Please check your attendance using Blackboard!

Lecture 3

Regular Languages and Regular Grammars

COSE215: Theory of Computation

Seunghoon Woo

Fall 2023

Contents

- **Regular expressions to finite automata**
- **DFA to regular expressions**
- **Regular Grammars**

Regular expression and finite automata

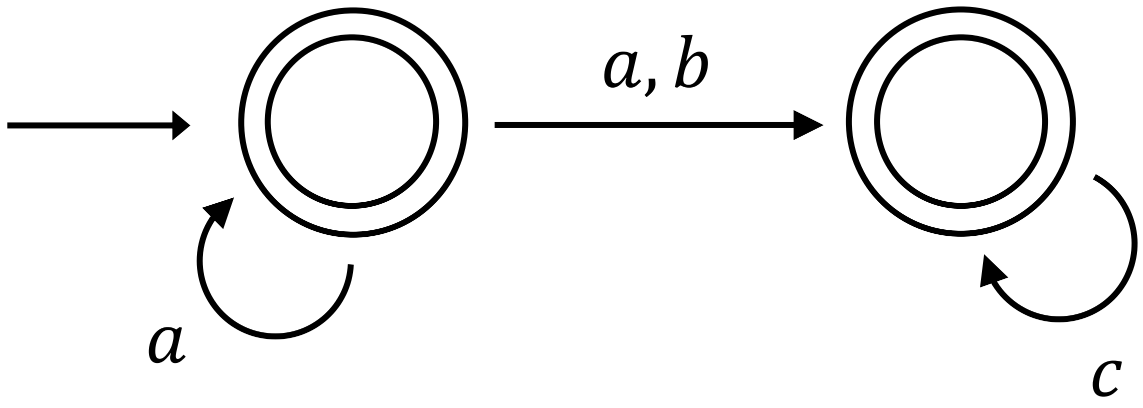
- **How can we represent regular expressions in finite automata?**

$$L(a^* + a^*(a+b)c^*)$$

Regular expression and finite automata

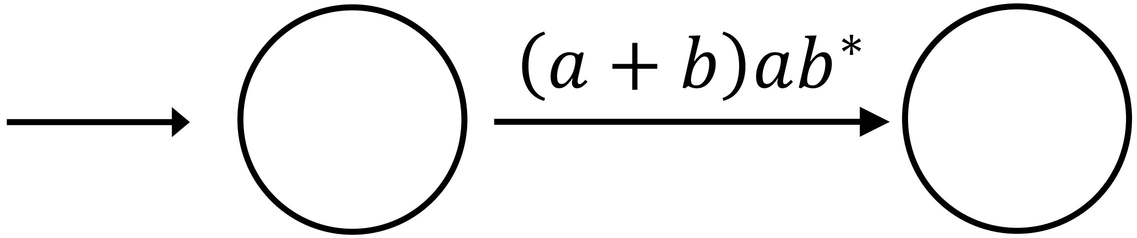
- **How can we represent regular expressions in finite automata?**

$$L(a^* + a^*(a + b)c^*)$$



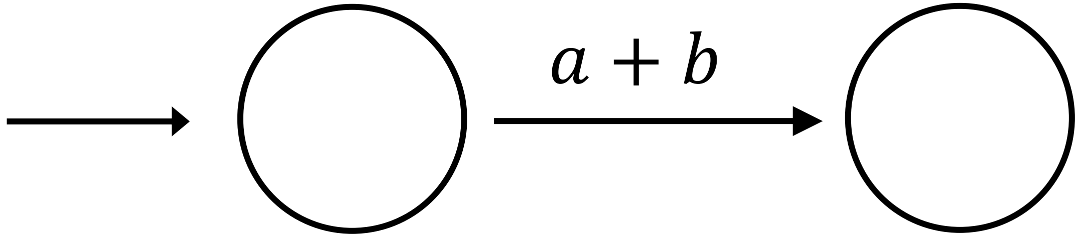
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - One way: using generalized NFA, aka, Generalized Transition Graph (GTG)
    - NFA with labels of "REs" instead of only members of Σ



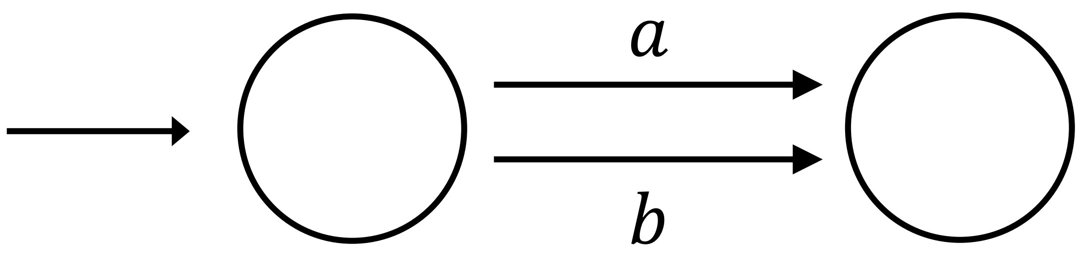
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ UNION operation



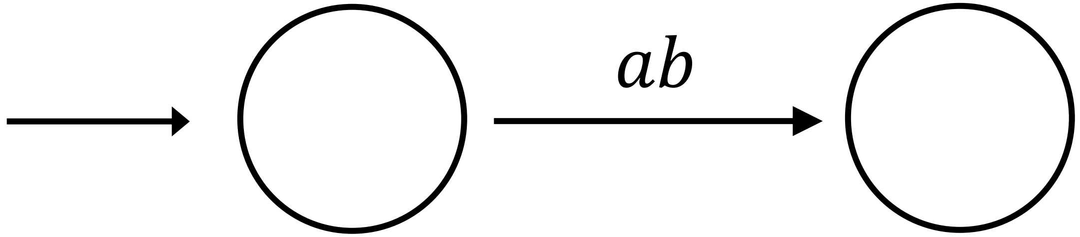
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ UNION operation



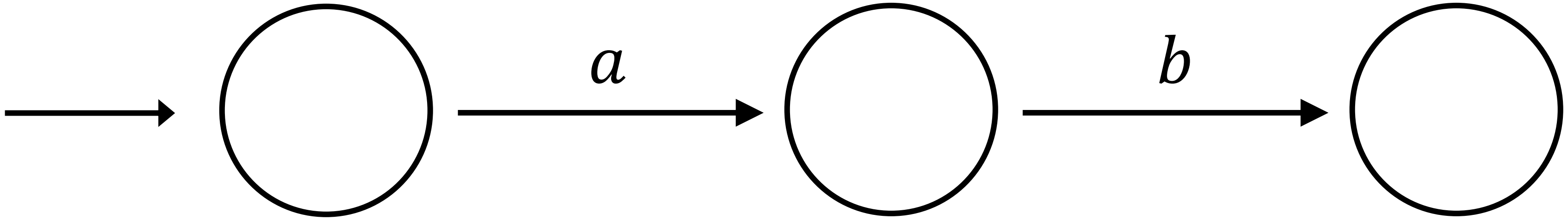
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ CONCATENATION operation



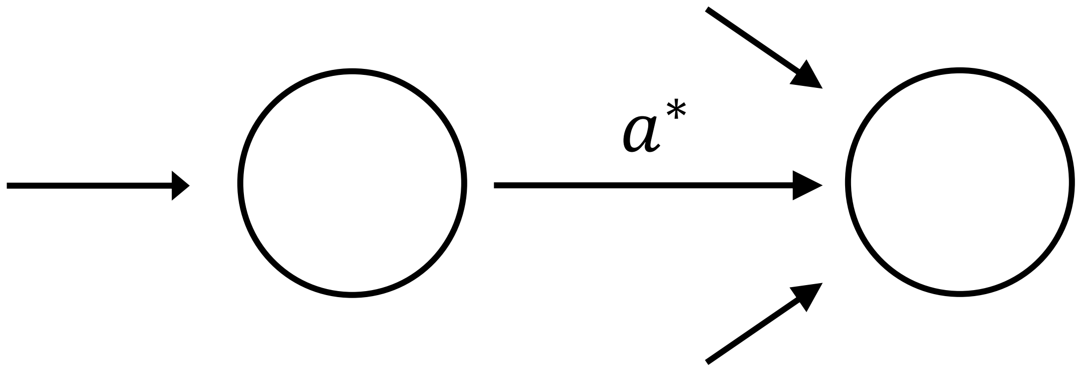
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ CONCATENATION operation



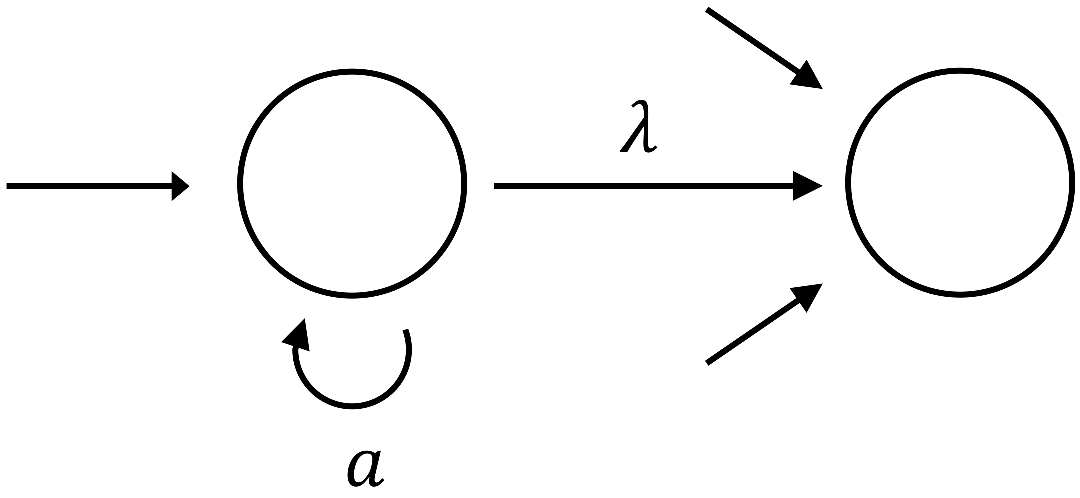
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ STAR operation
      - CASE 1) Only one outgoing edge at the left-most state



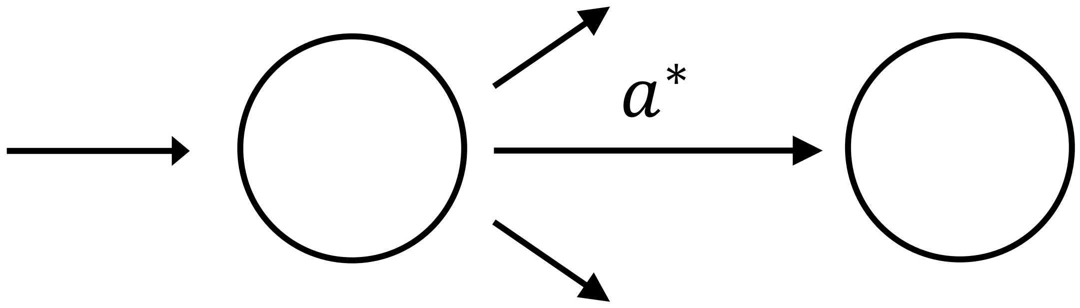
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - STAR operation
      - CASE 1) Only one outgoing edge at the left-most state



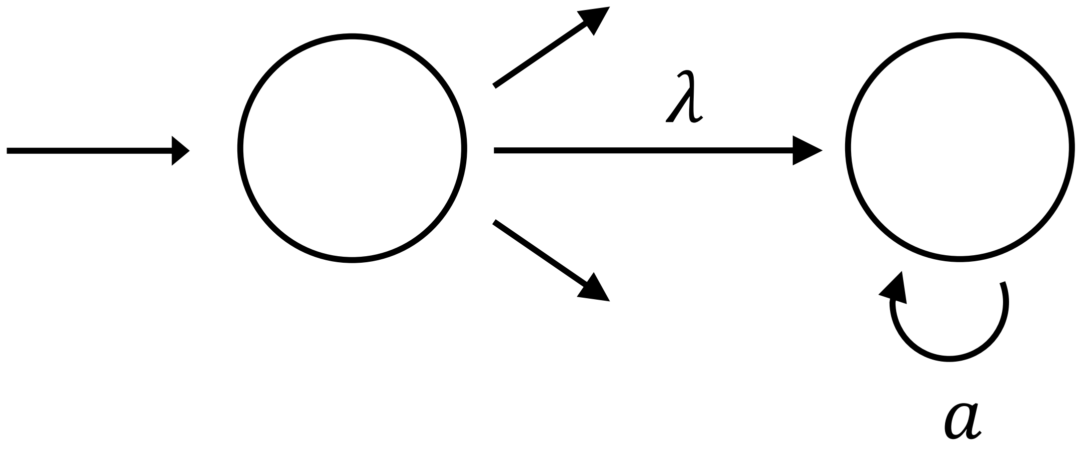
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ STAR operation
      - CASE 2) Only one incoming edge at the right-most state



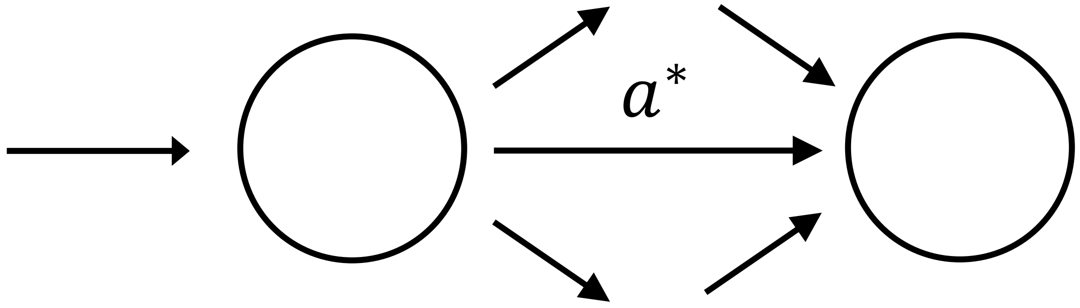
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ STAR operation
      - CASE 2) Only one incoming edge at the right-most state



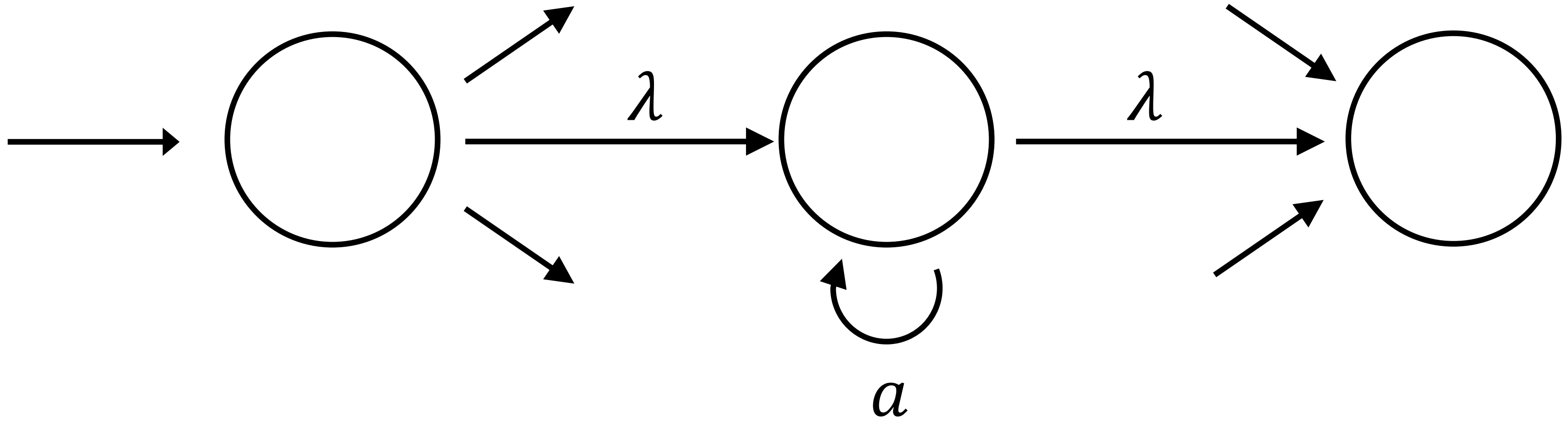
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ STAR operation
      - CASE 3) Remaining cases



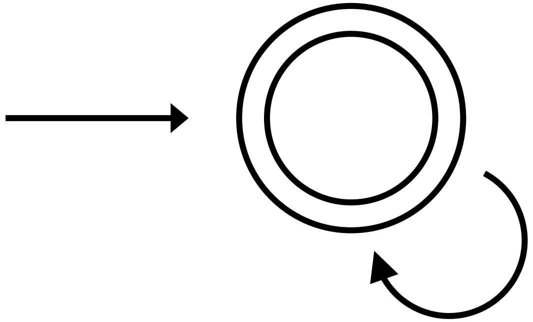
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Split regular expression based on the rules
    - ❖ STAR operation
      - CASE 3) Remaining cases: generating a new state



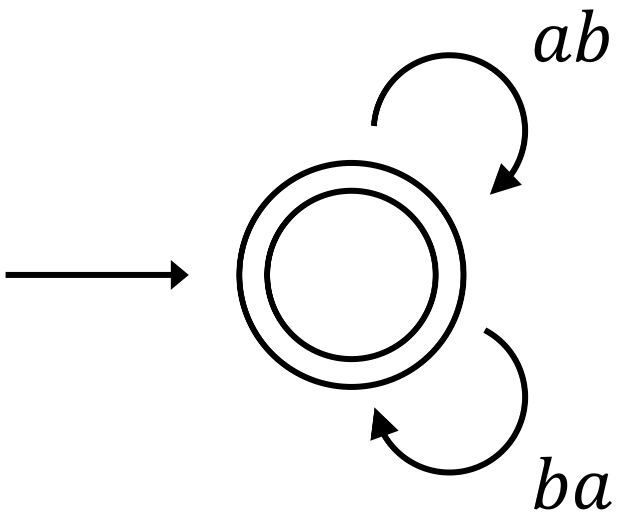
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $r = (ab + ba)^*$



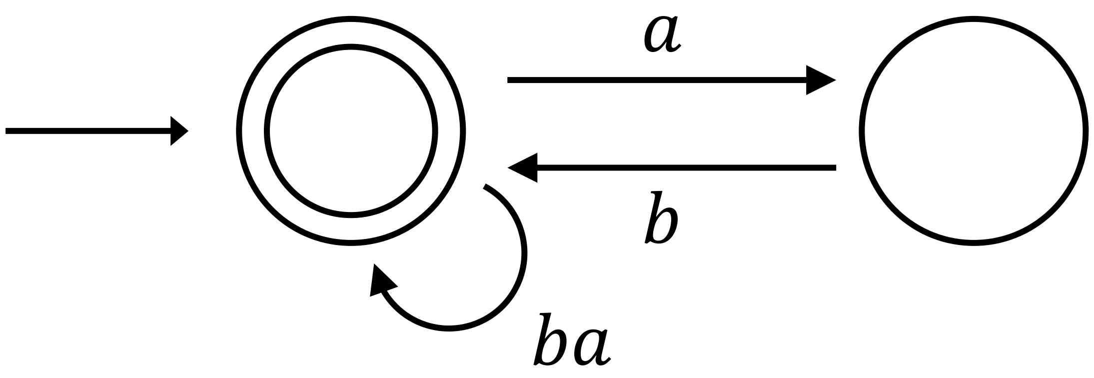
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $r = (ab + ba)^*$



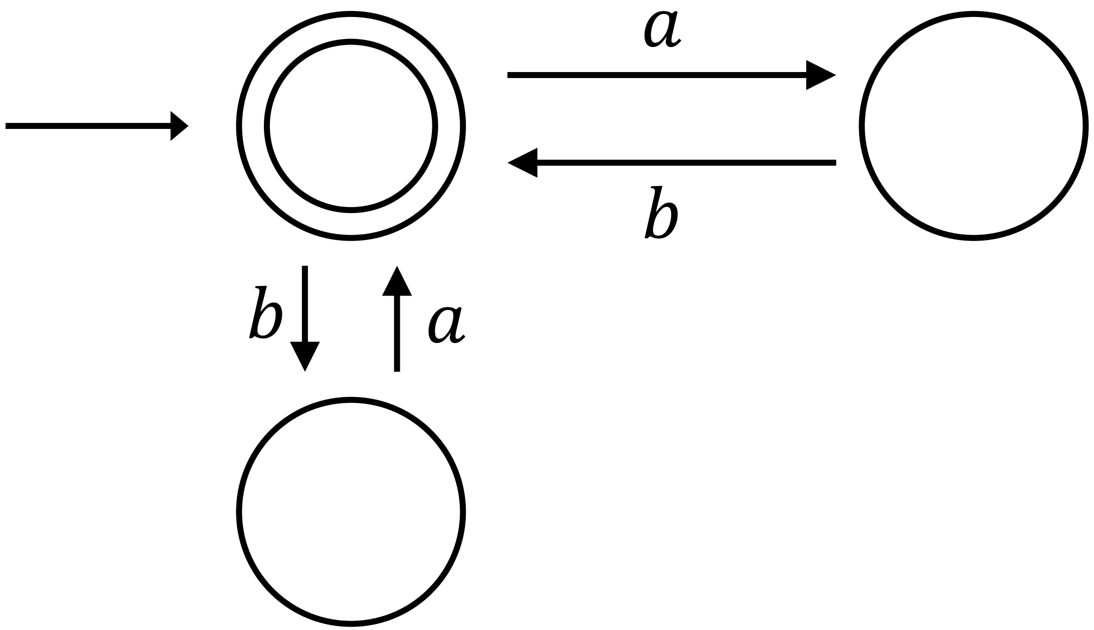
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $r = (ab + ba)^*$



Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $r = (ab + ba)^*$

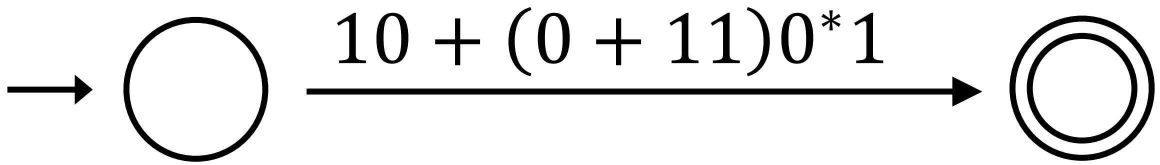


Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$

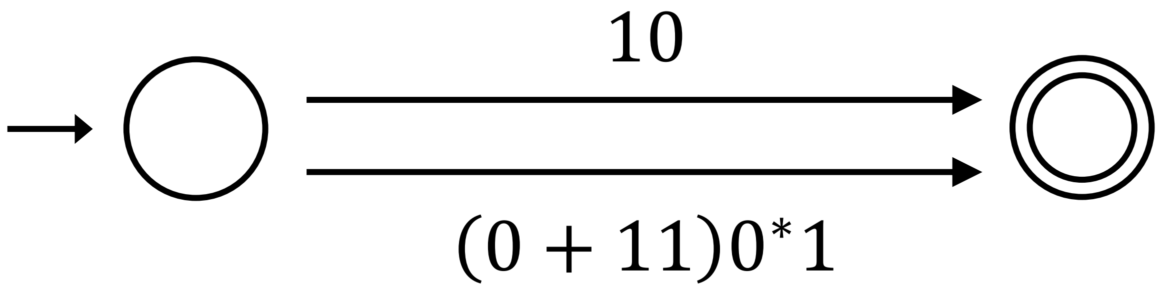
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$



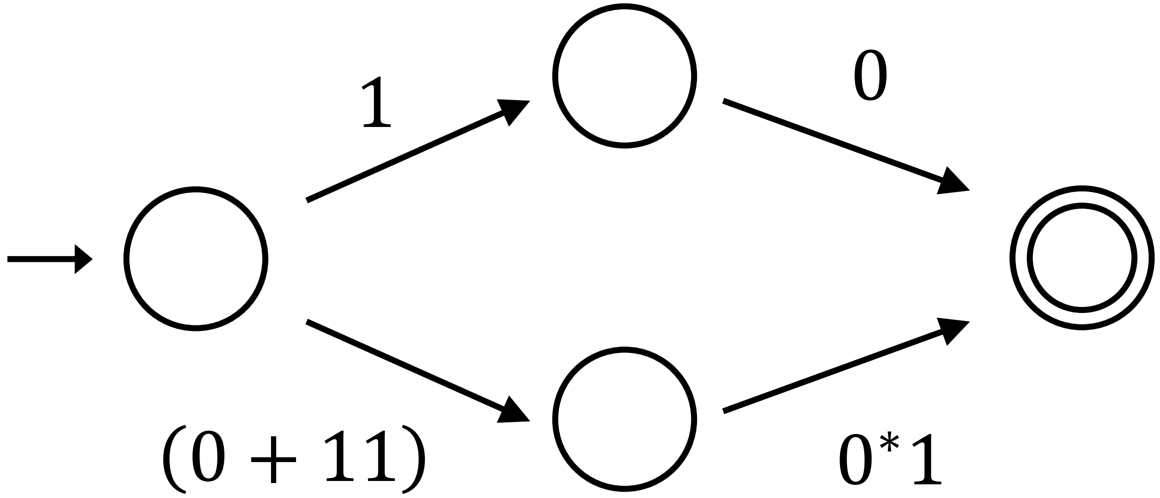
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$



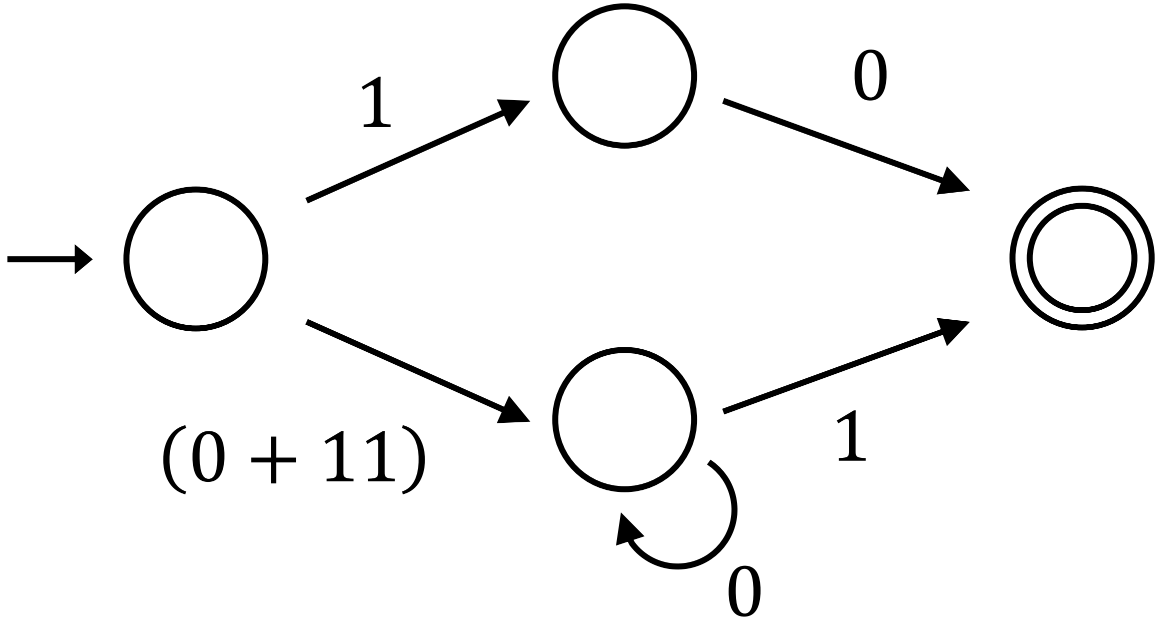
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$



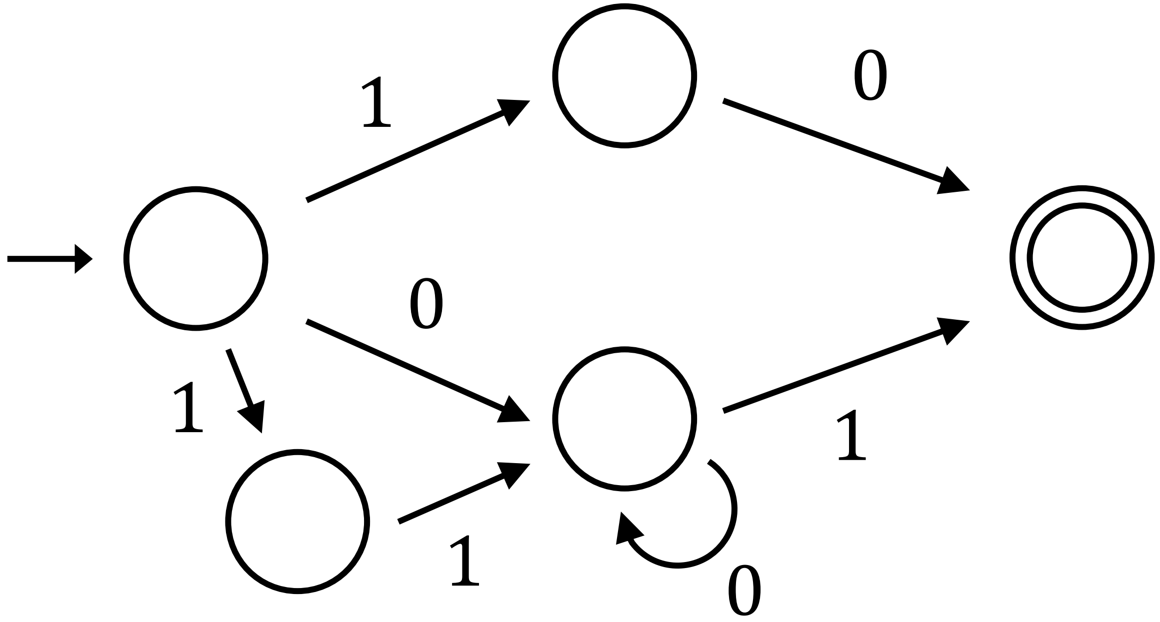
Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$



Regular expression and finite automata

- **How can we represent regular expressions in finite automata?**
  - Example: $10 + (0 + 11)0^*1$

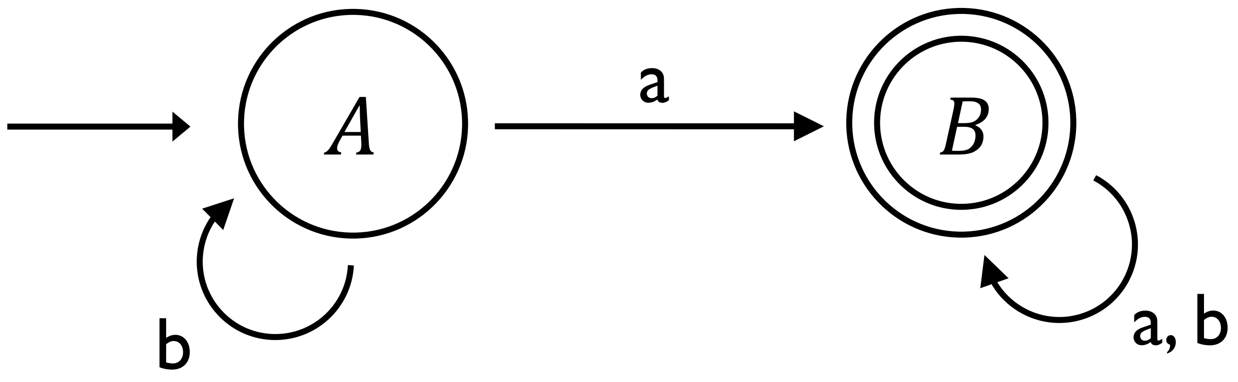


Regular expression and finite automata

- **Practice**
  - Example: $(a+b)^*ab^*a(a+ba)^*$

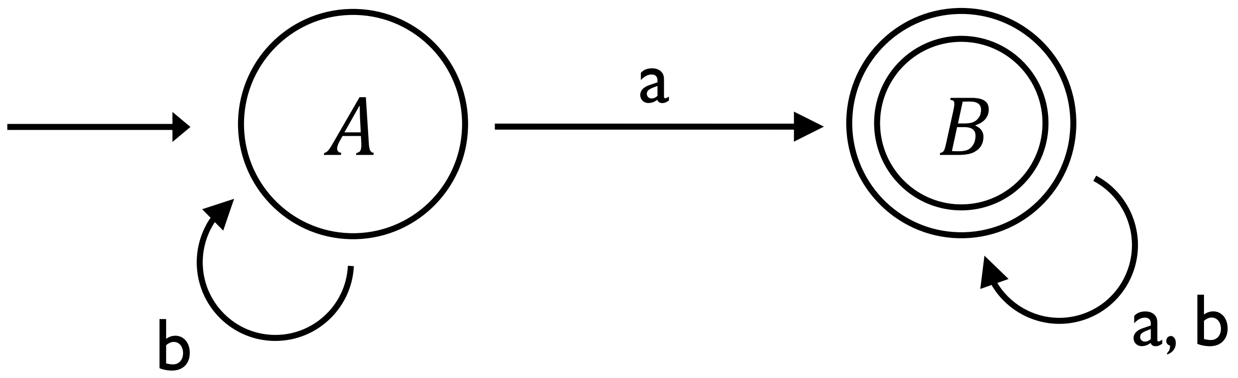
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Example



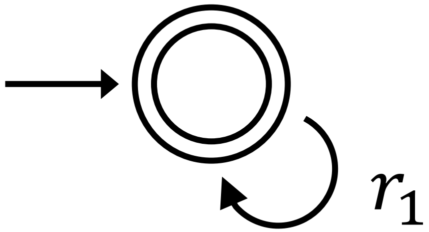
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Example
    - $b^*a(a+b)^*$



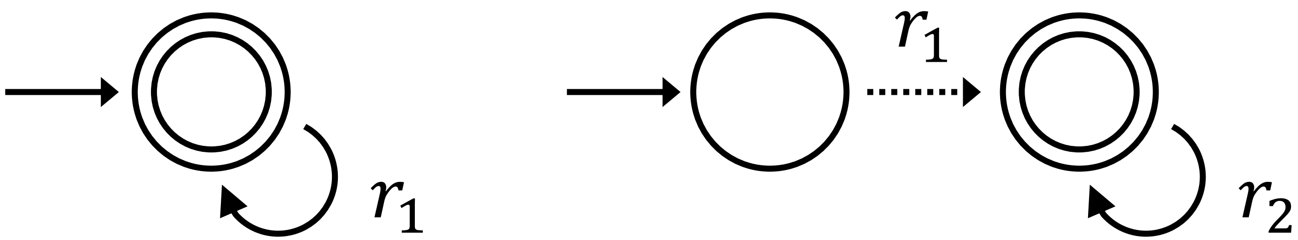
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Basic idea



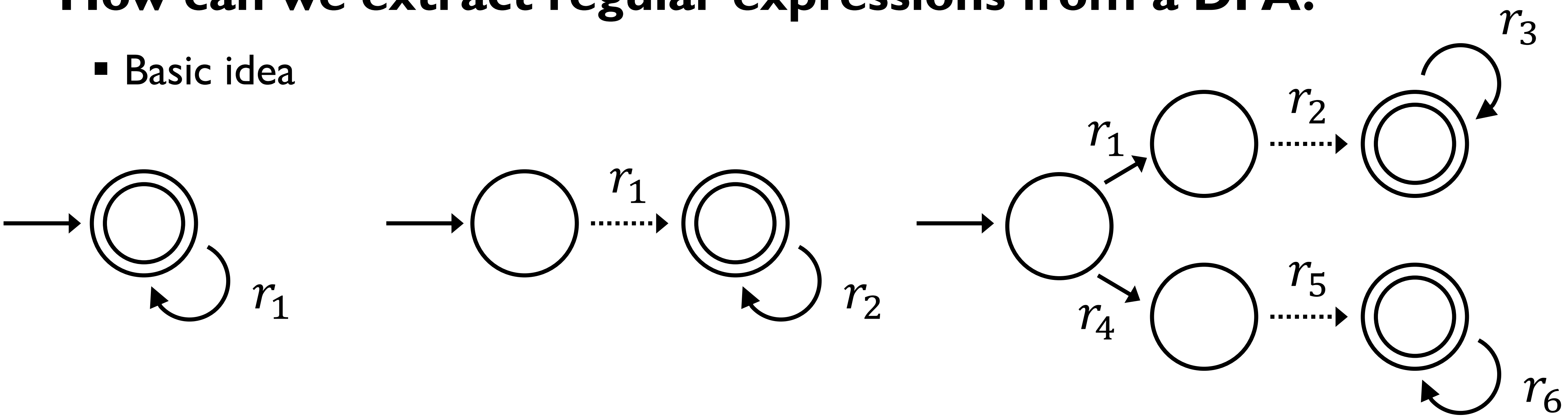
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Basic idea



DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Basic idea

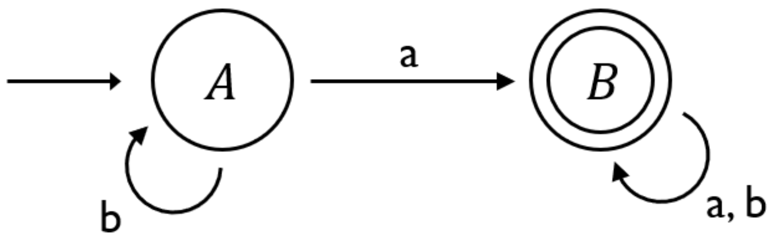


DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem (rule)
    - If P and Q are Regular Expressions over Σ,
      - Then the following equation in R given by $R = Q + RP$ has a unique solution $R = QP^*$

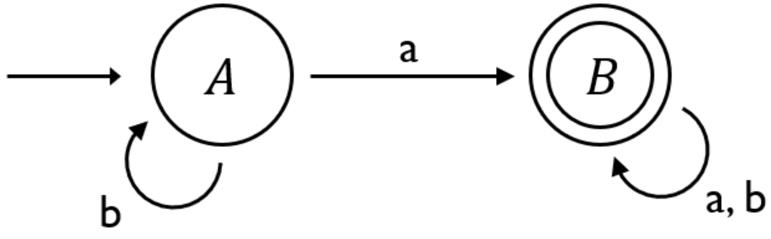
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem (rule)
    - If P and Q are Regular Expressions over Σ,
      - Then the following equation in R given by $R = Q + RP$ has a unique solution $R = QP^*$



DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem (rule)
    - ❖ If P and Q are Regular Expressions over Σ,
      - Then the following equation in R given by $R = Q + RP$ has a unique solution $R = QP^*$

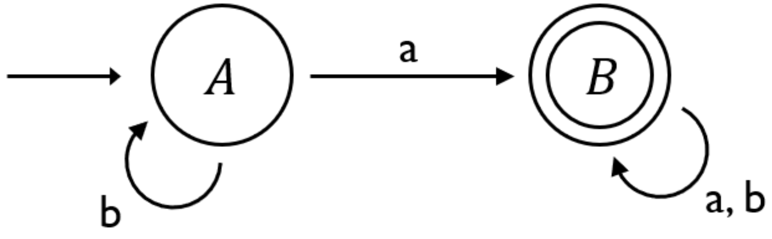


$$A = Ab + \lambda$$
$$B = Aa + Ba + Bb$$

※ Considering incoming edges

DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem (rule)
    - If P and Q are Regular Expressions over Σ,
      - Then the following equation in R given by $R = Q + RP$ has a unique solution $R = QP^*$

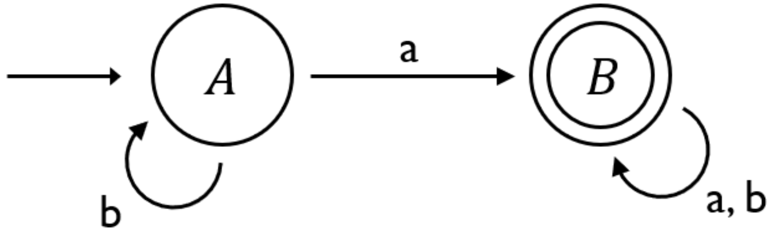


$$A = Ab + \lambda \qquad \Rightarrow A = \lambda \cdot b^* = b^*$$

$$B = Aa + Ba + Bb$$

DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem (rule)
    - If P and Q are Regular Expressions over Σ,
      - Then the following equation in R given by $R = Q + RP$ has a unique solution $R = QP^*$



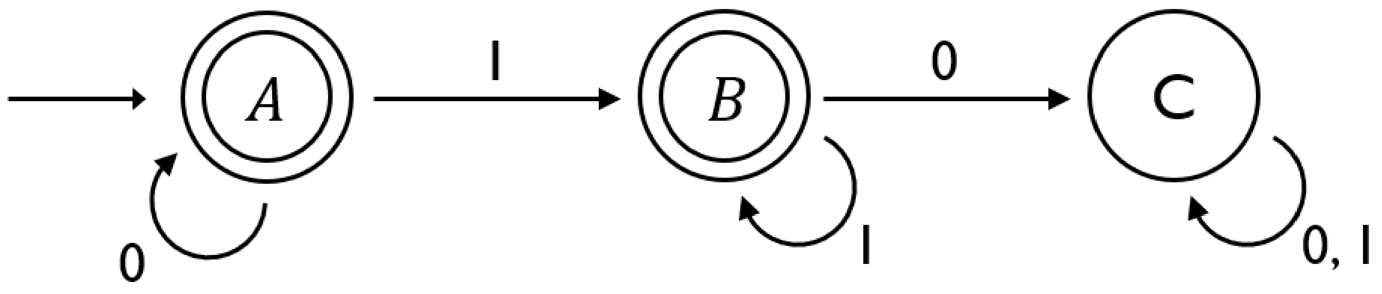
$$A = Ab + \lambda \qquad \Rightarrow A = \lambda \cdot b^* = b^*$$

$$B = Aa + Ba + Bb \qquad \Rightarrow B = b^*a + Ba + Bb$$

$$= \boldsymbol{b^*a(a+b)^*}$$

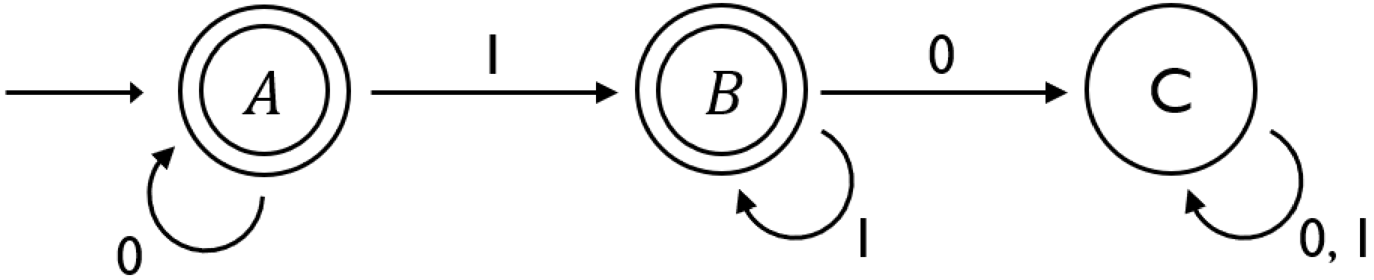
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



DFA to regular expressions

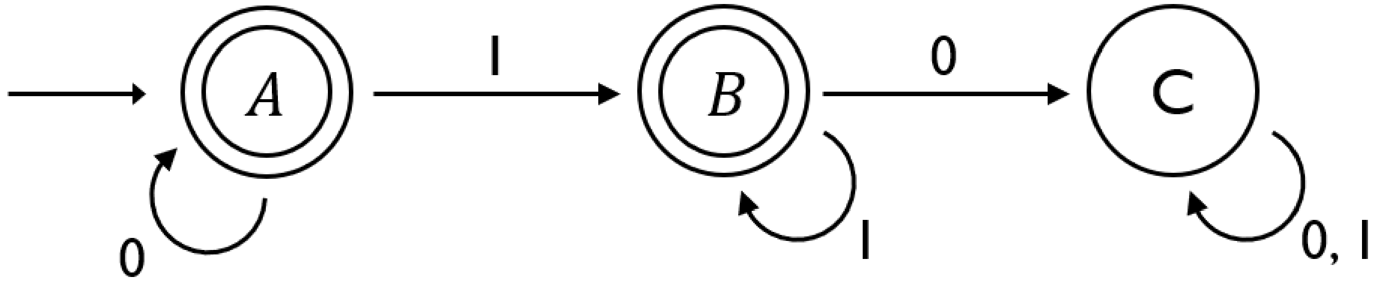
- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



$$A = A0 + \lambda$$
$$B = A1 + B1$$
$$C = B0 + C0 + C1$$

DFA to regular expressions

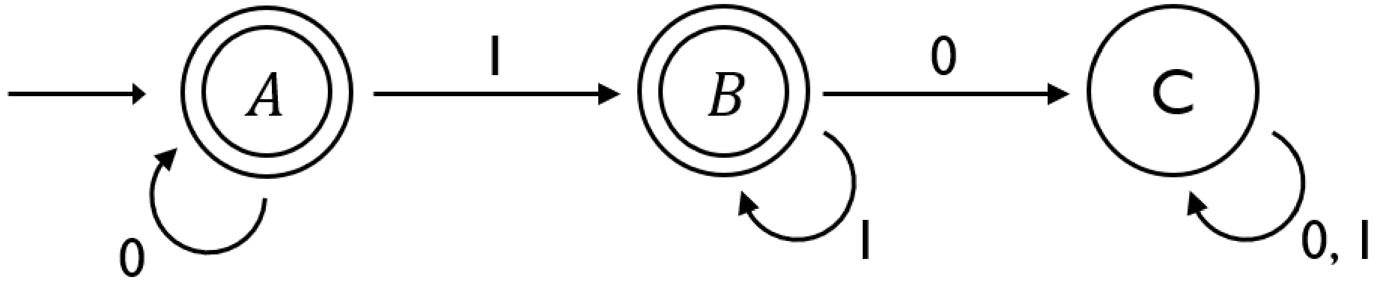
- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



$$A = A0 + \lambda \qquad A = 0^*$$
$$B = A1 + B1$$
$$C = B0 + C0 + C1$$

DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$

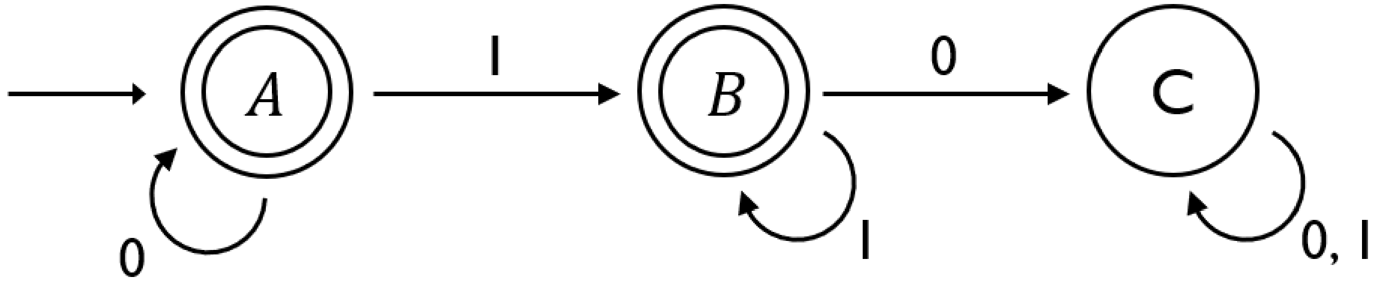


$$A = A0 + \lambda$$
$$B = A1 + B1$$
$$C = B0 + C0 + C1$$

$$A = 0^*$$
$$B = B1 + 0^*1 = 0^*11^* \; (0^*1^+)$$

DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



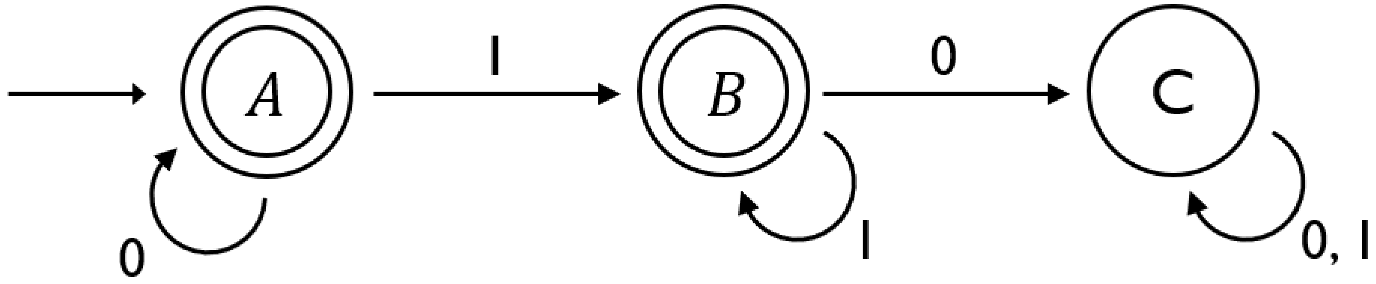
$$A = A0 + \lambda \qquad A = 0^*$$

$$B = A1 + B1 \qquad B = B1 + 0^*1 = 0^*11^* \ (0^*1^+)$$

$$C = B0 + C0 + C1 \qquad C \Rightarrow Trap\ state!$$

DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$

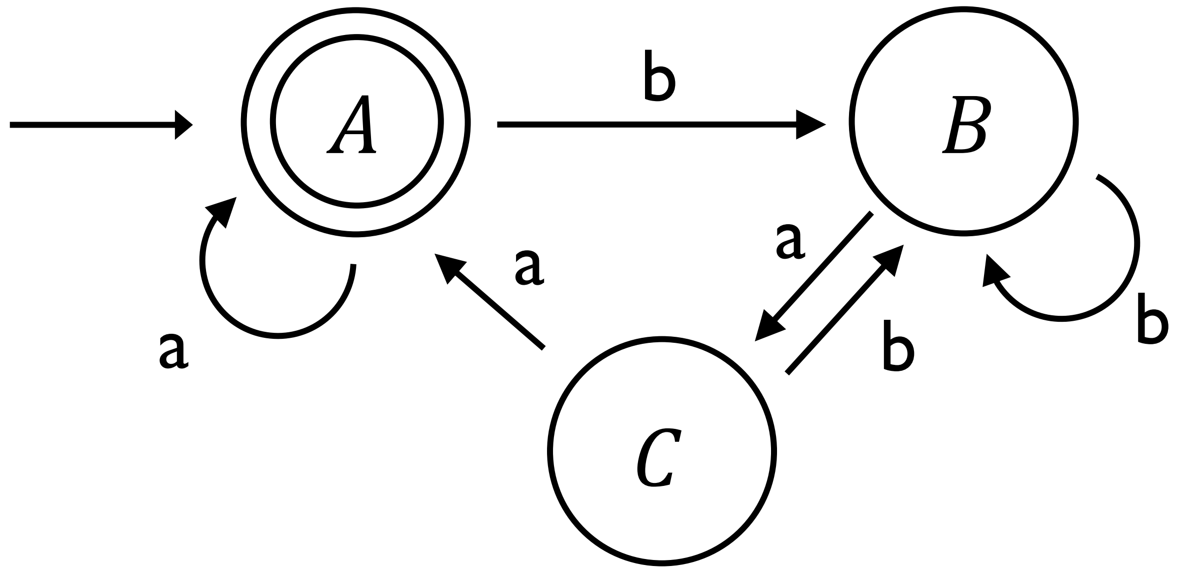


$$A = A0 + \lambda$$
$$B = A1 + B1$$
$$C = B0 + C0 + C1$$

$$A = 0^*$$
$$B = B1 + 0^*1 = 0^*11^* \ (0^*1^+)$$
$$\boldsymbol{A + B = 0^* + 0^*11^*}$$

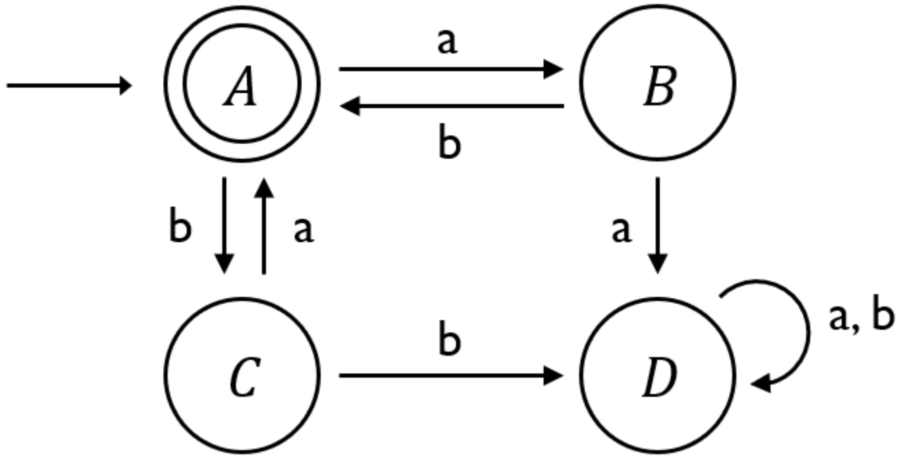
DFA to regular expressions

- **How can we extract regular expressions from a DFA?**
  - Another example
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



DFA to regular expressions

- **Practice**
  - Arden's theorem with multiple final states
    - ❖ R given by $R = Q + RP$ has a unique solution $R = QP^*$



Regular grammars

- **Regular languages can be described by using certain grammars**
  - First, introducing Right-Linear and Left-Linear grammars

Languages & grammars

- **Grammar (G)**
  - A set of rules used to define the structure of the strings in a language
  - G = (V, T, S, P)
    - V: Set of variables (non-empty)
    - T: Set of terminal symbols (non-empty; V and T are disjoint)
    - S: Start variable ($S \in V$)
    - P: Set of productions

Regular grammars

- **Regular languages can be described by using certain grammars**
  - First, introducing Right-Linear and Left-Linear grammars
    - A grammar G = (V,T, S, P) is said to be **right-linear** if all productions are of the form
      - $A \to xB$,
      - $A \to x$,

      where $A, B \in V$ and $x \in T^*$
    - A grammar G = (V,T, S, P) is said to be **left-linear** if all productions are of the form
      - $A \to Bx$,
      - $A \to x$
  - A regular grammar is one that is either right-linear or left-linear

Regular grammars

- **Regular languages can be described by using certain grammars**
  - Right-Linear
    - ❖ E.g., The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as $S \rightarrow abS \mid a$ is right-linear
  - Left-Linear
    - ❖ E.g., The grammar $G_2 = (\{S\}, \{a, b\}, S, P_2)$, with P_2 given as $S \rightarrow Sab \mid b$ is left-linear
  - Not a regular grammar
    - ❖ E.g., The grammar $G_3 = (\{S, A, B\}, \{a, b\}, S, P_3)$, with P_3 given as
      - $S \rightarrow A$
      - $A \rightarrow aB \mid \lambda$
      - $B \rightarrow Ab$

      is not regular

Regular grammars

- **Regular grammars to finite automata**
  - Construct a finite automaton recognizing $L(G)$ where $G = (\{S, A\}, \{a, b\}, S, P)$ with P given as
    - ❖ $S \rightarrow aS \mid bA \mid b$
    - ❖ $A \rightarrow aA \mid bS \mid a$

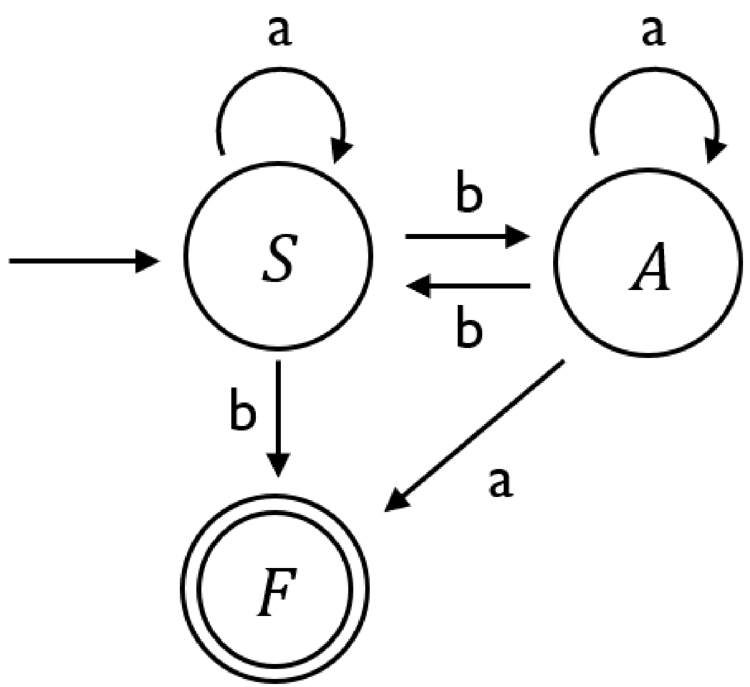
Regular grammars

- **Regular grammars to finite automata**
  - Construct a finite automaton recognizing $L(G)$ where $G = (\{S, A\}, \{a, b\}, S, P)$ with P given as
    - ❖ $S \rightarrow aS \mid bA \mid b$
    - ❖ $A \rightarrow aA \mid bS \mid a$

1. Each production $A_i \rightarrow aA_j$ induces a transition from q_i to q_j with label a
2. Each production $A_k \rightarrow a$ induces a transition from q_k to q_f (final) with label a

Regular grammars

- **Regular grammars to finite automata**
  - Construct a finite automaton recognizing $L(G)$ where $G = (\{S, A\}, \{a, b\}, S, P)$ with P given as
    - ❖ $S \rightarrow aS \mid bA \mid b$
    - ❖ $A \rightarrow aA \mid bS \mid a$

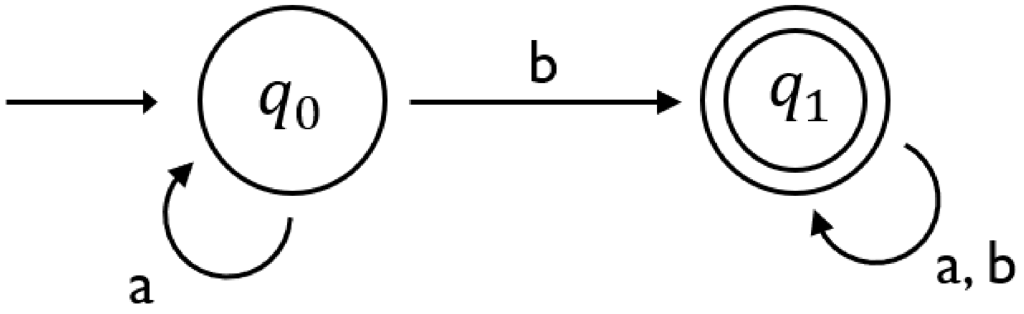


Regular grammars

- **Practice**
  - Construct a finite automaton recognizing $L(G)$ where $G = (\{S, A\}, \{a, b\}, S, P)$ with P given as
    - $S \rightarrow aS \mid bS \mid aA$
    - $A \rightarrow bB$
    - $B \rightarrow aC$
    - $C \rightarrow a$

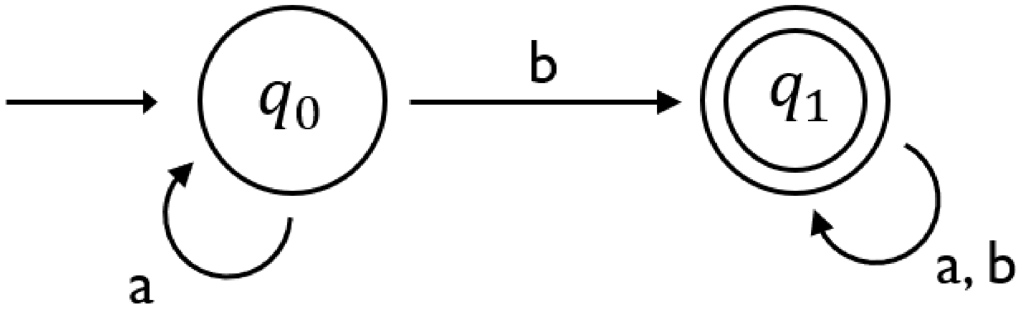
Regular grammars

- **DFA to regular grammar**
  - Construct a regular grammar from given DFA



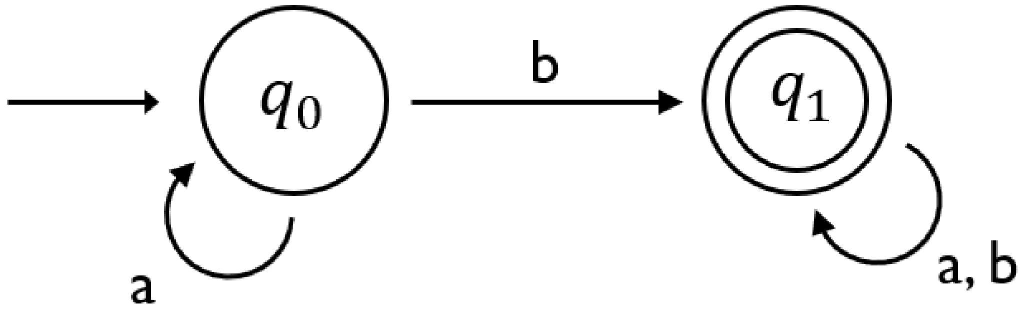
Regular grammars

- **DFA to regular grammar**
  - Construct a regular grammar from given DFA
    - ❖ $A_i \to aA_j$ constructed if $(q_i, a) = q_j$ where $q_j \notin F$
    - ❖ $A_i \to aA_j$ and $A_i \to a$ constructed if $(q_i, a) = q_j$ where $q_j \in F$

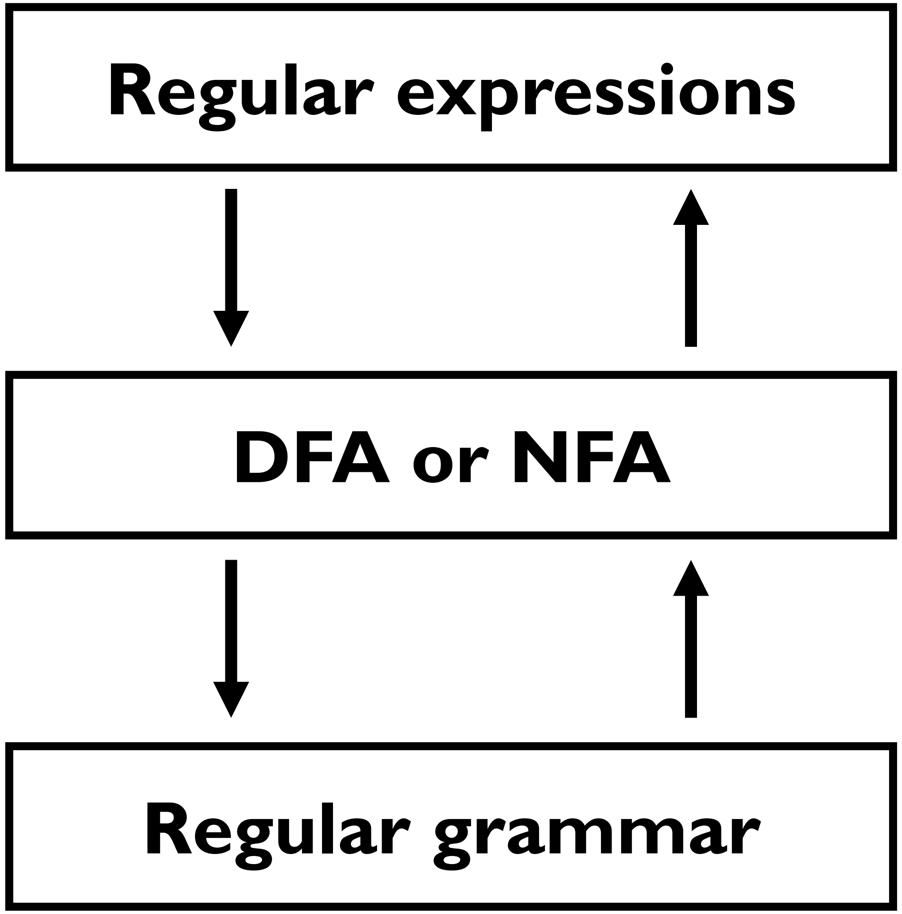


Regular grammars

- **DFA to regular grammar**
  - Construct a regular grammar from given DFA
    - ❖ $G = (V, T, P, S)$
      - $V = \{S, A\}$
      - $T = \{a, b\}$
      - P
        - $S \rightarrow aS \mid bA \mid b$
        - $A \rightarrow aA \mid bA \mid a \mid b$



Description of regular language



Next Lecture

- **Properties of Regular Languages**