REGULAR LANGUAGES
REFRESHER
Regular languages refresher

• Formal languages
  – Alphabet = finite set of letters
  – Word = sequence of letter
  – Language = set of words

• Regular languages defined equivalently by
  – Regular expressions
  – Finite-state automata
Regular expressions

• **Empty string:** $\epsilon$
• **Letter:** $a$
• **Concatenation:** $R_1 R_2$
• **Union:** $R_1 \mid R_2$
• **Kleene-star:** $R^*$
  – Shorthand: $R^+$ stands for $R \ R^*$
• **scope:** $(R)$
• **Example:** $(0^* \ 1^*) \mid (1^* \ 0^*)$
  – What is this language?
FINITE AUTOMATA
Finite automata: known results

- Types of finite automata:
  - Deterministic (DFA)
  - Non-deterministic (NFA)
  - Non-deterministic + epsilon transitions

- **Theorem**: translation of regular expressions to NFA+epsilon (linear time)

- **Theorem**: translation of NFA+epsilon to DFA
  - Worst-case exponential time

- **Theorem [Myhill-Nerode]**: For every DFA there is an equivalent unique minimal DFA
Finite automata

- An automaton $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ is defined by states and transitions.
Automaton running example

- Words are read left-to-right
Automaton running example

- Words are read left-to-right
Automaton running example

- Words are read left-to-right
Automaton running example

• Words are read left-to-right

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
</table>

word accepted

start

\[a \rightarrow b \rightarrow c \rightarrow \text{accepted}\]
Word outside of language 1

The sequence of characters is b, b, c.

The diagram shows a start state connected to states with transitions labeled a, b, and c.
• Missing transition means non-acceptance
Word outside of language 2

[Diagram]

- Start -> a -> b
- b -> c
- c to a
- a to b
- b to c

[Rectangle]

- a
- b
- b
Word outside of language 2
Word outside of language 2

- Final state is not an accepting state
Exercise - Question

• What is the language defined by the automaton below?
Exercise - Answer

• What is the language defined by the automaton below?
  – a b* c
  – Generally: all paths leading to accepting states
NONDETERMINISTIC FINITE AUTOMATA
Non-deterministic automata

- Allow multiple transitions from given state labeled by same letter
NFA run example
NFA run example

- Maintain set of states
NFA run example
NFA run example

• Accept word if any of the states in the set is accepting

![NFA Diagram]

- Start state
- Accepting states indicated by blue color
- Transitions for symbols a, b, and c
NFA+$\epsilon$ automata

- $\epsilon$ transitions can “fire” without reading the input
NFA+€ run example

start

a
b

ε

c

b

a

c

€
NFA+$\varepsilon$ run example

- Now $\varepsilon$ transition can non-deterministically take place
NFA+$\epsilon$ run example
NFA+€ run example
NFA+ɛ run example

- Word accepted
FROM REGULAR EXPRESSIONS TO NFA
From reg. exp. to automata

• **Theorem:** there is an algorithm to build an NFA+$\epsilon$ automaton for any regular expression

• **Proof:** by induction on the structure of the regular expression
  
  – For each sub-expression $R$ we build an automaton with exactly one start state and one accepting state
  
  – Start state has no incoming transitions
  
  – Accepting state has no outgoing transitions
From reg. exp. to $NFA+\epsilon$ automata

- **Theorem:** there is an algorithm to build an $NFA+\epsilon$ automaton for any regular expression

- **Proof:** by induction on the structure of the regular expression
Inductive constructions

\[ R = \varepsilon \]
\[ R = a \]
\[ R^* \]

\[ R_1 \mid R_2 \]
Running time of NFA+$\varepsilon$

- Construction requires $O(k)$ states for a reg-exp of length $k$
- Running an NFA+$\varepsilon$ with $k$ states on string of length $n$ takes $O(n \cdot k^2)$ time
  - Can we reduce the $k^2$ factor?

Each state in a configuration of $O(k)$ states may have $O(k)$ outgoing edges, so processing an input letter may take $O(n^2)$ time
From NFA+€ to DFA

• Construction requires $O(k)$ states for a reg-exp of length $k$

• Running an NFA+€ with $k$ states on string of length $n$ takes $O(n \cdot k^2)$ time
  – Can we reduce the $k^2$ factor?

• **Theorem:** for any NFA+€ automaton there exists an equivalent deterministic automaton

• **Proof:** determinization via subset construction
  – Number of states in the worst-case $O(2^k)$
  – Running time $O(n)$
NFA DETERMINIZATION
Subset construction

- For an NFA+∈ with states \( M=\{s_1,...,s_k\} \)
- Construct a DFA with one state per set of states of the corresponding NFA
  - \( M'=\{ [], [s_1], [s_1,s_2], [s_2,s_3], [s_1,s_2,s_3], ... \} \)
- Simulate transitions between individual states for every letter

\[
\text{NFA+∈}
\]

\[
\begin{align*}
S_1 & \xrightarrow{a} S_2 \\
S_4 & \xrightarrow{a} S_7
\end{align*}
\]

\[
\text{DFA}
\]

\[
\begin{align*}
[s_1,s_4] & \xrightarrow{a} [s_2,s_7]
\end{align*}
\]
Handling epsilon transitions

- Extend macro states by states reachable via $\epsilon$ transitions