Formal Language Theory & Finite State Automata

COMP90042 Lecture 16

THE UNIVERSITY OF MELBOURNE

Figures & examples from Ch 9, Eisenstein, 2018, draft textbook

Overview

- Languages and grammars
- Regular languages
- Finite state acceptors & transducers
- Modelling word morphology

What is a "language"?

- How to characterize valid English?

  ** were I earliest thoughtlyless?*
  # colourless green ideas sleep furiously
  It was the best of times, it was the worst of times,…

- Language = set of acceptable strings (e.g., sentences)
- Some examples
  - binary strings of that start with 0 and end with 1
    01, 001, 011, 0001, 0011, 0101, 0111, …
  - sentences of English words that start with wh-word and end in ?
    what ?, what is a dog ?, where are my pants ?, …
  - strings {a, b} of even length
  - HTML; your favourite programming language; …

Formal Language Theory

- Used to define membership:
  - is given string part of the language or not?
- Formal apparatus to answer this question automatically, using a **grammar**
- Today: regular languages
- Coming: context-free languages

Key Operations

- Membership
  - is the string part of the language? Y/N
- Scoring (requires weighting)
  - relax question to graded membership, how good an example of language is the string? (returning a number)
- Transduction
  - input one string, output another
  - a form of translation, but used extensively
    e.g., tagging = translating from words to tags

Regular Languages

- Regular languages the simplest class of languages
- Accepted by **regular expression** which supports 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

- Set of strings starting with 0 and ending in 1, with alphabet {0,1}
  - *0(0|1)*1*
- Question sentences: strings that start with wh-word, end in ?
  - *((what)|(where)|(why)|(which)|(whose)|(whom)) Σ* ?*
- Even length strings in {a, b}
  - *((aa)|(ab)|(ba)|(bb))**
- In practice regex libraries include several shortcuts

Properties of Regular Languages

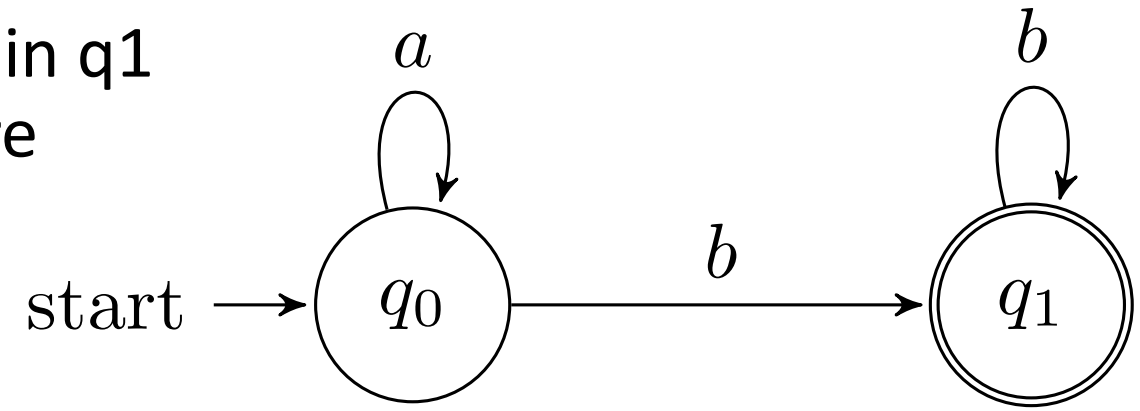
- Closure: if we take regular languages L1 and L2 and merge them, is the resulting language regular?
- RLs are closed under the following:
  - concatenation and union — follows from definition
  - intersection: strings that are valid in both L1 and L2
  - negation: strings that are not in L
- Extremely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

Finite State Acceptors

- RLs implemented by **finite state acceptors,** defined as:
  - 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 → 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

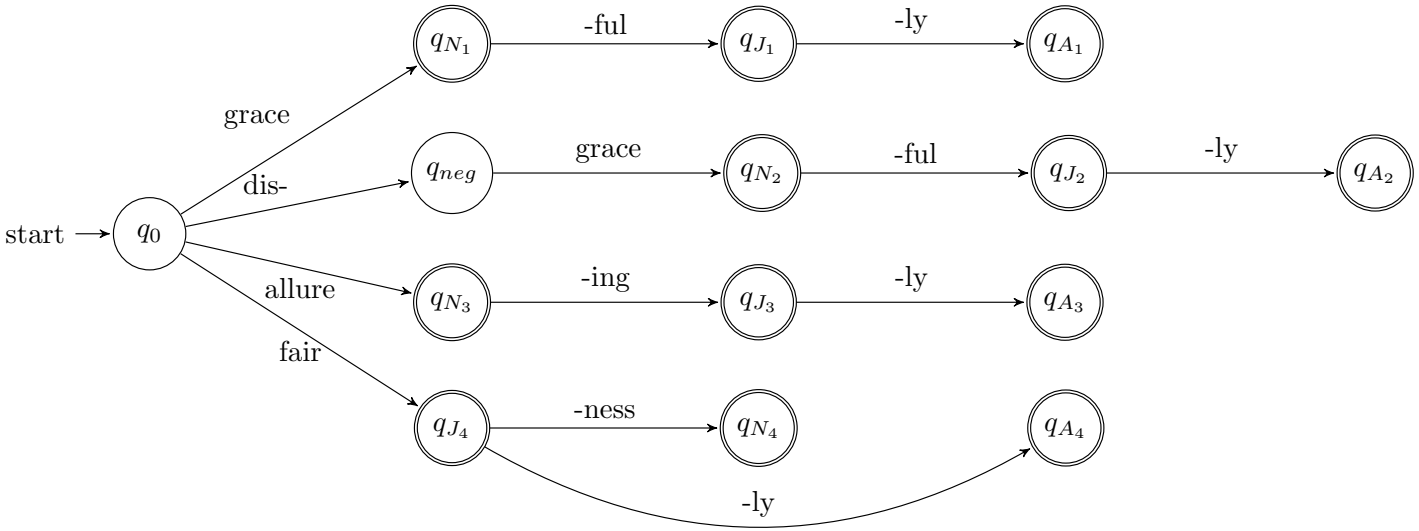
- Input alphabet {a, b}
- States {q0, q1}
- Start, final states q0, {q1}
- Transition function {(q0,a) → q0, (q0, b) → q1, (q1,b) → q1}
- Note: seeing a in q1 results in failure
- Accepts *a*bb**

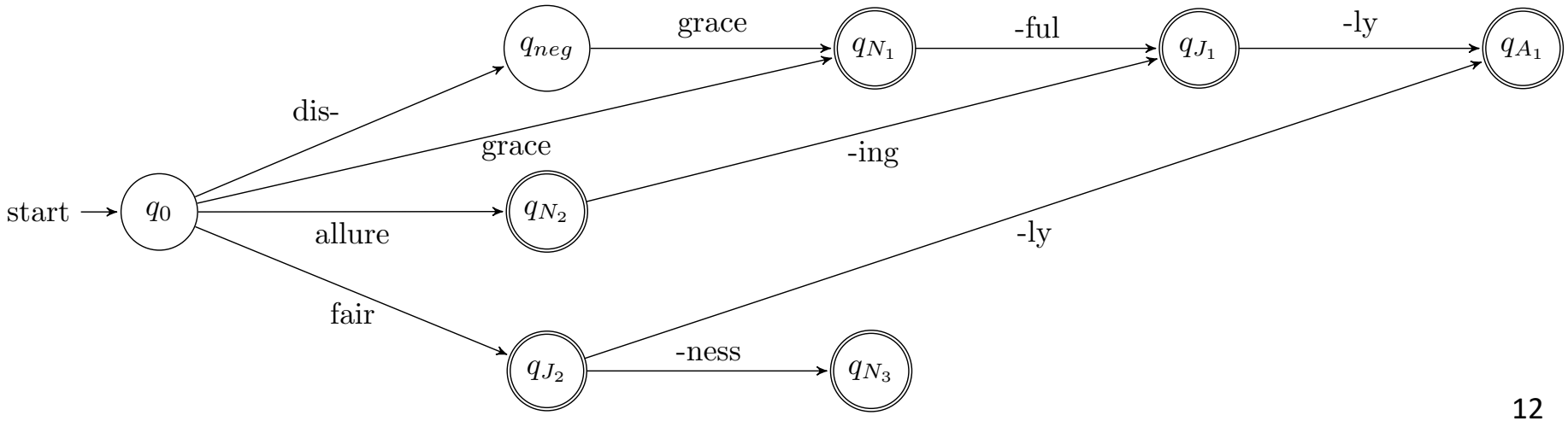


FSA for word morphology

- Morphology relates different word types, e.g., derivation
  - *grace (N) → graceful, gracefully, disgrace, disgracefully*
  - *allure (N) → alluring, alluringly*, *unalluring*
  - but not **disallure*, **allureful*, **disallure, *ungrace* etc
- (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





Weighted FSA

- How to handle:
  - unseen word forms? staycation, misspeak
  - understandable non-words? #fishful, #musicky
- Graded measure of acceptability — weighted FSA adds/changes the following:
  - start state weight function, $\lambda: Q \rightarrow \mathbb{R}$
  - final state weight function, $\rho: Q \rightarrow \mathbb{R}$
  - transition function, $\delta: (Q, \Sigma, Q) \rightarrow \mathbb{R}$

WSFA 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
  - often seek to *maximise* probability, so set λ, ρ, δ to *negative* log probabilities and *minimise*
- Looks a bit like Viterbi for HMMs?

N-gram LMs as WSFA

- Recall LM calculates score of string as follows

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

- Implemented as WSFA
  - Σ = set of word types
  - Q = Σ
  - $\lambda(q_i) = -\log P(w_1 = i|w_0 = \square)$
  - $\rho(q_i) = -\log P(w_{M+1} = \blacksquare|w_M = i)$
  - $\delta(q_i, w, q_j) = \begin{cases} -\log P(w_m = j\ |w_{m-1} = i) & \text{if } w = j \\ \infty & \text{otherwise} \end{cases}$

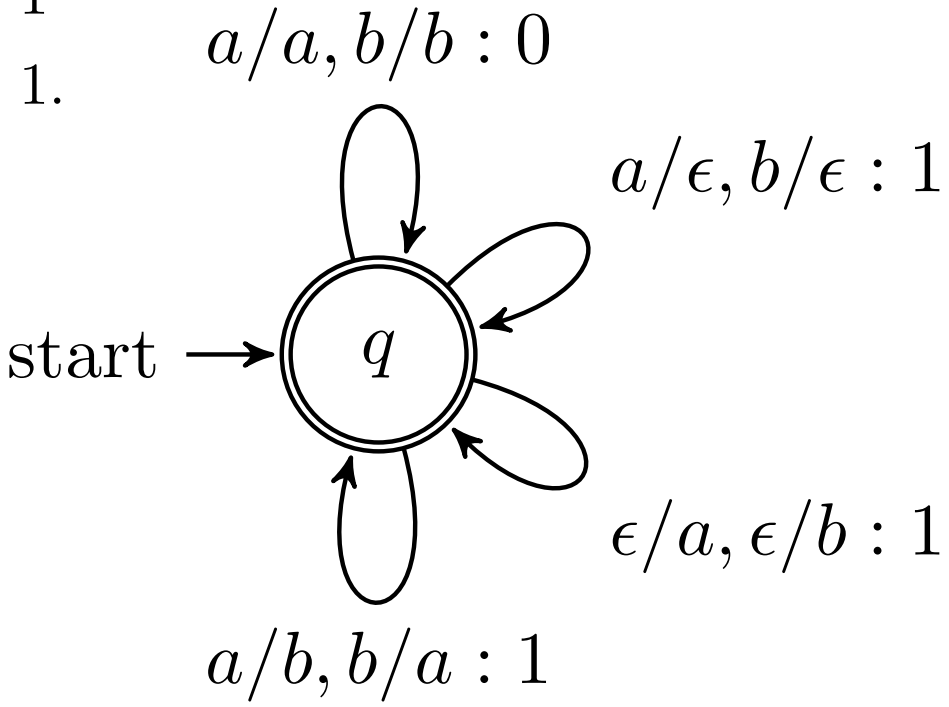
- How to extend to higher orders? HMMs?

Finite State Transducers

- Often don't want to just accept or score strings
  - want to translate them into another language, correct grammar, parse their structure, etc
- Transducers add string output capability to FSAs
  - includes an *output alphabet*
  - and transitions now take input symbol and *emit output symbol*
- E.g., edit distance as WFST which takes one string, and outputs the other
  - zero cost only if strings are identical

Edit distance automata

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



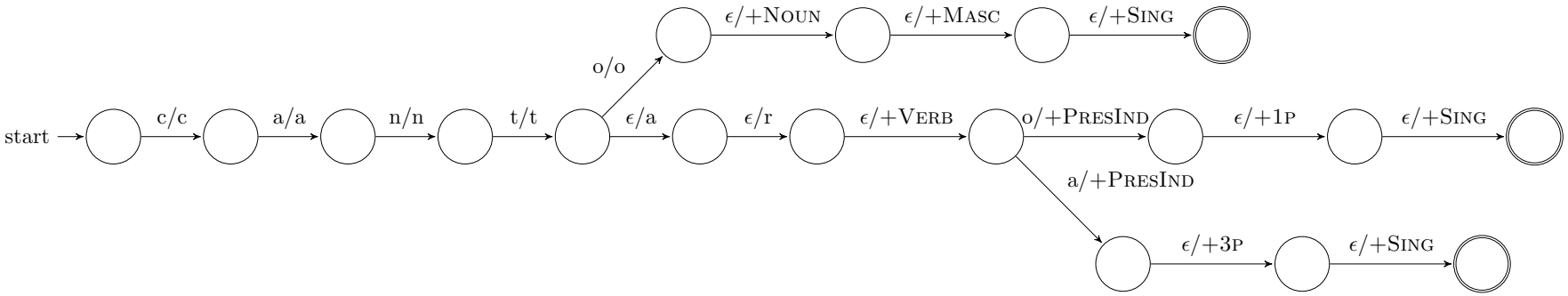
FST for Inflectional Morphology

- Verb inflections in Spanish must match the subject in person & number

	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

- Can we define finite state machine to accept any inflected form, and output infinitive + person/number?

FST for Spanish inflection



canto → *canto* +Noun +Masc +Sing

canto → *cantar* +Verb +PresInd +1P +Sign

FST Composition

- Compose two FSTs by taking output of one FST, S, and giving this as input to FST T
  - denoted S ⊙ T; and results in another FST
  - can also compose FST with FSA, resulting in a FSA
- Allows development of different processes as FSTs, e.g.,
  - morphological tagging
  - orthographic/phonological changes (bake+ed → baked)
  - translation (words or morphological tags)
  - word order changes …

But *is* language regular?

- Sometimes… e.g.,

  This is the house that Jack built.
  This is the malt that lay in the house that Jack built.
  This is the rat that ate the malt that lay in the house that Jack built.
  This is the cat that killed the rat that ate the malt that lay in the house that Jack built.
  This is the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.

  …

- Length is unbounded (*recursive*), but structure is local → can describe with FSA = Regular

But *is* language regular?

- But not in general: centre embedding of relative clauses
  - A man that a woman loves
  - A man that a woman that a child knows loves
  - …
  - **A man** that **a woman** that **a child** that **a bird** that I **heard saw knows loves**
- Need to remember the *n* subject nouns, to ensure *n* verbs follow (and that they agree etc)
  - can't be done with finite number of states
- Requires (at least) *context-free grammar* (next lectures)

Summary

- Concept of a language, and grammar
- Regular languages
- Finite state automata: acceptors, transducers
- Closure properties
- Application to edit distance, morphology

Reading

- Reading
  - Eisenstein, Chapter 9, “Formal Language Theory”