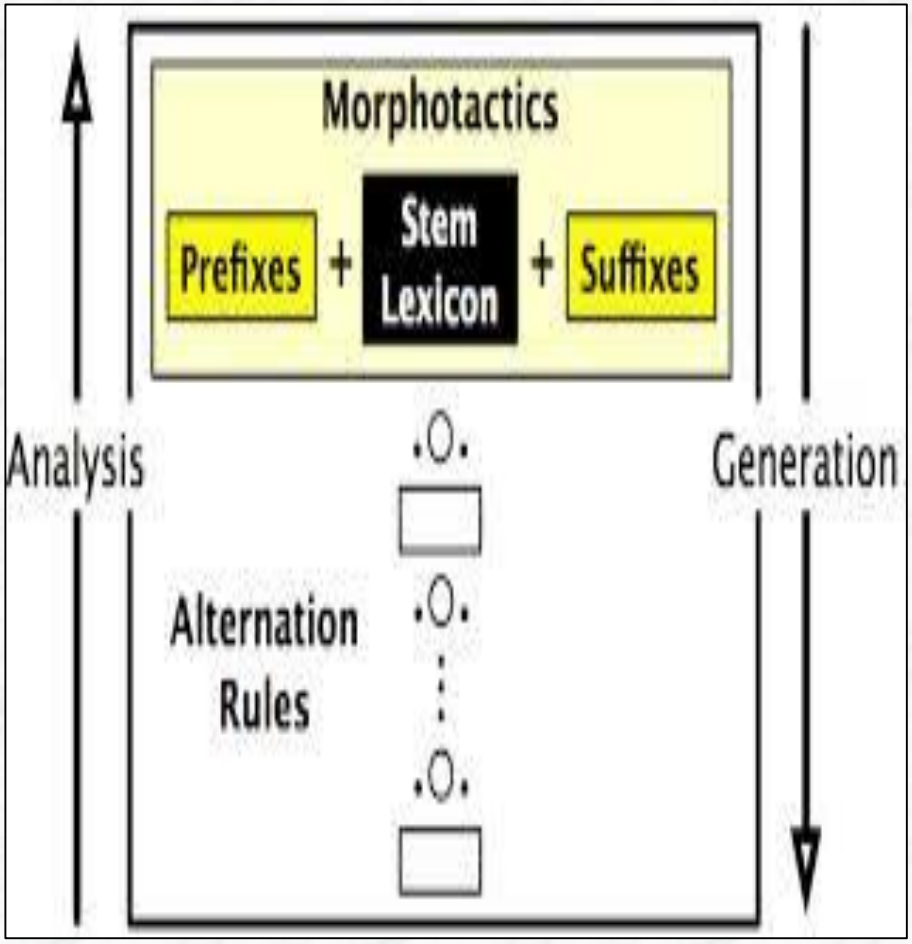
Morphological analysis



Words and Transducers

Instructor : Dr. Hanaa Bayomi Ali
Mail : h.mobarz @ fci-cu.edu.eg

Stages of language processing

1- Phonetics and phonology	Speech sound
2- Lexical Analysis	Dividing the whole chunk of txt into paragraphs, sentences, and words
3- Morphology & Lexicon	Words & their forms
4- Syntactic Analysis	Structure of sentences
5- Semantic Analysis	Meaning of words & sentences
6- Pragmatics	Meaning in context & for a purpose
7- Discourse	Connected sentence processing in a larger body of text

Words

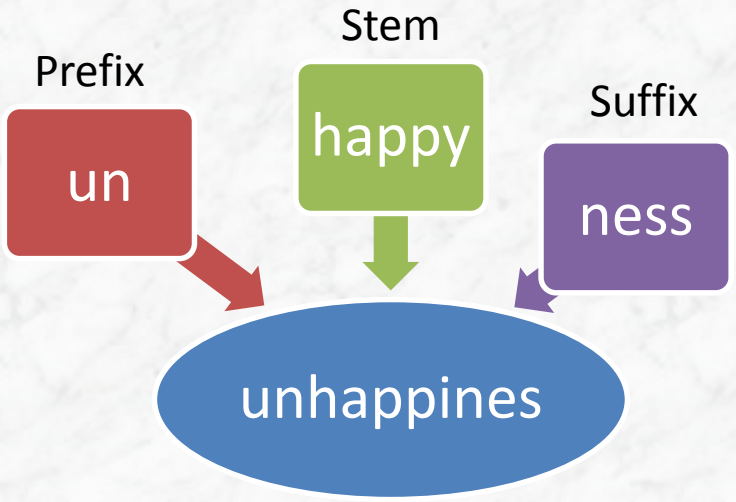
- In formal languages, words are arbitrary strings
- In natural languages, words are made up of meaningful subunits called morphemes
- morpheme = "minimal meaning-bearing unit in a language“
- Morphemes may be

Stems: the main morpheme of the word.

Affixes: convey the word’s role, number, gender, etc.

“Unhappiness” Contains 3 morphemes

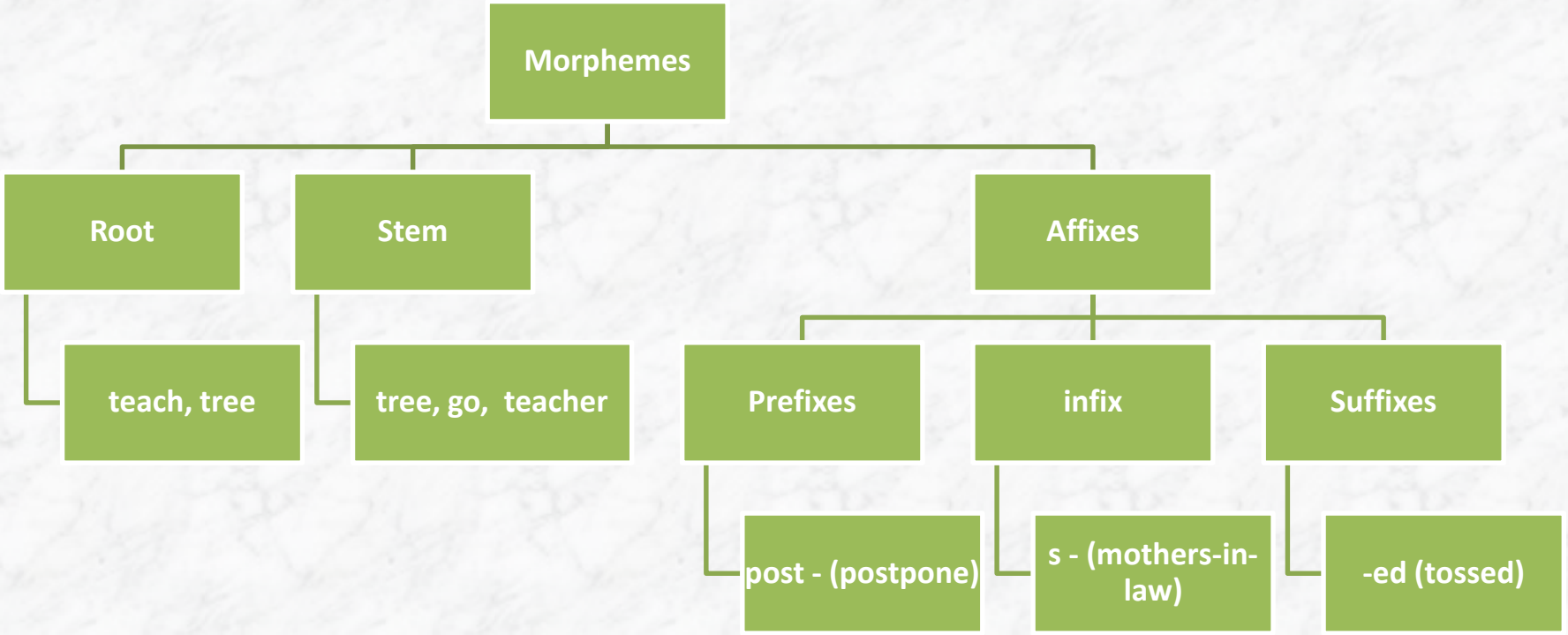
1- “*un*” means "not”
2- “*ness*” means "being in a state or condition"
3- “*Happy*” is a *free morpheme* (as a "word" in its own right)



Morphemes definitions

- ***Affix***
  - –A bound **morpheme** that is joined before, after, or within a root or stem
    - **Suffixes:** follow the root/stem "eat, eats"
    - **Prefixes:** precede the root/stem "happy, unhappy"
    - **Infixes:** inserted into the root/stem " s(plural), mother**s**-in-low"
  - –Affixes maybe derivational affixes, inflectional affixes or both
- ***Root***
  - The portion of the word that:
    - is common to a set of derived or inflected forms, if any, when all affixes are removed
    - is not further analyzable into meaningful elements
    - carries the principle portion of meaning of the words
- ***Stem***
  - The root of a word, together with any derivational affixes
  - the morpheme that forms the central meaning unit in a word

Morphemes definitions



Free and Bound Morpheme

Analysis at a morphological level is concerned with structural elements of meaning called **morphemes**. Morphemes are classified into two types:

1- Free Morphemes: girl, boy, mother, etc.

- These are words with a complete meaning, so they can stand alone as an *independent word* in a sentence.

2- Bound Morphemes: These are lexical items incorporated into a word as a *dependent part*. They cannot stand alone, but must be connected to another morpheme.

– Bound morphemes operates in the connection processes by means of **derivation, inflection,** and **compounding.**

EX. Bound morpheme: books

- Books = +morpheme +morpheme
- Books= +book +s

Morphological Analysis

Why do we need to do Morphological Analysis?

- determine meanings of individual word.

- To break word down into component morphemes and build a structured representation.

- Important for many tasks:
  - machine translation
  - information retrieval
  - Parsing
  - text simplification
  - Spelling correction

Morphological Analysis

- How words are constructed out of basic meaning unit
  - **Inflection** (number, person, tense , Gender …): same core meaning (don't change syntactic category) (**Stemmer**)
    - love + past tense →loved
  - **Derivation** (adjective-adverb, noun-verb): change meaning
    - un + friend + ly →unfriendly
  - **Compounding** (separate words or single word)
    - black + bird → blackbird
    - Foot + ball → football
- Arabic words is highly inflected

Morphological Analysis

- ***Inflection*** is the form variation of a word under certain **grammatical** conditions, these conditions consist notably of the number, gender, conjugation, or tense

  stem + gram. morpheme → same class
  Ex: help + ed → helped

- ***Derivation*** combines affixes to an existing root or stem to form a **new word**. ***Derivation is more irregular and complex than inflection***. It often results in a change in the **part of speech** for the derived word.

  stem + gram. morpheme → different class
  Ex: civilization

Can you divide these words?

- teaches
- Happiness
- Unbelievable
- Teacher
- Monster
- Rattlesnake

Can you divide these words?

• Teaches	**teach-s**	**Inflection**
• Happiness	**happi-ness**	***Derivation***
• Unbelievable	**un-believ-able**	***Derivation***
• Teacher	**teach-er**	***Derivation***
• Monster	**monster**	**Inflection**
• Rattlesnake	**rattle-snake**	**Compound**

Distinguishing inflection from derivation

main criteria:

1- Category change: Inflection does not change grammatical category; derivation sometimes does (thereby creating new words).

Form 1	Form 2	Category change	Der/Inf
car	cars	No	(s) Inflection
Play	plays	no	(s) Inflection
Formal(adj)	Formalize (V)	yes	(ize) Derivation
Read (V)	Readable(adj)	yes	(able) Derivation

Distinguishing inflection from derivation

Three main criteria:

2- Order: Derivational affixes must combine with the base before an inflectional affix does (root - aff_{der}- aff_{inf} " $teach_{root}$-er_{der}-s_{inf}).

- Inflectional affixes always have a regular meaning.
  - the plural 's *in word-forms like* bicycles*,* dogs*,* shoes, tins*,* trees*, and so on*

'more than one'

- Derivational affixes may have irregular meaning.
  - write :: writer (one who writes)
  - paint :: painter (one who paints)
  - cut :: cutter? (an instrument used to cut)
  - Mix :: Mixer (an instrument used to mix)

More on Inflection

Noun inflectional suffixes	•Plural marker -s •Possessive marker ‘s
Verb inflectional suffixes	•Third person present singular marker -s •Past tense marker -ed •Progressive marker -ing •Past participle markers -en or –ed
Adjective inflectional suffixes	•Comparative marker -er •Superlative marker -est

Inflectional Suffixes in English

Morphological Parsing

- **Parsing** means taking an input and producing some sort of structure for it.

- **Morphological parsing** means breaking down a word form into its constituent morphemes.
  - e.g. Colourful $\Rightarrow$ **colour+ ful**

- Mapping of a word form to its baseform is called **stemming**.
  - e.g. Teacher $\Rightarrow$ **Teach**

Finite-State Morphological Parsing

- In order to build a parser we need the following:
  - a **lexicon** containing the stems and affixes,
  - **morphotactics**, i.e. the model of morpheme ordering, e.g. un + friend + ly instead of ly+ friend + un
  - a set of **rules** (orthographic, etc.), i.e. the model of changes that occur in a word, usually when two morphemes combine, e.g. *city + s* $\Rightarrow$ *cities*.

Finite-State Automaton for Inflection of English Verbs

Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	Jump	Parse	Fry	Sob	Eat	Bring	Cut
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
–ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

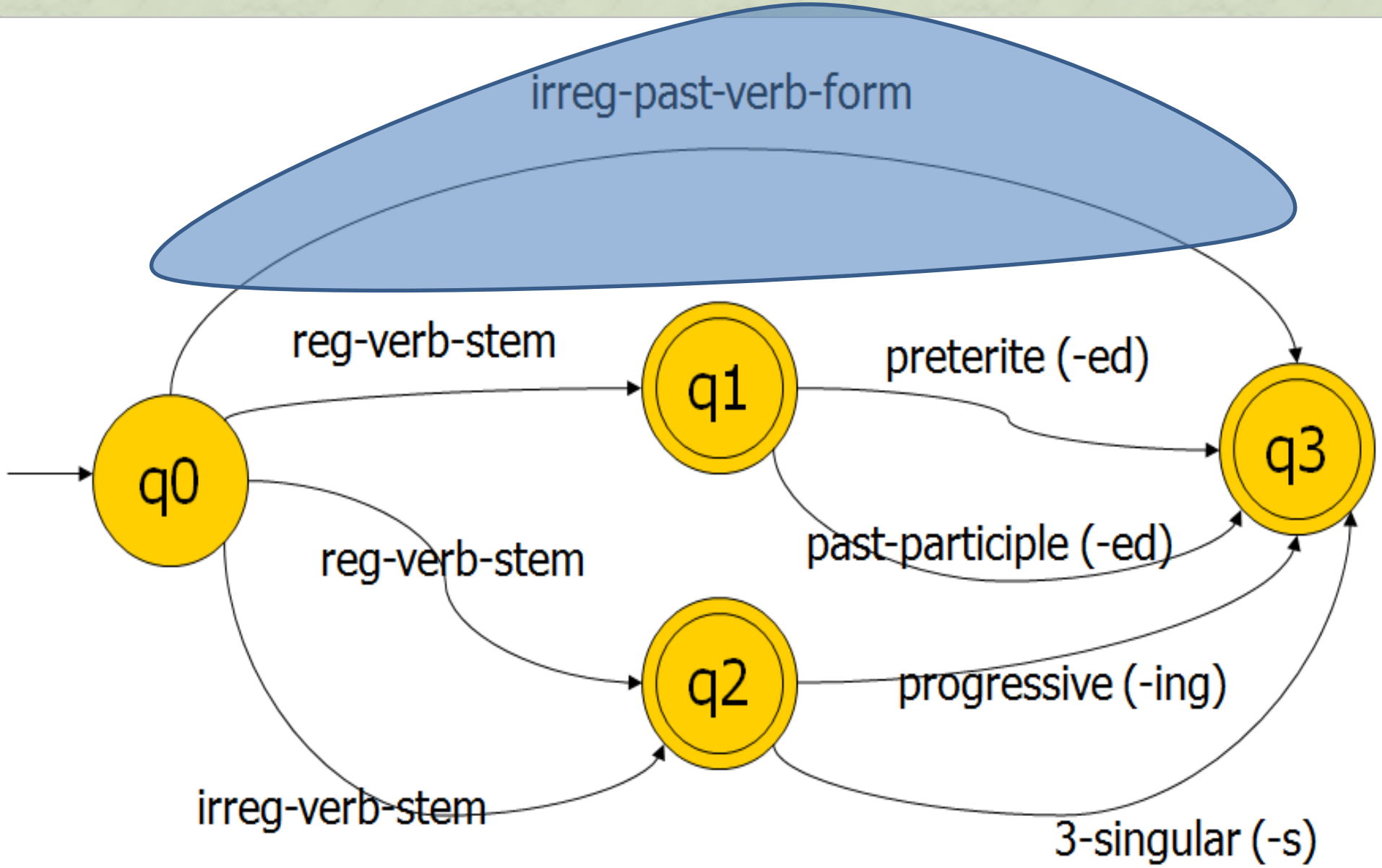
Finite-State Automaton for Inflection of English Verbs

Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	**Jump**	**Parse**	**Fry**	**Sob**	**Eat**	**Bring**	**Cut**
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
–ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

Irreg-past –verb-form

Finite-State Automaton for Inflection of English Verbs



Finite-State Automaton for Inflection of English Verbs

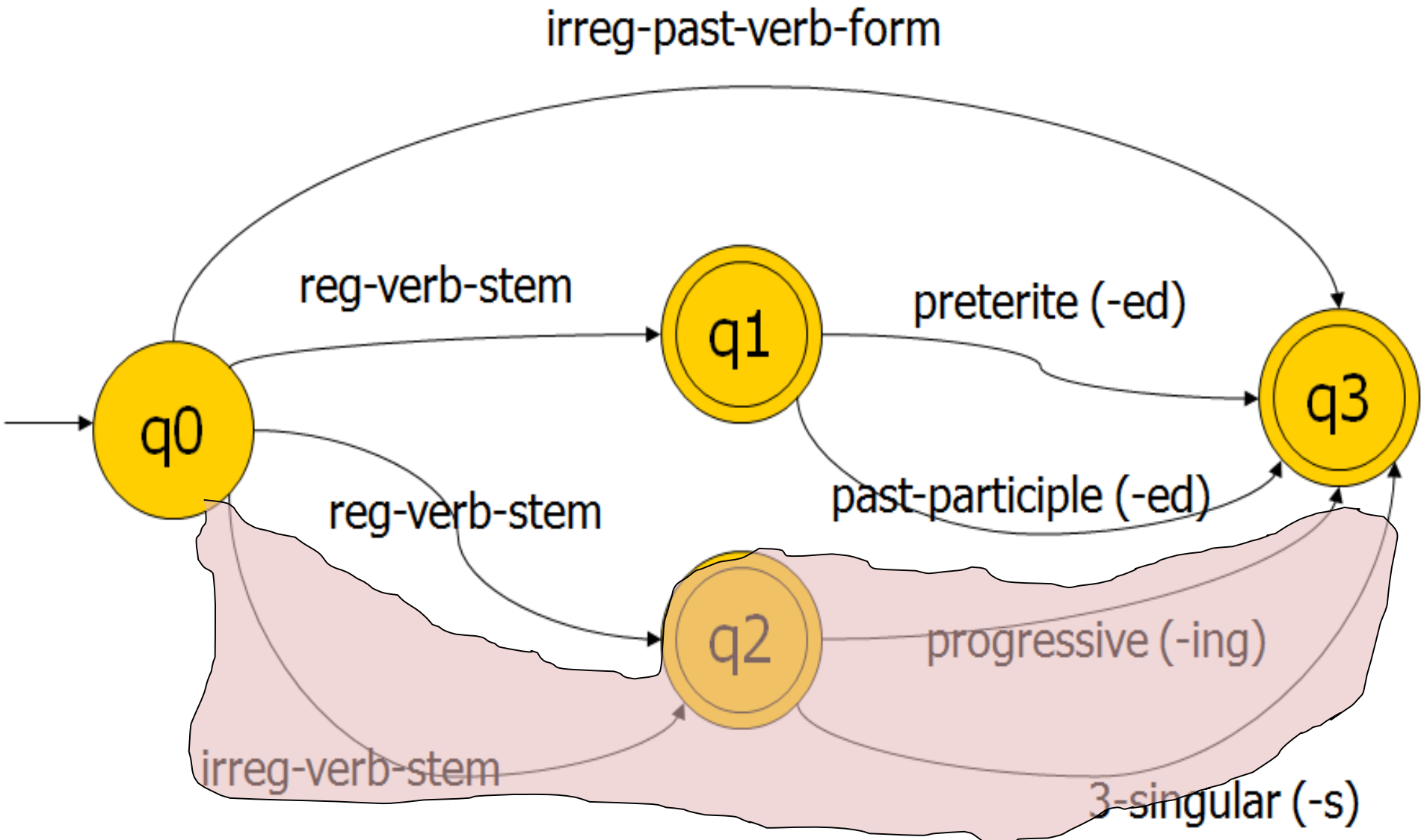
Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	Jump	Parse	Fry	Sob	Eat	Bring	Cut
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
–ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

Irreg-verb –progressive-

3 singular

Finite-State Automaton for Inflection of English Verbs



Finite-State Automaton for Inflection of English Verbs

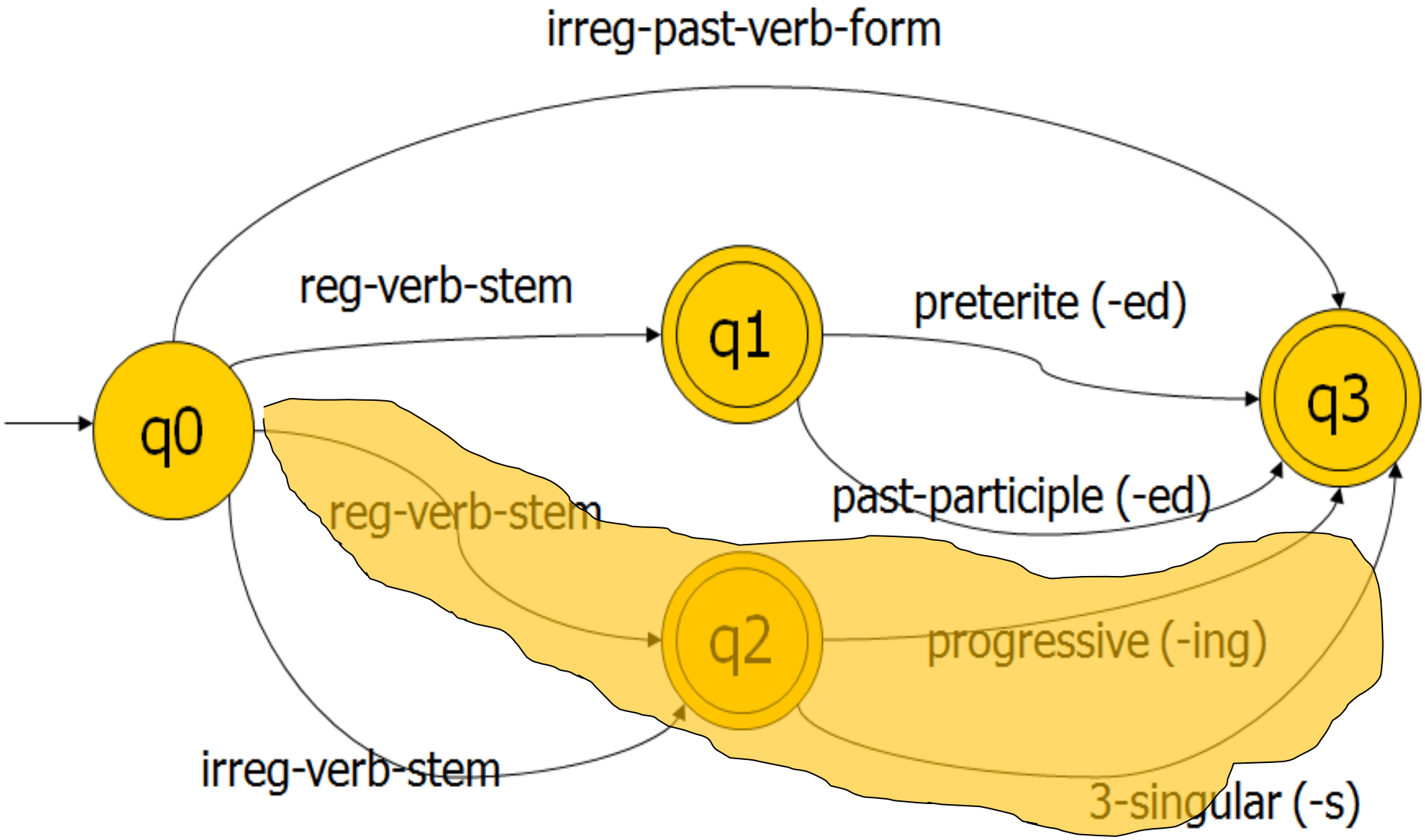
Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	Jump	Parse	Fry	Sob	Eat	Bring	Cut
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
–ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

reg-verb –progressive-

3 singular

Finite-State Automaton for Inflection of English Verbs



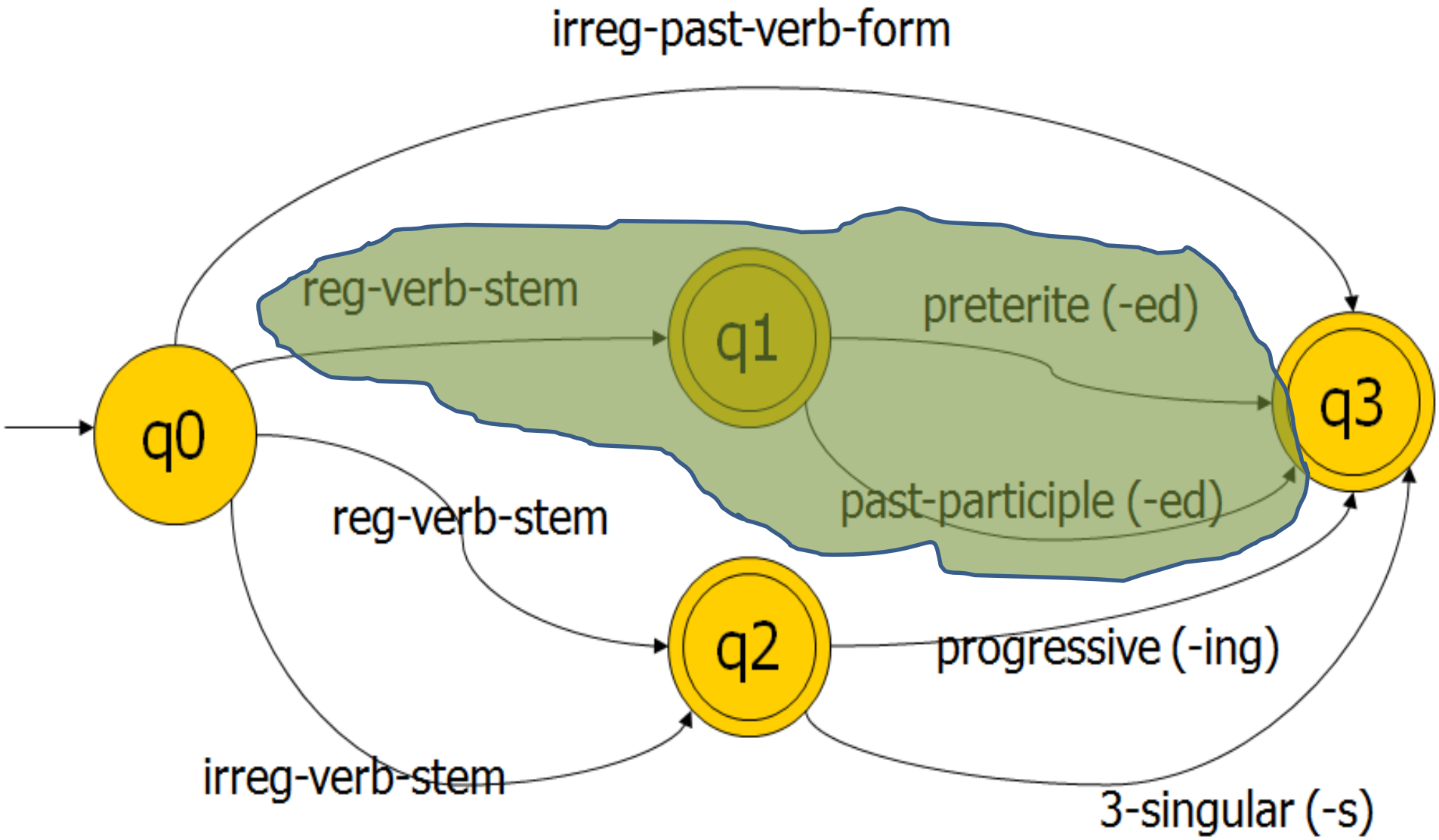
Finite-State Automaton for Inflection of English Verbs

Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	Jump	Parse	Fry	Sob	Eat	Bring	Cut
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
–ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

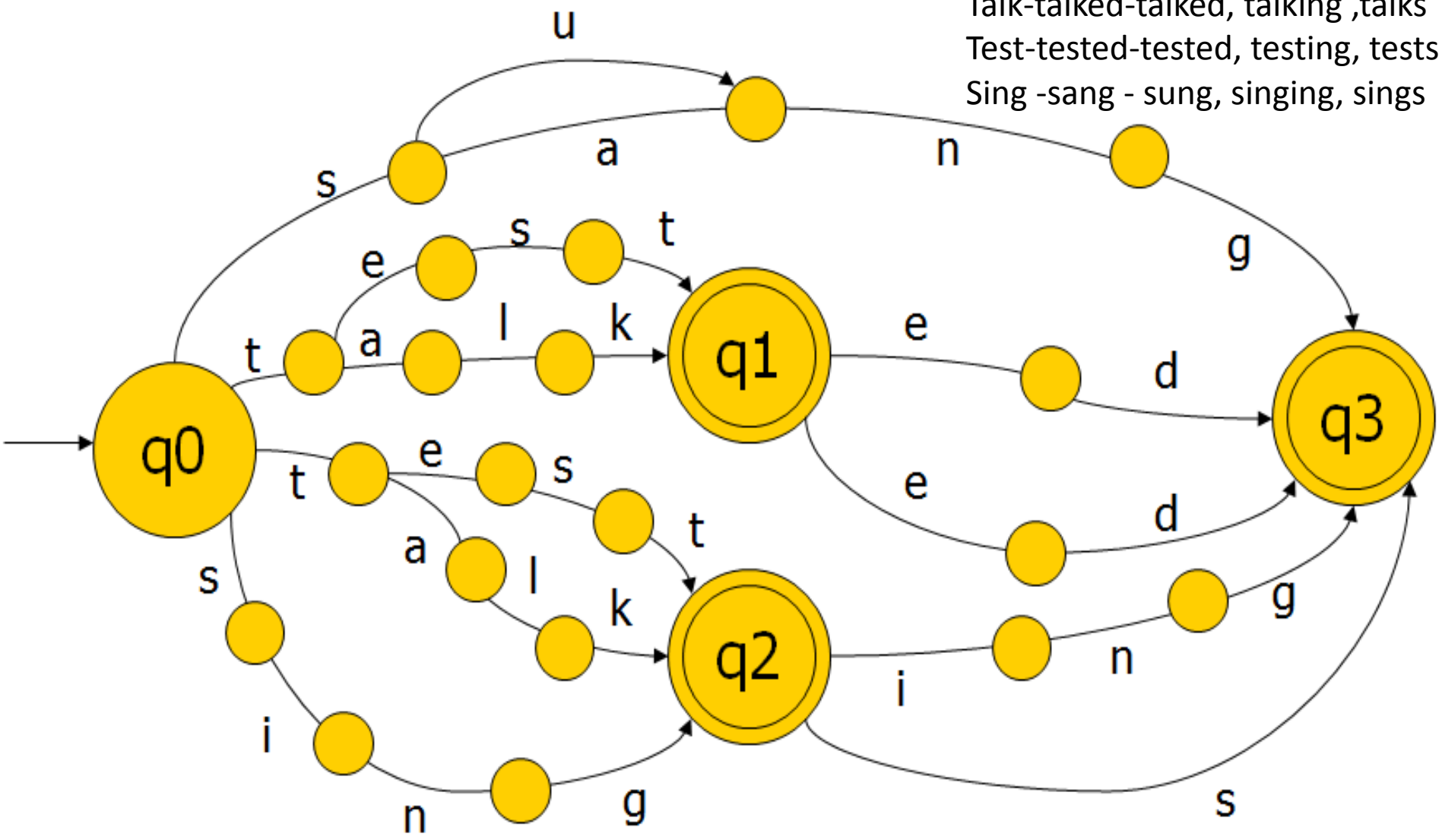
reg- past form

Finite-State Automaton for Inflection of English Verbs



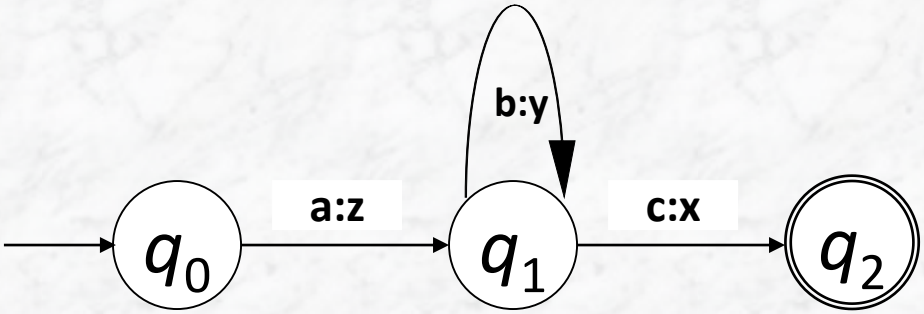
Finite-State Automaton for Inflection of the Verbs ’talk’, ’test’ and ’sing’

Talk-talked-talked, talking ,talks
Test-tested-tested, testing, tests
Sing -sang - sung, singing, sings



Finite State Transducer (FST)

- Transduce: to convert from a form to another
- The input symbol is transduced into the output symbol as a transition occurs on the arc
- FST accepts and generates string (to define **internal structure** of the word <Analysis>)



Accepts
abbbc
and generates
zyyyx

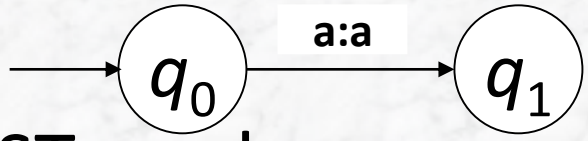
Finite State Transducer (Cont.)

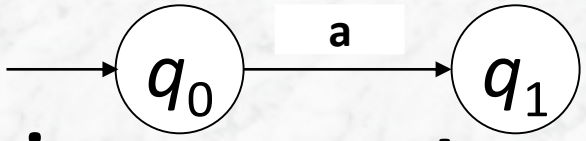
- Finite-state transducers have a formal definition, which is similar to that of finite state automata. A FST consists of five components $(Q, \Sigma, q_0, F, \delta)$ where:
  - Q is a finite set of states
  - Σ is a finite set of symbol or character pairs $i : o$, where i is a symbol of the input alphabet and o of the output alphabet
  - q_0 is the start state, $q_0 \in Q$
  - F is the set of final states, $F \subseteq Q$
  - δ is a relation from states and symbols to states.

    $\delta: Q \times \Sigma \times Q$

Finite-State Transducer

- A transducer maps between one set of symbols and another; a finite state transducer does this via a finite automaton.
- Where an FSA accepts a language stated over a finite alphabet of single symbols, e.g. Σ={*a*, *b*, *c*, ...}, an FST accepts a language stated over **pairs of symbols**, e.g. Σ={*a:a*, *b:b*, *a:c*, *a:*ε, ε:ε, ...}
- In two-level morphology, we call pairs like *a:a* **default pairs**, and refer to them by a single symbol *a*.





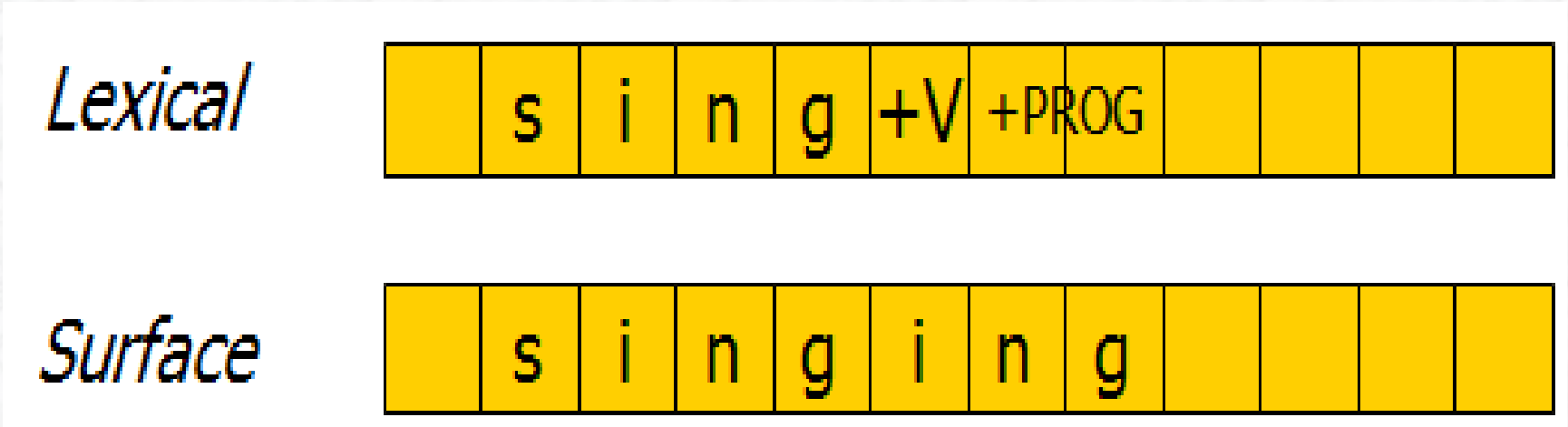
- An FST can be seen as a **recognizer**, **generator**, **translator** or a **set relator**.

Two-Level Morphology

- Two-level morphology represents a word as a correspondence between :-

  1- a **lexical level**, which represents a simple **concatenation of morphemes** making up a word

  2- the **surface level**, which represents the **actual spelling** of the final word.

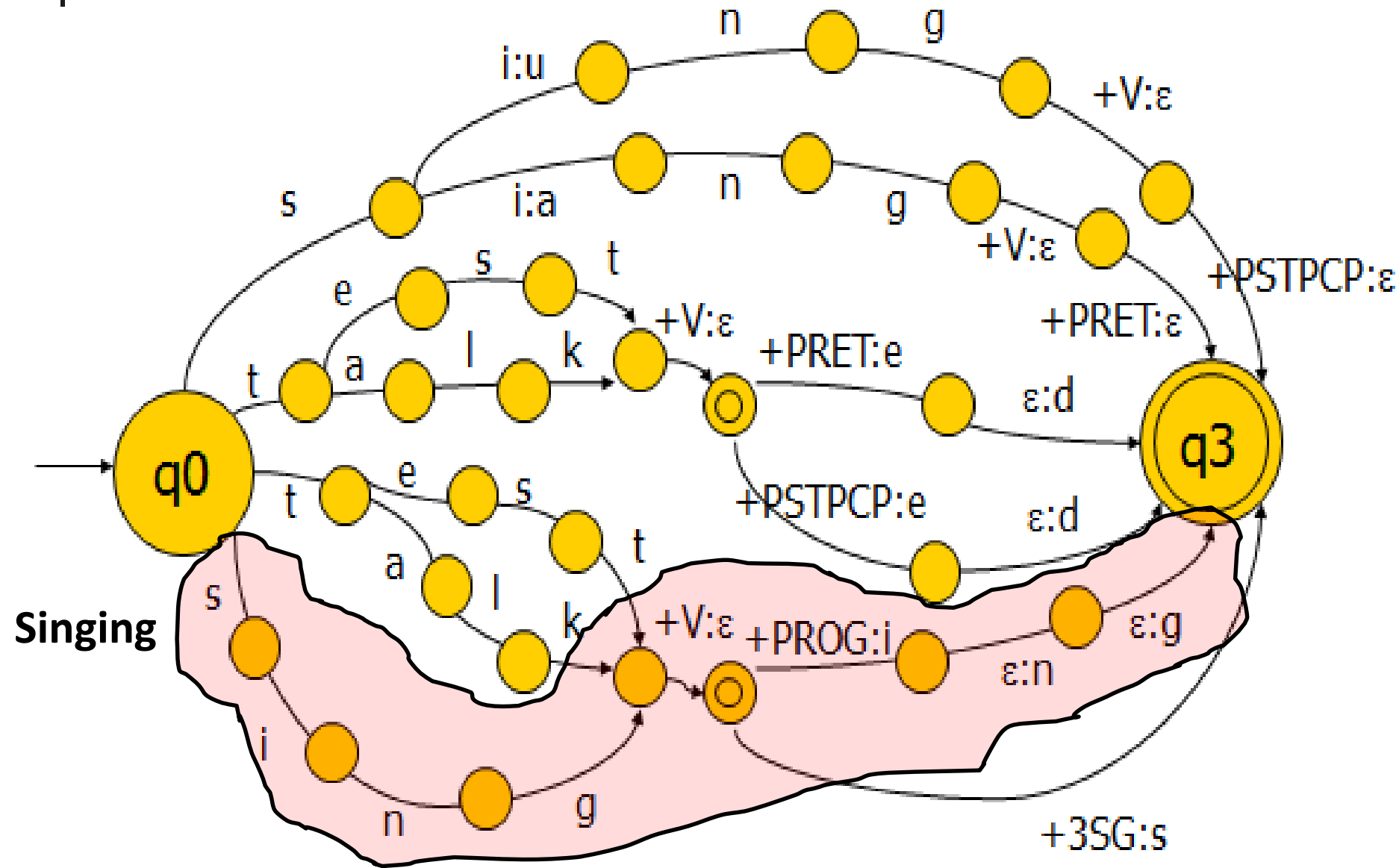


Examples

Lexical form	*Surface form*
sing +V +3SG	sings
test +V +PROG	testing
talk +V +PRET	talked
sing +V +PRET	sang
talk +V +PSTPCP	talked
sing +V +PSTPCP	sung

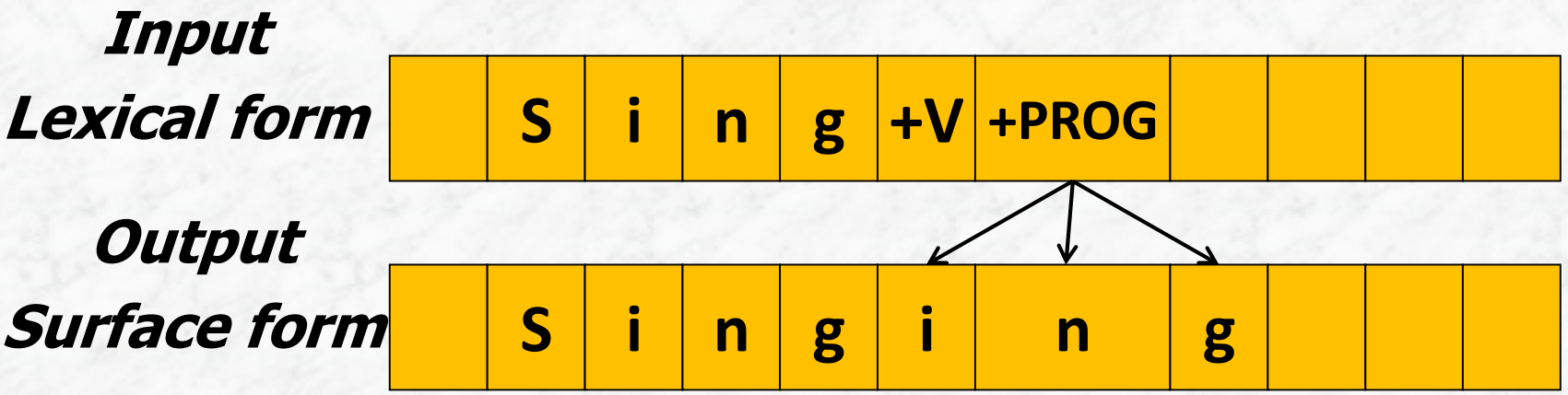
- ε epsilon ⇨ Do Nothing

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



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

Trace



Useful FST Operations

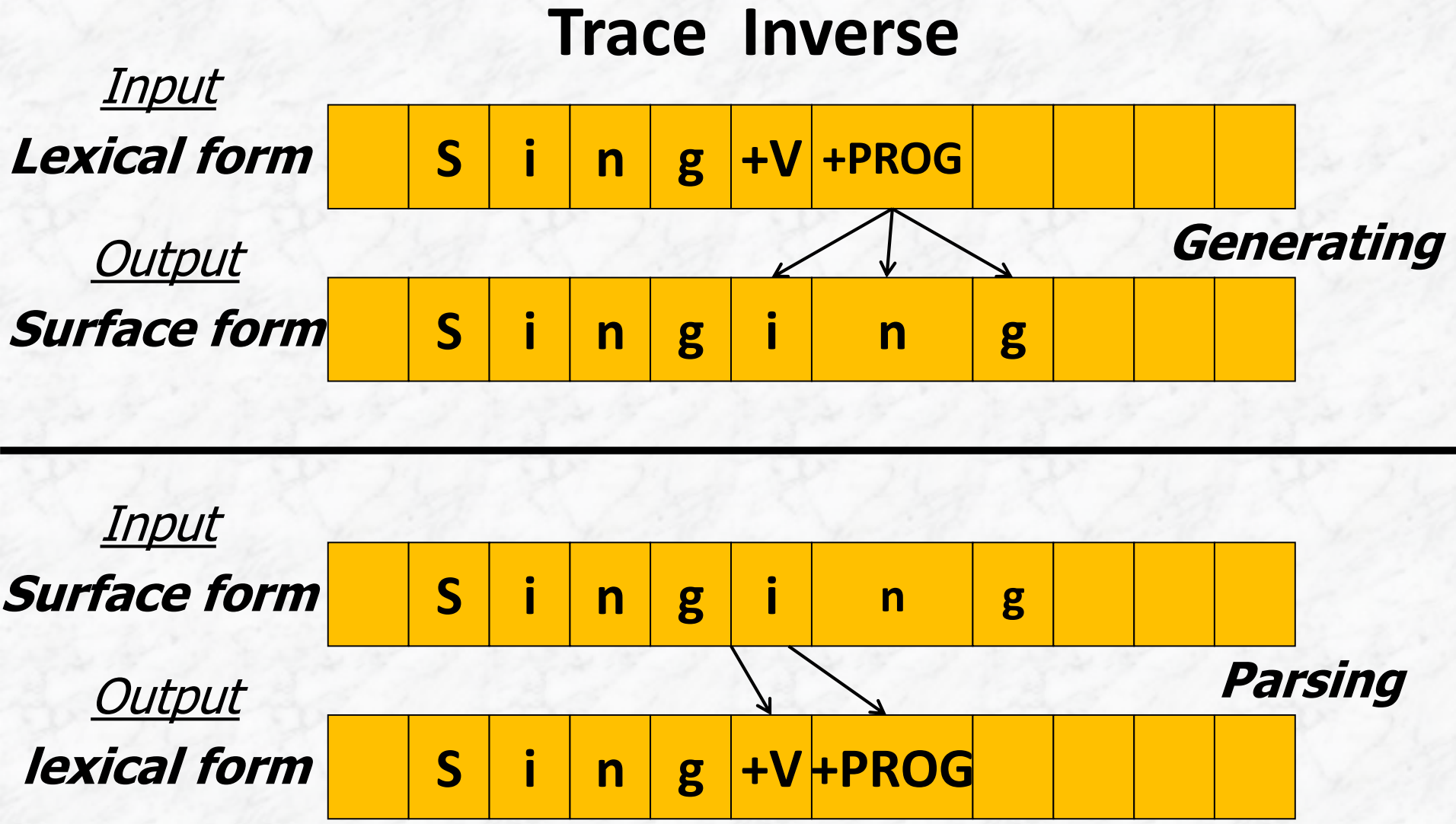
- **Inversion**: Switch input and output labels.

- e.g. $\Sigma(T)=\{a:b, c:d\} \Rightarrow \Sigma(inv(T))=\{b:a, d:c\}$
- This can be used to switch from generating words to parsing words.

- **Intersection**: Only sequences of pairs accepted by **both** transducer*T1* **and** transducer*T2* are accepted by transducer *T1^T2.*

- **Composition**: The output of transducer *T1* serves as input to *T2*. This is marked as ***T1 ° T2*** or ***T2(T1)***.

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



Finite-state transducers

A FST $T = L_{in} \times L_{out}$ *defines a relation between two regular languages* L_{in} *and* L_{out}*:*

L_{in} = { ***cat*** , ***cats*** , ***fox*** , ***foxes*** , ...}

L_{out} = {***cat+N+sg, cat+N+pl, fox+N+sg, fox+N+PL ...***}

T = { ***<cat, cat+N+sg>,***
<cats, cat+N+pl>,
<fox, fox+N+sg>,
<foxes, fox+N+pl> }

English spelling rules

English spelling (orthography) is funny:
The underlying morphemes (*plural-s, etc.) can have different* orthographic surface realizations (-s, -es)

Spelling changes at morpheme boundaries:
– E-insertion: fox +s = fox**es**
– E-deletion: mak**e +ing = making**

Intermediate representations

English plural -s: cat ⇒ cats dog ⇒ dogs

but: fox ⇒ foxes, bus ⇒ buses , buzz ⇒ buzzes

We define an intermediate representation which **captures**

- morpheme boundaries (^) and word boundaries (#).

Lexicon:	***cat+N+PL***	***fox+N+PL***
Intermediate representation:	***cat^s#***	***fox^s#***
Surface string:	***cats***	***foxes***

- *Intermediate-to-Surface Spelling Rule:*

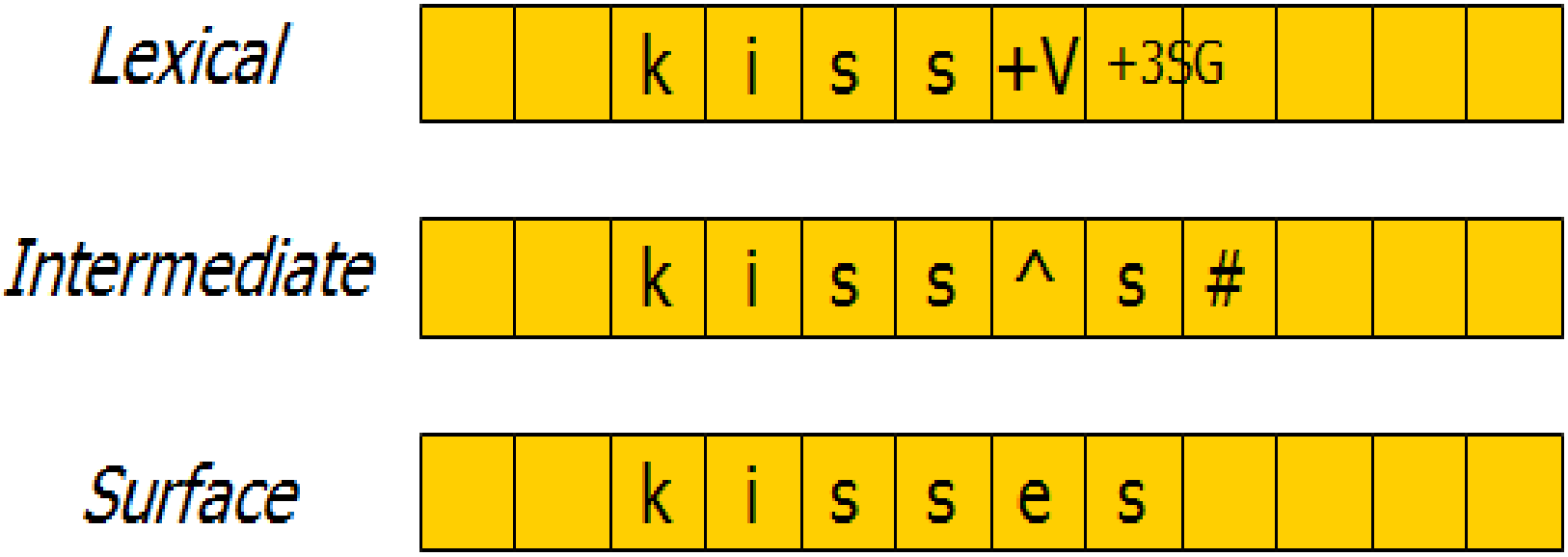
  If plural ***'s' follows a morpheme ending in 'x','z' or 's', insert 'e'.***

Spelling Rules and FSTs

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*, and to *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel + -c* add *-k*	panic/panicked

Three levels

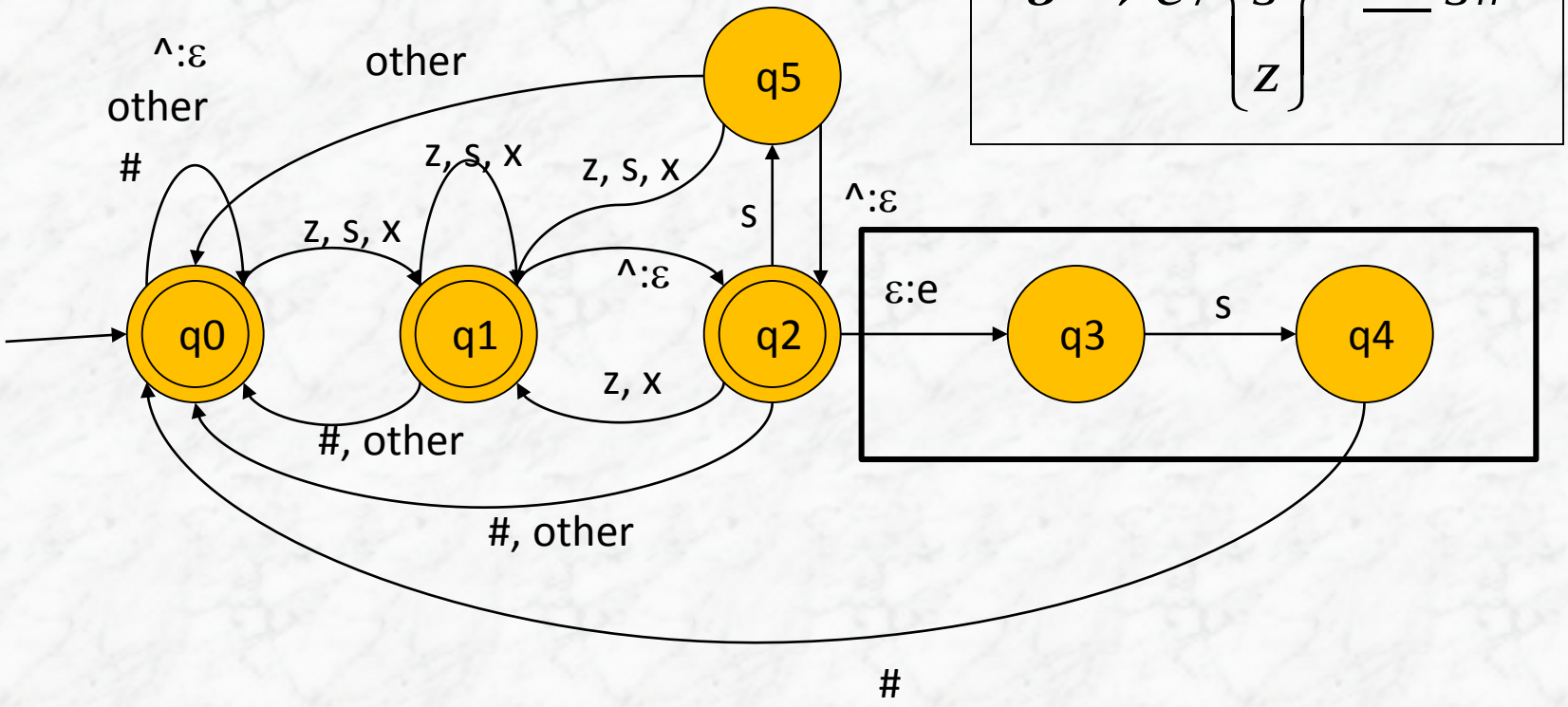
- Add an intermediate level between the lexical and surface levels



FST for the E-insertion Rule

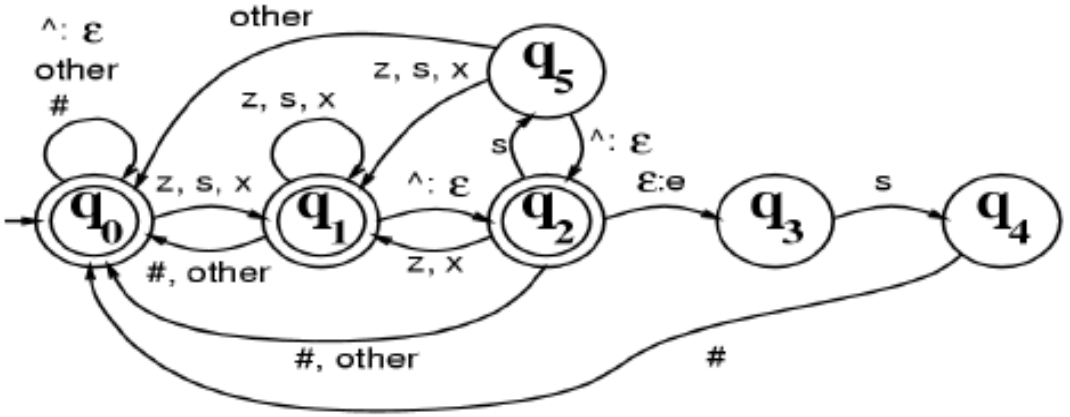
fox^s#

$$\varepsilon \rightarrow e / \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \wedge \_\_ \; s\#$$



The state-transition table for E-insertion rule

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



The state-transition table for E-insertion rule

f	o	x	^	s	#
f	o	x	e	s	#

Intermediate Tape

Surface Tape

- Trace:
  - generating foxes# from fox^s#:
    q0-f->q0-o->q0-x->q1-^:ε->q2-ε:e->q3-s->q4-#->q0
  - generating foxs# from fox^s#:
    q0-f->q0-o->q0-x->q1-^:ε->q2-s->q5-#->FAIL
  - generating salt# from salt#:
    q0-s->q1-a->q0-l->q0-t>q0-#->q0
  - parsing assess#:
    q0-a->q0-s->q1-s->q1-^:ε->q2-ε:e->q3-s->q4-s->FAIL

    q0-a->q0-s->q1-s->q1-e->q0-s->q1-s->q1-#->q0

State\Input	s:s	x:x	z:z	^:ε	ε:e	#	othe
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