Formal Languages: Recap

• Symbols: $a, b, c$

• Alphabet: finite set of symbols $\Sigma = \{a, b\}$

• String: sequence of symbols $bab$

• Empty string: $\varepsilon$ Define: $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$

• Set of all strings: $\Sigma^*$ cf. The Library of Babel, Jorge Luis Borges

• (Formal) Language: a set of strings

  $\{a^n b^n : n > 0\}$
Regular Languages

• The set of regular languages: each element is a regular language

• Each regular language is an example of a (formal) language, i.e. a set of strings
e.g. \{ a^m b^n : m, n \text{ are +ve integers} \}
Regular Languages

• Defining the set of all regular languages:
  • The empty set and \{a\} for all \(a\) in \(\Sigma^e\) are regular languages
  • If \(L_1\) and \(L_2\) and \(L\) are regular languages, then:
    \[L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}\] (concatenation)
    \[L_1 \cup L_2\] (union)
    \[L^* = \bigcup_{i=0}^{\infty} L^i\] (Kleene closure)
  are also regular languages
• There are no other regular languages
Formal Grammars

• A formal grammar is a concise description of a formal language
• A formal grammar uses a specialized syntax
• For example, a regular expression is a concise description of a regular language

\[(ab)^*abb\] : is the set of all strings over the alphabet \(\{a, b\}\) which end in \(abb\)
Regular Expressions: Definition

• Every symbol of $\Sigma \cup \{ \varepsilon \}$ is a regular expression

• If $r_1$ and $r_2$ are regular expressions, so are
  – Concatenation: $r_1 r_2$
  – Alternation: $r_1 | r_2$
  – Repetition: $r_1^*$

• Nothing else is.
  – Grouping re’s: e.g. aalbc vs. ((aa)lb)c
Regular Expressions: Examples

• Alphabet \{ V, C \}  V: vowel  C: consonant
• A set of consonant-vowel sequences  \((CV|CCV)^*\)
• All strings that do not contain “VC” as a substring \(C^*V^*\)
• Need a decision procedure: does a particular regular expression (regexp) accept an input string
• Provided by: Finite State Automata
Finite Automata: Recap

- A set of states $S$
  - One start state $q_0$, zero or more final states $F$
- An alphabet $\Sigma$ of input symbols
- A transition function:
  - $\delta: S \times \Sigma \Rightarrow S$
- Example: $\delta(1, a) = 2$
Finite Automata: Example

• What regular expression does this automaton accept?

Answer: $(0|1)^*00$
NFAs

- NFA: like a DFA, except
  - A transition can lead to more than one state, that is, \( \delta: S \times \Sigma \rightarrow 2^S \)
  - One state is chosen non-deterministically
  - Transitions can be labeled with \( \varepsilon \), meaning states can be reached without reading any input, that is, \( \delta: S \times \Sigma \cup \{ \varepsilon \} \rightarrow 2^S \)
Recognition of strings (NFAs)

- Input string: aba#
- Recognition problem: Is input string in the language generated by the NFA?
- Recognition (without conversion to DFA) is also called simulation of NFA
Recognition of strings (NFAs)

\[ \begin{aligned}
\text{A} & \xrightarrow{a} \text{B} \\
\text{B} & \xrightarrow{b} \text{D} \\
\text{B} & \xrightarrow{b} \text{C} \\
\text{C} & \xrightarrow{a} \text{B} \\
\text{D} & \xrightarrow{a} \text{B} \\
\end{aligned} \]

- **Input tape:** 0 a 1 b 2 a 3 # 4
- **Start State:** A  
  **Agenda:** \( \{ (A, 0) \} \)
- **Pop** (A, 0) from Agenda
- **q(A, a) = B,**  
  **Agenda:** \( \{ (B, 1) \} \)
- **Pop** (B, 1) from Agenda
- **q(B, b) = \{ D, C \},**  
  **Agenda:** \( \{ (D, 2), (C, 2) \} \)
Recognition of strings (NFAs)

- Input tape: 0 a 1 b 2 a 3 # 4
- Pop (D, 2) from Agenda
- q(D, a) = { B } Agenda: { (B, 3), (C, 2) }
- Pop (B, 3) from Agenda: B is not a final state
- Pop (C, 2) from Agenda: if Agenda empty, reject
- q(C, a) = { D } Agenda: { (D, 3) }
Recognition of strings (NFAs)

- Input tape: 0 a 1 b 2 a 3 # 4
- Pop (D, 3) from Agenda
- Is (D, 3) an accept item?
- Yes: D is a final state and 3 is index of the end-of-string marker #

Return accept
Recognition of strings (NFAs)

function NDRecognize (tape[], q):
    Agenda = { (start-state, 0) }
    Current = (state, index) = pop(Agenda)
    while (true) {
        if (Current is an accept item) return accept
        else Agenda = Agenda ∪ GenStates(q, state, tape[index])
        if (Agenda is empty) return reject
        else Current = (state, index) = pop(Agenda)
    }

function GenStates (q, state, index):
    return { (q’, index) : for all q’ = q(state, ε) } ∪
            { (q’, index+1) : for all q’ = q(state, tape[index+1]) }
Algorithms for FSMs
(finite-state machines)

• Recognition of a string in a regular language: is a string accepted by an NFA?
• Conversion of regular expressions to NFAs
• Determinization: converting NFA to DFA
• Converting an NFA into a regular expression
• Other useful closure properties: union, concatenation, Kleene closure, intersection