03-0: **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

03-1: **DFA Example**
Example Deterministic Finite Automaton

03-2: **DFA Example**
DFA for all strings over \{a,b\} that contain exactly 2 a’s

03-3: **DFA Example**
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?

03-7: DFA Components

• What makes up a DFA?

03-8: DFA Components

• States

03-9: DFA Components

• Alphabet (characters than can occur in strings accepted by DFA)
• Transitions

03-10: DFA Components

• What makes up a DFA?

03-11: DFA Components

• Initial State

03-12: DFA Definition

• A DFA is a 5-tuple $M = (K, \Sigma, \delta, s, F)$
  
  • $K$ Set of states
  
  • $\Sigma$ Alphabet

• Final State(s) (there can be > 1)
• $\delta : (K \times \Sigma) \mapsto K$ is a Transition function
• $s \in K$ Initial state
• $F \subseteq K$ Final states

03-13: DFA Definition

$K = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\delta = \{(q_0, a, q_1), (q_0, b, q_0), (q_1, a, q_2), (q_1, b, q_1),$
$(q_2, a, q_3), (q_2, b, q_2), (q_3, a, q_3), (q_3, b, q_3)\}$
$s = q_0$
$F = \{q_2\}$

03-14: Fun with DFA
Create a DFA for:
• All strings over $\{0, 1\}$ that contain the substring 1001

03-15: Fun with DFA
Create a DFA for:
• All strings over $\{0, 1\}$ that contain the substring 1001

03-16: Fun with DFA
Create a DFA for:
• All strings over $\{0, 1\}$ that end with 111

03-17: Fun with DFA
Create a DFA for:
• All strings over $\{0, 1\}$ that end with 111
03-18: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin with 111

03-19: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin with 111

03-20: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin or end with 111

03-21: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin or end with 111
03-22: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin \textit{and} end with 111

03-23: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that begin \textit{and} end with 111

03-24: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that contain 1001 \textit{or} 0110

03-25: **Fun with DFA**  
Create a DFA for:  
- All strings over \{0, 1\} that contain 1001 \textit{or} 0110
03-26: **Fun with DFA**
Create a DFA for:

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

03-27: **Fun with DFA**
Create a DFA for:

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

03-28: **Why DFA?**

- 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)

03-29: **DFA Configuration & ↓^M**

- Way to describe the computation of a DFA
- **Configuration**: What state the DFA is currently in, and what string is left to process
• ∈ K × Σ*
• (q₂, abba) Machine is in state q₂, has abba left to process
• (q₈, bba) Machine is in state q₈, has bba left to process
• (q₄, ε) Machine is in state q₄ at the end of the computation (accept iff q₄ ∈ F)

03-30: DFA Configuration & ⊢ₘ

• Way to describe the computation of a DFA

• Configuration: What state the DFA is currently in, and what string is left to process
  • ∈ K × Σ*

• Binary relation ⊢ₘ: What machine M yields in one step
  • ⊢ₘ ⊆ (K × Σ*) × (K × Σ*)
  • ⊢ₘ = {((q₁, aw), (q₂, w)) : q₁, q₂ ∈ Kₘ, w ∈ Σₘ, a ∈ Σ, ((q₁, a), q₂) ∈ δₘ}

03-31: DFA Configuration & ⊢ₘ

Given the following machine M:

03-32: DFA Configuration & ⊢ₘ

03-33: DFA Configuration & ⊢ₘ
03-34: 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:

$$(s_M, w) \vdash_M^* (f, \epsilon) \text{ for some } f \in F_M$$

03-35: DFA & Languages

- 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 $L_{REG}$