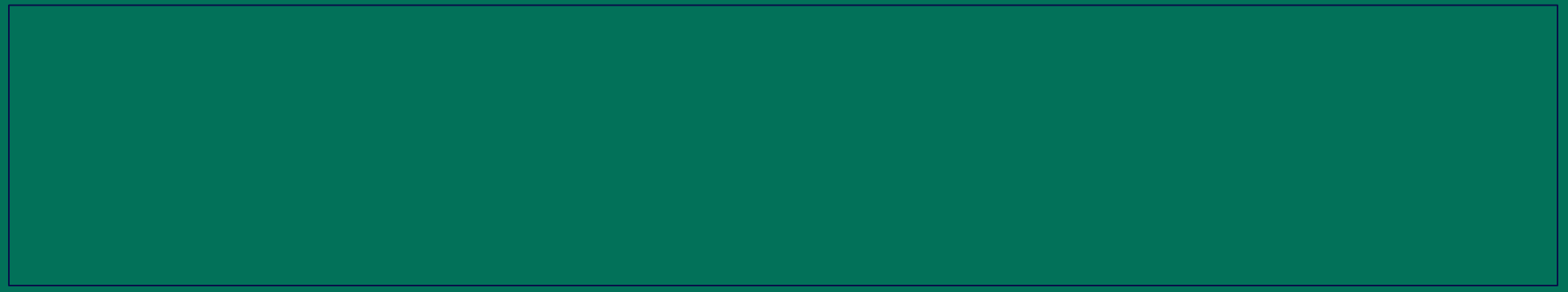
Nepali lemmatizer

```
def nepali_lemmatize(word):

    # Step 1: Remove plural suffix
    plural_suffixes = ["हरू", "हरु"]
  for suf in plural_suffixes:
        if word.endswith(suf):
            word = word[:-len(suf)]

    # Step 2: Remove common case
markers (postpositions)
    case_markers = ["ले", "लाई", "मा", "बाट",
"तिर", "हरू"]
  for cm in case_markers:
        if word.endswith(cm):
            word = word[:-len(cm)]

    # Step 3: Verb endings (very
simplified)
    verb_suffixes = ["एको", "एकी", "एकै", "एछ",
"एछौ", "एछन", "छ", "छन्",
            "दै", "ने", "नु", "यो", "एउ", "ए"]

  for vs in verb_suffixes:
        if word.endswith(vs):
            word = word[:-len(vs)]
            # assume Nepali verb root
ends with नु
            return word + "नु"

  return word
```

```
words = ["खानेछ", "खाएको", "खेल्दै", "खेलेको",
    "मानिसहरूले", "पुस्तकहरू", "कुकुरलाई", "स्कुलबाट"]

for w in words:
    print(w, "→", nepali_lemmatize(w))
```



2.3 Introduction to FSM and FST



Finite State Machine in morphological representation

- A Finite State Machine (FSM) is an efficient way to model morphological rules.
- It works like a **word-building machine**, moving through a sequence of **states** as it reads morphemes.
- Finite state machine are quite useful as a language recognizer.
- For example, NFA for the words ‘boy’ and ‘bat’ is shown in the Figure below.

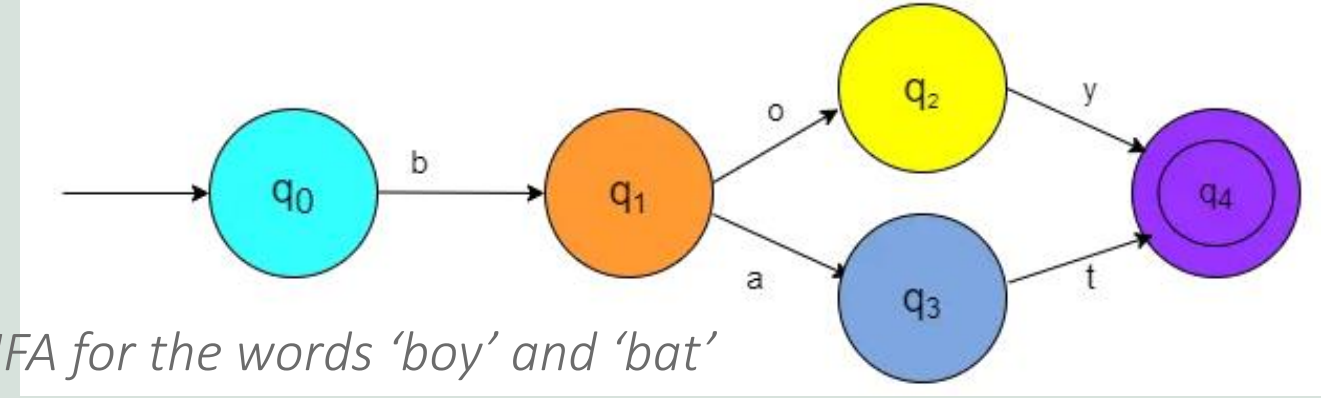


Figure: NFA for the words ‘boy’ and ‘bat’

FSM

- **Finite State Machine (FSM)**
  - Recognizes sequences of symbols.
  - Used to check whether a word *fits* allowable patterns.
- A **Finite State Machine** is a model with:
  - **States**: nodes representing conditions
  - **Transitions**: arrows labeled with input symbols
  - **Start state**: where processing begins
  - **Final/accepting state**: indicates a valid sequence
- **Use case in NLP:**
  - Check whether a **word fits a certain pattern**.

1. Finite Automation (FA):

A finite automaton is a collection of 5-tuple (Q, ∑, δ, q0, F), where:

- Q: finite set of states
- ∑: finite set of the input symbol
- q0: initial state
- F: final state
- δ: Transition function

Or the transition function can be written as:

δ(a, 0) → a
δ(a, 1) → b
δ(b, 0) → c
δ(b, 1) → a
δ(c, 0) → b

Example

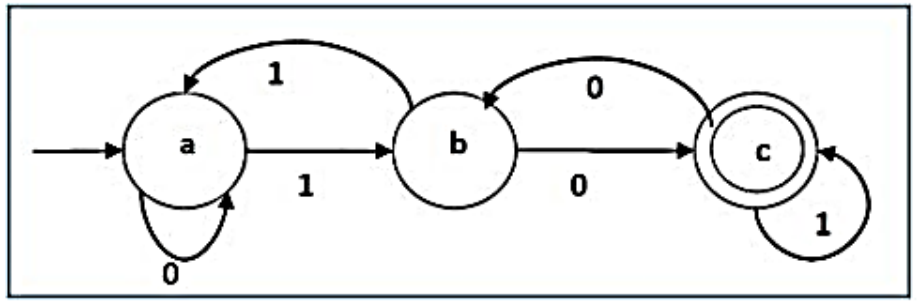
Let a deterministic finite automaton be →

$Q = \{a, b, c\}$,
$\sum = \{0, 1\}$,
$q_0 = \{a\}$,
$F = \{c\}$, and

Transition function δ as shown by the following table –

Q	0	1
a	a	b
b	c	a

Its graphical representation would be as follows –



Classwork

1. Design a DFA that accepts the language given by: L= {w ∈ {a, b}* : w has even numbers of b}

Numerical:

1. **Design a DFA that accepts the language given by: L= {w ∈ {a, b}* : w has even numbers of b}**

Sol^{n}: Here, L = { abb, bb, bbbb, bab, bba, bbabb, babbab, ---- }

First let us design for bb

Again design for bbbb

Hence the DFA is

$M = \{ Q, \Sigma, \delta, q_0, F \}$

where,

$Q = \{ q_0, q_1, q_2 \}$

$\Sigma = \{ a, b \}$

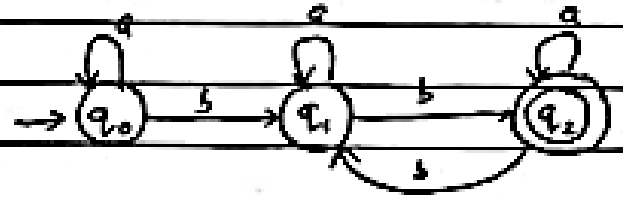
$q_0 = \{ q_0 \}$

$F = \{ q_2 \}$

δ :

Q	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_1

State diagram is :



2. Design a DFA that accepts the language given by: L= {w ∈ {1, 0}* : w ends with 100 }

Here,

$L = \{100, 1100, 0100, 10100, 010100, \ldots\}$

Let $M = \{Q, \Sigma, \delta, q_0, F\}$ where

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

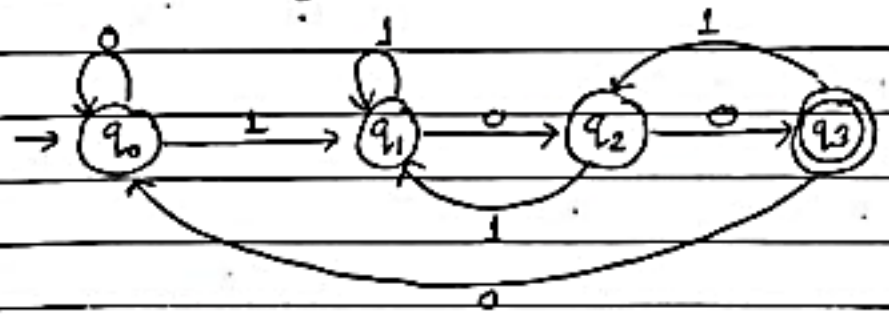
$q_0 = \{q_0\}$

$F = \{q_3\}$

δ:

Q	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_3	q_1
q_3	q_0	q_1

Here, the state diagram is:

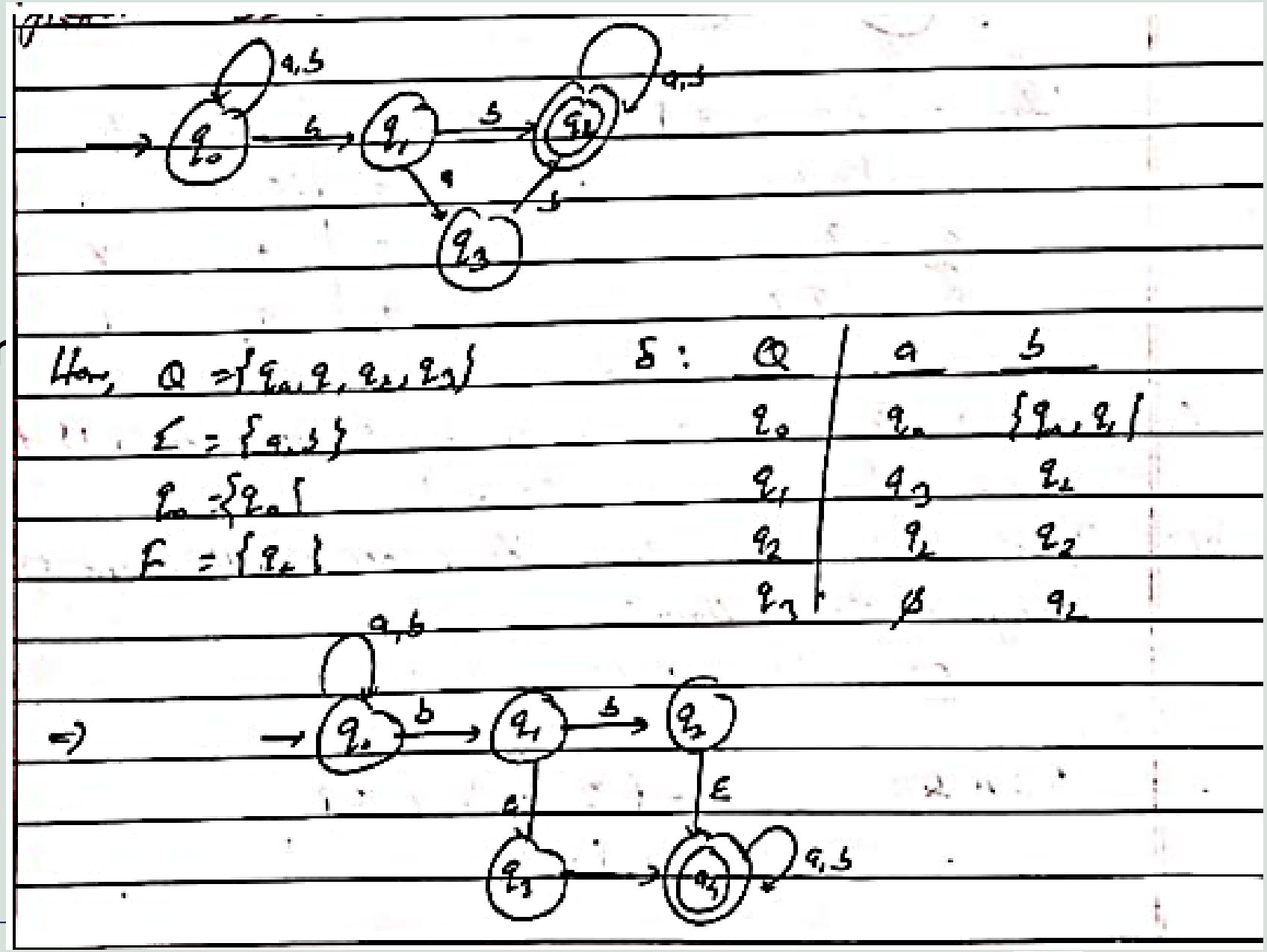


Verification:

1 0 1 0 0 → Verified ends at q_3, accepted

1 0 1 0 1 → ends at q_1, rejected

1. Design a NFA that accepts the set of strings containing occurrence of patterr bb or bab

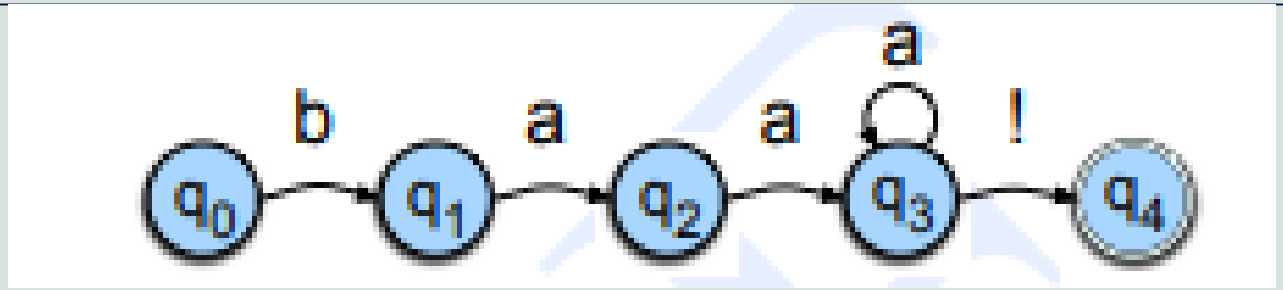


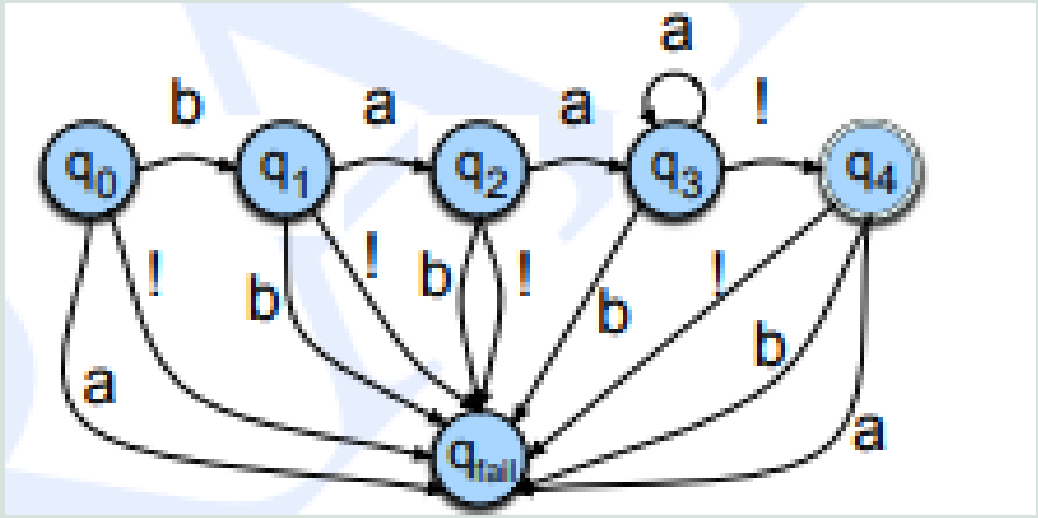
Assignment:

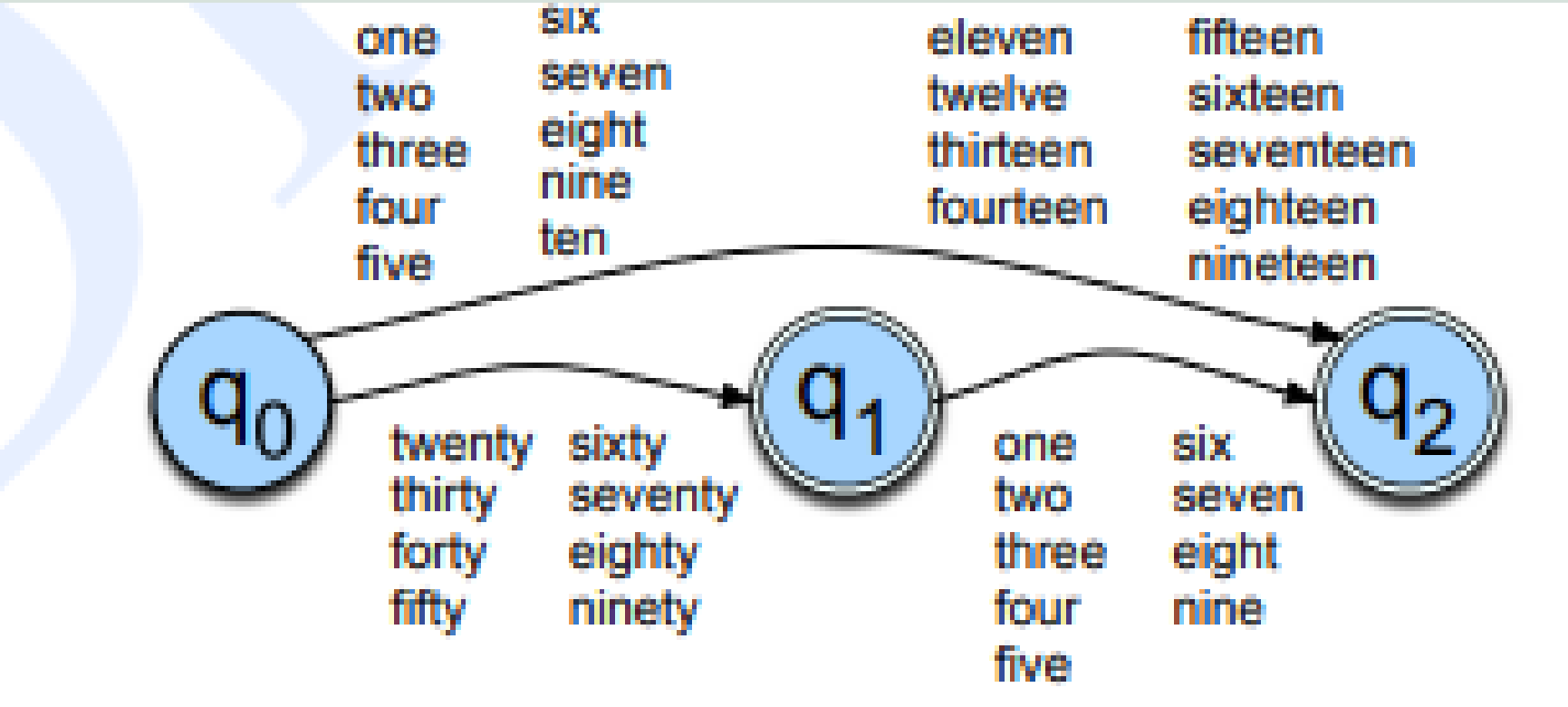
1. Construct a DFA starting with aba, ∑ = a, b.
2. Design a DFA that accepts the language given by: L= {w ∈ {a, b}* : w does not contain three consecutive b }
3. Design a DFA that accepts the language given by: L= {w ∈ {0, 1}* : w starts with 01 having even length}
4. Design a DFA that accepts the language given by: L= {w ∈ {0, 1}* : w starts with 01 }

A finite-state automation for talking sheep

baa!
baaa!
baaaa!
baaaaa!
baaaaaa!
...

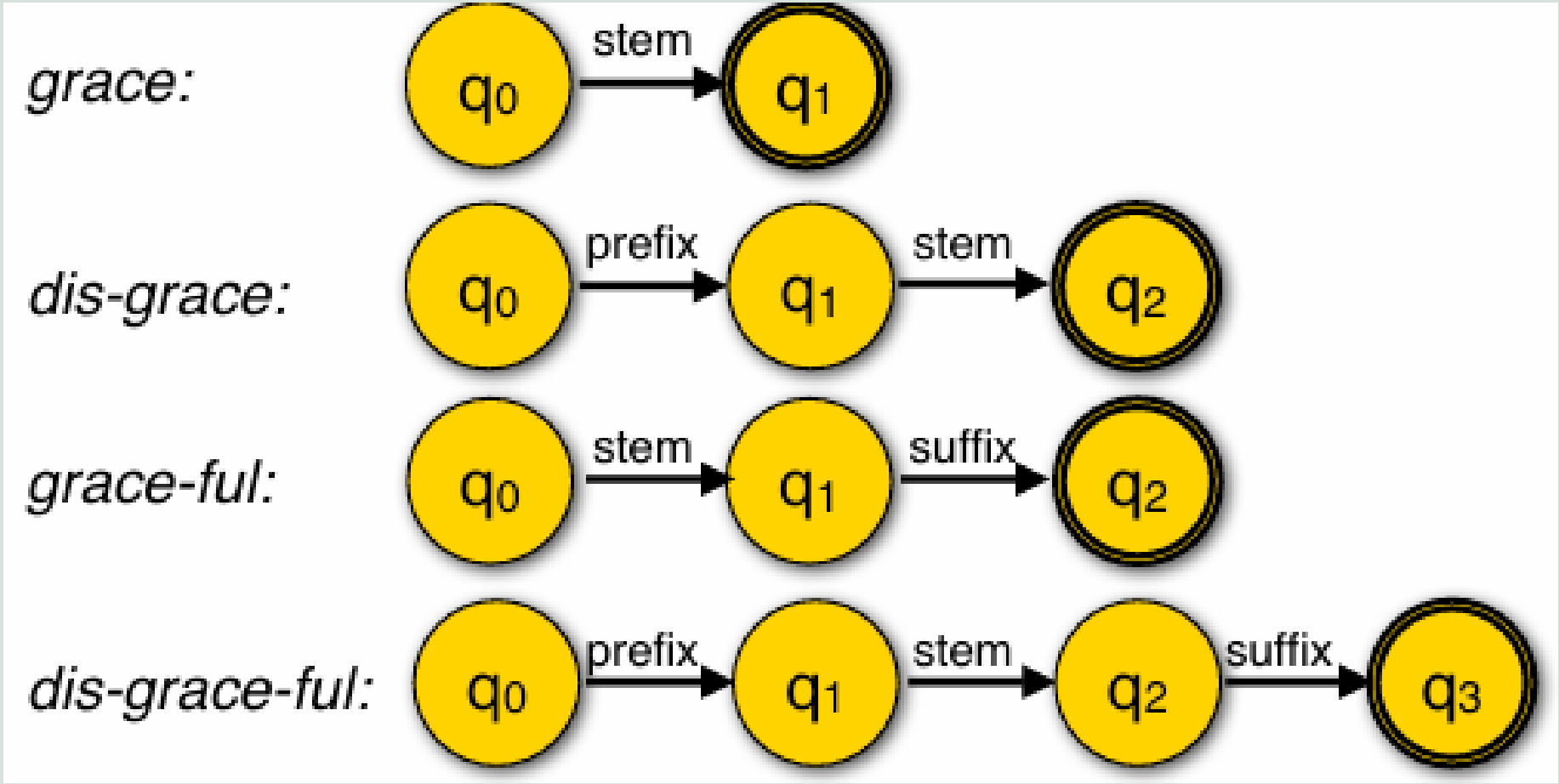






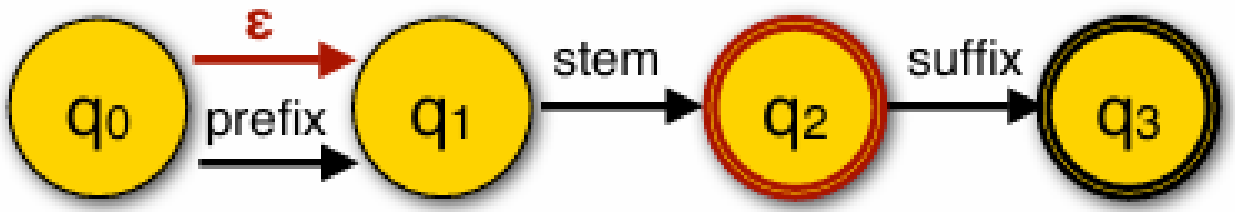
An FSA for the words for English numbers 1-99

Finite state automata for morphology

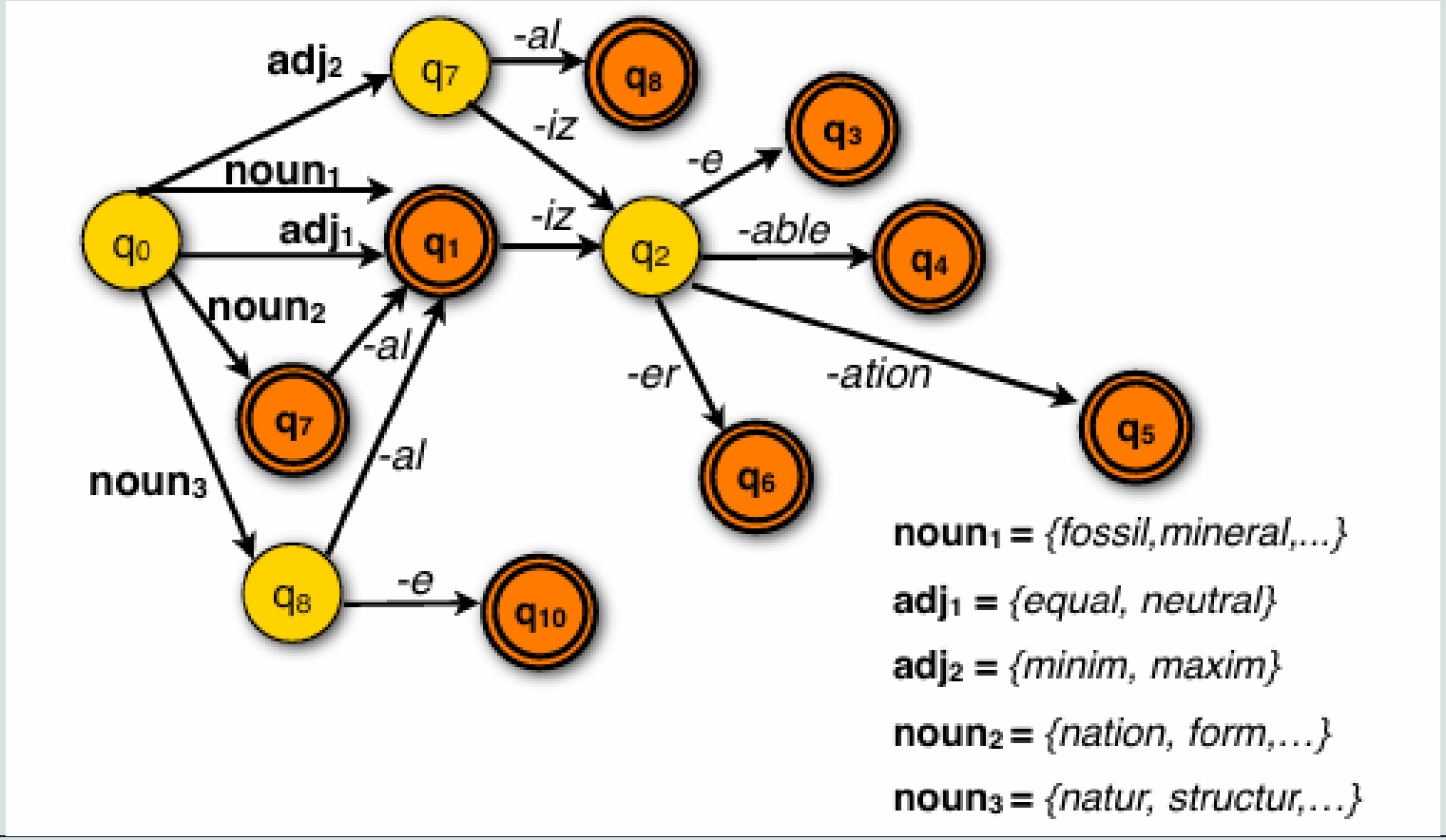


- **Union: Merging Automata**

grace,
dis-grace,
grace-ful,
dis-grace-ful



- **FSAs for derivational morphology**



Assignment:

- Design a **Finite-State Automaton (FSA)** that recognizes **all English plural nouns ending with “-s” or “-es.”**

Finite-State Transducers (FST)

- A **Finite State Transducer (FST)** is an extension of the **Finite State Machine (FSM)**
- It **maps between two levels of representation** — usually **lexical (abstract)** and **surface (actual)** word forms.
- In simple words:
  - An **FSM** *accepts or rejects* a string (pattern recognition).
  - An **FST** *translates* or *transduces* one string into another.

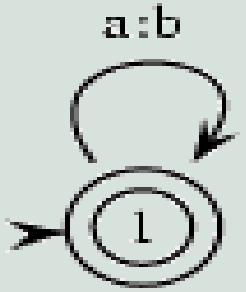
Structure of an FST

- An FST has:
  - **States** (like FSM)
  - **Transitions** between states
  - **Input symbols** (lexical level, e.g., “play+PAST”)
  - **Output symbols** (surface level, e.g., “played”)
  - **Start** and **final states**
- Each transition carries **two symbols**:

  (input symbol : output symbol)
  - Example: (y : i)

    means the input symbol “y” is changed to “i” (as in “study → studied”).

- **an augmentation of FSAs** in which **there are two tapes**
  - A two-tape automaton,
  - the upper tape an "underlying representation" tape
  - and the lower a "surface representation“



Lexical Tape | | a | | | | | | (upper tape)

Surface Tape | | b | | | | | | (lower tape)

a:b at the arc means that in this transition the transducer reads **a** from the first tape and writes **b** onto the second.

FST Operation Example

- **Example 1: Simple word “cats”**
- **Lexical input:** cat + PLURAL
  Surface output: cats

State	Input	Output	Next
S0	c	c	S1
S1	a	a	S2
S2	t	t	S3
S3	+PLURAL	s	S4 (final)

- FST produces cats as output.

- **Example 2: “studied”**
- **Lexical form:** study + PAST
  Surface form: studied

State	Input	Output	Note
S0	s	s	—
S1	t	t	—
S2	u	u	—
S3	d	d	—
S4	y	i	Rule: “y → i”
S5	+PAST	ed	—
Final	—	—	Valid surface form

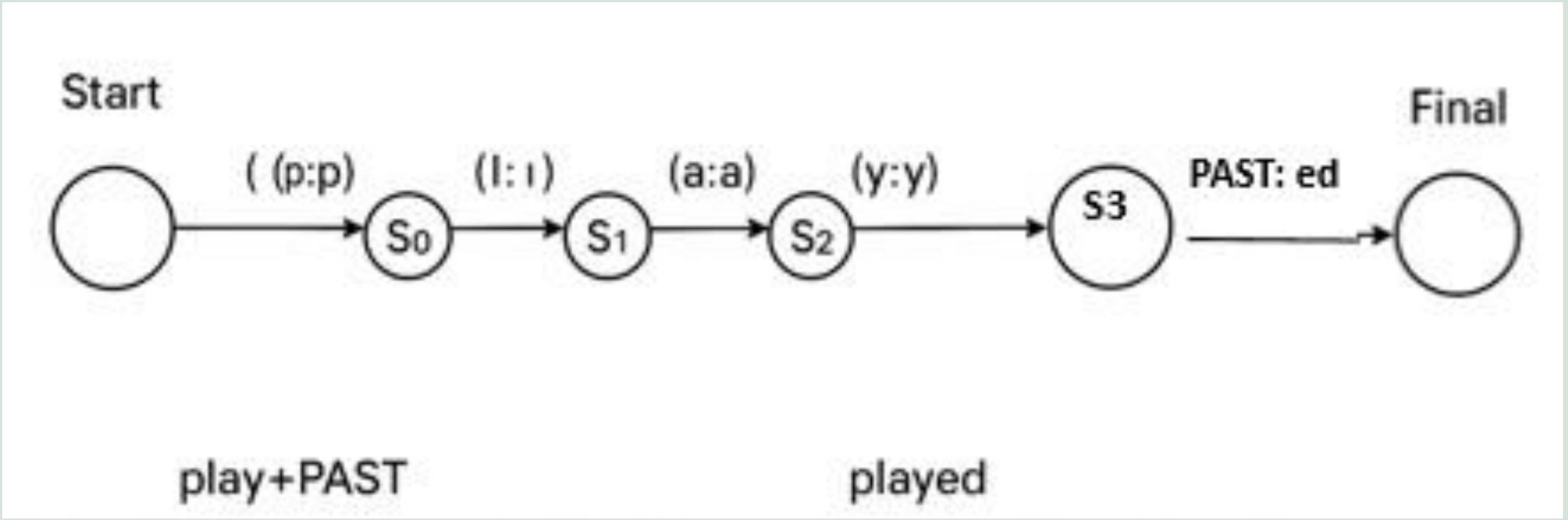


Figure: FST

FST operations

- FSTs support several key operations that make them powerful for **building modular morphological analyzers**.
  1. Concatenation
  2. Union
  3. Kleene Closure (Iteration)
  4. Composition

1. Concatenation

- This operation links two FSTs **in sequence**.
- **Definition**
  - Given FST A and FST B: **Concatenation(A, B)** creates a new machine that first accepts strings from A and then from B.
- **Example in Morphology**
  - FST for **noun stem**
  - FST for **plural suffix**
- Concatenate them:
  - cat + s = cats dog + s = dogs
- Useful for building **morphotactic sequences** like:
  - Prefix + Stem + Suffix

2. Union

- Union combines two FSTs to accept **either** machine's strings.
- **Definition**
  - Union(A, B) accepts any pair (input/output) accepted by A or B.
- **Example in Morphology**
  - Union of plural rules for English:
    - *-s* suffix FST
    - *-es* suffix FST
    - vowel alternation FST (*foot* → *feet*)
  - Union merges all three into a single plural analyzer.

3. Kleene Closure (Iteration)

- Kleene star (*) allows **repetition** of an FST zero or more times.
- **Definition**
  - Kleene*(A) = A repeated 0, 1, 2, … times.
- **Example in Morphology**
- For languages with repeated affixes or reduplication:
  - Indonesian plural FST for reduplication
    - buku → buku-buku
  - English derivational chaining:
    - quick → quick-ly → quick-ly-ness

4. Composition

- **The most important operation for NLP morphology.**
- **Definition**
- Composition(A, B) = an FST that maps input → intermediate → output.
- If A maps:
  - surface → lexical
- And B maps:
  - lexical → morphological features
- Then Composition(A, B) directly maps:
  - surface → features
- **Morphological Example**
  - FST for morphotactics
    - cat +N +PL
  - FST for orthographic rules
    - +PL → "s"

Applications of FST in NLP

- **Morphological analyzers and generators**
  → Recognizing or producing valid word forms
- **Spell checkers / correctors**
  → Understanding valid morphology
- **Machine translation**
  → Generating target-language morphology
- **Text-to-speech**
  → Generating correct pronunciations
- **Speech recognition**
  → Mapping spoken words to possible lexical forms

End of chapter