Automata Theory
CS411 20145-03
Deterministic Finite Automata

David Galles

Department of Computer Science
University of San Francisco
Generators vs. Checkers

- Regular expressions are one way to specify a formal language
  - **String Generator** Generates strings in the language
- Deterministic Finite Automata (DFA) are another way to specify a language
  - **String Checker** Given any string, determines if that string is in the language or not
Example Deterministic Finite Automaton
DFA for all strings over \{a,b\} that contain exactly 2 a’s
DFA for all strings over \{a,b\} that contain exactly 2 a’s
03-4: DFA Example

- All strings over a, b that have length 3.
03-5: DFA Example

- All strings over a, b that have length 3.
03-6: DFA Components

- What makes up a DFA?
• What makes up a DFA?

• States
What makes up a DFA?

Alphabet (characters than can occur in strings accepted by DFA)
What makes up a DFA?

Transitions
03-10: DFA Components

- What makes up a DFA?

- Initial State
• What makes up a DFA?

• Final State(s) (there can be > 1)
A DFA is a 5-tuple $M = (K, \Sigma, \delta, s, F)$

- $K$ Set of states
- $\Sigma$ Alphabet
- $\delta : (K \times \Sigma) \rightarrow K$ is a Transition function
- $s \in K$ Initial state
- $F \subseteq K$ Final states
DFA Definition

\[ K = \{ q_0, q_1, q_2, q_3 \} \]

\[ \Sigma = \{ a, b \} \]

\[ \delta = \{(q_0, a), q_1\}, ((q_0, b), q_0), ((q_1, a), q_2), ((q_1, b), q_1),
\]
\[ ((q_2, a), q_3), ((q_2, b), q_2), ((q_3, a), q_3), ((q_3, b), q_3)\} \]

\[ s = q_0 \]

\[ F = \{ q_2 \} \]
Create a DFA for:

- All strings over \{0, 1\} that contain the substring 1001
Create a DFA for:

- All strings over \( \{0, 1\} \) that contain the substring 1001
Create a DFA for:

- All strings over \{0, 1\} that end with 111
Create a DFA for:

- All strings over \{0, 1\} that end with 111
Create a DFA for:

- All strings over \( \{0, 1\} \) that begin with 111
Create a DFA for:

- All strings over \{0, 1\} that begin with 111
Create a DFA for:

- All strings over \{0, 1\} that begin or end with 111
Create a DFA for:

- All strings over \{0, 1\} that begin or end with 111
Create a DFA for:

- All strings over \( \{0, 1\} \) that begin \( \text{and} \) end with 111
Create a DFA for:

- All strings over \{0, 1\} that begin \textit{and} end with 111

![DFA Diagram]
Create a DFA for:

- All strings over \( \{0, 1\} \) that contain 1001 or 0110
Create a DFA for:

- All strings over \{0, 1\} that contain 1001 or 0110
Create a DFA for:

- All strings over \{a, b\} that begin and end with the same letter
Create a DFA for:

- All strings over \{a, b\} that begin and end with the same letter
Why are these machines called “Deterministic Finite Automata”

- **Deterministic** Each transition is completely determined by the current state and next input symbol. That is, for each state / symbol pair, there is exactly one state that is transitioned to
- **Finite** Every DFA has a finite number of states
- **Automata** (singular automaton) means “machine”

(From Merriam-Webster Online Dictionary, definition 2: A machine or control mechanism designed to follow automatically a predetermined sequence of operations or respond to encoded instructions)
Way to describe the computation of a DFA

**Configuration**: What state the DFA is currently in, and what string is left to process

- \( \in K \times \Sigma^* \)
- \((q_2, abba)\) Machine is in state \(q_2\), has \(abba\) left to process
- \((q_8, bba)\) Machine is in state \(q_8\), has \(bba\) left to process
- \((q_4, \epsilon)\) Machine is in state \(q_4\) at the end of the computation (accept iff \(q_4 \in F\))
Way to describe the computation of a DFA

**Configuration**: What state the DFA is currently in, and what string is left to process
  - $\in K \times \Sigma^*$

**Binary relation $\vdash_M$**: What machine $M$ yields in one step
  - $\vdash_M \subseteq (K \times \Sigma^*) \times (K \times \Sigma^*)$
  - $\vdash_M = \{(q_1, a w, q_2, w) : q_1, q_2 \in K_M, w \in \Sigma_M^*, a \in \Sigma_M, ((q_1, a), q_2) \in \delta_M\}$
Given the following machine $M$:

\[ ((q_0, abba), (q_2, bba)) \in \vdash_M \]

- can also be written $(q_0, abba) \vdash_M (q_2, bba)$
03-32: DFA Configuration & $\vdash_M$

\[
(q_0, 11101) \vdash_M (q_1, 1101) \\
\vdash_M (q_2, 101) \\
\vdash_M (q_3, 01) \\
\vdash_M (q_0, 1) \\
\vdash_M (q_1, \epsilon)
\]
03-33: DFA Configuration & ⊢ \_M

(q₀, 10111) ⊢ \_M (q₁, 0111)

⊢ \_M (q₀, 111)

⊢ \_M (q₁, 11)

⊢ \_M (q₂, 1)

⊢ \_M (q₃, \epsilon)
DFA Configuration & $\vdash^*_M$

- $\vdash^*_M$ is the reflexive, transitive closure of $\vdash_M$
  - Smallest superset of $\vdash_M$ that is both reflexive and transitive
  - “yields in 0 or more steps”
- Machine $M$ accepts string $w$ if:
  $$(s_M, w) \vdash^*_M (f, \epsilon)$$ for some $f \in F_M$$
• Language accepted by a machine $M = L[M]$
  • $\{ w : (s_M, w) \vdash^*_M (f, \epsilon) \text{ for some } f \in F_M \}$

• DFA Languages, $L_{DFA}$
  • Set of all languages that can be defined by a DFA
  • $L_{DFA} = \{ L : \exists M, L[M] = L \}$

• To think about: How does $L_{DFA}$ relate to $L_{REG}$