Data Structures and Algorithms 2

Marquette University 2018

Regular Expressions and Finite State Systems

Transition Diagram

A man with a pet-wolf, a pet-goat, and a cabbage wants to cross a river in a boat that can only carry him and one passenger. If the man leaves the wolf and the goat alone, the wolf will eat the goat. If the man leaves the goat and the cabbage alone, then the goat will eat the cabbage. How can the man transport all three possessions to the other side of the river?

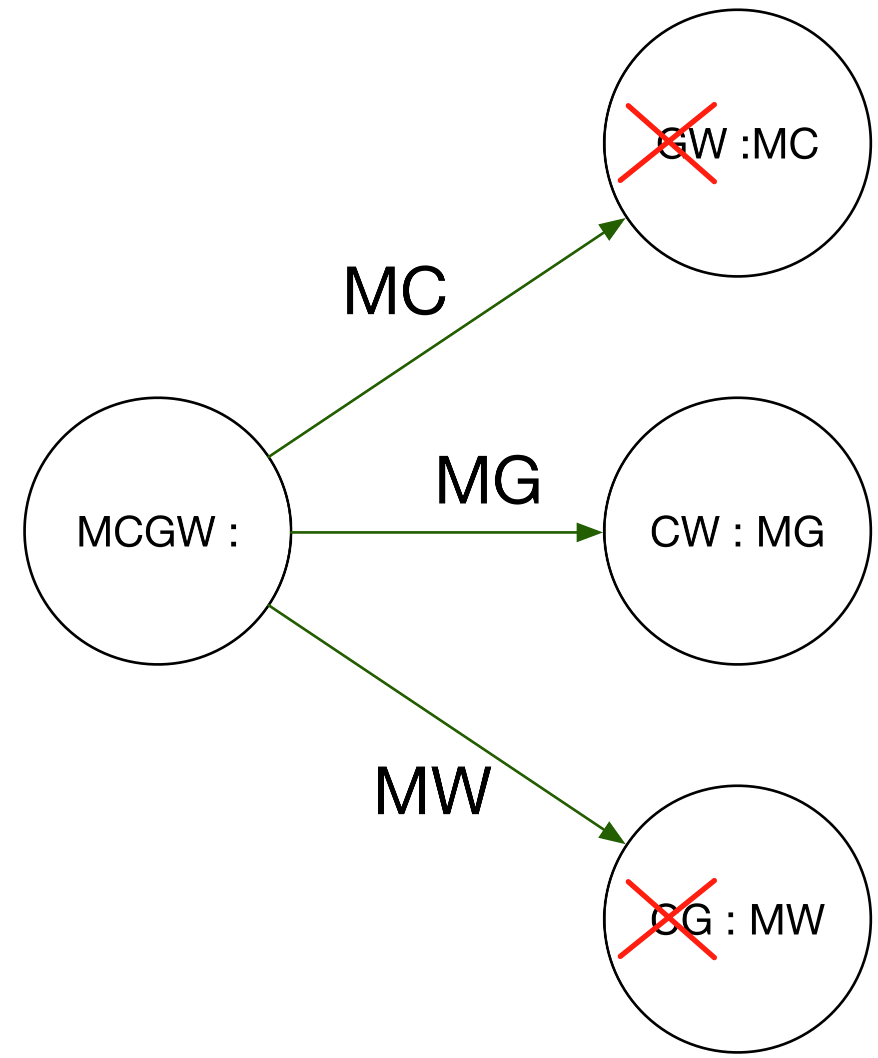
Transition Diagram

A man with a pet-wolf, a pet-goat, and a cabbage wants to cross a river in a boat that can only carry him and one passenger. If the man leaves the wolf and the goat alone, the wolf will eat the goat. If the man leaves the goat and the cabbage alone, then the goat will eat the cabbage. How can the man transport all three possessions to the other side of the river?

- Introduce a set of states describing possible configurations
- ab:cd
  - a and b are on the original side of the river
  - c and d are on the other side
  - M — man, W — wolf, G— goat, C — cabbage

Transition Diagram

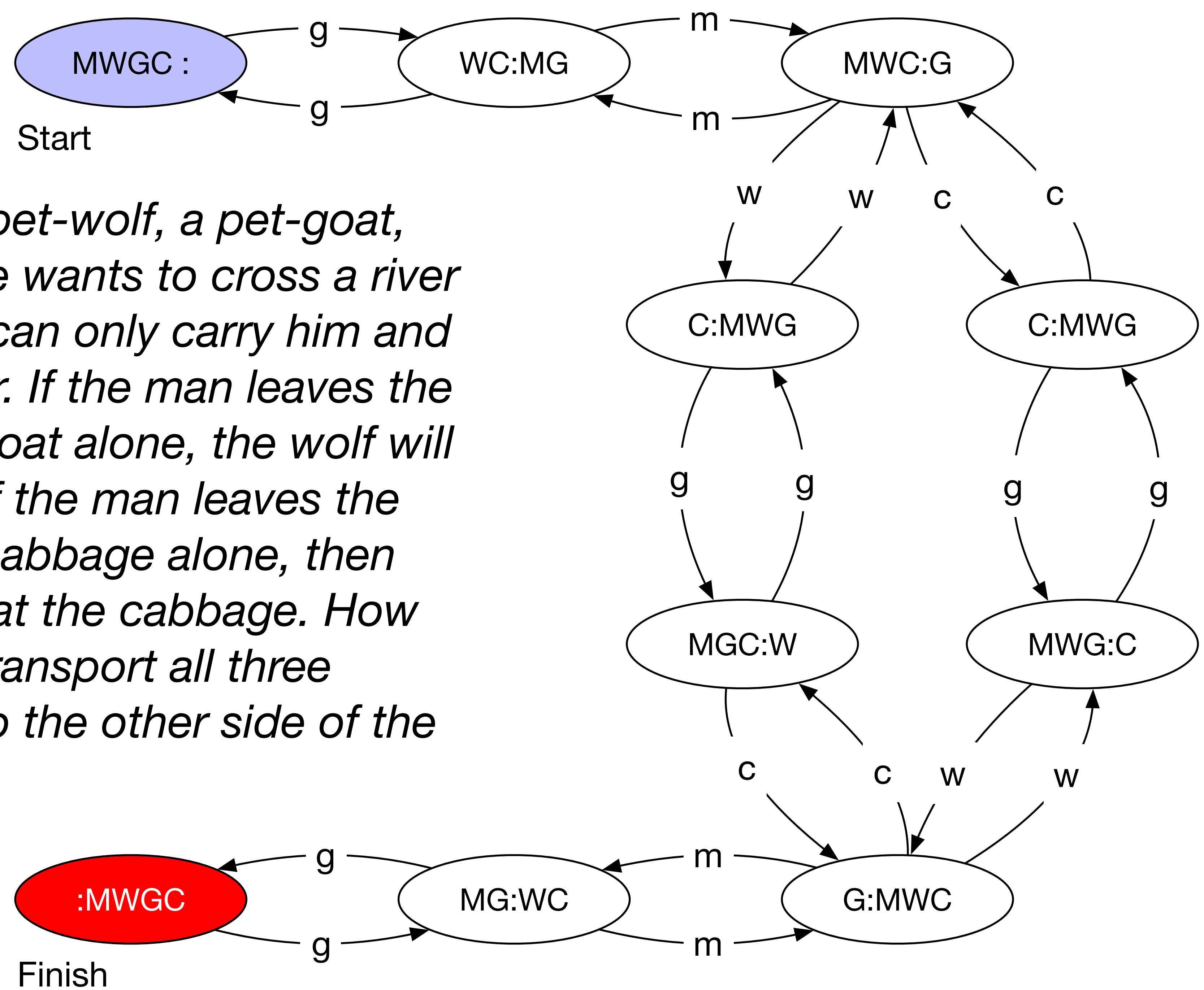
- Original State : MCGW
- Three possible transaction: Men moves cabbage, goat, or wolf to other side
- Results in three different states, but not all of them are feasible because without the man, the wolf eats the goat and the goat eats the cabbage.



Transition Diagram

- Transition diagram
  - States are circles
  - Transitions move system from one state to the other
  - Each movement is associated with a letter
    - The letter is the pet that the man selects to transport from one side of the river to the other

Transition Diagrams



A man with a pet-wolf, a pet-goat, and a cabbage wants to cross a river in a boat that can only carry him and one passenger. If the man leaves the wolf and the goat alone, the wolf will eat the goat. If the man leaves the goat and the cabbage alone, then the goat will eat the cabbage. How can the man transport all three possessions to the other side of the river?

Finite Automata

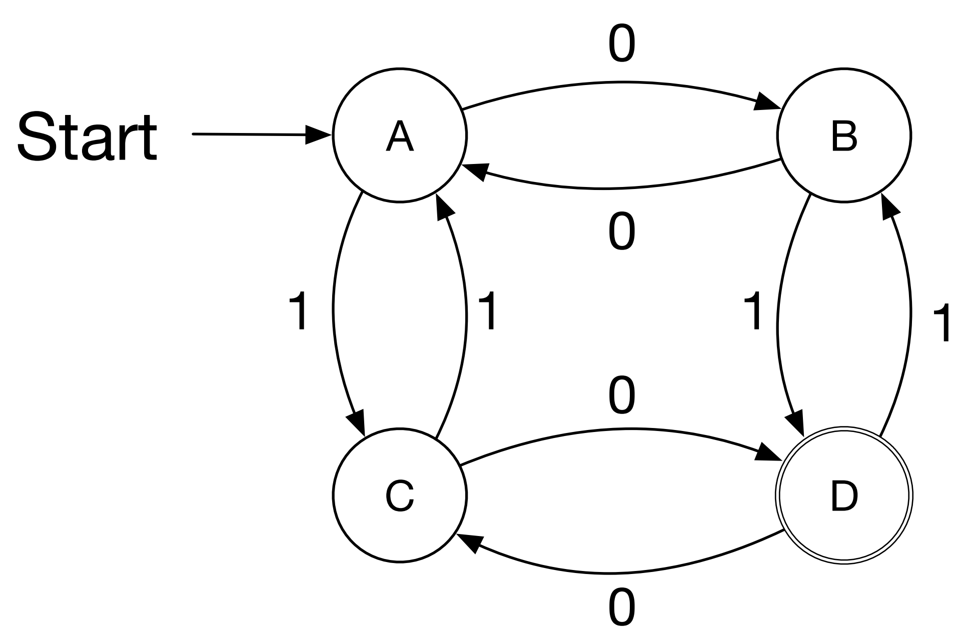
- A finite automaton consists of
  - A finite set of states
  - A finite alphabet of inputs
  - An initial state
  - A set of final states
  - A set of transitions
    - Each transition is between two states and labelled with an input. Only one transition with a certain label can leave a state.
- A string is accepted by a finite automaton if it corresponds to a path from the starting state to a final state

Finite Automata

- Formal definition
  - A finite automaton is a quintuple $(Q, q_0, Q_f, \Sigma, \delta)$
    - where Q is a finite “set of states”
    - $q_0 \in Q$ is the “start state”
    - $Q_f \subset Q$ is the “set of final or accepting states”
    - Σ is a finite set, the “alphabet”
    - $\delta : Q \times \Sigma \to Q$ is the “transition function”

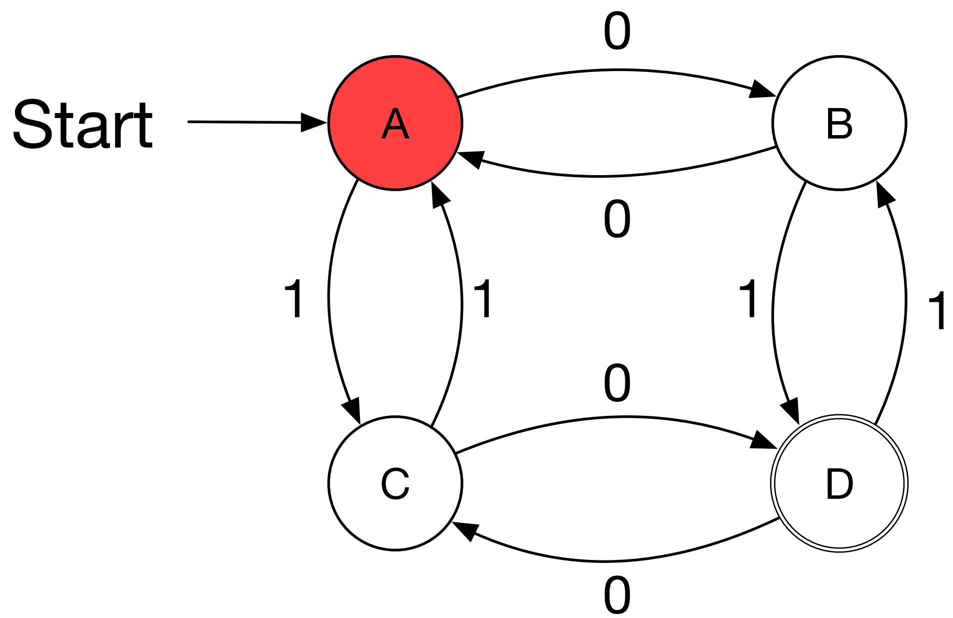
Finite Automata

- A sequence of letters is processed by a series of transitions.
  - Assume the following automaton and the series 0001001011.



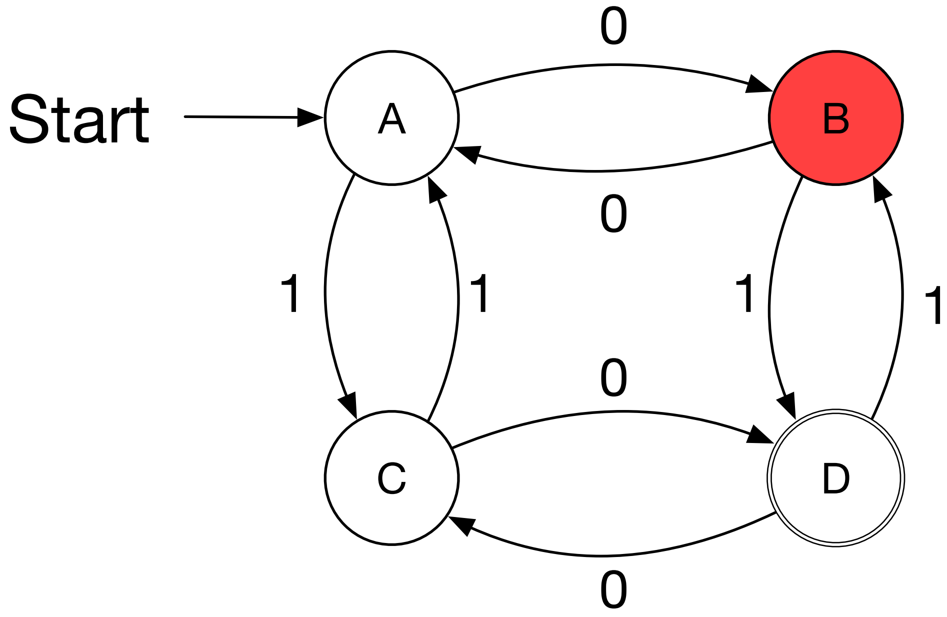
Finite Automata

Before processing anything of
0001001011

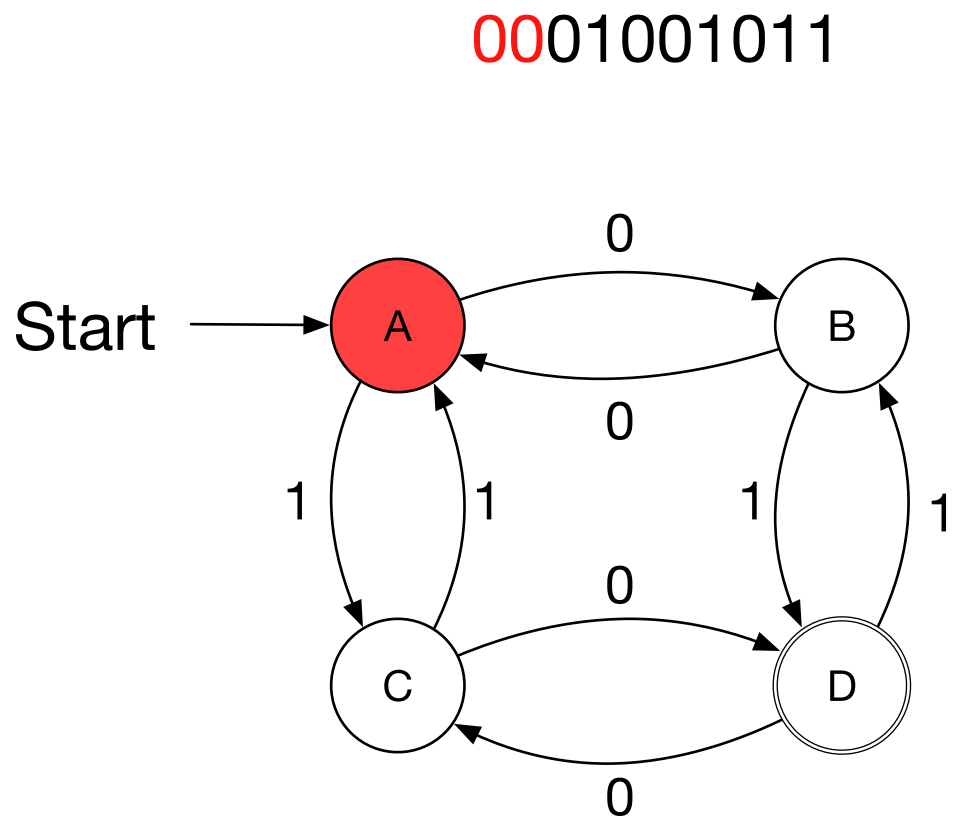


Finite Automata

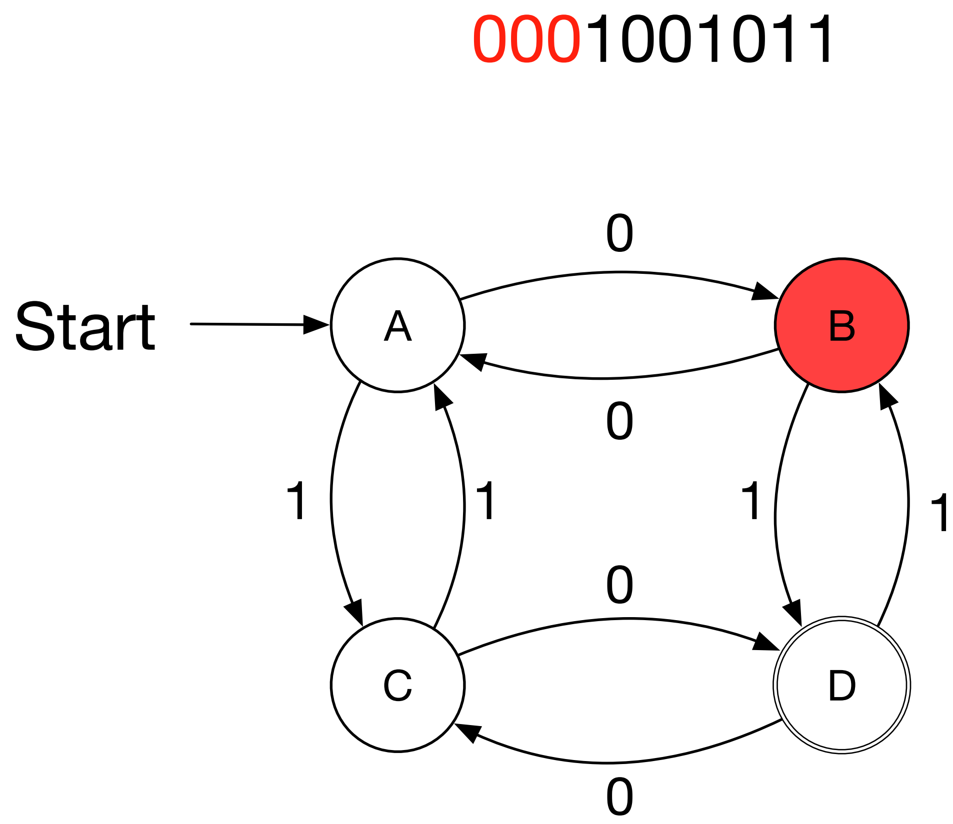
0001001011



Finite Automata

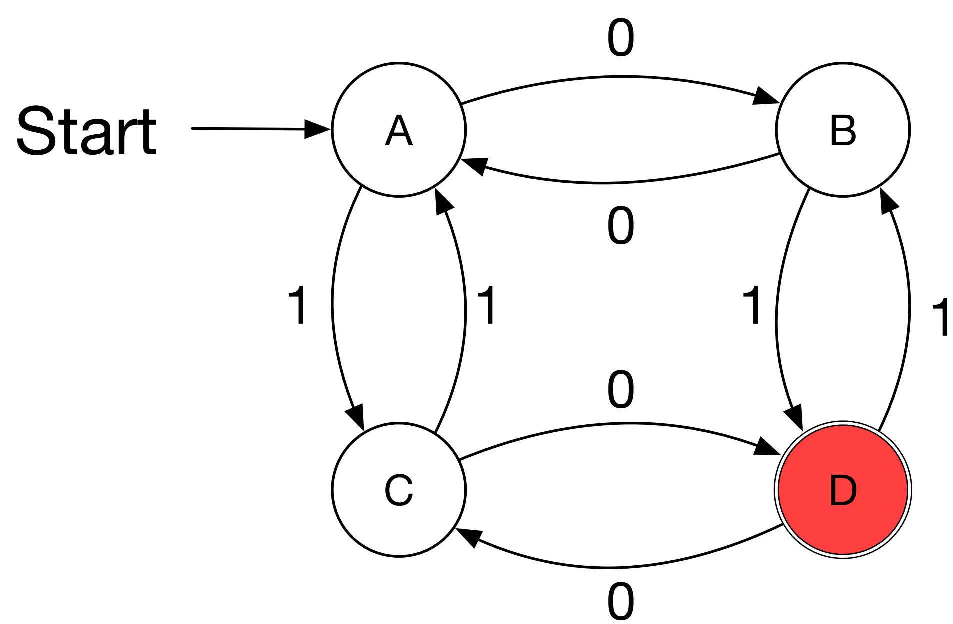


Finite Automata



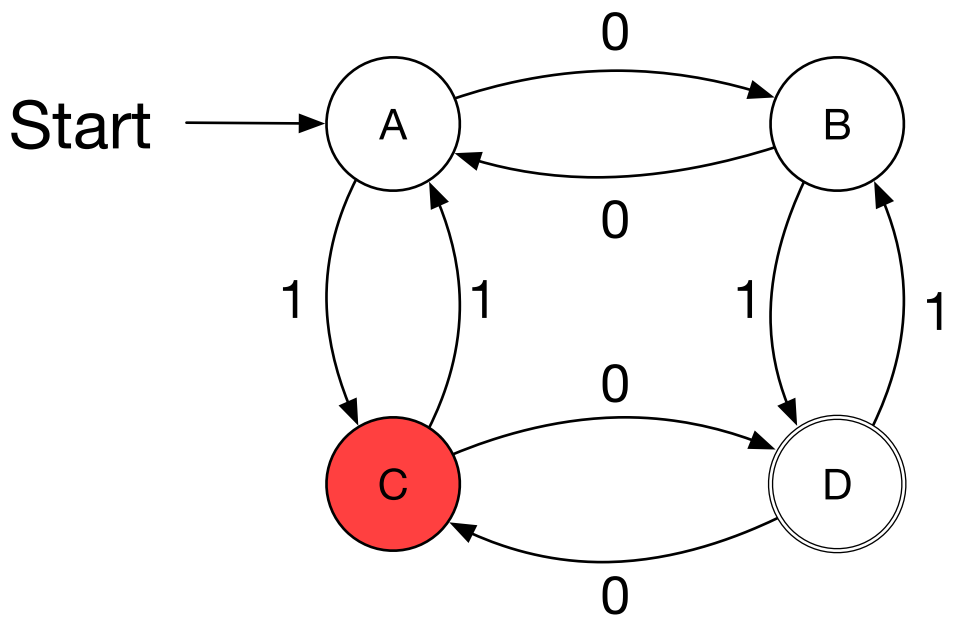
Finite Automata

0001001011



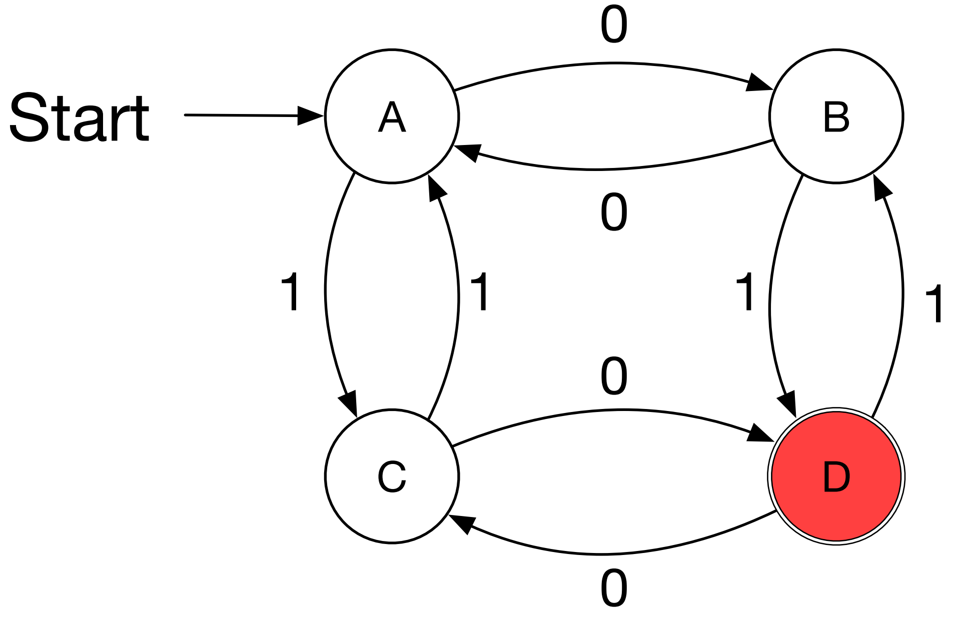
Finite Automata

0001001011



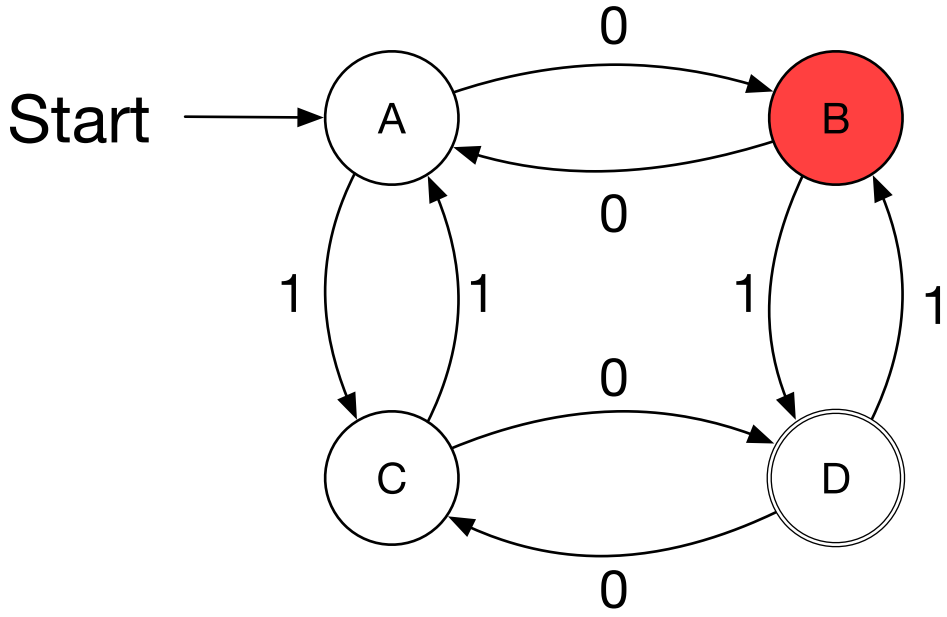
Finite Automata

0001001011



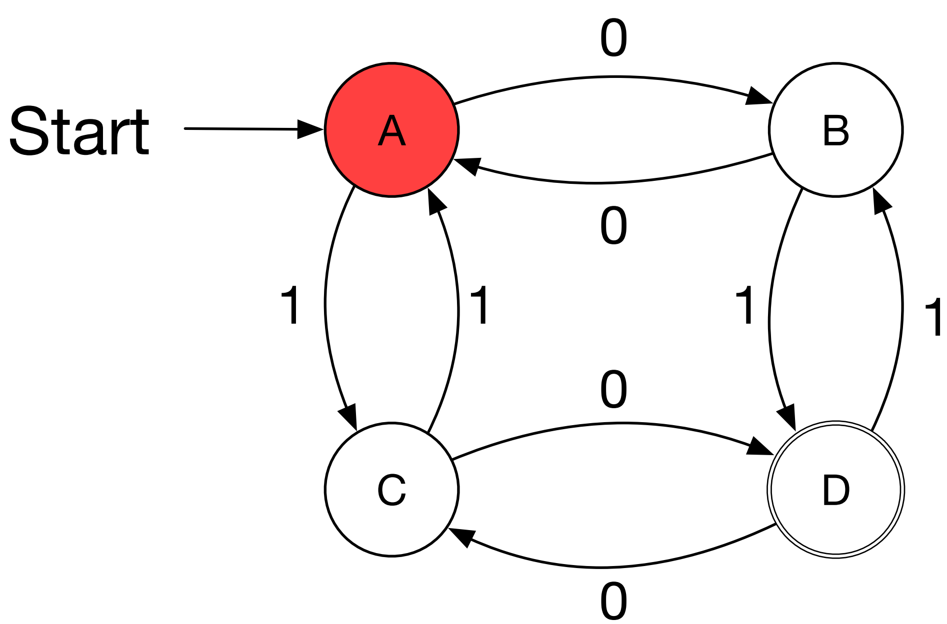
Finite Automata

0001001011



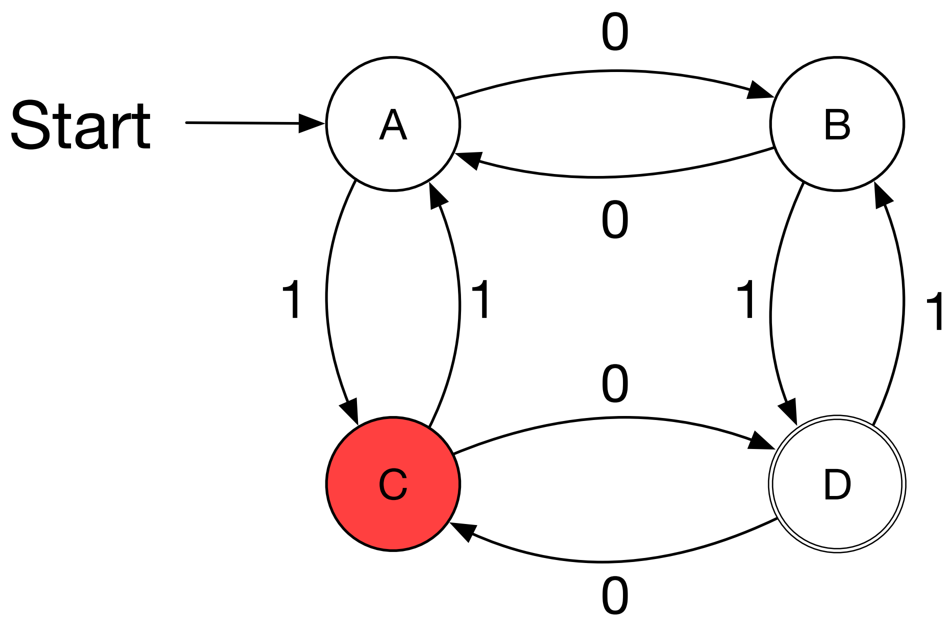
Finite Automata

00010010 11



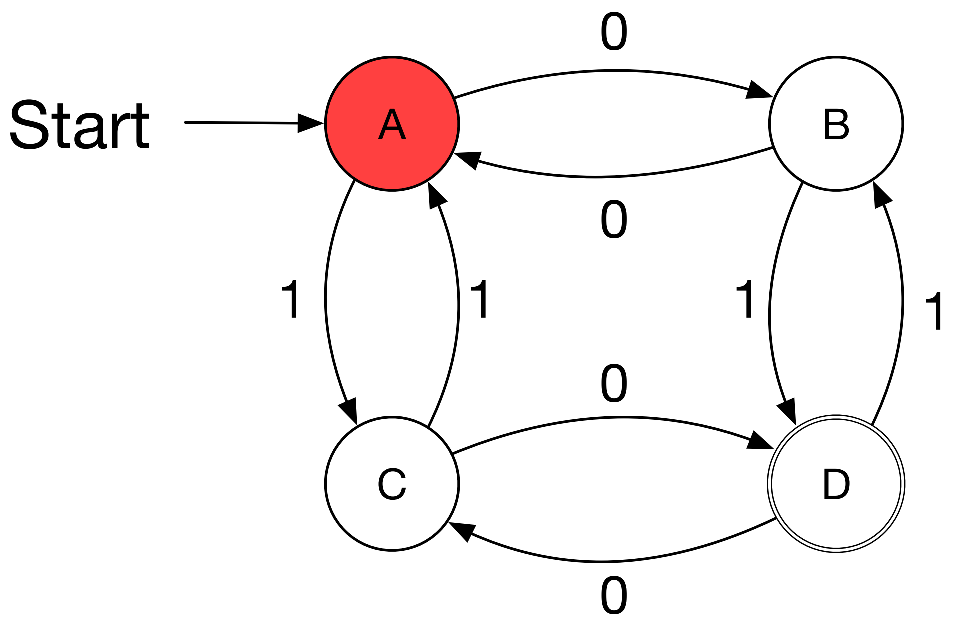
Finite Automata

0001001011



Finite Automata

0001001011



Finite Automata

- We can extend this to create a mapping
  - Argument:
    - Any state
    - A string
  - Image
    - The state in which we end up processing the string from the state

Finite Automata

- Let Σ^* be the set of strings with letters in the alphabet Σ
  - ϵ is the empty string
- Extend δ to $\hat{\delta} : Q \times \Sigma^* \to Q$ by defining

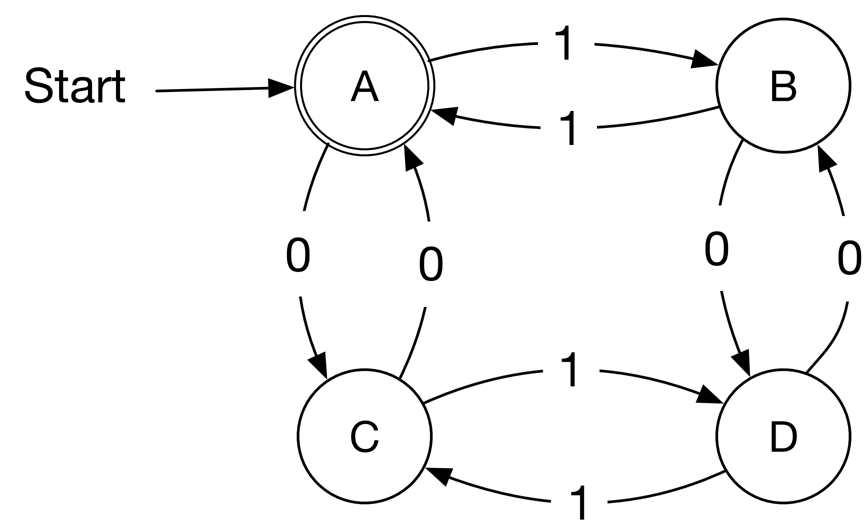
$$\forall q \in Q : \hat{\delta}(q, \epsilon) = q$$

$$\forall a \in \Sigma \; \forall q \in Q \; : \; \hat{\delta}(q, a) = \delta(q, a)$$

$$\forall w \in \Sigma^* \; \forall a \in \Sigma \; \forall q \in Q \; : \; \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

Finite Automata

- Example:

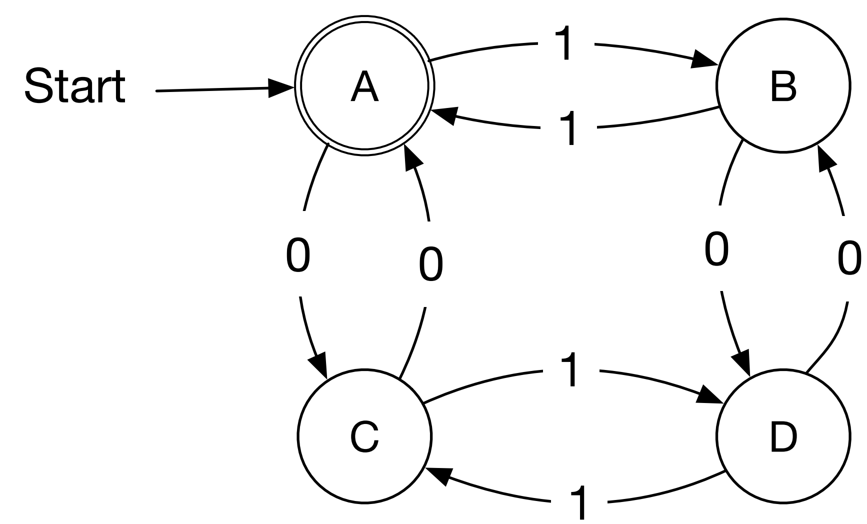


- Then strings of length 2

	A	B	C	D
ϵ	A	B	C	D
0	C	D	A	B
1	B	A	D	D

Finite Automata

- Example:

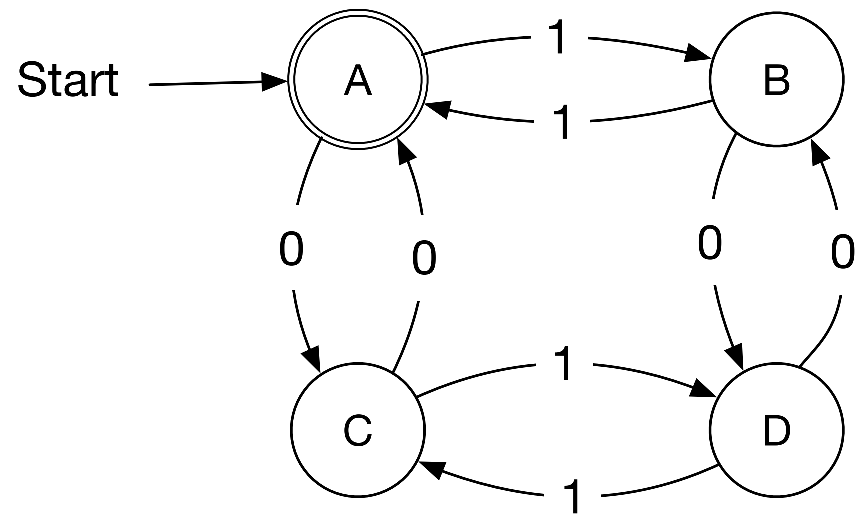


- First for the empty string and strings of length 1

$\hat{\delta}$	A	B	C	D
ϵ	A	B	C	D
0	C	D	A	B
1	B	A	D	D
00	A	B	C	D
01	D	C	B	A
10	D	C	B	A
11	A	B	C	D

Finite Automata

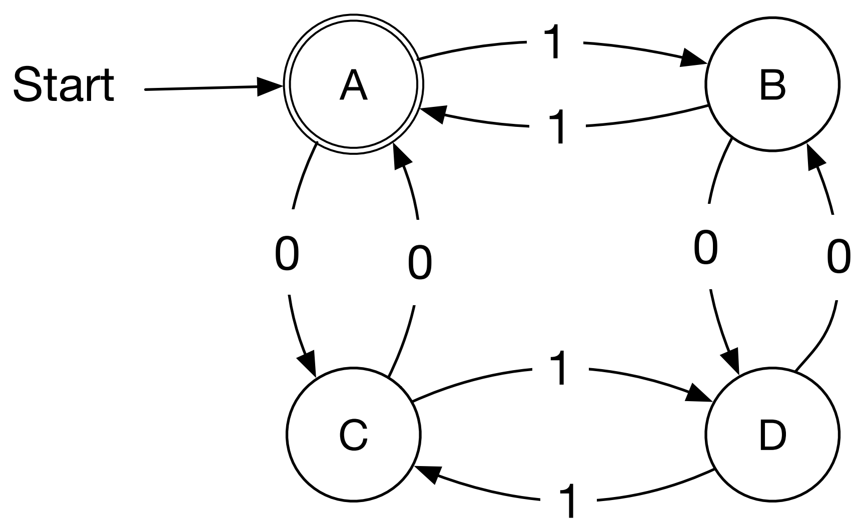
- Example:



- First for the empty string and strings of length 1

$\hat{\delta}$	A	B	C	D
ϵ	A	B	C	D
0	C	D	A	B
1	B	A	D	D
00	A	B	C	D
01	D	C	B	A
10	D	C	B	A
11	A	B	C	D

Finite Automata



$$
\begin{aligned}
\hat{\delta}(A, 0011) &= \delta(\hat{\delta}(A, 001), 1) \\
&= \delta(\delta(\hat{\delta}(A, 00), 1), 1) \\
&= \delta(\delta(\delta(\hat{\delta}(A, 0), 0), 1), 1) \\
&= \delta(\delta(\delta(\delta(A, 0), 0), 1), 1) \\
&= \delta(\delta(\delta(C, 0), 1, 1) \\
&= \delta(\delta(A, 1), 1) \\
&= \delta(B, 1) \\
&= A
\end{aligned}
$$

Finite Automata

- Acceptance
  - A string *w* is accepted by a finite automaton iff

    $$\hat{\delta}(q_0, w) \in Q_f$$

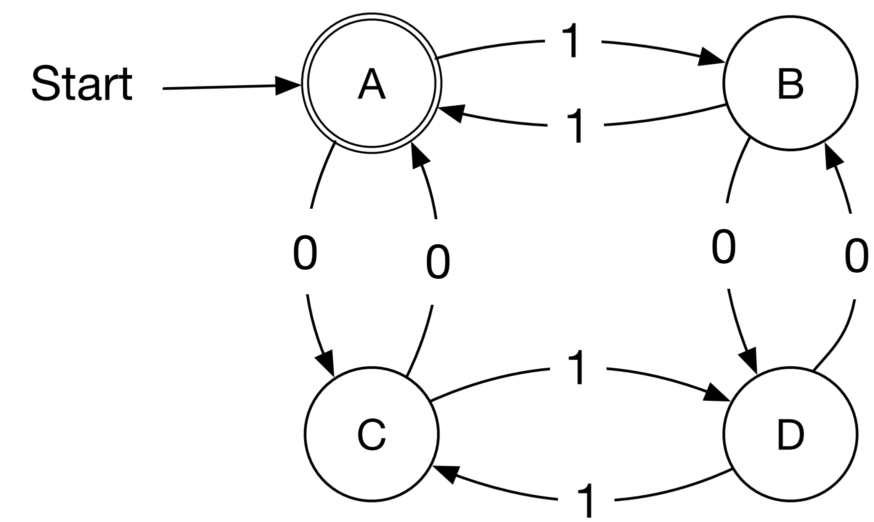
  - With other words, the string transitions from the starting state to an accepting state

Finite Automata

- Programming Finite State Machines
  - Easy if you can use goto and labels
  - Unfortunately, too many language designers decided that you should not be lead into temptation
  - Otherwise:
    - Use an enumeration data structure for the states
    - Express the transition function as a dictionary
      - Or in Java, as an array
    - Keep track of the current state

Finite Automata

- This finite automaton accepts those strings that have an even number of ones and an even number of zeroes
- Lemma: The automaton is in a left state iff it has seen an even number of ones
- Lemma: The automaton is in a upper state iff it has seen an even number of zeroes
- Proofs by induction on the length of a string

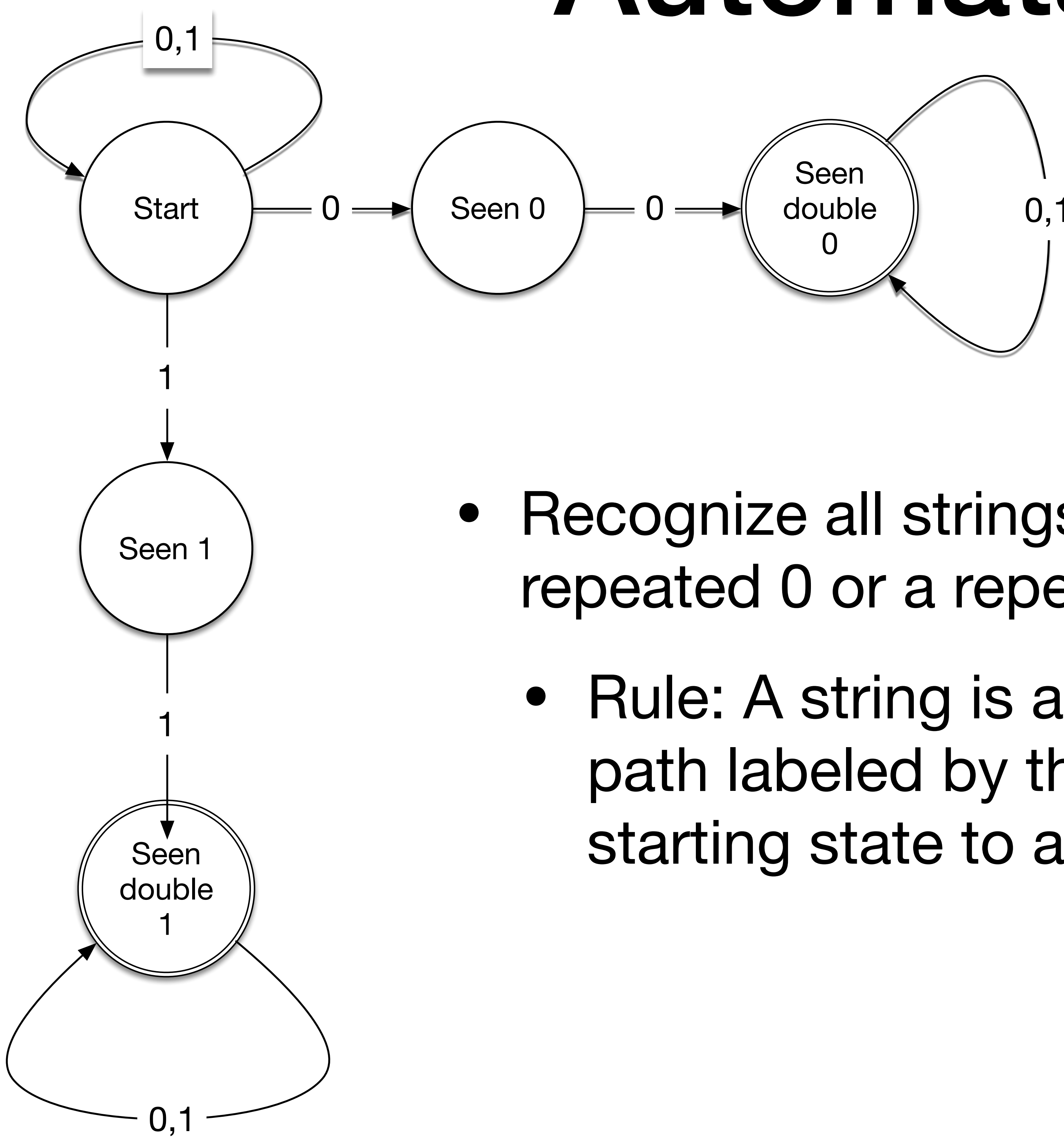


Non-deterministic Finite Automata

- Non-determinism can make it easier to design finite automata
- The transition function can be multivalued
    - It is a function whose values are subsets of Q

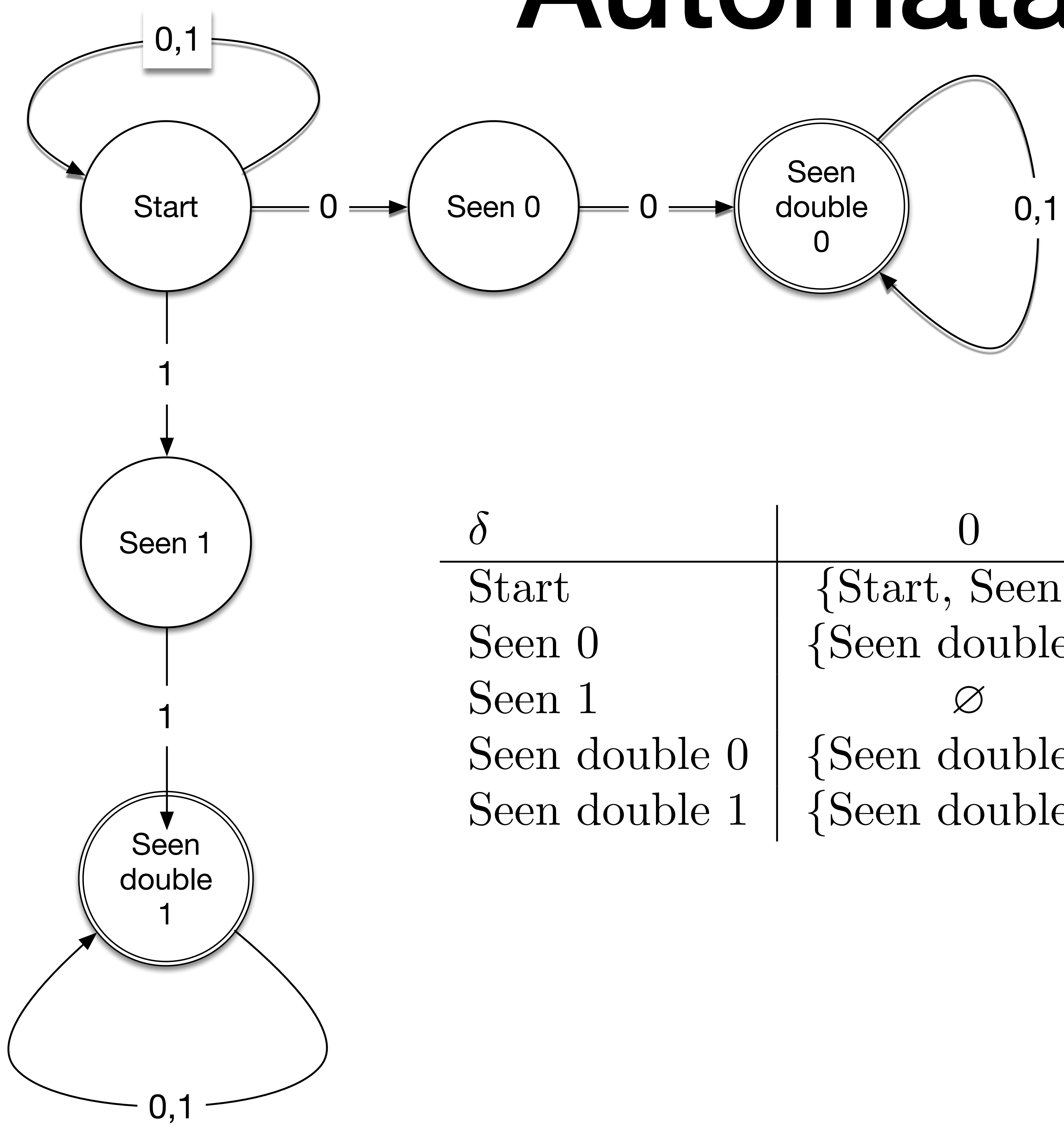
$$\delta : Q \times \Sigma \longrightarrow 2^Q$$

Non-deterministic Finite Automata



- Recognize all strings in $\{0,1\}^*$ with a repeated 0 or a repeated 1
  - Rule: A string is accepted if there is a path labeled by the string from the starting state to an accepting state

Non-deterministic Finite Automata



δ	0	1
Start	{Start, Seen 0}	{Start, Seen 1}
Seen 0	{Seen double 0}	$\varnothing$
Seen 1	$\varnothing$	{ Seen double 1 }
Seen double 0	{Seen double 0}	{ Seen double 0}
Seen double 1	{Seen double 1}	{Seen double 1}

Non-deterministic Finite Automata

- As before, extend transition function to all strings

$$\forall q \in Q \;:\; \hat{\delta}(q, \epsilon) = \{q\}$$

$$\forall a \in \Sigma \; \forall q \in Q \;:\; \hat{\delta}(q, a) = \delta(q, a)$$

$$\forall w \in \Sigma^* \; \forall a \in \Sigma \; \forall q \in Q \;:\; \hat{\delta}(q, wa) =$$

$$\{p \in Q | \exists r \in Q \;:\; r \in \hat{\delta}(q, w) \text{ and } p \in \delta(r, a)\}$$

Non-deterministic Finite Automata

Theorem: Let L be a set accepted by a non-deterministic finite automaton. Then there exists a deterministic finite automaton that also accepts L.

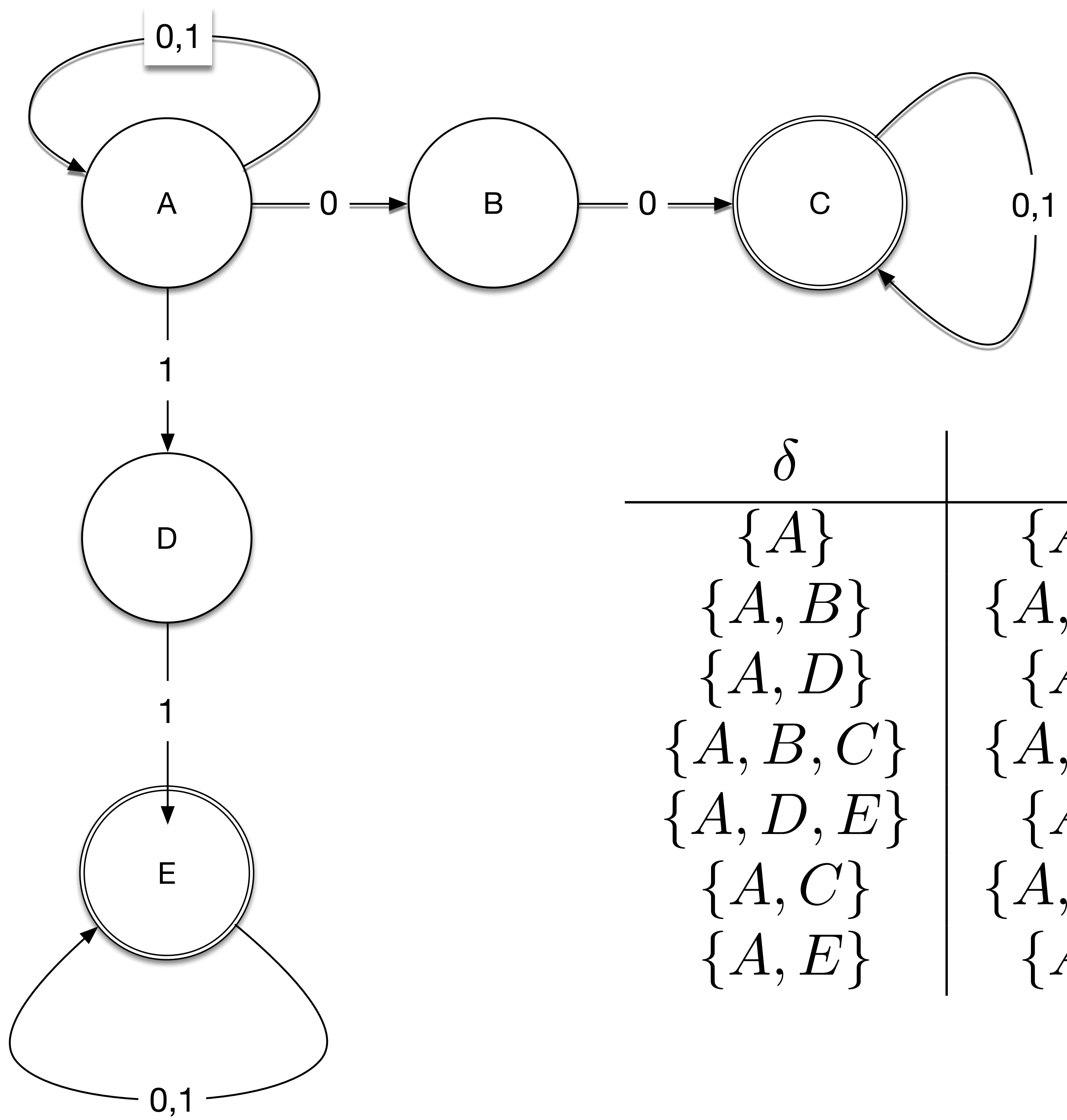
Non-deterministic Finite Automata

- Proof sketch:
  - Key idea: The states of the deterministic automaton are the subsets of the non-deterministic automaton
  - To calculate a transition from a subset *X* of states, form

$$\bigcup_{q \in X} \delta(q, a)$$

  - Accepting states: Those subsets that contain an accepting state
  - Can show: A string is accepted in the NFA only if it is accepted in the DFA

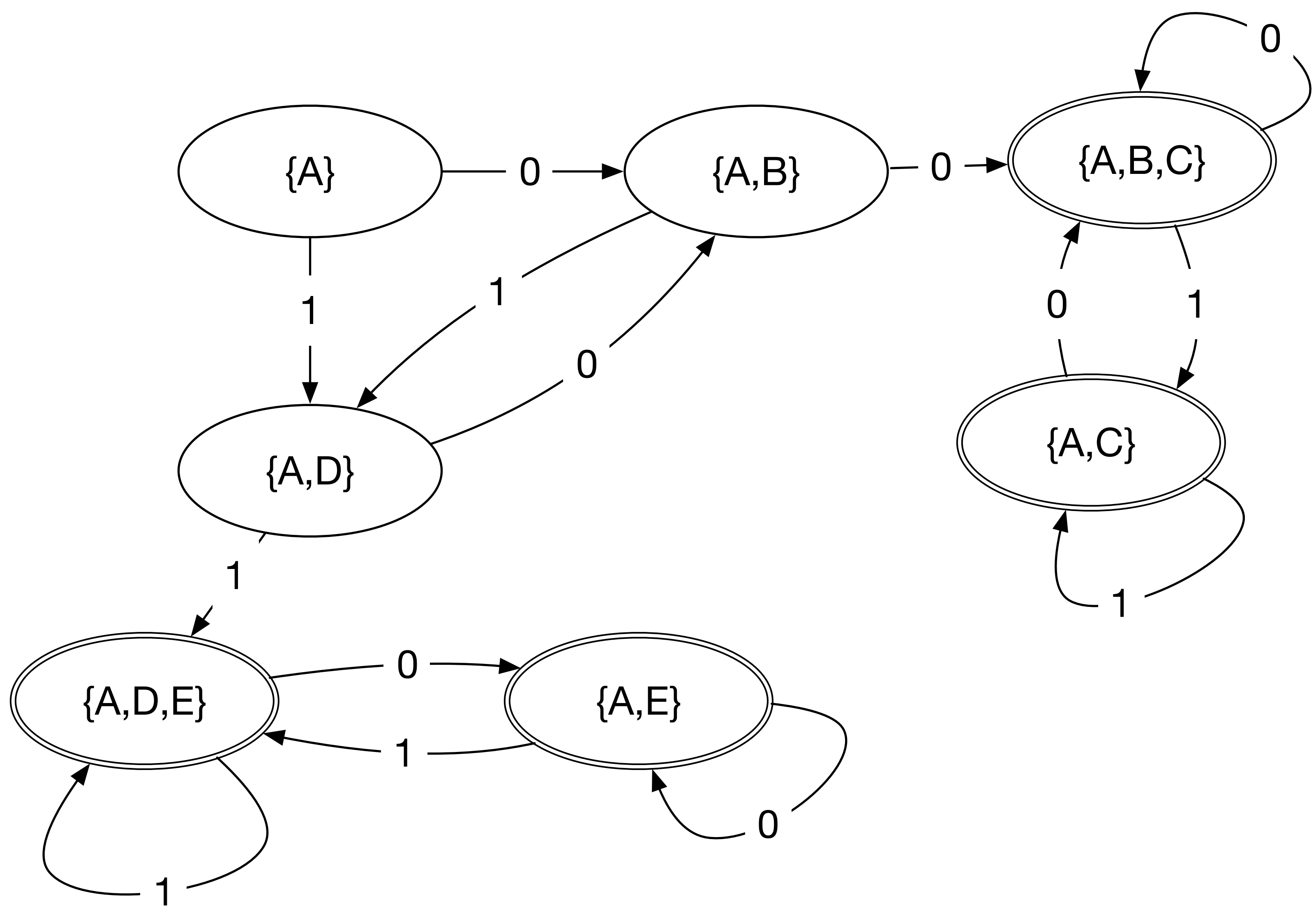
Non-deterministic Finite Automata



δ	0	1
$\{A\}$	$\{A, B\}$	$\{A, D\}$
$\{A, B\}$	$\{A, B, C\}$	$\{A, D\}$
$\{A, D\}$	$\{A, B\}$	$\{A, D, E\}$
$\{A, B, C\}$	$\{A, B, C\}$	$\{A, C\}$
$\{A, D, E\}$	$\{A, E\}$	$\{A, D, E\}$
$\{A, C\}$	$\{A, B, C\}$	$\{A, C\}$
$\{A, E\}$	$\{A, E\}$	$\{A, D, E\}$

Non-deterministic Finite Automata

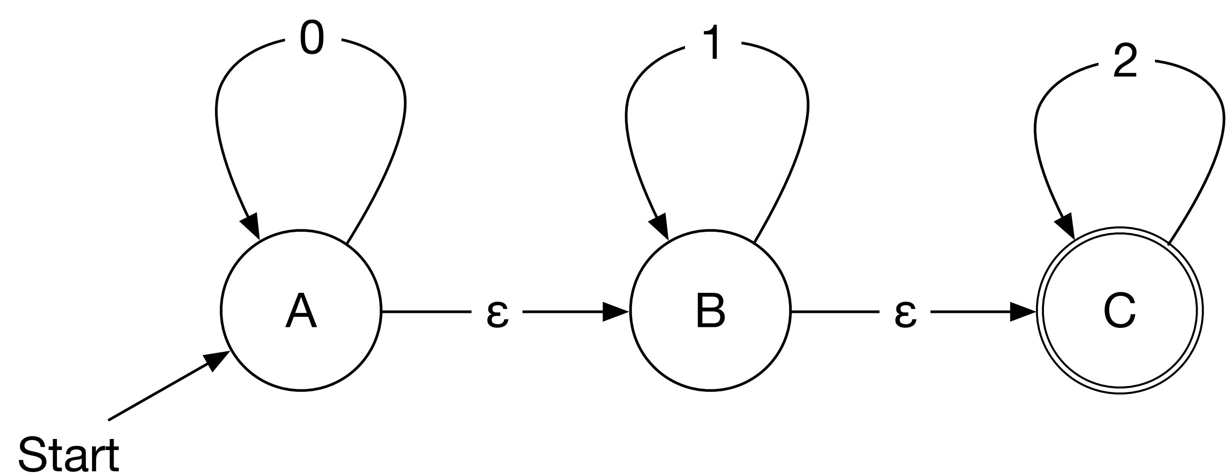
- Resulting DFA



A string without a double 0 or a double 1 needs to have alternative 0s and 1s, oscillating between the upper left three states.

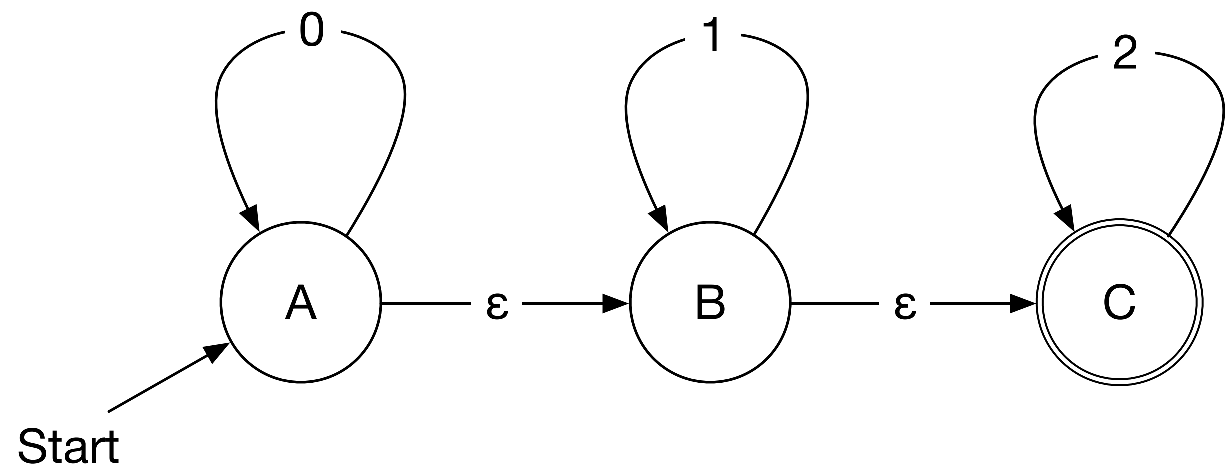
Non-deterministic finite automata with ε-moves

- A further generalization of non-deterministic finite automata are non-deterministic automata with ε - moves
- Example: Strings in $\{0, 1, 2\}^*$ whose digits only increase.



How does this automaton accept 0000222?

Non-deterministic finite automata with ε-moves

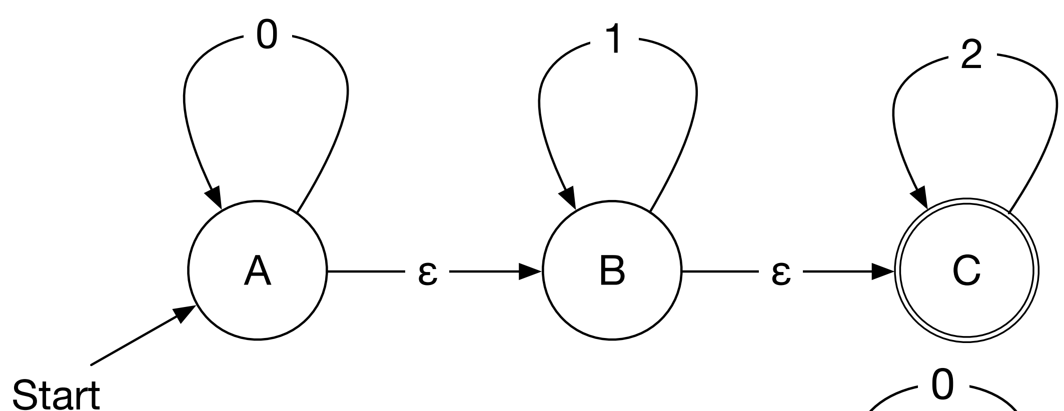


- Transition function

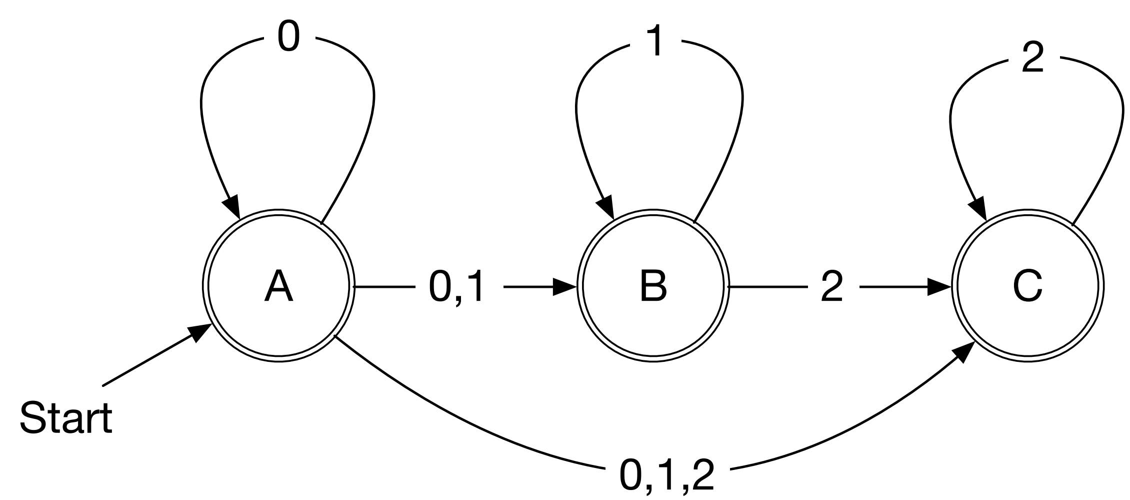
δ	ϵ	0	1	2
A	$\{B\}$	$\{A\}$	$\emptyset$	$\emptyset$
B	$\{C\}$	$\emptyset$	$\{B\}$	$\emptyset$
C	$\emptyset$	$\emptyset$	$\emptyset$	$\{\text{C}\}$

Non-deterministic finite automata with ε-moves

- We can reduce non-deterministic finite automata with ε-moves to non-deterministic automata
  - For any state and letter of the alphabet Σ, calculate the states that can be reached by using ε-transitions and a single transition with the letter.
  - Any state that can reach an accepting state by ε-moves is accepting



δ	0	1	2
A	$\{A, B, C\}$	$\{B, C\}$	$\{C\}$
B	$\emptyset$	$\{B, C\}$	$\{C\}$
C	$\emptyset$	$\emptyset$	$\{\mathrm{C}\}$



Regular Expressions

- Regular expressions define subsets of strings in an finite alphabet Σ
  - Concatenation:

$$L_1, L_2 \subset \Sigma^* : L_1 L_2 = \{x.y | x \in L_1, y \in L_2\}$$

  - Powers:

$$L \subset \Sigma^* : \qquad \begin{aligned} L^0 &:= \{\epsilon\} \\ L^1 &:= L \\ L^{n+1} &:= L^n L \text{ for } n \in \mathbb{N} \end{aligned}$$

  - Kleene Closure

$$L \subset \Sigma^* : \qquad L^* := \bigcup_{i \in \mathbb{N}} L^i$$

Regular Expressions

- Example: $L = \{01, 10\} \subset \{0,1\}^*$

$$L^0 = \{\epsilon\}$$
$$L^1 = \{01, 10\}$$
$$L^2 = \{0101, 0110, 1001, 1010\}$$
$$L^3 = \{010101, 010110, 011001, 011010, 100101, 100110, 101001, 101010\}$$

Regular Expressions

- Regular expressions are defined by induction
  - $\emptyset$ is a regular expression and denotes the empty set
  - ϵ is a regular expression and denotes the set $\{\epsilon\}$
  - If $a \in \Sigma^*$ then a is a regular expression and denotes the set $\{a\}$
  - If r, s are regular expressions for the sets R, S then so are

$$
\begin{array}{rcl}
r + s & \text{for} & R \cup S \\
rs & \text{for} & RS \\
r^* & \text{for} & R^*
\end{array}
$$

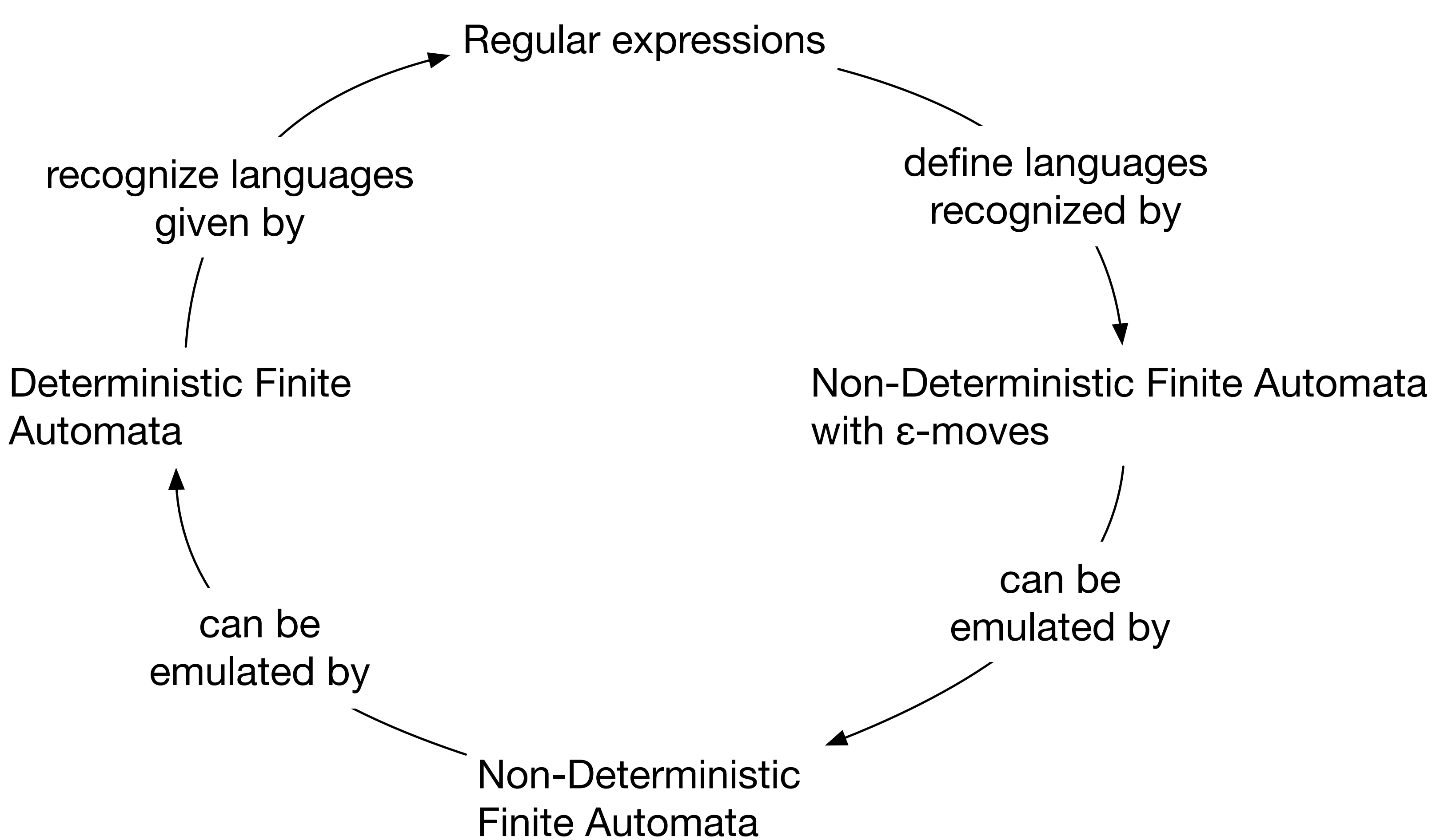
Regular Expressions

- Examples for $\Sigma = \{0, 1\}$

$$\begin{aligned}
01 &\quad \text{for} \quad \{01\} \\
0+1 &\quad \text{for} \quad \{0, 1\} \\
(0+1)^* &\quad \text{for} \quad \{\text{strings with characters 0 and 1}\} \\
1^*01^* &\quad \text{for} \quad \{\text{strings with one 0 and any number of 1s}\}
\end{aligned}$$

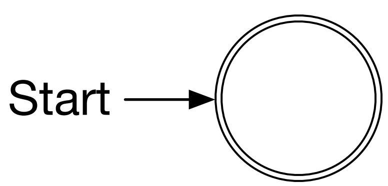
Regular Expressions and Deterministic Finite Automata

- We want to show that regular expressions are exactly those recognized by a finite automaton.
    - The proof follows a simple scheme



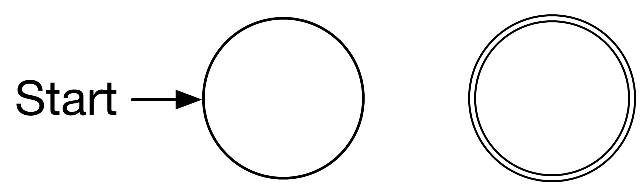
Regular Expressions and Deterministic Finite Automata

- Regular expressions are recognized by non-deterministic finite automata with ε-transitions
  - Base steps



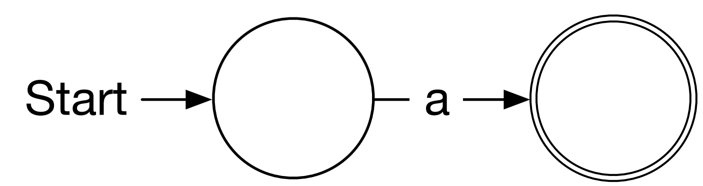
ϵ

$\{\epsilon\}$



$\emptyset$

$\emptyset$

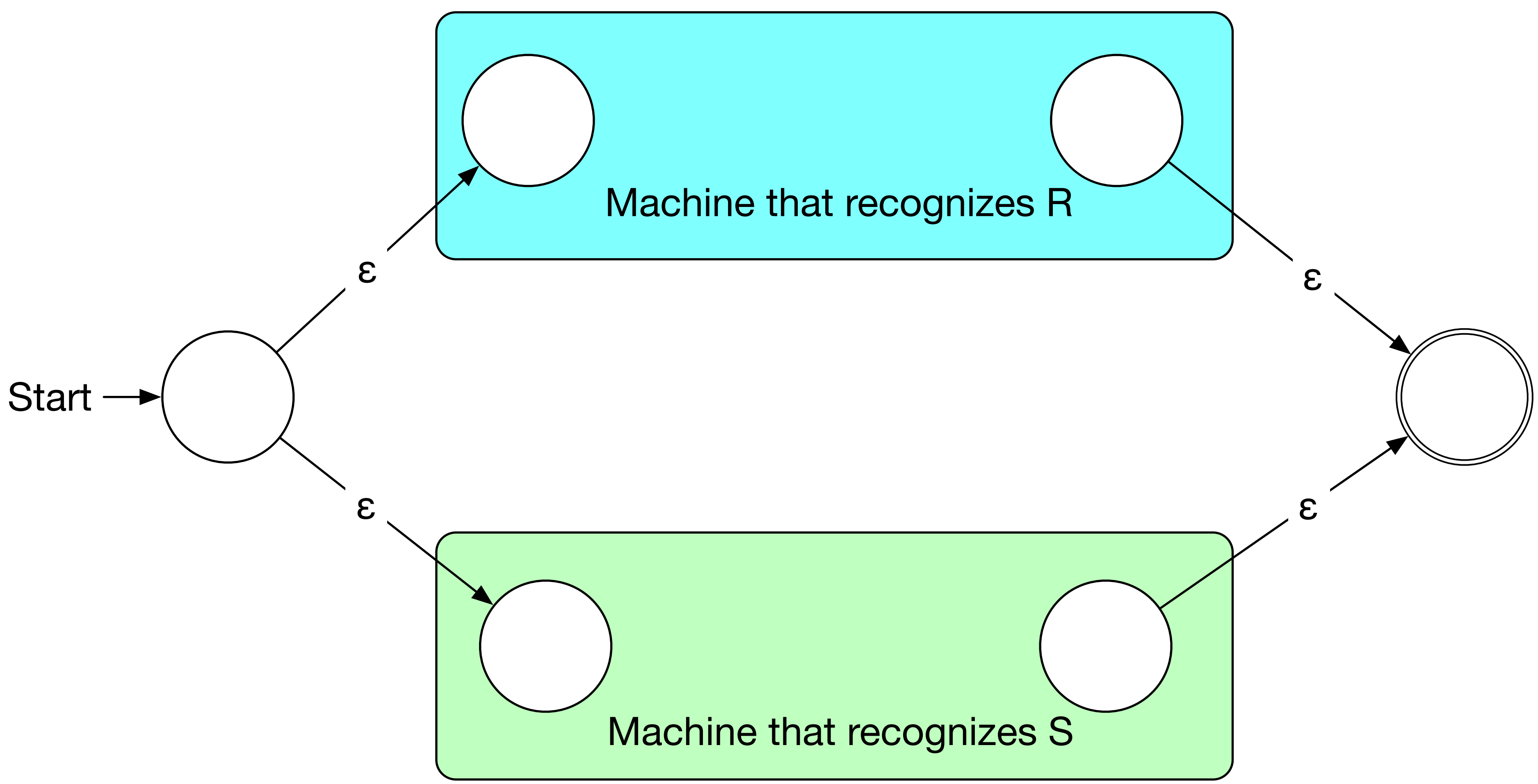


a

$\{a\}$

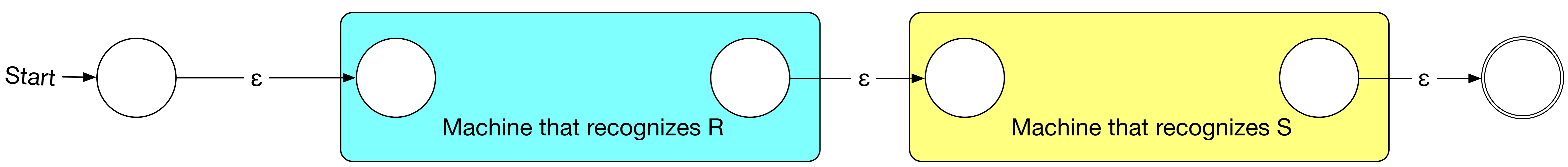
Regular Expressions and Deterministic Finite Automata

- Union r+s: Get two machines that recognize r and s
  - Connect a new start state to the start states of the two machines
  - Connect all final states with a new, single final state



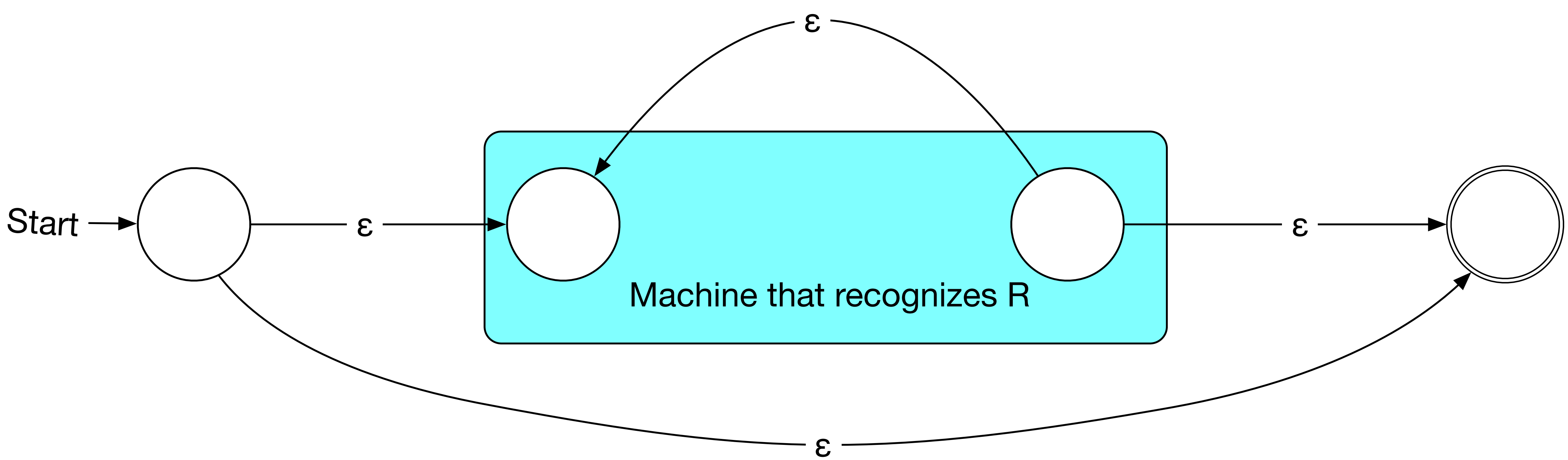
Regular Expressions and Deterministic Finite Automata

- Concatenation



Regular Expressions and Deterministic Finite Automata

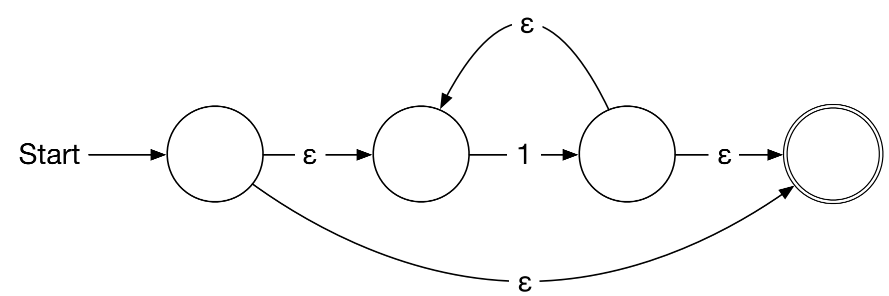
- Closure



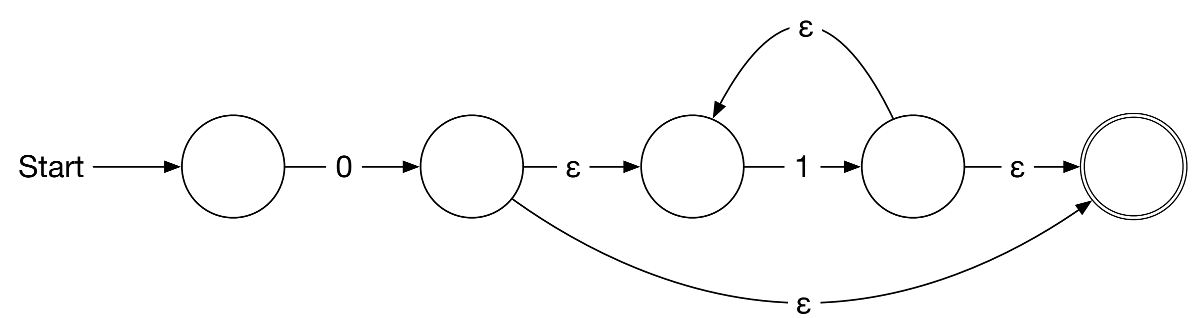
Regular Expressions and Deterministic Finite Automata

- Example $01^* + 1$

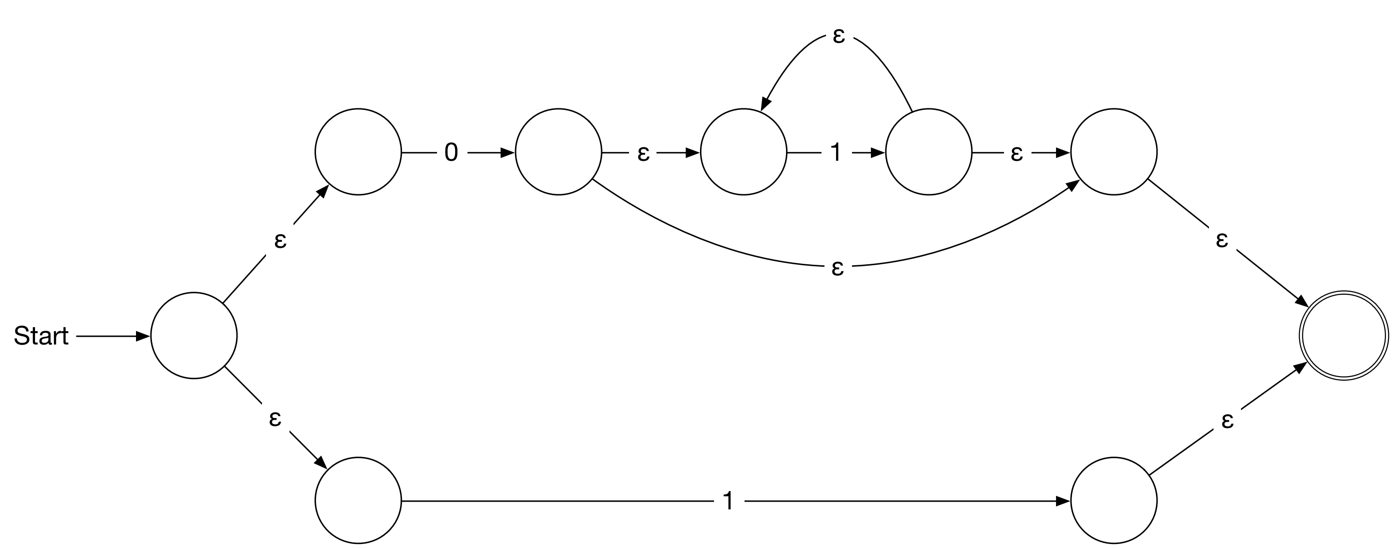
1^*



01^*



$01^* + 1$



Regular Expressions and Deterministic Finite Automata

- Now, we need to show that every language accepted by a deterministic finite automaton is regular.
  - Given a DFA $M = (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$
  - Define $R^k_{i,j}$
    - Set of strings that go from State i to State j without going through any state numbered higher than *k*
    - We can define $R^k_{i,j}$ by recursion

$$R^0_{i,i} = \{a \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\}$$

$$R^0_{i,j} = \{a \mid \delta(q_i, a) = q_j\} \qquad \text{if } i \neq j$$

$$R^k_{i,j} = R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j} \cup R^{k-1}_{i,j}$$

Regular Expressions and Deterministic Finite Automata

- Observation: $R^k_{i,j}$ is given by a regular expression
  - Proof by induction on *k*
  - Base: *k* = 0
    - $R^0_{i,j}$ is a finite set of strings with a single symbol or ε
  - Induction step: *k* —> *k*+1
    - By induction hypothesis, we have regular expressions such that $\mathcal{L}(R^{k-1}_{i,j}) = r^{k-1}_{i,j}$
    - Simply define a regular expression for $R^k_{i,j}$ by

$$r^{k-1}_{k,i}(r^{k-1}_{k,k})^*(r^{k-1}_{k,j}) + r^{k-1}_{i,j}$$

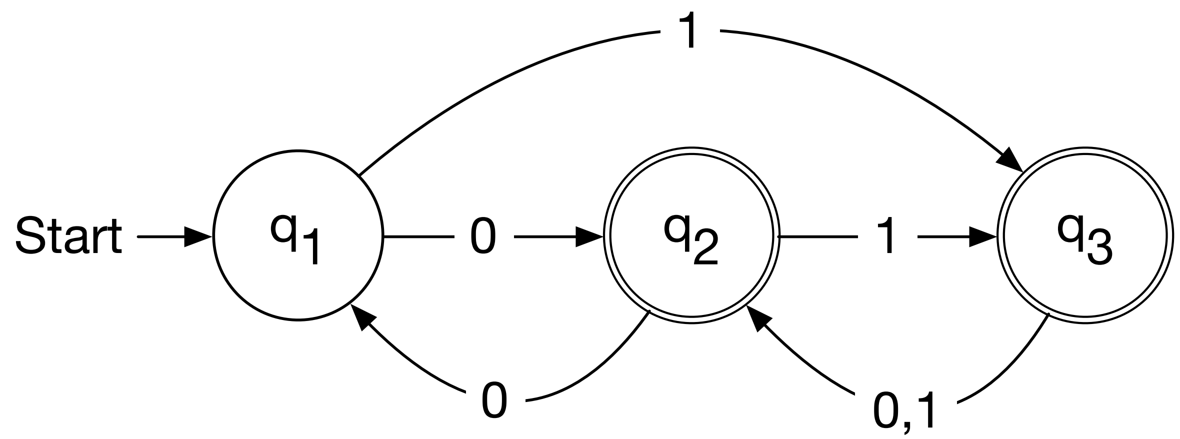
Regular Expressions and Deterministic Finite Automata

- It follows that the language accepted by a DFA is regular:

$$\mathcal{L}(M) = \bigcup_{q_j \in F} R_{1,j}^n = \sum_{q_j \in F} r_{1,j}^n$$

Regular Expressions and Deterministic Finite Automata

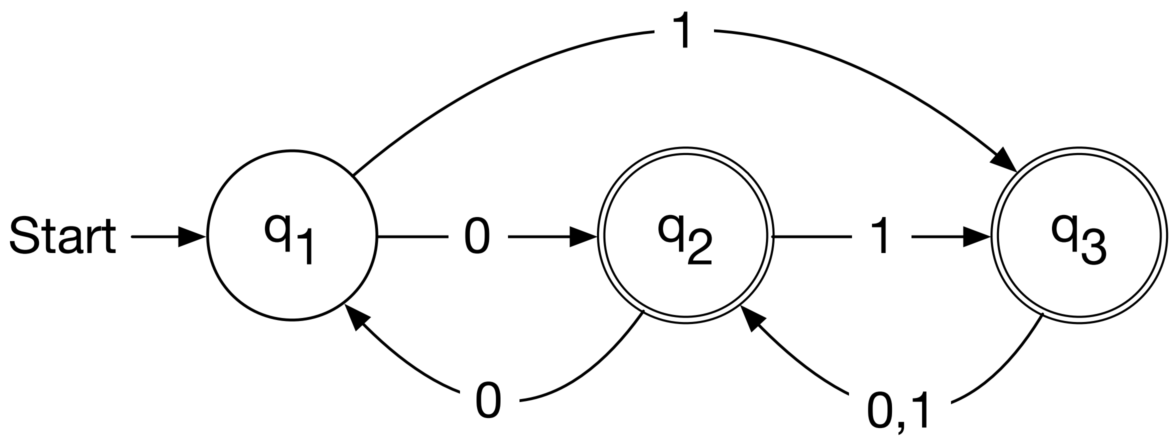
- Example:



	k=0	k=1	k=2
$r_{1,1}^k$	ϵ		
$r_{1,2}^k$	0		
$r_{1,3}^k$	1		
$r_{2,1}^k$	0		
$r_{2,2}^k$	ϵ		
$r_{2,3}^k$	1		
$r_{3,1}^k$	$\emptyset$		
$r_{3,2}^k$	0+1		
$r_{3,3}^k$	ϵ		

Regular Expressions and Deterministic Finite Automata

- Example:



$$r^1_{2,2} = r^0_{2,1}(r^0_{1,1})^* r^0_{1,2} + r^0_{2,2} = 0\epsilon^*0 + \epsilon = 00 + \epsilon$$

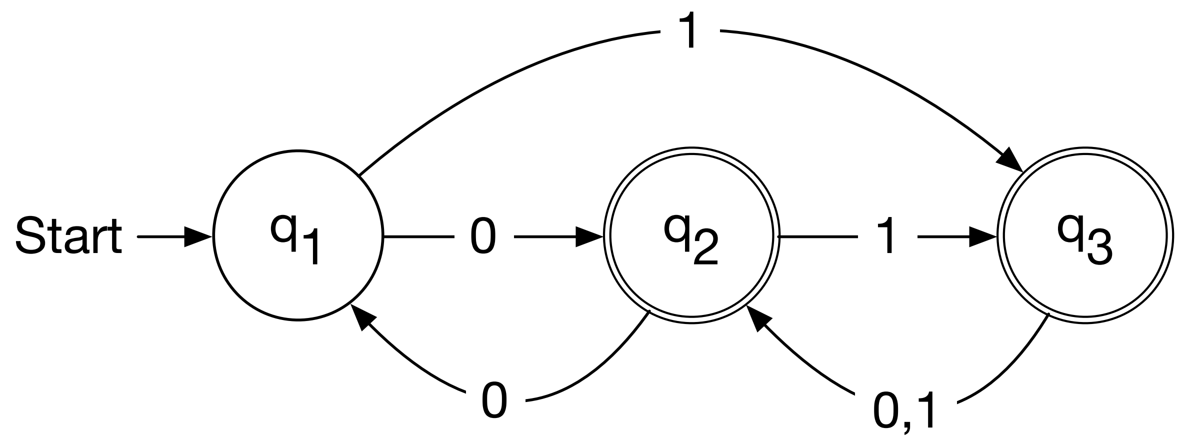
$$r^1_{2,3} = r^0_{2,1}(r^0_{1,1})^* r^0_{1,3} + r^0_{2,3} = 0\epsilon^*1 + 1 = 01 + 1$$

$$r^1_{3,2} = r^0_{3,1}(r^0_{1,1})^* r^0_{1,2} + r^0_{3,2} = \emptyset\epsilon^*0 + 1 = 0 + 1$$

	k=0	*k*=1	*k*=2
$r^k_{1,1}$	ϵ	ϵ	
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

Regular Expressions and Deterministic Finite Automata

- Example:

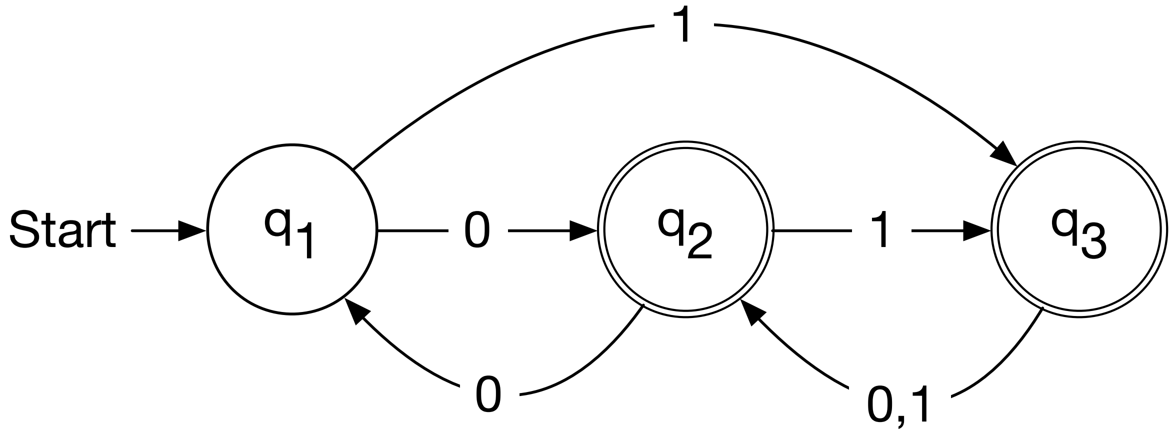


	k=0	*k*=1	*k*=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

$$r^2_{1,1} = r^1_{1,2}(r^1_{2,2})^* r^1_{2,1} + r^1_{1,1} = 0(\epsilon + 00)^* 0 + \epsilon = (00)^+ + \epsilon = (00)^*$$

Regular Expressions and Deterministic Finite Automata

- Example:

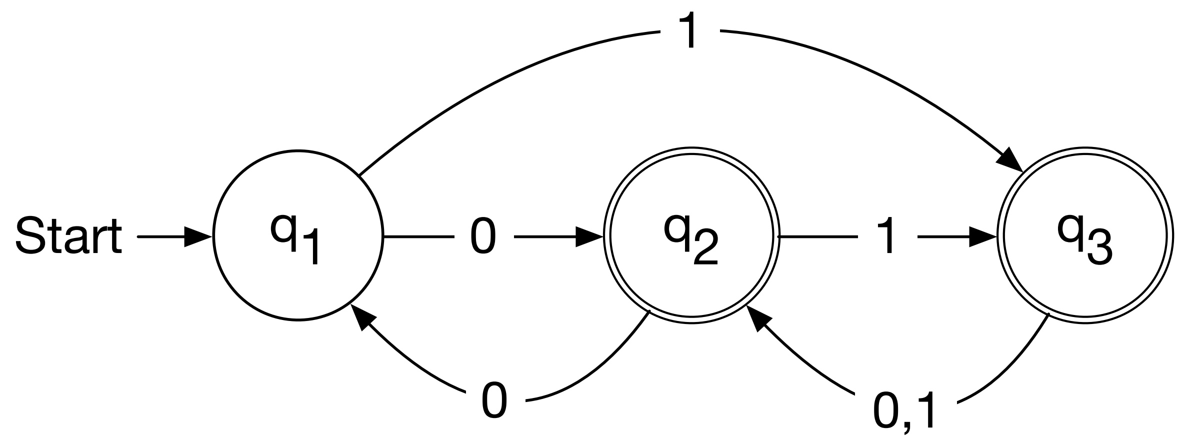


$$
\begin{aligned}
r_{1,2}^2 &= r_{1,2}^1 (r_{2,2}^1)^* r_{2,2}^1 + r_{1,2}^1 \\
&= 0(\epsilon + 00)^*(\epsilon + 00) + 0 \\
&= 0(00)^*\epsilon + 0(00)^*(00) + 0 \\
&= 0(00)^* + 0 \\
&= 0(00)^*
\end{aligned}
$$

	k=0	*k*=1	*k*=2
$r_{1,1}^k$	ϵ	ϵ	$(00)^*$
$r_{1,2}^k$	0	0	
$r_{1,3}^k$	1	1	
$r_{2,1}^k$	0	0	
$r_{2,2}^k$	ϵ	$\epsilon + 00$	
$r_{2,3}^k$	1	$1 + 01$	
$r_{3,1}^k$	$\emptyset$	$\emptyset$	
$r_{3,2}^k$	$0 + 1$	$0 + 1$	
$r_{3,3}^k$	ϵ	ϵ	

Regular Expressions and Deterministic Finite Automata

- Example:

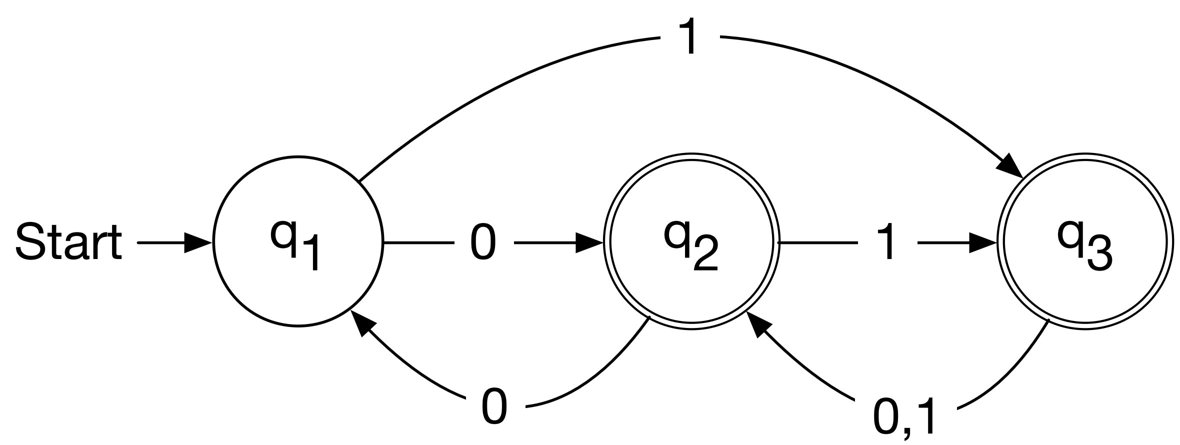


	k=0	k=1	k=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

$$\begin{aligned} r^2_{1,3} &= r^1_{1,2}(r^1_{2,2})^* r^1_{2,3} + r^1_{1,3} \\ &= 0(\epsilon + 00)^*(1 + 01) + 1 \\ &= 0(00)^*(1 + 01) + 1 \\ &= 0(00)^*1 + 0(00)^*01 + 1 \\ &= 0 * 1 \end{aligned}$$

Regular Expressions and Deterministic Finite Automata

- Example:

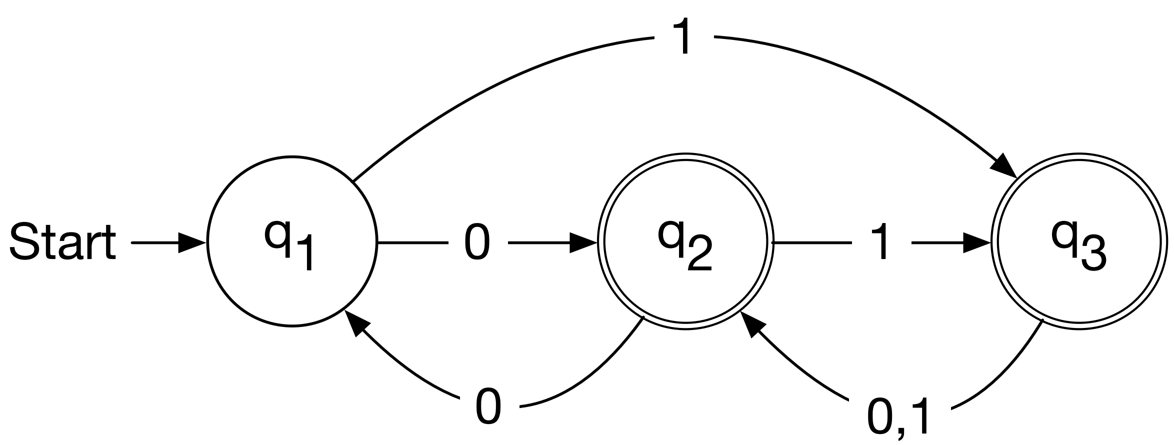


$$
\begin{aligned}
r_{2,1}^{2} &= r_{2,2}^{1}(r_{2,2}^{1})^{*}r_{2,1}^{1} + r_{2,1}^{1} \\
&= (\epsilon + 00)(\epsilon + 00)^{*}0 + 0 \\
&= (00)^{*}0 + 0 \\
&= 0(00)^{*}
\end{aligned}
$$

	k=0	*k*=1	*k*=2
$r_{1,1}^{k}$	ϵ	ϵ	$(00)^{*}$
$r_{1,2}^{k}$	0	0	
$r_{1,3}^{k}$	1	1	
$r_{2,1}^{k}$	0	0	
$r_{2,2}^{k}$	ϵ	$\epsilon + 00$	
$r_{2,3}^{k}$	1	$1 + 01$	
$r_{3,1}^{k}$	$\emptyset$	$\emptyset$	
$r_{3,2}^{k}$	$0 + 1$	$0 + 1$	
$r_{3,3}^{k}$	ϵ	ϵ	

Regular Expressions and Deterministic Finite Automata

- Example:

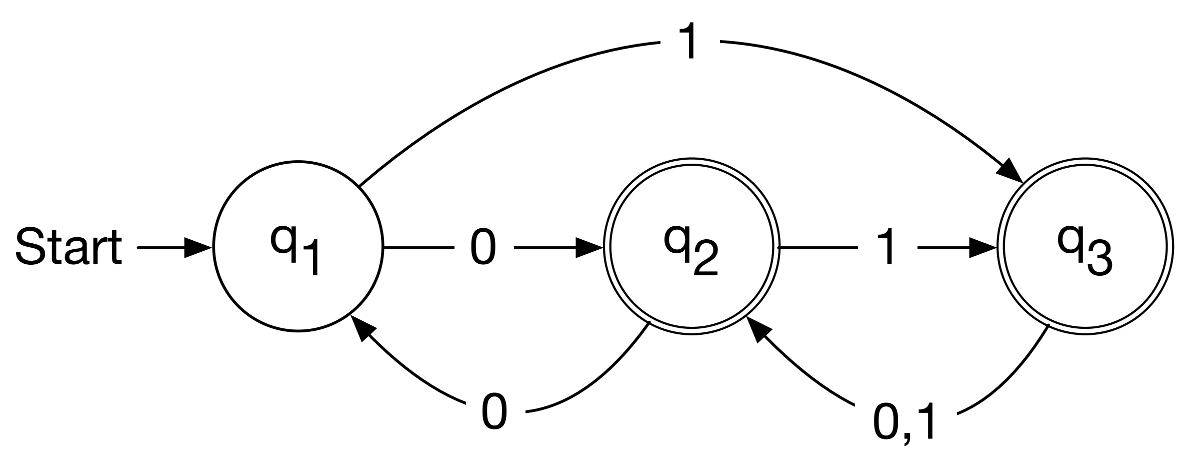


	k=0	k=1	k=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

$$
\begin{aligned}
r^2_{2,2} &= r^1_{2,2}(r^1_{2,2})^* r^1_{2,2} + r^1_{2,2} \\
&= (\epsilon + 00)(\epsilon + 00)^*(\epsilon + 00) + (\epsilon + 00) \\
&= (00)^*
\end{aligned}
$$

Regular Expressions and Deterministic Finite Automata

- Example:

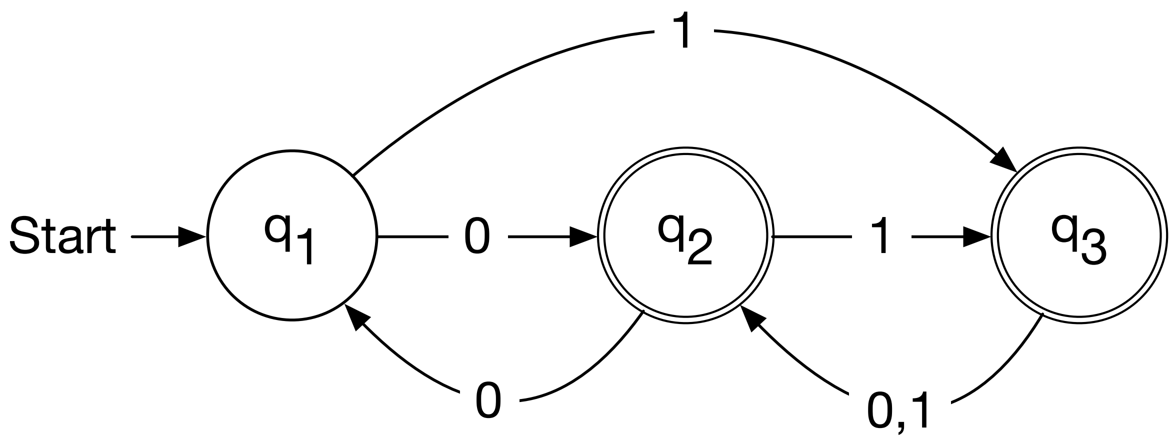


$$
\begin{aligned}
r^2_{2,3} &= r^1_{2,2}(r^1_{2,2})^* r^1_{2,3} + r^1_{2,3} \\
&= (\epsilon + 00)(\epsilon + 00)^*(1 + 01) + (1 + 01) \\
&= (00)^*(1 + 01) \\
&= (00)^*1 + (00)^*01 \\
&= 0^*1
\end{aligned}
$$

	k=0	*k*=1	*k*=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

Regular Expressions and Deterministic Finite Automata

- Example:

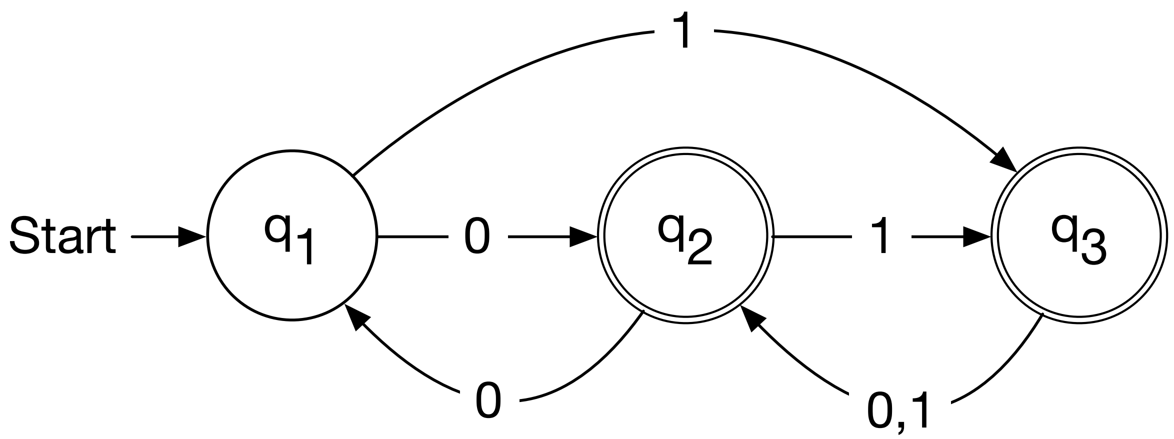


$$
\begin{aligned}
r^2_{3,1} &= r^1_{3,2}(r^1_{2,2})^* r^1_{2,1} + r^1_{3,1} \\
&= (0+1)(\epsilon+00)^* 0 + \emptyset \\
&= (0+1)(00)^* 0
\end{aligned}
$$

	k=0	*k*=1	*k*=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ϵ	$\epsilon + 00$	
$r^k_{2,3}$	1	$1 + 01$	
$r^k_{3,1}$	$\emptyset$	$\emptyset$	
$r^k_{3,2}$	$0 + 1$	$0 + 1$	
$r^k_{3,3}$	ϵ	ϵ	

Regular Expressions and Deterministic Finite Automata

- Example:

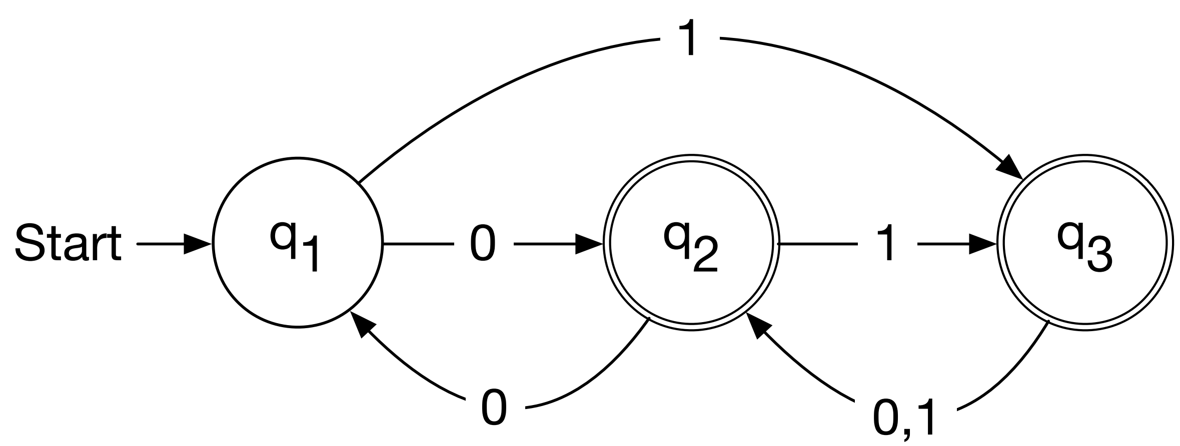


	k=0	k=1	k=2
$r_{1,1}^k$	ϵ	ϵ	$(00)^*$
$r_{1,2}^k$	0	0	
$r_{1,3}^k$	1	1	
$r_{2,1}^k$	0	0	
$r_{2,2}^k$	ϵ	$\epsilon + 00$	
$r_{2,3}^k$	1	$1 + 01$	
$r_{3,1}^k$	$\emptyset$	$\emptyset$	
$r_{3,2}^k$	$0 + 1$	$0 + 1$	
$r_{3,3}^k$	ϵ	ϵ	

$$
\begin{aligned}
r_{3,2}^2 &= r_{3,2}^1 (r_{2,2}^1)^* r_{2,2}^1 + r_{3,2}^1 \\
&= (0+1)(\epsilon + 00)^*(\epsilon + 00) + (0+1) \\
&= (0+1)(00)^*
\end{aligned}
$$

Regular Expressions and Deterministic Finite Automata

- Example:

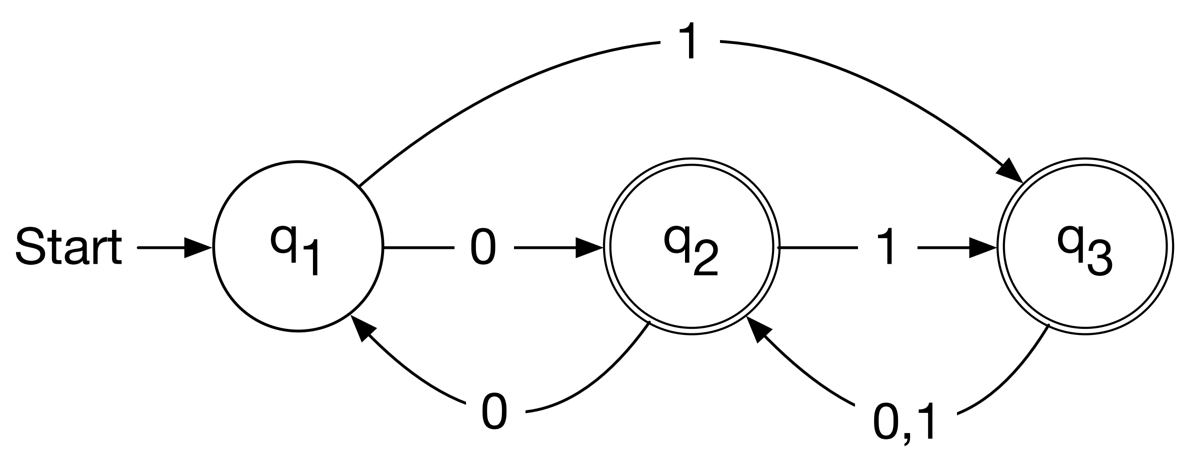


	k=0	k=1	k=2
$r_{1,1}^k$	ϵ	ϵ	$(00)^*$
$r_{1,2}^k$	0	0	$0(00)^*$
$r_{1,3}^k$	1	1	0^*1
$r_{2,1}^k$	0	0	$0(00)^*$
$r_{2,2}^k$	ϵ	$\epsilon+00$	$(00)^*$
$r_{2,3}^k$	1	$1+01$	0^*1
$r_{3,1}^k$	$\emptyset$	$\emptyset$	$(0+1)(00)^*0$
$r_{3,2}^k$	$0+1$	$0+1$	$(0+1)(00)^*$
$r_{3,3}^k$	ϵ	ϵ	$\epsilon+(0+1)0^*1$

$$\begin{aligned} r_{3,3}^2 &= r_{3,2}^1(r_{2,2}^1)^* r_{2,3}^1 + r_{3,3}^1 \\ &= (0+1)(\epsilon+00)^*(1+01)+\epsilon \\ &= (0+1)\left((00)^*1+(00)^*01\right)+\epsilon \\ &= (0+1)0^*1+\epsilon \end{aligned}$$

Regular Expressions and Deterministic Finite Automata

- Example:

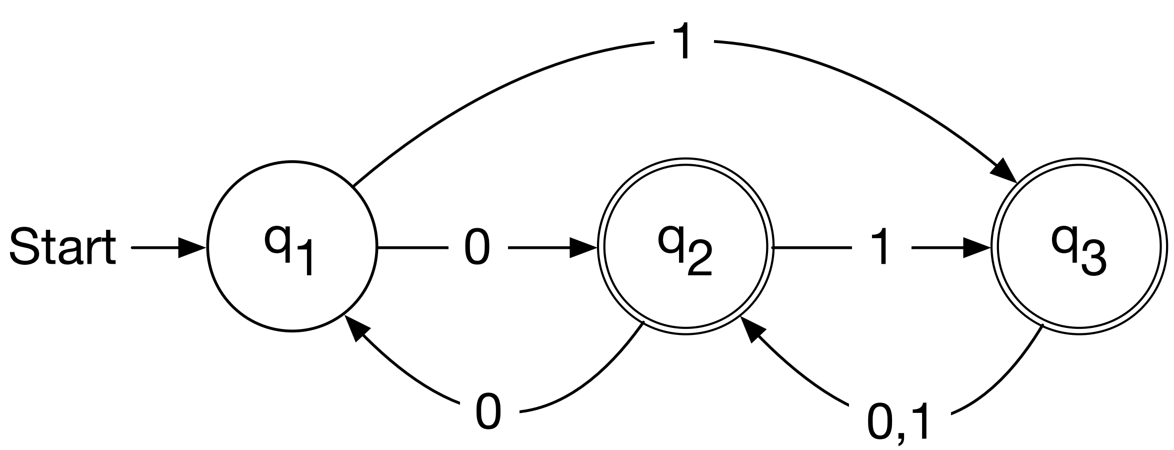


	k=0	k=1	k=2
$r_{1,1}^k$	ϵ	ϵ	$(00)^*$
$r_{1,2}^k$	0	0	$0(00)^*$
$r_{1,3}^k$	1	1	0^*1
$r_{2,1}^k$	0	0	$0(00)^*$
$r_{2,2}^k$	ϵ	$\epsilon + 00$	$(00)^*$
$r_{2,3}^k$	1	$1 + 01$	0^*1
$r_{3,1}^k$	$\emptyset$	$\emptyset$	$(0+1)(00)^*0$
$r_{3,2}^k$	$0 + 1$	$0 + 1$	$(0+1)(00)^*$
$r_{3,3}^k$	ϵ	ϵ	$\epsilon + (0+1)0^*1$

$$
\begin{aligned}
r_{1,2}^3 &= r_{1,3}^2(r_{3,3}^2)^* r_{3,2} + r_{1,2}^2 \\
&= 0^*1\big(\epsilon + (0+1)0^*1\big)^*(0+1)(00)^* + 0(00)^* \\
&= 0^*1\big((0+1)0^*1\big)^*(0+1)(00)^* + 0(00)^*
\end{aligned}
$$

Regular Expressions and Deterministic Finite Automata

- Example:



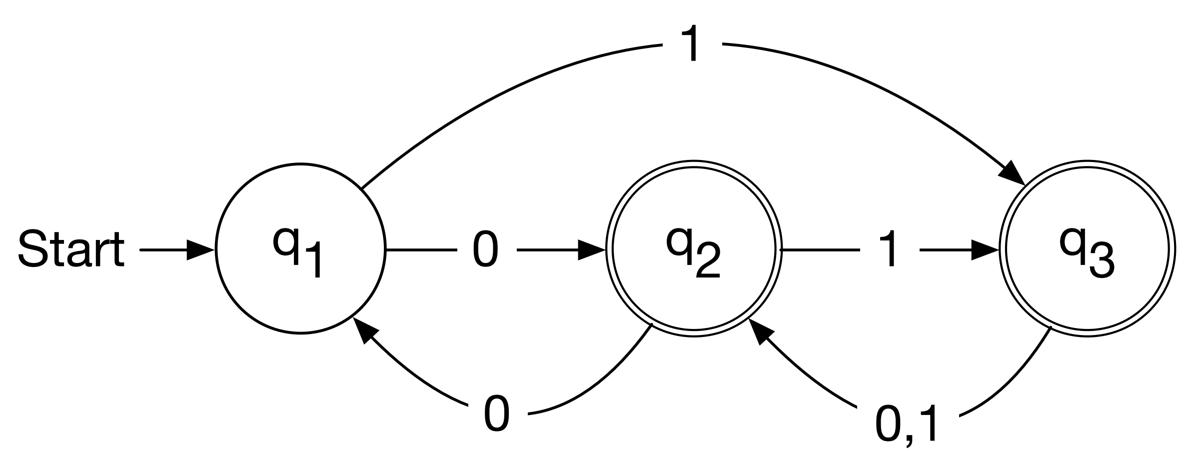
	k=0	k=1	k=2
$r^k_{1,1}$	ϵ	ϵ	$(00)^*$
$r^k_{1,2}$	0	0	$0(00)^*$
$r^k_{1,3}$	1	1	0^*1
$r^k_{2,1}$	0	0	$0(00)^*$
$r^k_{2,2}$	ϵ	$\epsilon + 00$	$(00)^*$
$r^k_{2,3}$	1	$1 + 01$	0^*1
$r^k_{3,1}$	$\emptyset$	$\emptyset$	$(0+1)(00)^*0$
$r^k_{3,2}$	$0+1$	$0+1$	$(0+1)(00)^*$
$r^k_{3,3}$	ϵ	ϵ	$\epsilon + (0+1)0^*1$

$$
\begin{aligned}
r^3_{1,3} &= r^2_{1,3}(r^2_{3,3})^* r_{3,3} + r^2_{1,3} \\
&= 0^*1\big(\epsilon + (0+1)0^*1\big)^*\big(\epsilon + (0+1)0^*1\big) + 0^*1 \\
&= 0^*1\big((0+1)0^*1\big)^* + 0^*1 \\
&= 0^*1\big((0+1)0^*1\big)^*
\end{aligned}
$$

Regular Expressions and Deterministic Finite Automata

- Therefore

$$\begin{aligned}\mathcal{L}(M) &= r_{1,2}^3 + r_{1,3}^3 \\ &= 0^*1\big((0+1)0^*1\big)^*(\epsilon + (0+1)(00)^*) + 0(00)^*\end{aligned}$$



Finite Automata with Output

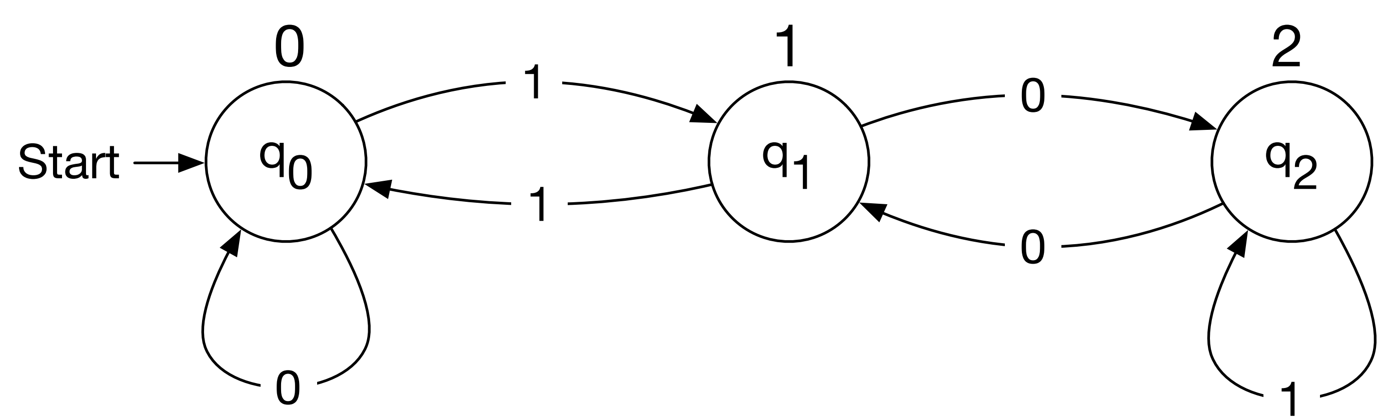
- Moore machines
  - Whenever the machine is in state *i* it outputs a symbol depending on the state

    - Example:
      - A Moore machine that calculates the remainder modulo 3 of a binary number
      - To derive the formula, consider

$$
\begin{aligned}
a.x \pmod 3 &\equiv 2a + x \pmod 3 \\
&\equiv \Big(2a \pmod 3\Big) + \Big(x \pmod 3\Big) \\
&\equiv 2\Big(a \pmod 3\Big) + \Big(x \pmod 3\Big)
\end{aligned}
$$

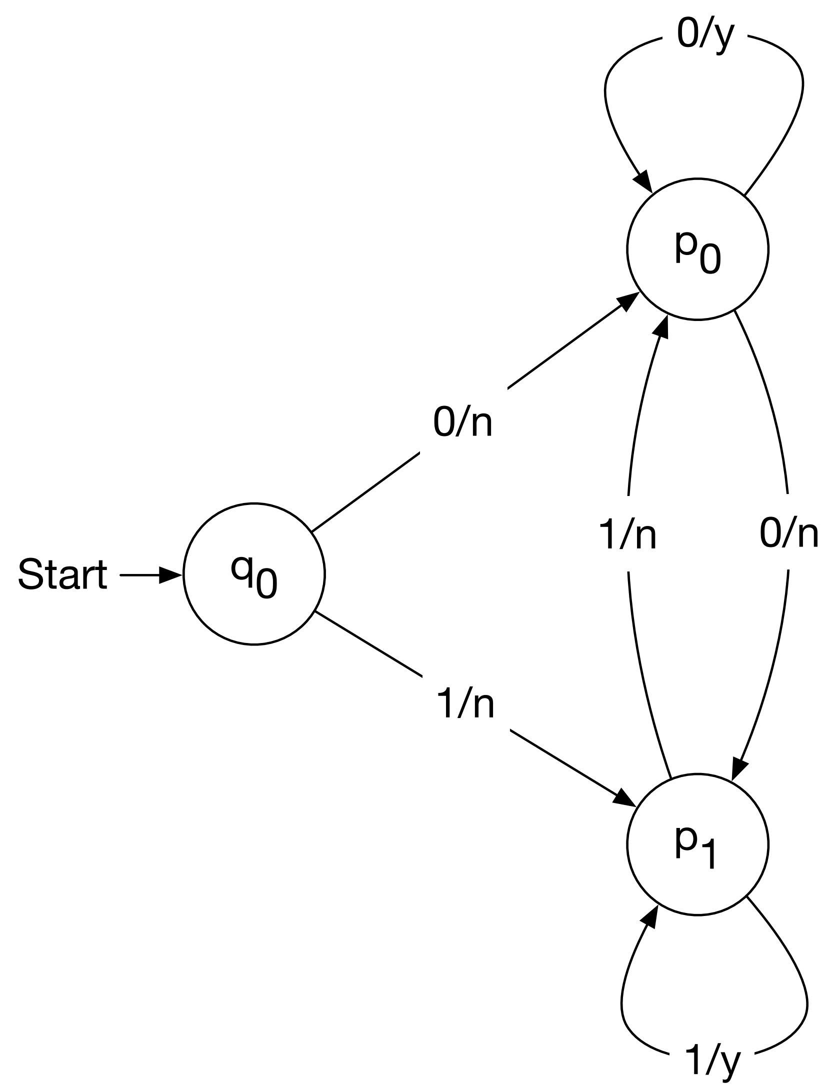
Finite Automata with Output

$a \pmod 3$	$x \pmod 3$	$a.x \pmod 3$
0	0	0
0	1	1
1	0	2
1	1	0
2	0	1
2	1	2



Finite Automata with Output

- Mealy Machines
  - Output depends on the current state and the transition



Finite Automata with Output

- It can be shown that Mealy and Moore machines are equivalent