Unit 1: Sequence Models

Lectures 2-3: Finite-State Acceptors/Transducers
This Week: Finite-State Machines

• Finite-State Acceptors and Languages
  • DFAs (deterministic)
  • NFAs (non-deterministic)
• Finite-State Transducers
• Applications in Language Processing
  • part-of-speech tagging, morphology, text-to-sound
  • word alignment (machine translation)
• Next Week: putting probabilities into FSMs
Q1: how to formally define a *language*?

- a language is a *set of strings*
  - could be finite, but often infinite (due to recursion)
  - \( L = \{ \text{aa, ab, ac, \ldots, ba, bb, \ldots, zz} \} \) (finite)

- English is the set of *grammatical English sentences*

- variable names in C is set of alphanumeric strings

Q2: how to *describe* a (possibly infinite) language?

- use a finite (but recursive) representation
- finite-state acceptors (FSAs) or regular-expressions
English Adjective Morphology

Figure 3.4 An FSA for a fragment of English adjective morphology: Antworth’s Proposal #1.

exceptions?
Finite-State Acceptors

- $L_1 = \{ \text{aa, ab, ac, ..., ba, bb, ..., zz} \}$ (finite)
  - start state, final states

- $L_2 = \{ \text{all letter sequences} \}$ (infinite)
  - recursion (cycle)

- $L_3 = \{ \text{all alphanumeric strings} \}$
More Examples

- $L_4 = \{ \text{all letter strings with at least a vowel} \}$

- $L_5 = \{ \text{all letter strings with vowels in order} \}$

- $L_6 = \{ \text{all 01 strings with even number of 0's and even number of 1's} \}$
English Adjective Morphology

Figure 3.4 An FSA for a fragment of English adjective morphology: Antworth’s Proposal #1.

Figure 3.5 An FSA for a fragment of English adjective morphology: Antworth’s Proposal #2.
More English Morphology

Figure 3.6 An FSA for another fragment of English derivational morphology.
Membership and Complement

- deterministic FSA: iff no state has two exiting transitions with the same label. (DFA)
- the language \( L \) of a DFA \( D \): \( L = L(D) \)
- how to check if a string \( w \) is in \( L(D) \) ? (membership)
  - linear-time: follow transitions, check finality at the end
  - no transition for a char means “into a trap state”
- how to construct complement DFA? \( L(D') = \neg L(D) \)
  - super easy: just reverse the finality of states :)
  - note that “trap states” also become final states
END OF WEEK I (half week)