Formal Language Theory & Finite State Automata

COMP90042

Natural Language Processing

Lecture 13

Semester 1 2022 Week 7

Jey Han Lau

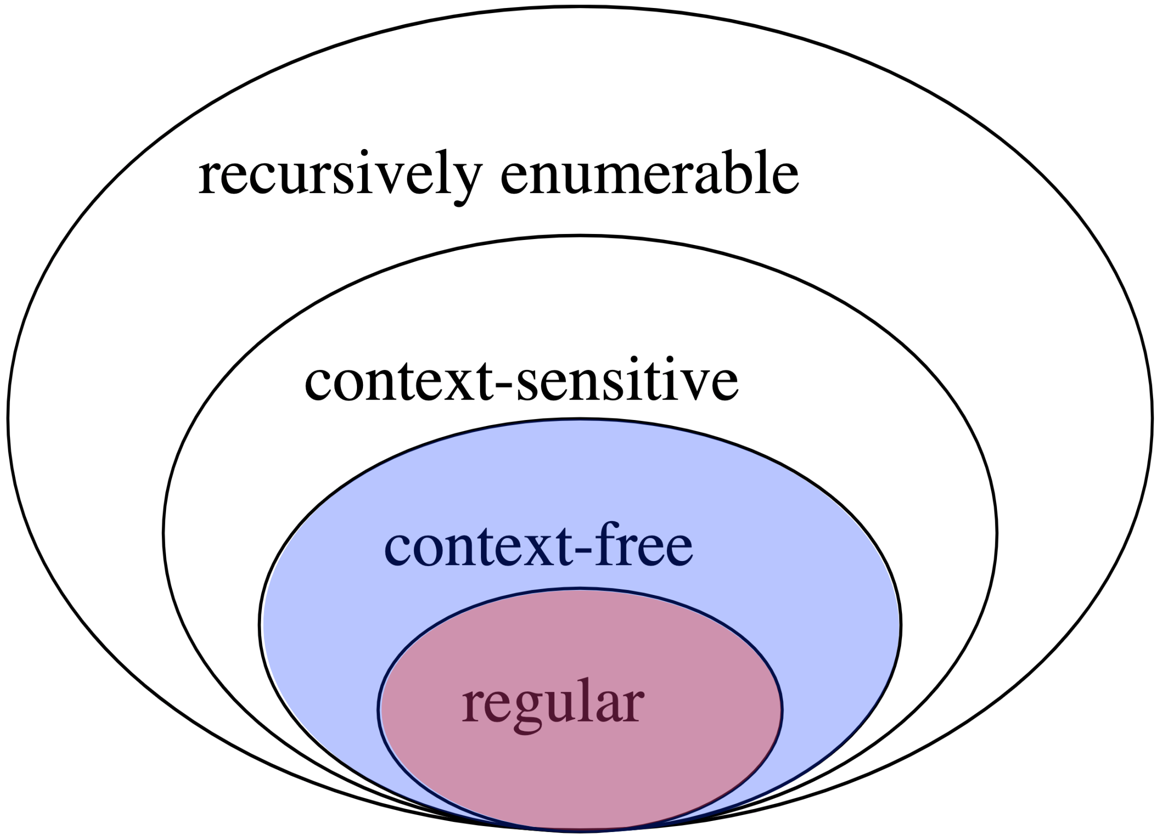
THE UNIVERSITY OF MELBOURNE

What Have We Learnt?

- Methods to process sequence of words:
  - N-gram language Model
  - Hidden Markov Model
  - Recurrent Neural Networks
- Nothing is fundamentally linguistic about these models

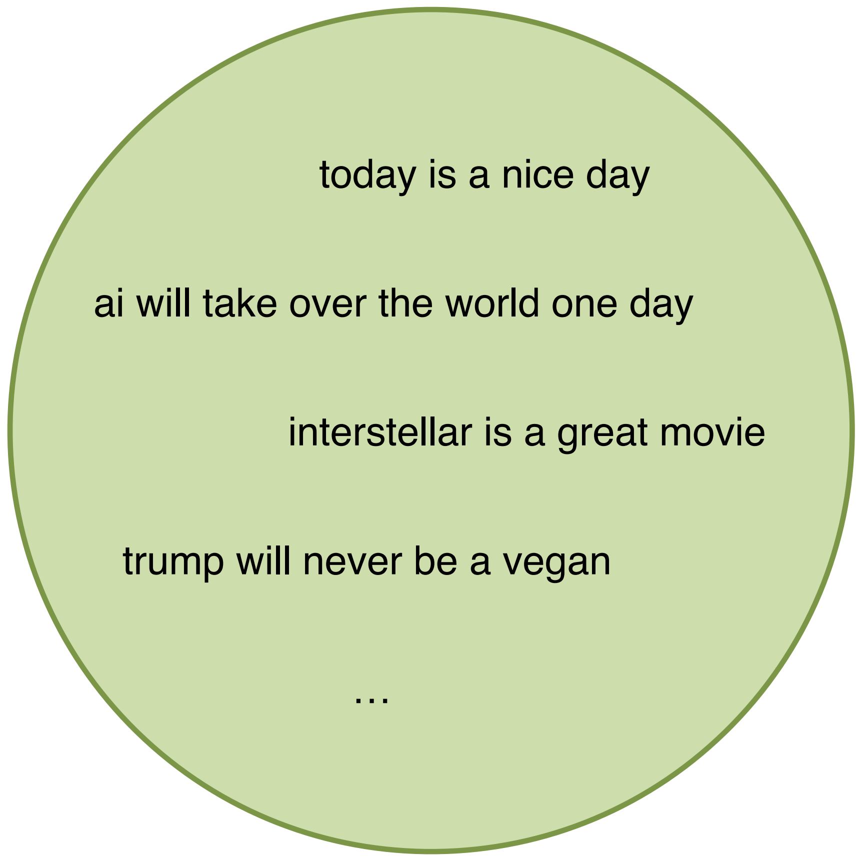
Formal Language Theory

- Studies **classes of languages** and their computational properties
  - Regular language (this lecture)
  - Context free language (next lecture)



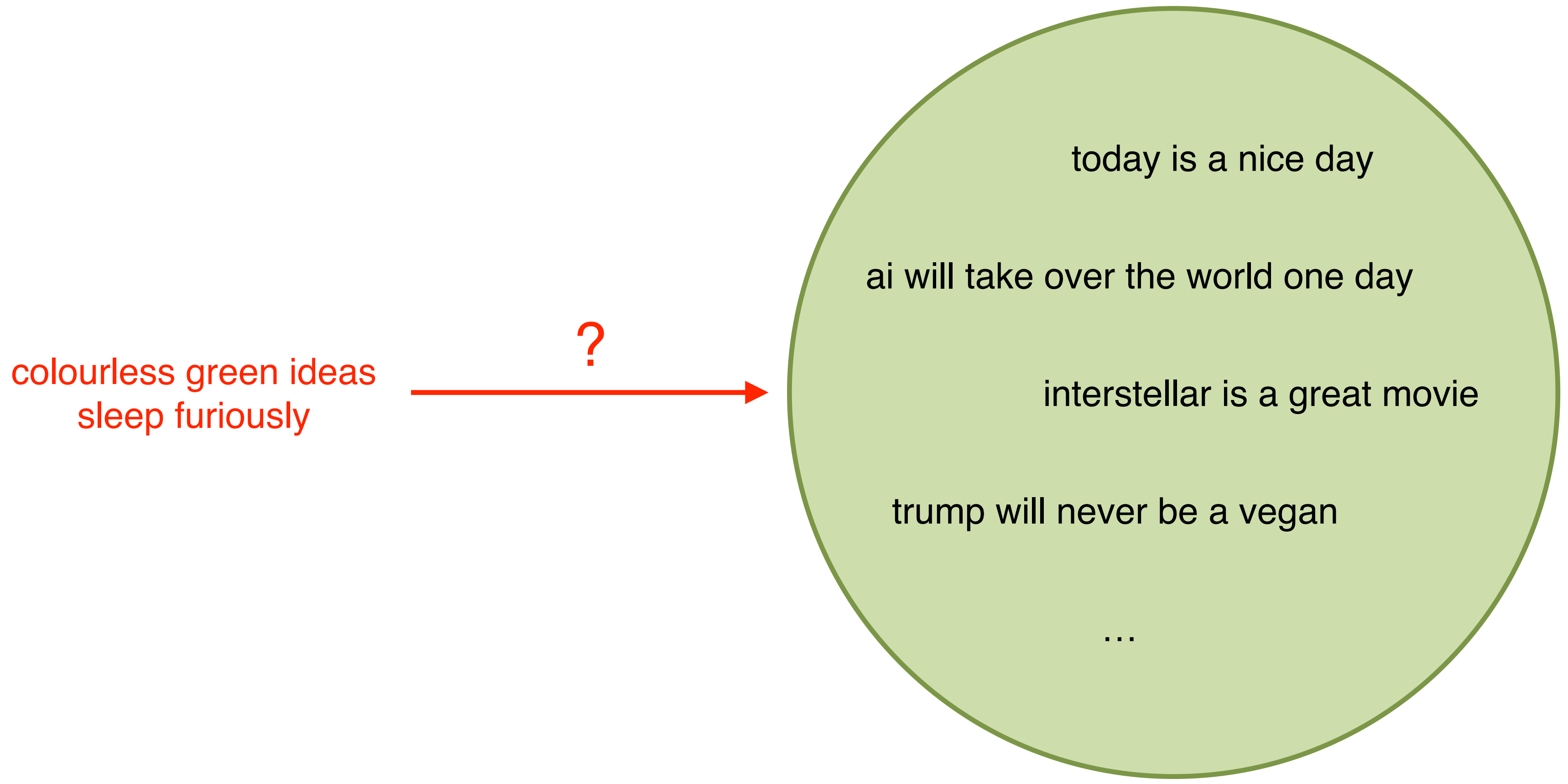
Formal Language Theory

- A language = set of **strings**
- A string = sequence of **elements** from a finite **alphabet** (AKA vocabulary)


today is a nice day
ai will take over the world one day
interstellar is a great movie
trump will never be a vegan
…

Why?

- Main goal is to solve the **membership problem**
  - Whether a string is in a language
- How? By defining its **grammar**


colourless green ideas sleep furiously
?
today is a nice day
ai will take over the world one day
interstellar is a great movie
trump will never be a vegan
…

Examples of Language

- Binary strings that start with 0 and end with 1
  - { 01, 001, 011, 0001, … } ✓
  - (1, 0, 00, 11, 100, … } ✗
- Even-length sequences from alphabet {*a*, *b*}
  - { *aa*, *ab*, *ba*, *bb*, *aaaa*, … } ✓
  - { aaa, aba, bbb, … } ✗
- English sentences that start with *wh*-word and end in *?*
  - { *what ?*, *where my pants ?*, … } ✓
  - { *hello how are you?*, *why is the dog so cute!* }

Beyond Membership Problem…

- Membership
  - Is the string part of the language? Y/N
- Scoring
  - Graded membership
  - “How acceptable is a string?” (language models!)
- Transduction
  - “Translate” one string into another (stemming!)

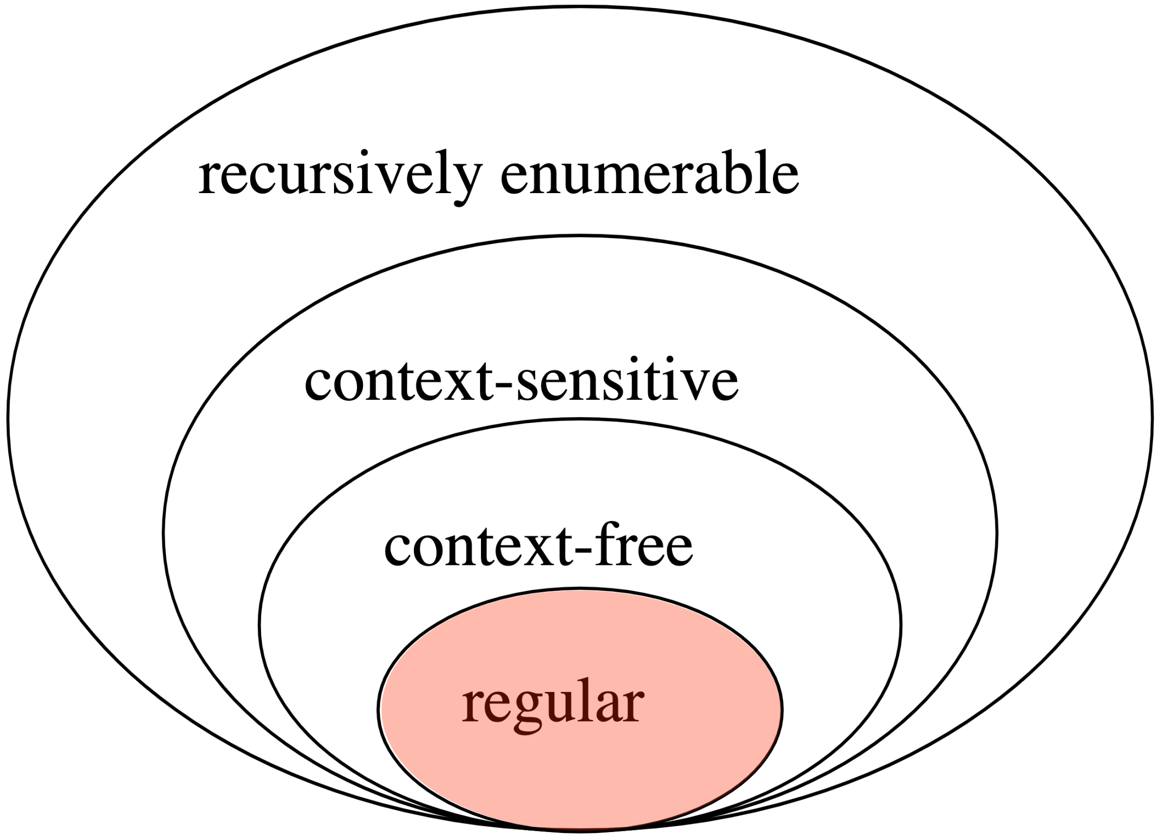
Outline

- Regular language
- Finite state acceptor
- Finite state transducer

Regular Language

Regular Language

- The simplest class of languages
- Any **regular expression** is a regular language
  - Describes what strings are part of the language (e.g. ‘0(0|1)*1’)



Regular Languages

- Formally, a regular expression includes the following operations/definitions:
  - Symbol drawn from alphabet, Σ
  - Empty string, ε
  - Concatenation of two regular expressions, RS
  - Alternation of two regular expressions, R|S
  - Kleene star for 0 or more repeats, R*
  - Parenthesis () to define scope of operations

Examples of Regular Languages

- Binary strings that start with 0 and end with 1
  - *0(0|1)*1*
- Even-length sequences from alphabet {*a*, *b*}
  - *((aa)|(ab)|(ba)|(bb))**
- English sentences that start with *wh*-word and end in *?*
  - *((what)|(where)|(why)|(which)|(whose)|(whom)) Σ* ?*

Properties of Regular Languages

- **Closure**: if we take regular languages L1 and L2 and merge them, is the resulting language regular?
- RLs are closed under the following:
  - **concatenation and union**
  - **intersection**: strings that are valid in both L1 and L2
  - **negation**: strings that are not in L
- Extremely versatile! Can have RLs for different properties of language, and use them together

Finite State Acceptor

Finite State Acceptor

- Regular expression defines a regular language
- But it doesn’t give an algorithm to check whether a string belongs to the language
- **Finite state acceptors** (FSA) describes the computation involved for membership checking

Finite State Acceptors

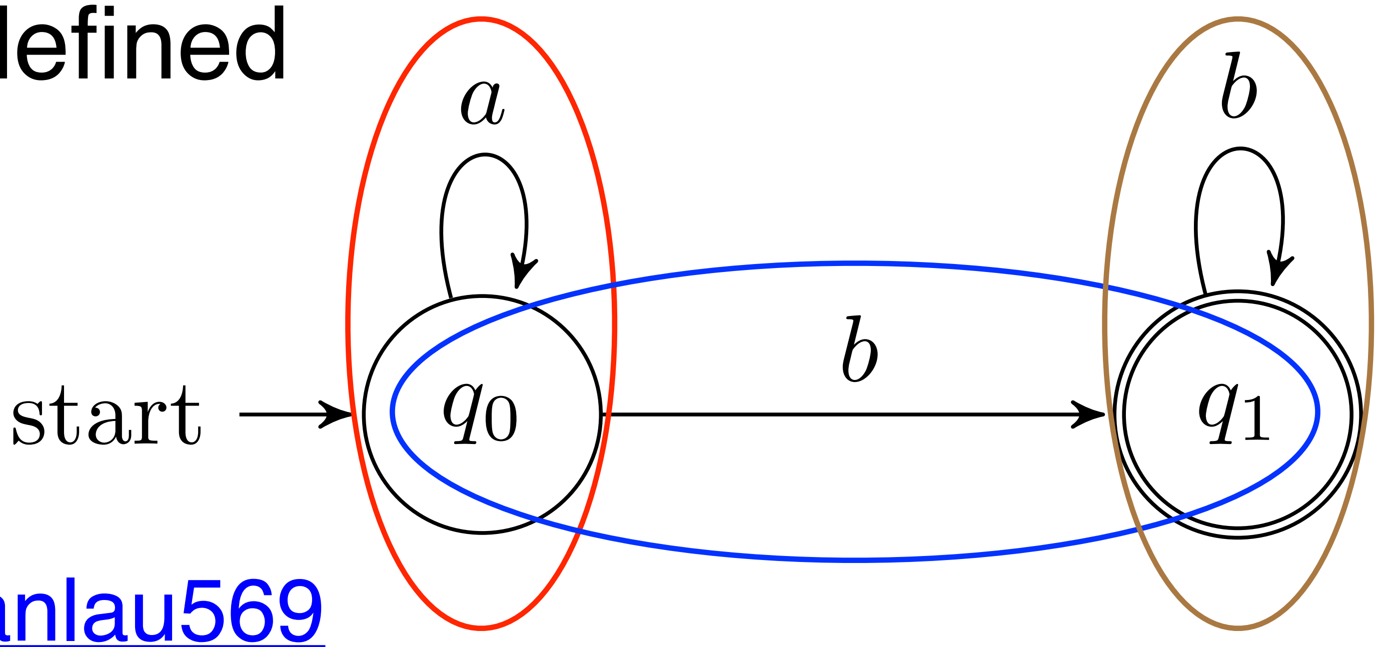
- FSA consists:
  - alphabet of input symbols, Σ
  - set of states, Q
  - start state, $q_0 \in Q$
  - final states, $F \subseteq Q$
  - transition function: symbol and state $\rightarrow$ next state
- Accepts strings if there is **path** from q_0 to a final state with transitions matching each symbol
  - Djisktra’s shortest-path algorithm, O(V log V + E)

Example FSA

abb ✓
aba ✗
aab ✓
b ✓
aaa ✗

• Input alphabet	{a, b}
• States	{q0, q1}
• Start, final states	q0, {q1}
• Transition function	{(q0,a) → q0, (q0, b) → q1, (q1,b) → q1}

- Regular expression defined by this FSA?



PollEv.com/jeyhanlau569

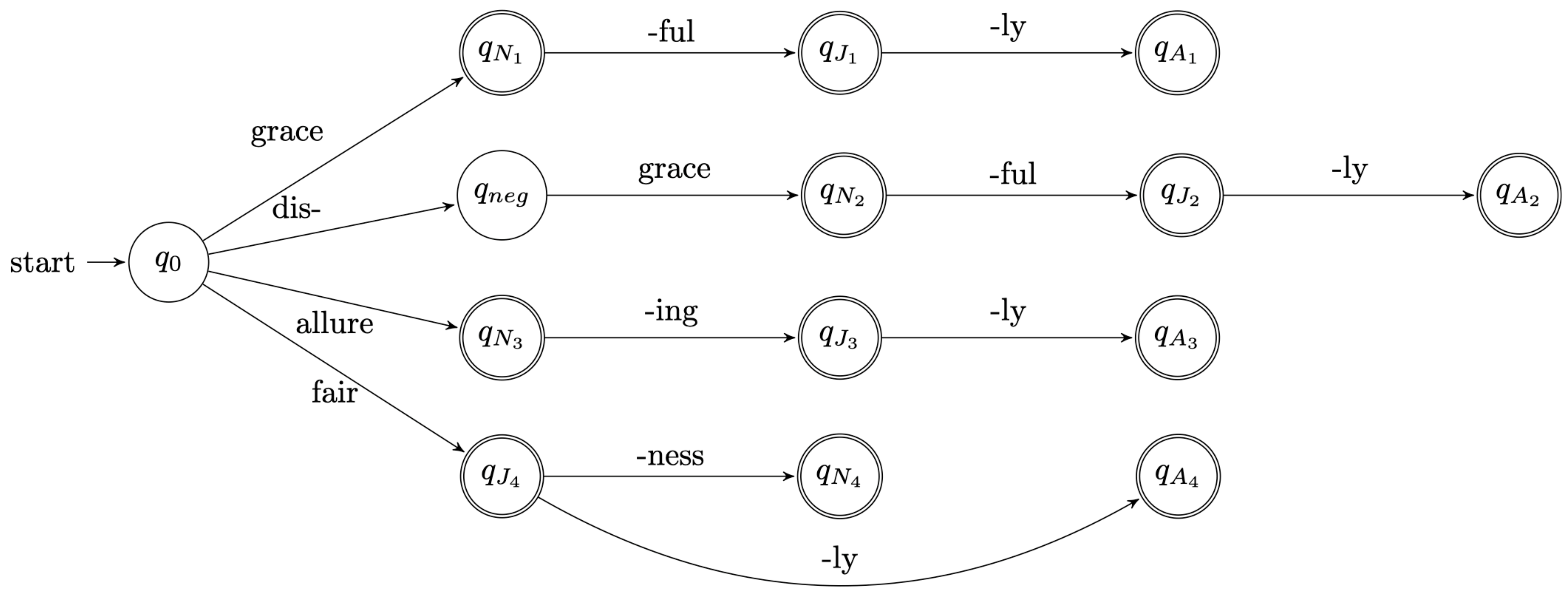
Derivational Morphology

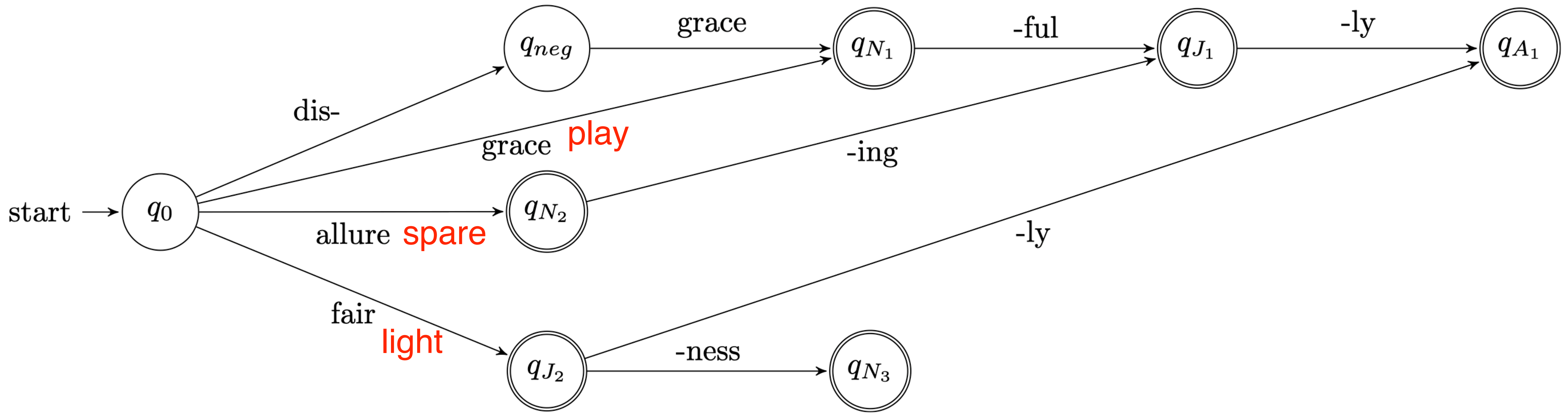
- Use of affixes to change word to another grammatical category
- *grace* → *graceful* → *gracefully*
- *grace* → *disgrace* → *disgracefully*
- *allure* → *alluring* → *alluringly*
- *allure* → **allureful*
- *allure* → **disallure*

FSA for Morphology

- Fairly consistent process
    - want to accept valid forms (*grace* → *graceful)*
    - reject invalid ones (*allure* → **allureful*)
    - generalise to other words, e.g., nouns that behave like *grace* or *allure*

FSA for Word Morphology





Weighted FSA

- Some words are more **plausible** than others
  - *fishful* vs. *disgracelyful*
  - *musicky* vs. *writey*
- Graded measure of acceptability — weighted FSA changes the following:
  - start state weight function, $\lambda: Q \rightarrow \mathbb{R}$
  - final state weight function, $\rho: Q \rightarrow \mathbb{R}$
  - transition function, $\delta: (Q, \Sigma, Q) \rightarrow \mathbb{R}$

WFSA Shortest-Path

- Total score of a path $\pi = t_1, \ldots, t_N$

$$\lambda(t_0) + \sum_{i=1}^{N} \delta(t_i) + \rho(t_N)$$

  - t is an edge

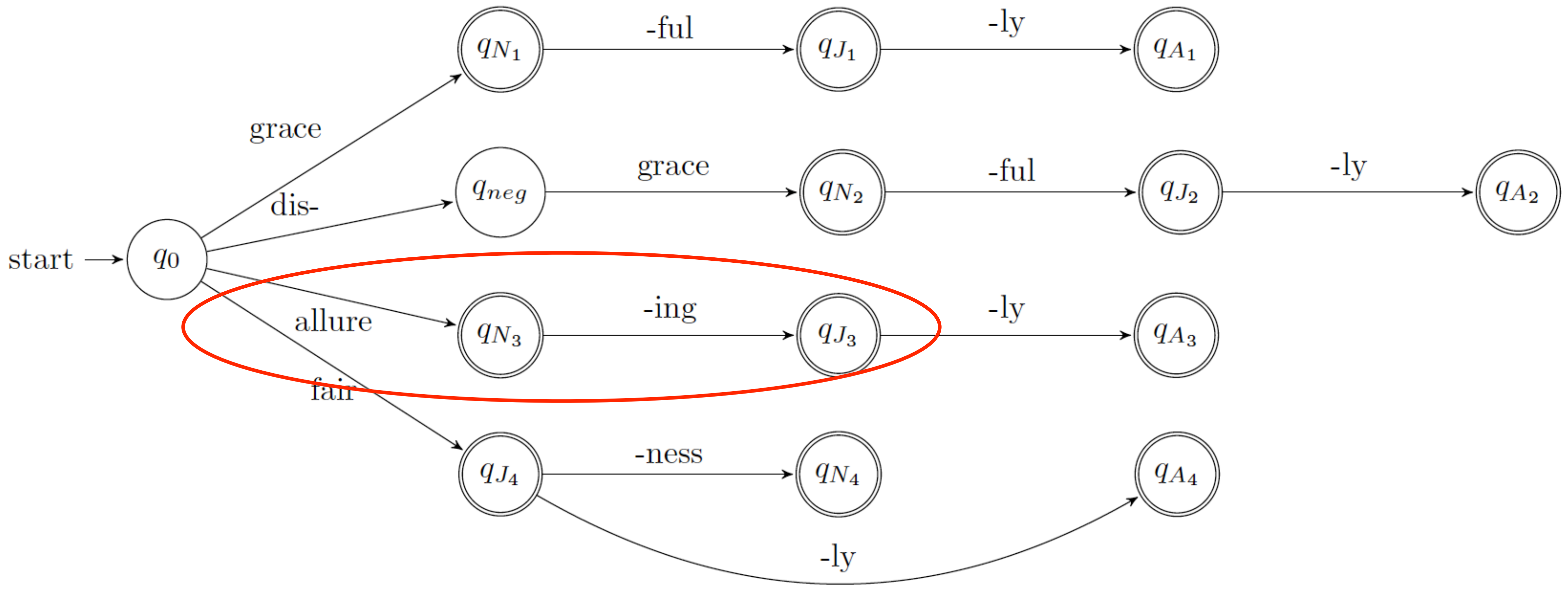
- Use **shortest-path algorithm** to find π with minimum cost
  - O(V log V + E), as before

Finite State Transducer

Finite State Transducers (FST)

- Often don't want to just accept or score strings
    - want to translate them into another string

FSA for Word Morphology



Finite state **acceptor**: allure + ing = allureing

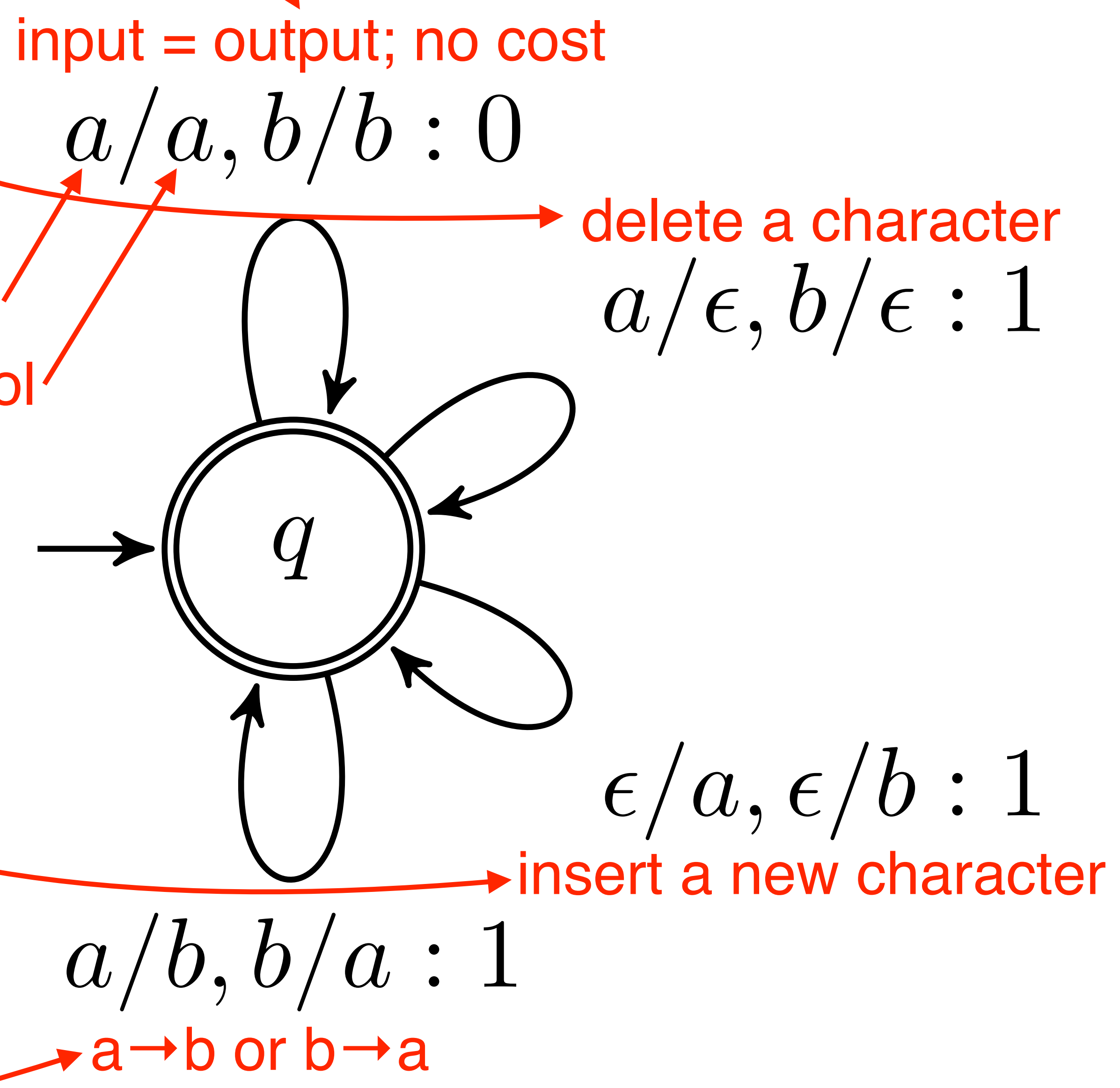
Finite state **transducer**: allure + ing = alluring

Finite State Transducer

- FST add string output capability to FSA
  - includes an **output alphabet**
  - transitions now take input symbol and **emit output symbol** (Q, Σ, Σ, Q)
- Can be **weighted** = WFST
  - Graded scores for transition
- E.g., edit distance as WFST
  - distance to transform one string to another

Edit Distance Automata

$$\delta(q, a, a, q) = \delta(q, b, b, q) = 0$$
$$\delta(q, a, b, q) = \delta(q, b, a, q) = 1$$
$$\delta(q, a, \epsilon, q) = \delta(q, b, \epsilon, q) = 1$$
$$\delta(q, \epsilon, a, q) = \delta(q, \epsilon, b, q) = 1.$$



ab → bb: 1

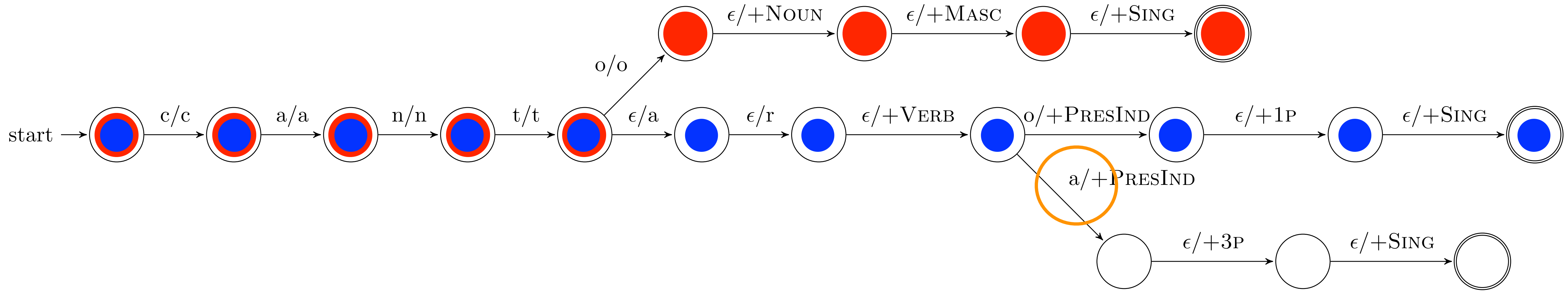
ab → aaab: 2

FST for Inflectional Morphology

- Verb inflections in Spanish must match the subject in person & number
- Goal of **morphological analysis**:
  - *canto* → *cantar*+VERB+present+1P+singular

	cantar	**to sing**
1P singular	yo canto	I sing
2P singular	tu cantas	you sing
3P singular	ella canta	she sings
1P plural	nostotros cantamos	we sing
2P plural	vosotros cantáis	you sing
3P plural	ellas cantan	they sing

FST for Spanish Inflection



canto →

- *canto*+Noun+Masc+Sing
- *cantar*+Verb+PresInd+1P+Sing

canta → *cantar*+VERB+PresInd+3P+Sing

Is natural language regular?

- Yes
- No
- Maybe
- Don’t know

PollEv.com/jeyhanlau569

Sometimes…

- Example:

  the mouse that ran.

  the cat that killed the mouse that ran.

  …

  the lion that bullied the hyena that bit the dog that chased the cat that killed the mouse that ran

  …

- Length is unbounded, but structure is local
  - (Det Noun Prep Verb)*
  - Can describe with FSA

Non-Regular Languages

- Arithmetic expressions with balanced parentheses
  - (a + (b x (c / d)))
  - Can have arbitrarily many opening parentheses
  - Need to remember how many open parentheses, to produce the same number of closed parentheses
  - Can’t be done with finite number of states
- $a^n b^n$

Center Embedding

- Center embedding of relative clauses
  - The cat loves Mozart
  - The cat the dog chased loves Mozart
  - The cat the dog the rat bit chased loves Mozart
  - The cat the dog the rat the elephant admired bit chased loves Mozart
- Need to remember the *n* subject nouns, to ensure *n* verbs follow (and that they agree etc)
- Requires (at least) **context-free grammar** (next lecture!)

Summary

- Concept of a language
- Regular languages
- Finite state automata: acceptors, transducers
- Weighted variants
- Application to edit distance, morphology

Reading

- E18, Chapter 9.1