Formal Language Theory & Finite State Automata

COMP90042

Natural Language Processing

Lecture 13

What is a Language?

- Methods to process sequence of symbols:
  - Language Model
  - Hidden Markov Model
  - Recurrent Neural Networks
- Nothing is fundamentally linguistic about these models

Formal Language Theory

- A language = set of **strings**
- A string = sequence of **elements** from a finite **alphabet**

Motivation

- Formal language theory studies **classes** of languages and their computational properties
  - Regular language (this lecture)
  - Context free language (next lecture)
- Main goal is to solve the **membership problem**
  - Whether a string is in a language or not
- How? By defining its **grammar**

Examples

- 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 ?*, … } ✓

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!)

Overview

- Regular languages
- Finite state acceptors & transducers
- Modelling word morphology

Regular Languages

- Regular language: the simplest class of languages
- Any **regular expression** is a regular language
- The regular expression itself is the **grammar**
  - Evaluates whether a string is part of the language

Regular Languages

- Formally, a regular expression includes the following operations:
  - 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 — follows from definition
  - 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
  - core algorithms will still apply

Finite State Acceptors

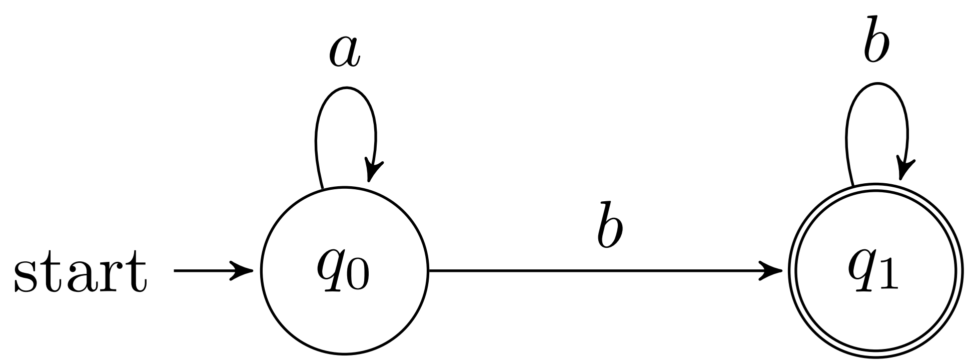
- Regular expression defines a regular language
- But it doesn’t give an algorithm to check whether a string belongs to the language
- **Finite state acceptors** (FSA) describes the computation involved for membership checking

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

- Input alphabet {a, b}
- States {q0, q1}
- Start, final states q0, {q1}
- Transition function {(q0,a) → q0, (q0, b) → q1, (q1,b) → q1}
- Note: seeing a in q1 results in failure
- Accepts *a*bb**



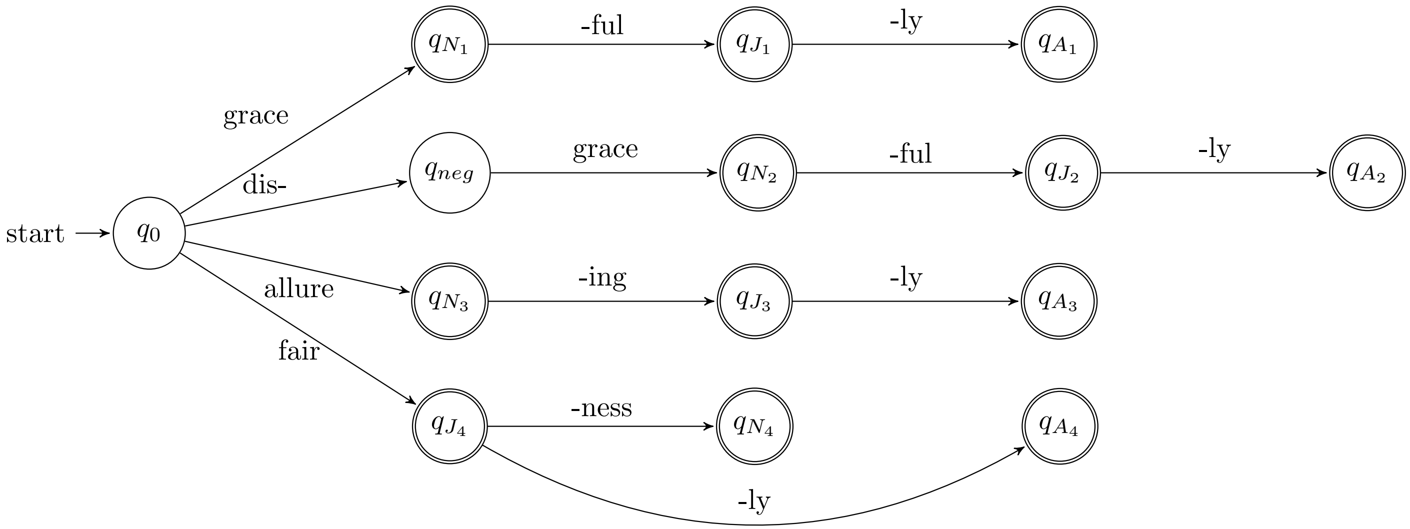
Derivational Morphology

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

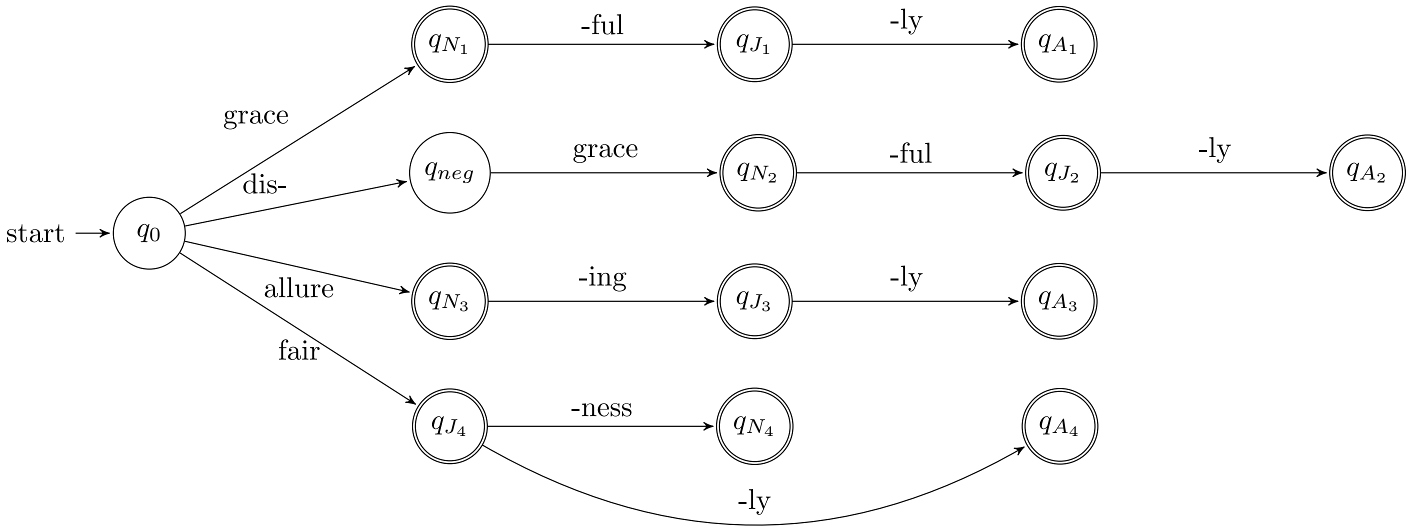
FSA for Morphology

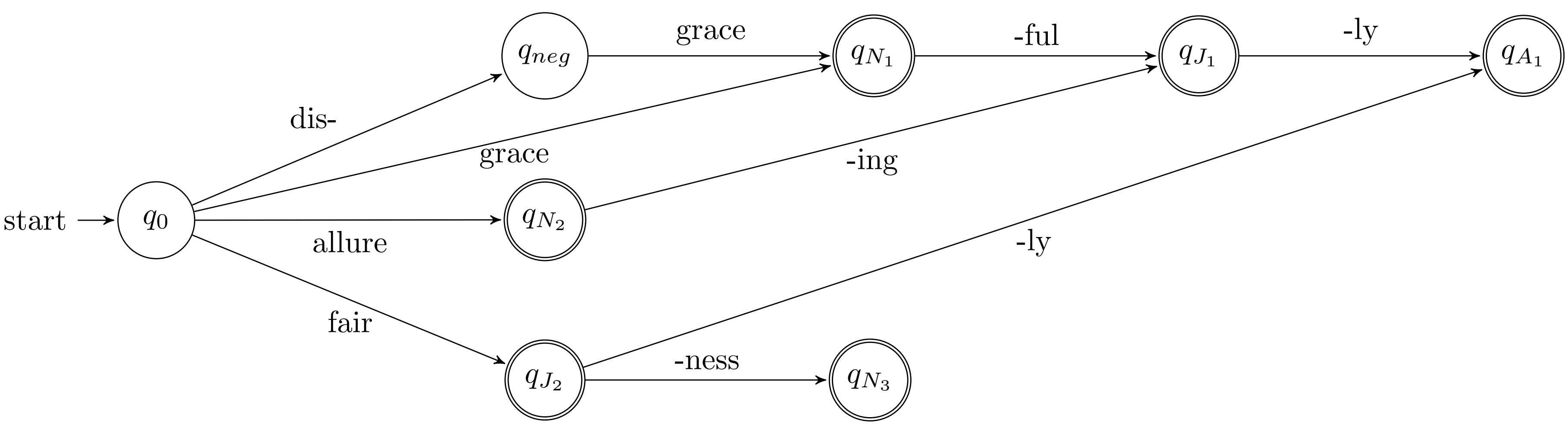
- (Fairly) consistent process— can we describe this as a regular language?
  - want to accept valid forms, and reject invalid ones [flagged with *]
  - generalise to other words, e.g., nouns that behave like *grace* or *allure*

FSA for Word Morphology



FSA for Word Morphology





Weighted FSA

Weighted FSA

- Some words are more **possible** than others
  - *fishful* vs. *disgracelyful*
  - *musicky* vs. *writey*
- Graded measure of acceptability — weighted FSA adds/changes the following:
  - set of states Q
  - alphabet of input symbols Σ
  - 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 $\pi = t_1, \ldots, t_N$ now

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

each *t* is an edge, so more formally using from &/or to states and edge label in score calculation

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

N-gram LMs as WFSA

- Recall LM calculates score of string as follows

$$p(w_1, \ldots, w_M) \approx \prod_{m=1}^{M} p_n(w_m \mid w_{m-1}, \ldots, w_{m-n+1}).$$

- Unigram language model:
  - One state: q_0
  - State transition score: $\delta(q_0, \omega, q_0) = \log p_1(\omega)$
  - Initial and final state scores = 0
  - Path score for $w_1, w_2, \ldots, w_M$:

$$0 + \sum_{m}^{M} \delta(q_0, w_m, q_0) + 0 = \sum_{m}^{M} \log p_1(w_m).$$

Bigram LM

- Bigram sentence probability:

$$P(w_1, w_2, .. w_M) = \prod_{i=1}^{M} P(w_i \mid w_{i-1}) \quad \textit{(bigram)}$$

- Implemented as WFSA
  - Σ = set of word types
  - Q = Σ (no. of states = no. of word types)

$$\delta(q_i, \omega, q_j) = \begin{cases} \log \Pr(w_m = j \mid w_{m-1} = i), & \omega = j \\ -\infty, & \omega \neq j \end{cases}$$

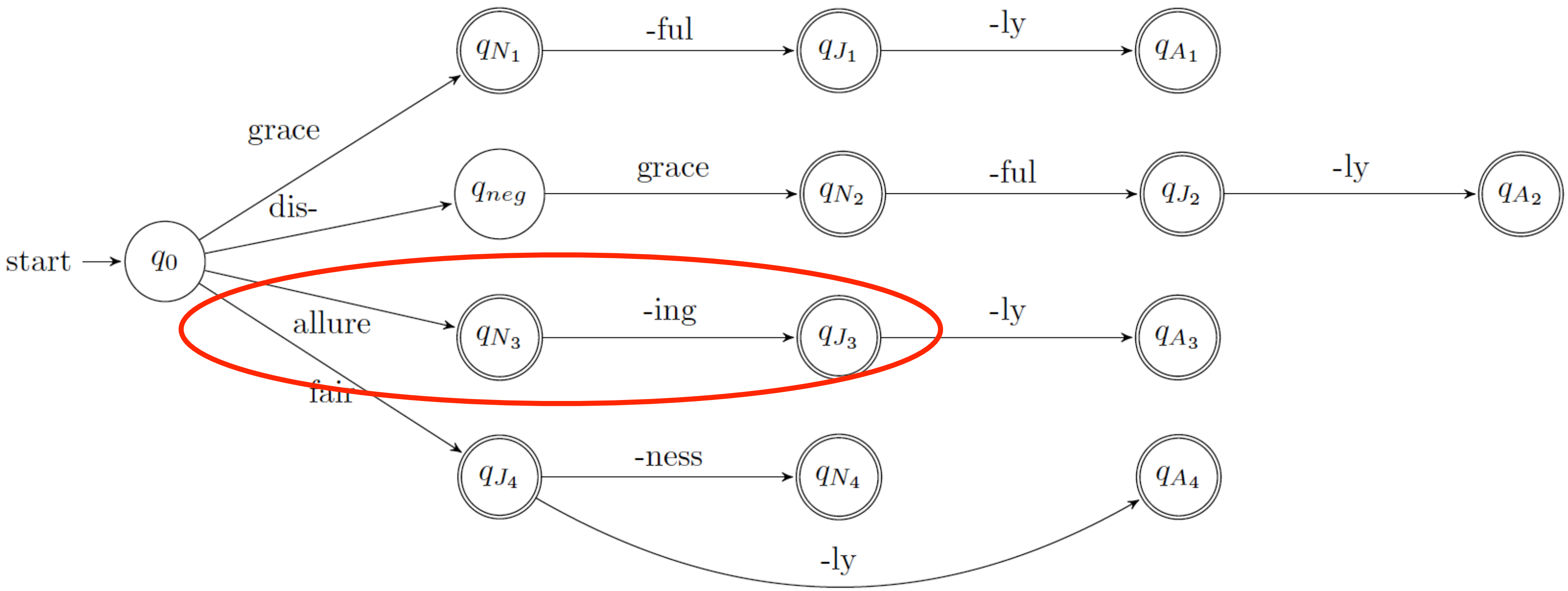
$$\lambda(q_i) = \log \Pr(w_1 = i \mid w_0 = \square)$$
$$\rho(q_i) = \log \Pr(w_{M+1} = \blacksquare \mid w_M = i).$$

Finite State Transducer

Finite State Transducers (FST)

- Often don’t want to just accept or score strings
  - want to translate them into another language, correct grammar, parse their structure, etc

FSA for Word Morphology



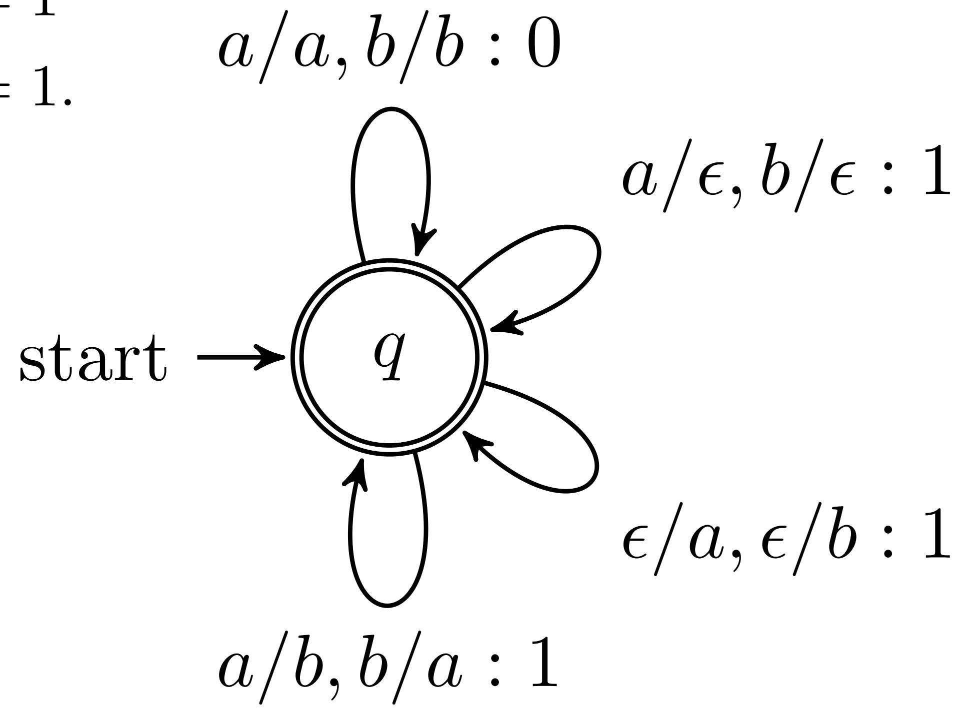
FSA: allure + ing = allureing
FST: allure + ing = alluring

Finite State Transducers

- FST add string output capability to FSAs
  - includes an *output alphabet*
  - and 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** which takes one string, and outputs the other
  - zero cost only if strings are identical

Edit Distance Automata

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



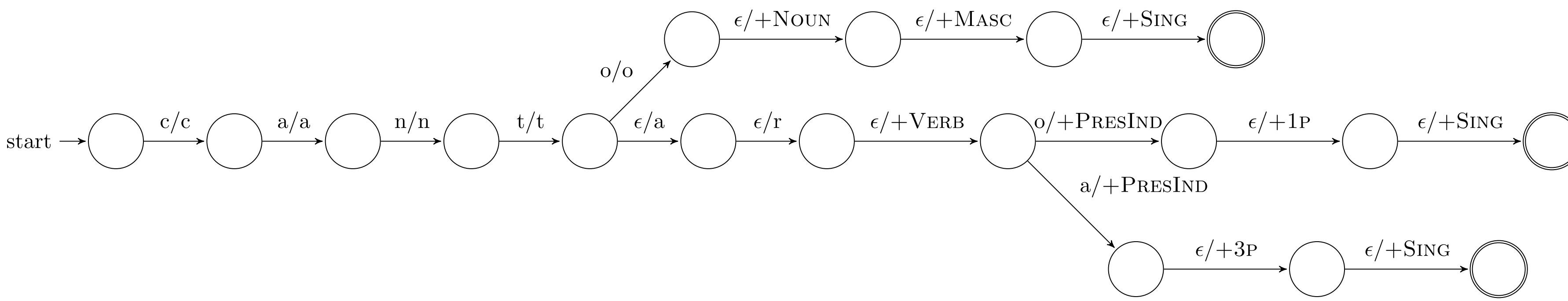
ab → bb: 1
ab → aaab: 2

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

canto → *cantar*+Verb+PresInd+1P+Sing

FST Composition

- Compose two FSTs by taking output of one FST, T1, and giving this as input to FST T2
  - denoted T1 ○ T2; and results in another FST
  - can also compose FST with FSA, resulting in a FST

FST Composition Example

- Allows development of different processes as FSTs:
  - *-ed* added to signal past tense in English
  - *cook* → *cooked*, *want* → *wanted*
  - But when the word ends with '*e*', then we want to avoid producing a spelling with consecutive *e*'s (*bakeed*)
  - Solution: build 2 FSTs
  - FST T1: bake+PAST → bake+ed
  - FST T2: bake+ed → baked

Is Natural Language Regular?

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 (*recursive*), but structure is local → can describe with FSA = Regular
- (Det Noun Prep Verb)*

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, and grammar
- Regular languages
- Finite state automata: acceptors, transducers
- Closure properties
- Weighted variants & shortest path inference
- Application to edit distance, morphology

Reading

- Reading
  - E18, Chapter 9.1