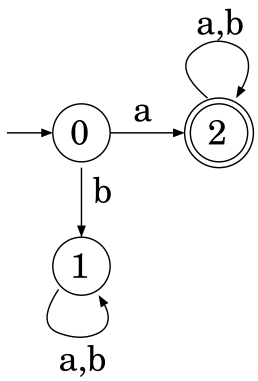
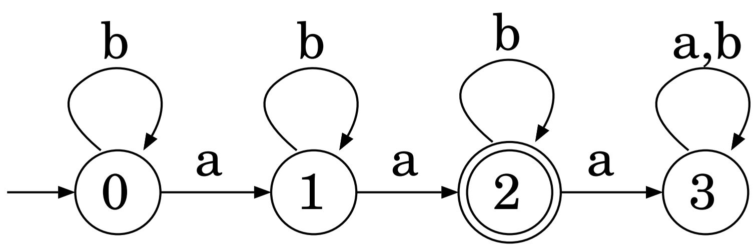
03-0: **Generators vs. Checkers**

- 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

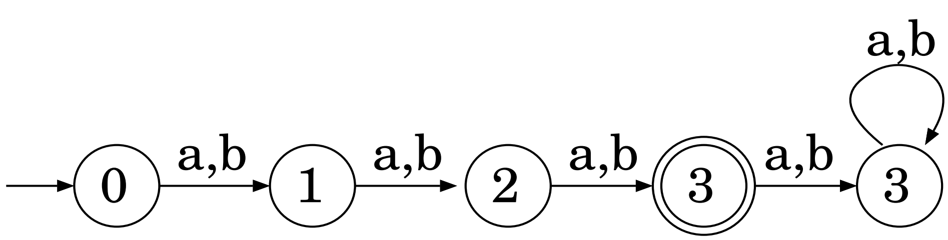


03-4: **DFA Example**

- All strings over a, b that have length 3.

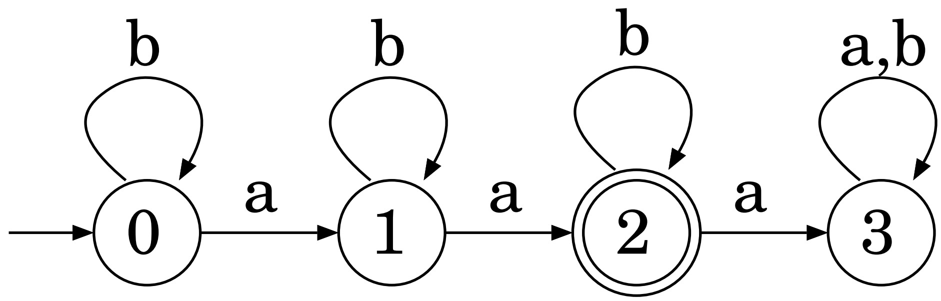
03-5: **DFA Example**

- All strings over a, b that have length 3.



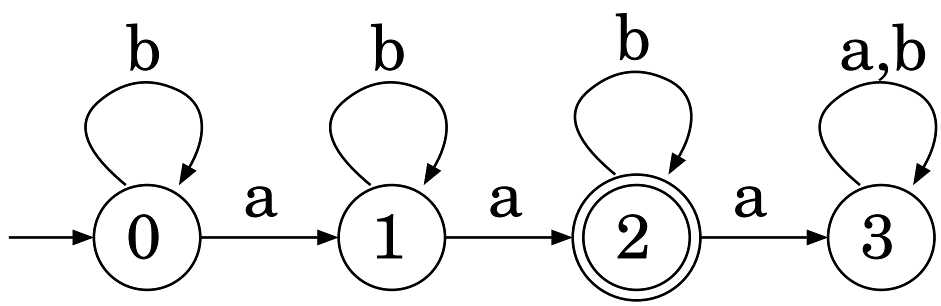
03-6: **DFA Components**

- What makes up a DFA?



03-7: **DFA Components**

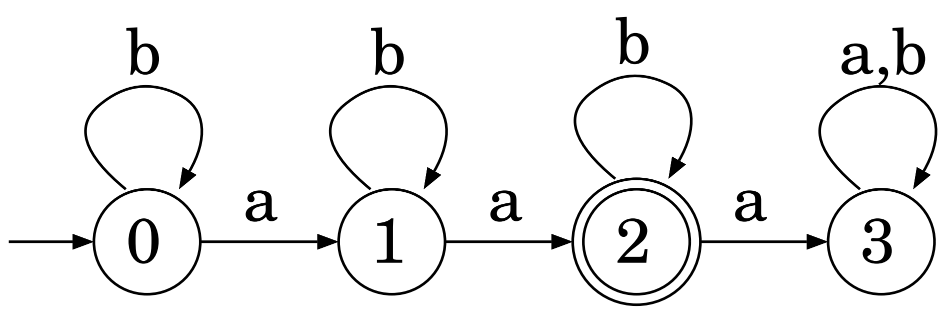
- What makes up a DFA?



- States

03-8: **DFA Components**

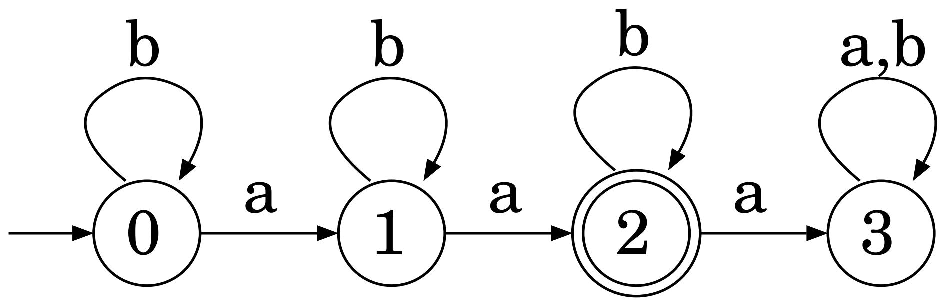
- What makes up a DFA?



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

03-9: **DFA Components**

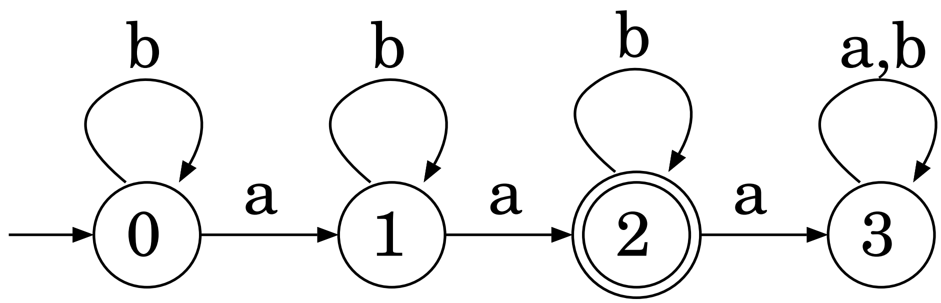
- What makes up a DFA?



- Transitions

03-10: **DFA Components**

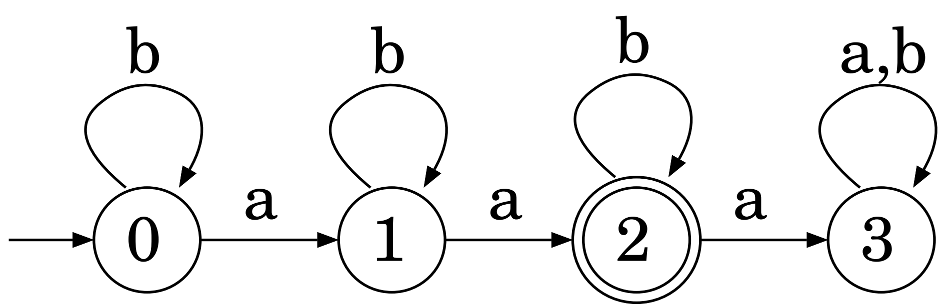
- What makes up a DFA?



- Initial State

03-11: **DFA Components**

- What makes up a DFA?



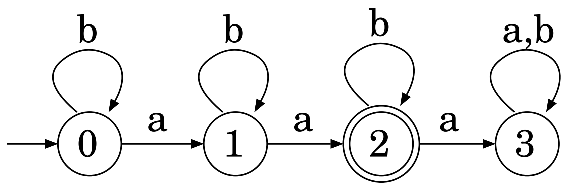
- Final State(s) (there can be > 1)

03-12: **DFA Definition**

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

- $\delta : (K \times \Sigma) \mapsto K$ is a Transition function
- $s \in K$ Initial state
- $F \subseteq K$ Final states

03-13: **DFA Definition**



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

03-14: **Fun with DFA**

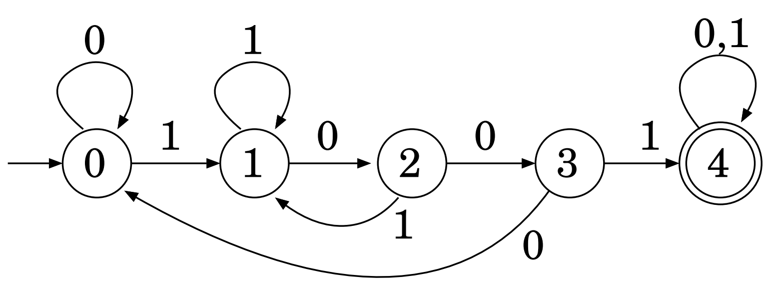
Create a DFA for:

- 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**

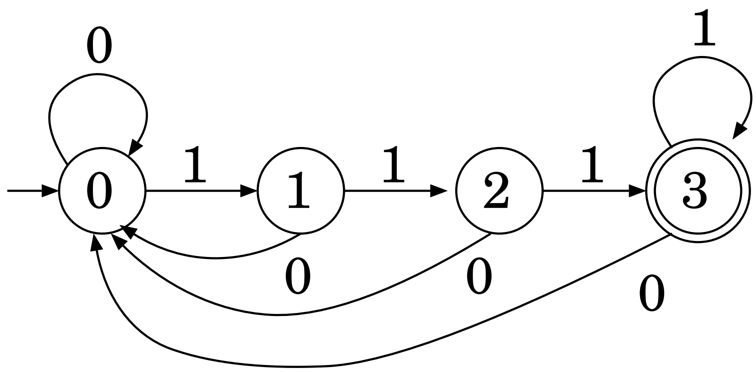
Create a DFA for:

- 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**

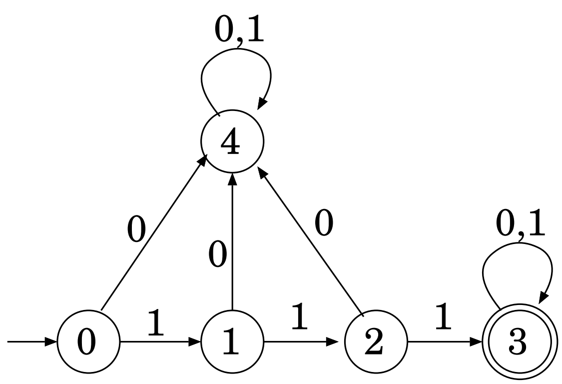
Create a DFA for:

- 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**

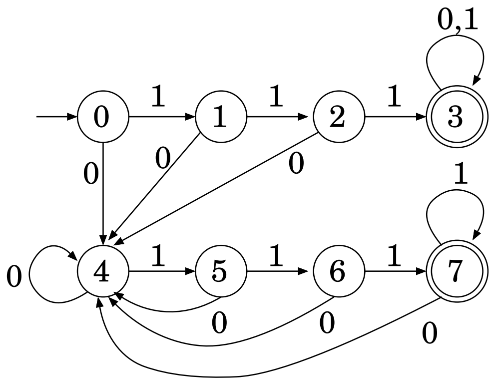
Create a DFA for:

- 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**

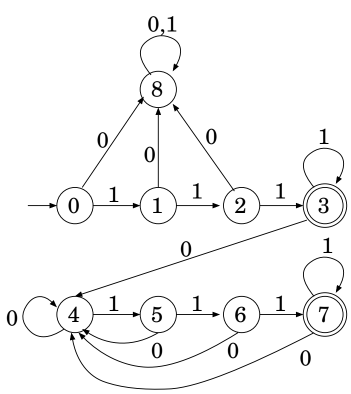
Create a DFA for:

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

03-23: **Fun with DFA**

Create a DFA for:

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



03-24: **Fun with DFA**

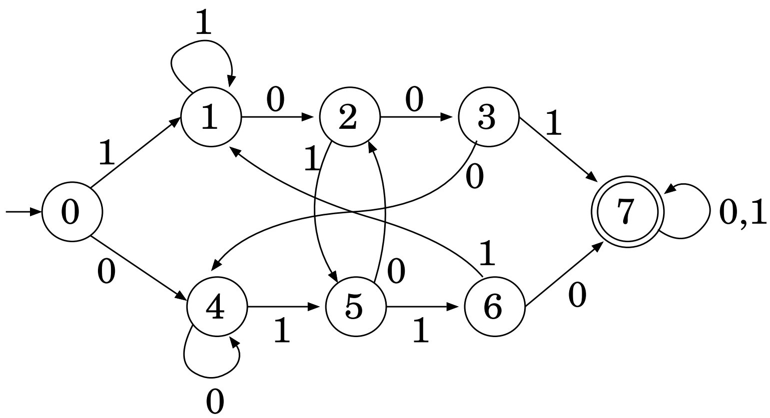
Create a DFA for:

- All strings over $\{0, 1\}$ that contain 1001 *or* 0110

03-25: **Fun with DFA**

Create a DFA for:

- All strings over $\{0, 1\}$ that contain 1001 *or* 0110



03-26: **Fun with DFA**

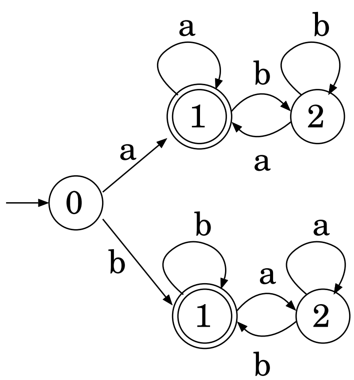
Create a DFA for:

- 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 & $\vdash_M$**

- Way to describe the computation of a DFA
- **Configuration**: What state the DFA is currently in, and what string is left to process

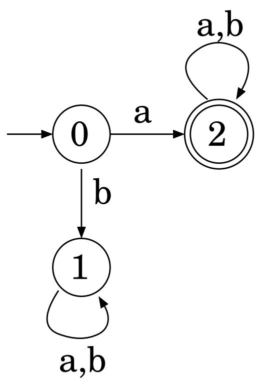
- $\in K \times \Sigma^*$
- $(q_2, abba)$ Machine is in state q_2, has $abba$ left to process
- (q_8, bba) Machine is in state q_8, has bba left to process
- (q_4, ϵ) Machine is in state q_4 at the end of the computation (accept iff $q_4 \in F$)

03-30: **DFA Configuration &** $\vdash_M$

- 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
  - $\vdash_M \subseteq (K \times \Sigma^*) \times (K \times \Sigma^*)$
  - $\vdash_M = \{((q_1, aw), (q_2, w)) : q_1, q_2 \in K_M, w \in \Sigma_M^*, a \in \Sigma_M, ((q_1, a), q_2) \in \delta_M\}$

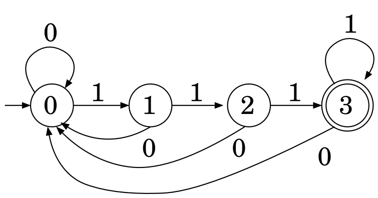
03-31: **DFA Configuration &** $\vdash_M$

Given the following machine M:



- $((q_0, \text{abba}), (q_2, \text{bba})) \in \vdash_M$
  - can also be written $(q_0, \text{abba}) \vdash_M (q_2, \text{bba})$

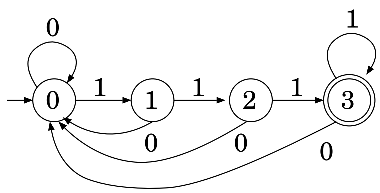
03-32: **DFA Configuration &** $\vdash_M$



$$
\begin{aligned}
(q_0, 11101) &\vdash_M (q_1, 1101)\\
&\vdash_M (q_2, 101)\\
&\vdash_M (q_3, 01)\\
&\vdash_M (q_0, 1)\\
&\vdash_M (q_1, \epsilon)
\end{aligned}
$$

03-33: **DFA Configuration &** $\vdash_M$

03-34: **DFA Configuration & $\vdash_M^*$**



$$\begin{aligned}(q_0, 10111) &\vdash_M (q_1, 0111)\\ &\vdash_M (q_0, 111)\\ &\vdash_M (q_1, 11)\\ &\vdash_M (q_2, 1)\\ &\vdash_M (q_3, \epsilon)\end{aligned}$$

- $\vdash_M^*$ 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 $f \in F_M$

03-35: **DFA & Languages**

- Language accepted by a machine $M = L[M]$
    - $\{w : (s_M, w) \vdash_M^* (f, \epsilon)$ 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}