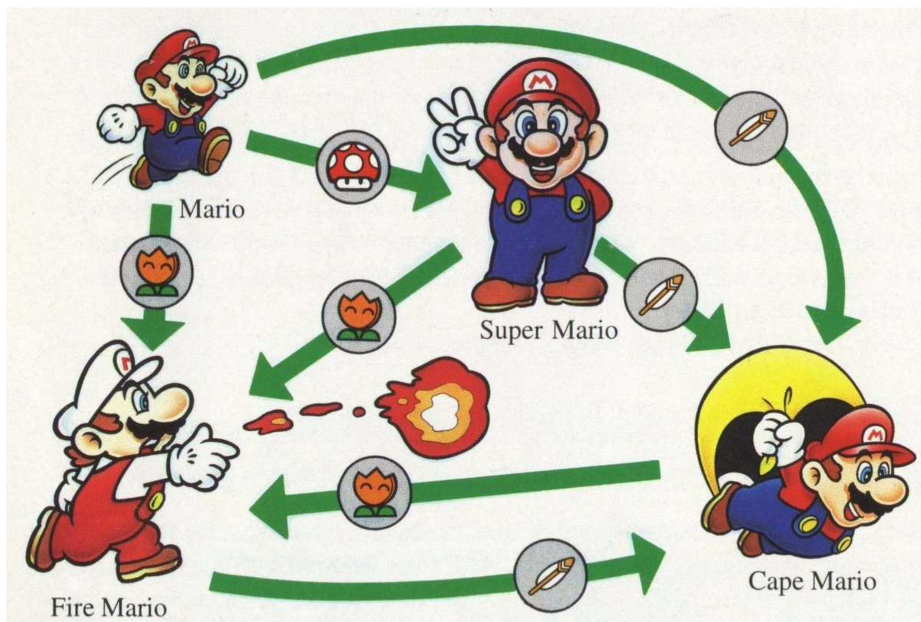
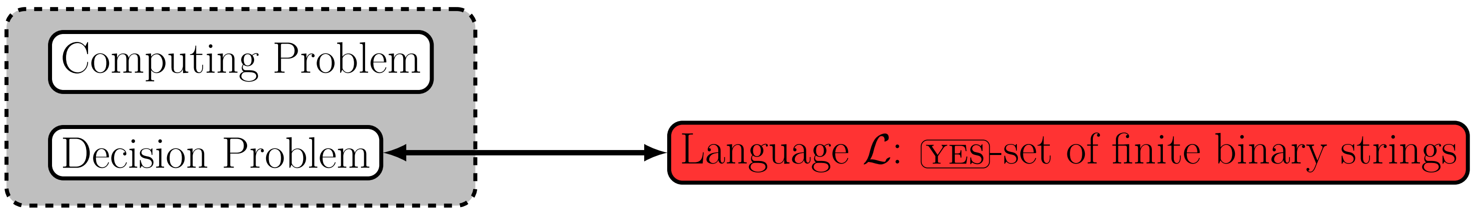
Foundations of Computer Science
Lecture 24

Deterministic Finite Automata (DFA)

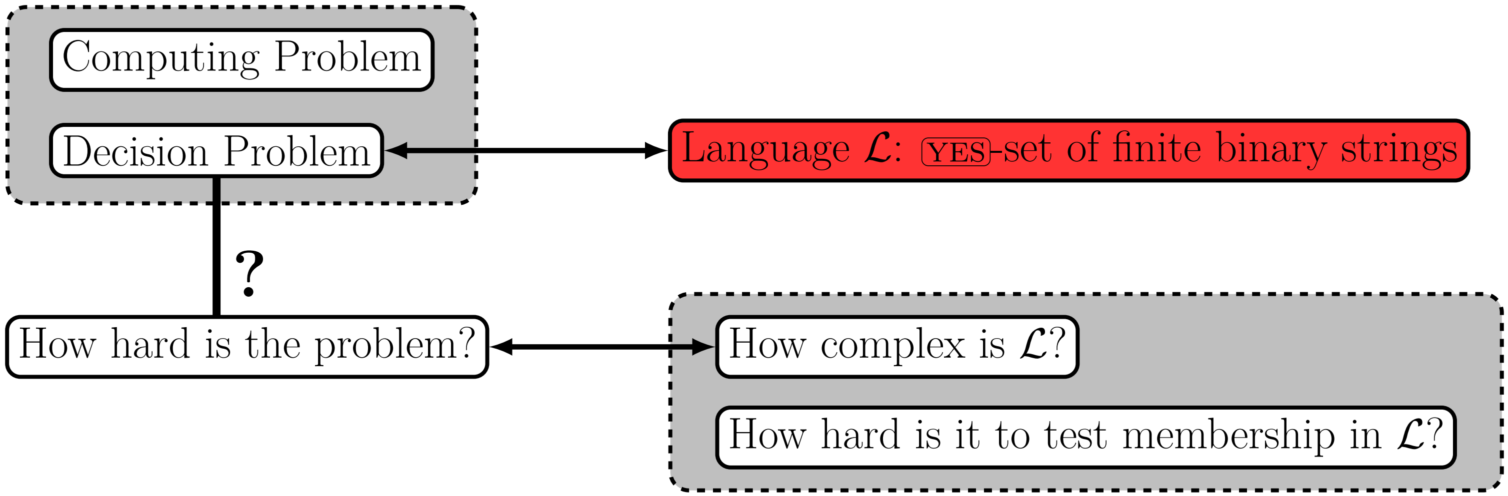
A Simple Computing Machine: A CPU with States and Transitions
What Problems Can It Solve: Regular Languages
Is There A Problem It *Can't* Solve?



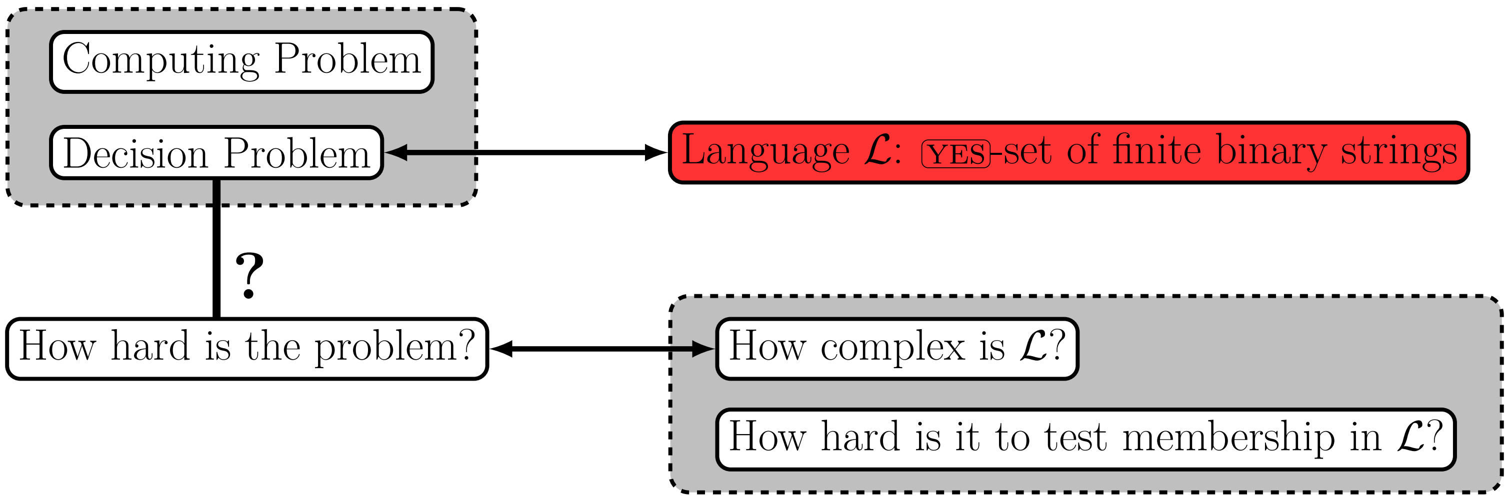
Computing Problems and Their Difficulty



Computing Problems and Their Difficulty



Computing Problems and Their Difficulty

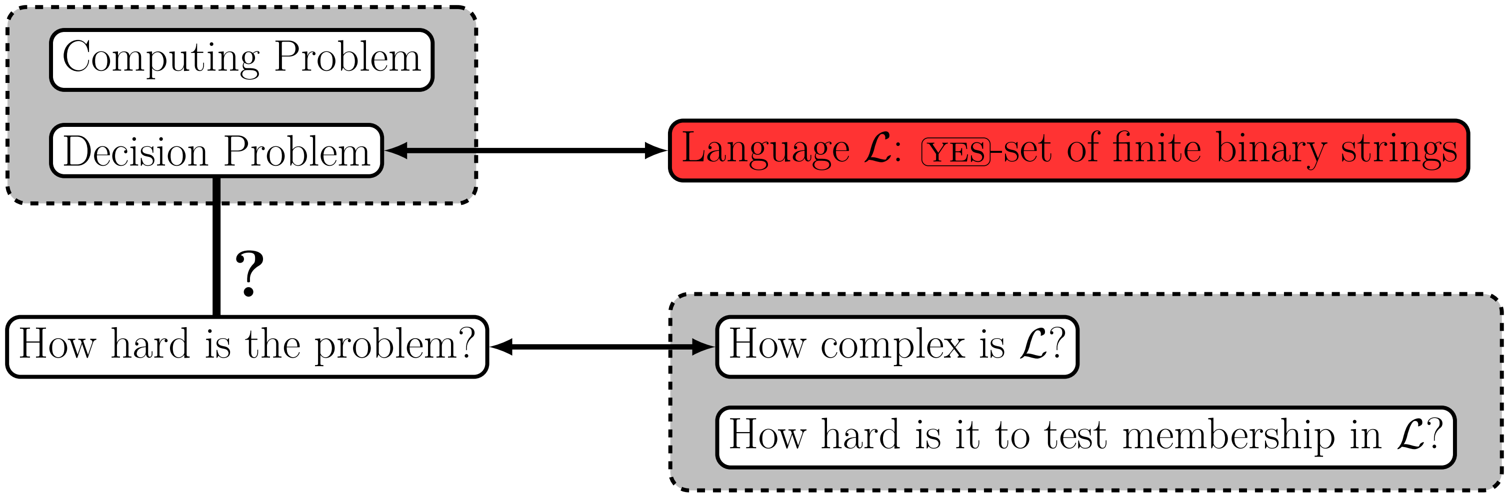


A problem can be harder in two ways.

1. The problem needs more resources. For example, the problem can be solved with a similar machine to ours, except with more states.

2. The problem needs a different *kind* of computing machine, with superior capabilities.

Computing Problems and Their Difficulty



A problem can be harder in two ways.

1. The problem needs more resources. For example, the problem can be solved with a similar machine to ours, except with more states.
2. The problem needs a different *kind* of computing machine, with superior capabilities.

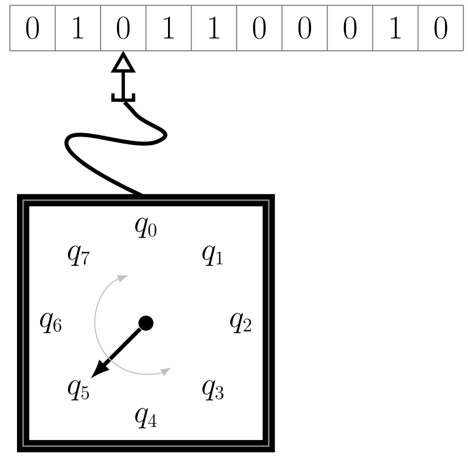
The first type of “harder” is the focus of a follow-on algorithms course.

We focus on what *can and can't be solved* on a particular kind of machine.

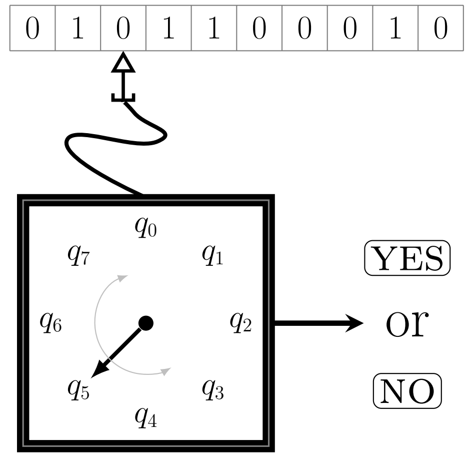
Today: Deterministic Finite Automata (DFA)

1. A simple computing machine.
   - States.
   - Transitions.
   - No scratch paper.
2. What computing problems can this simple machine solve?
   - Vending machine.
3. Regular languages.
   - Closed under all the set operations: union, intersection, complement, concatenation, Kleene-star.
4. Are there problems that cannot be solved?

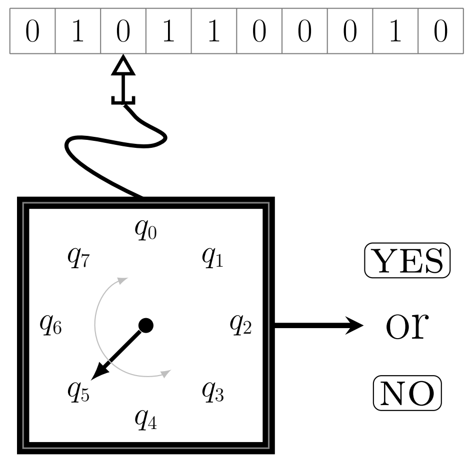
A Simple Computing Machine

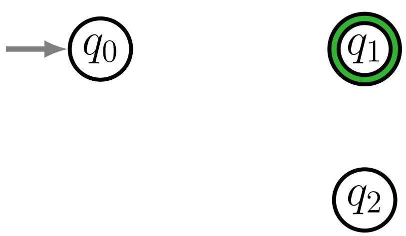


A Simple Computing Machine



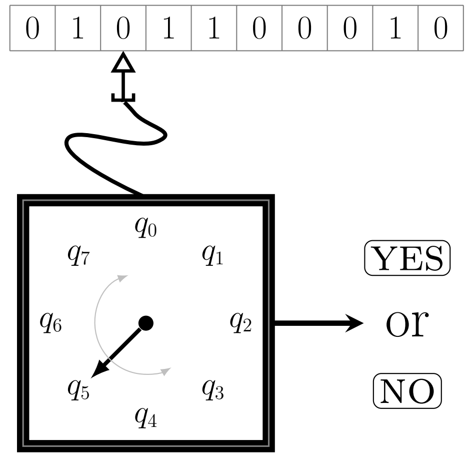
A Simple Computing Machine

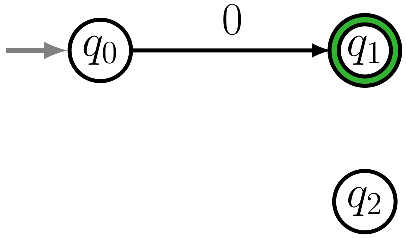




states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

A Simple Computing Machine



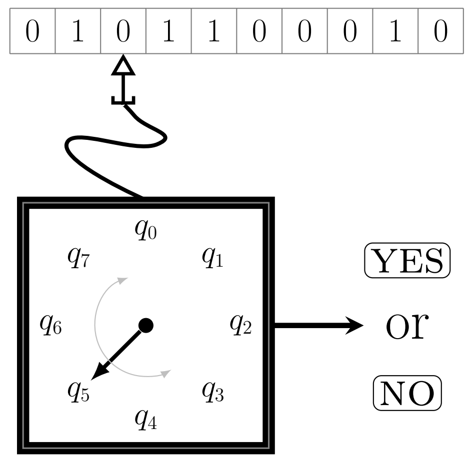


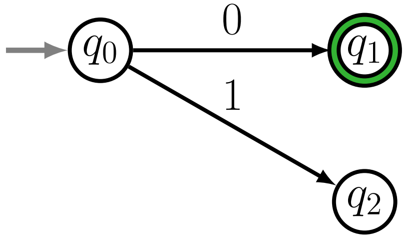
	states	
$\rightarrow$	q_0	NO
	q_1	YES
	q_2	NO

transitions
$1: \; q_0 \;\; 0 \;\; q_1$

$\leftarrow$ In state q_0, if you read 0, transition to q_1

A Simple Computing Machine



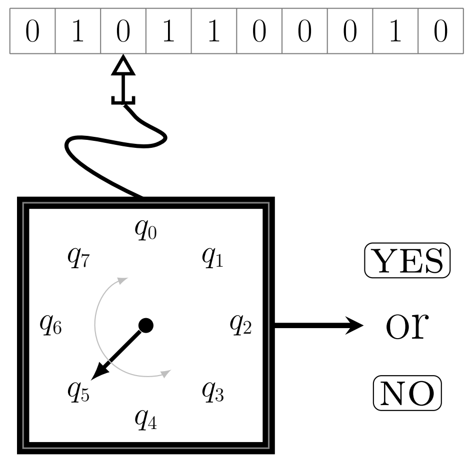


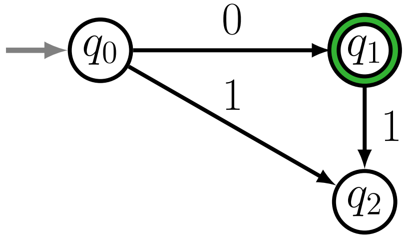
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

A Simple Computing Machine



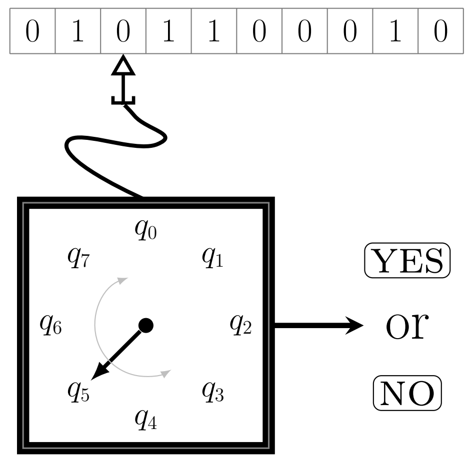


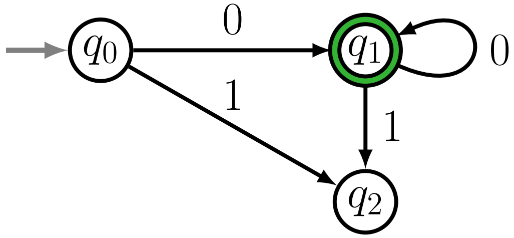
states	
$\to q_0$	NO
q_1	YES
q_2	NO

transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1

$\leftarrow$ In state q_0, if you read 0, transition to q_1

A Simple Computing Machine



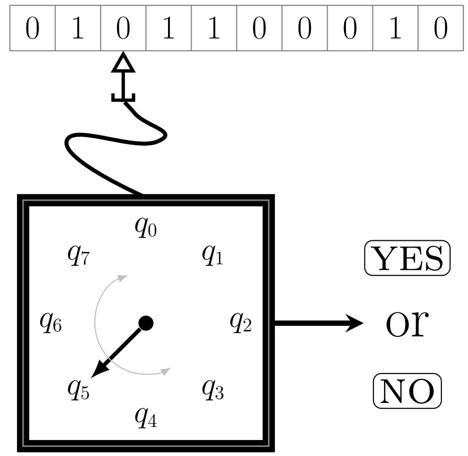


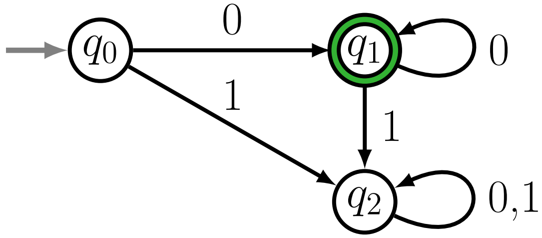
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1
4 :	q_1	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

A Simple Computing Machine



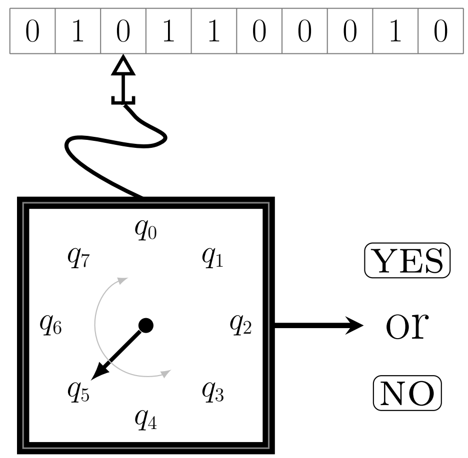


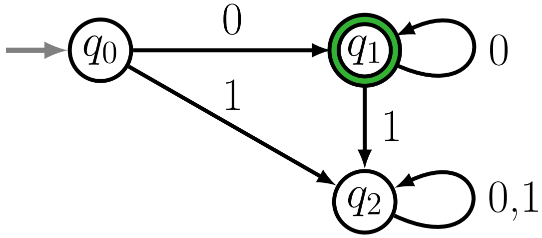
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1
4 :	q_1	1	q_2
5 :	q_2	0	q_2
6 :	q_2	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

A Simple Computing Machine





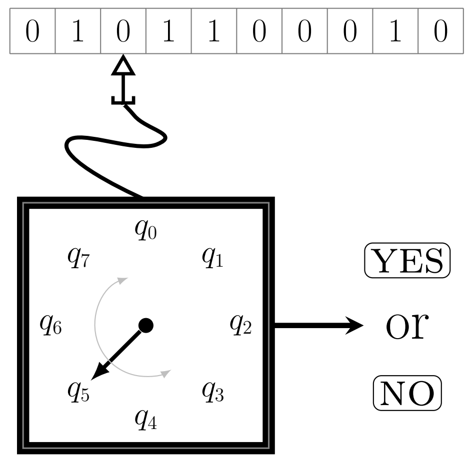
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

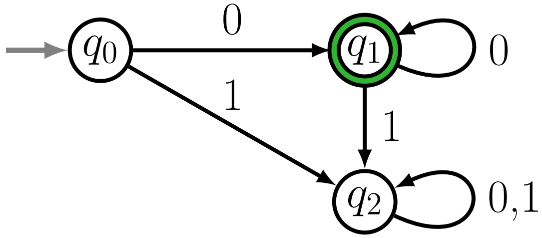
transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1
4 :	q_1	1	q_2
5 :	q_2	0	q_2
6 :	q_2	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

1: Process the input string (left-to-right) starting from the initial state q_0.

A Simple Computing Machine





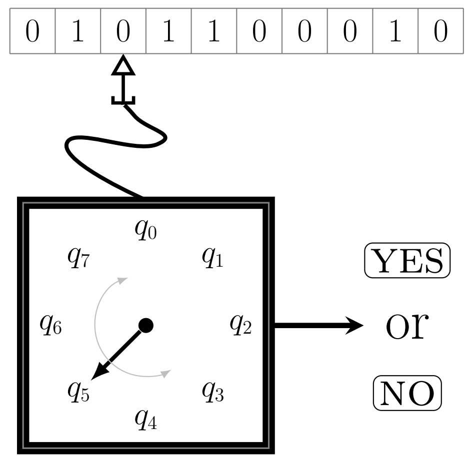
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

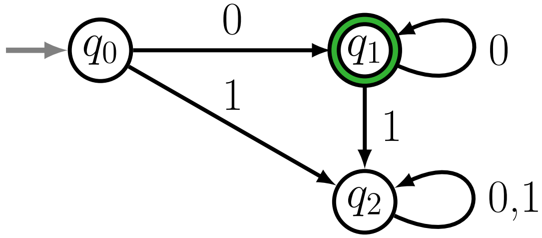
transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1
4 :	q_1	1	q_2
5 :	q_2	0	q_2
6 :	q_2	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

1: Process the input string (left-to-right) starting from the initial state q_0.
2: Process one bit at a time, each time transitioning from the current state to the next state according to the transition instructions.

A Simple Computing Machine





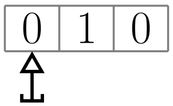
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

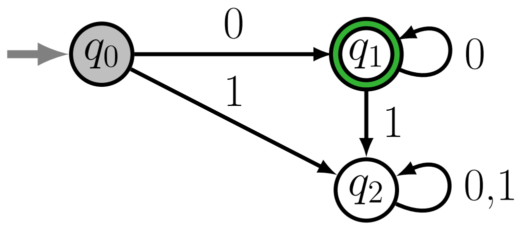
transitions			
1 :	q_0	0	q_1
2 :	q_0	1	q_2
3 :	q_1	0	q_1
4 :	q_1	1	q_2
5 :	q_2	0	q_2
6 :	q_2	1	q_2

$\leftarrow$ In state q_0, if you read 0, transition to q_1

1: Process the input string (left-to-right) starting from the initial state q_0.
2: Process one bit at a time, each time transitioning from the current state to the next state according to the transition instructions.
3: When done processing every bit, output YES if the final resting state of the DFA is a YES-state; otherwise output NO.

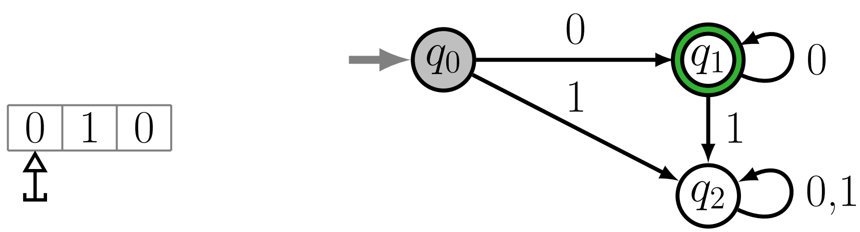
Running the Machine on an Input



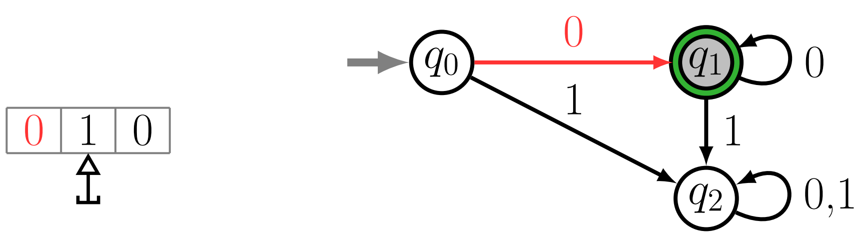


$q_0|\triangleright 010$

Running the Machine on an Input



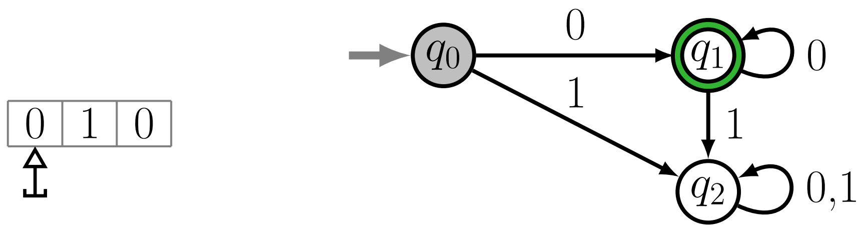
$q_0|\triangleright 010$



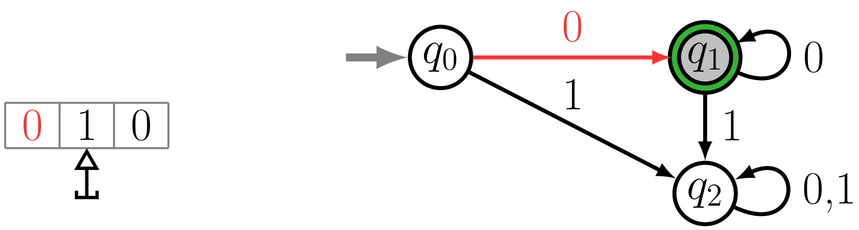
$q_0|\triangleright 010 \quad \overset{M}{\mapsto} \quad q_1|0\triangleright 10$

$\left(\begin{array}{l} M \text{ is the name} \\ \text{of our “Machine”} \end{array} \right)$

Running the Machine on an Input

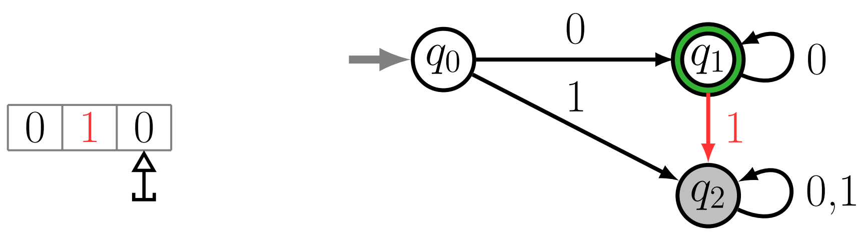


$q_0|\triangleright 010$



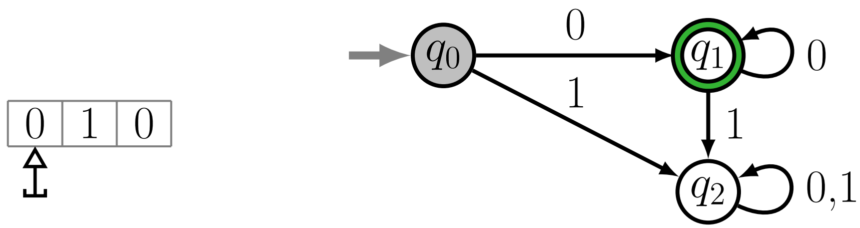
$q_0|\triangleright 010 \ \overset{M}{\mapsto}\ q_1|0\triangleright 10$

(M is the name of our “Machine”)

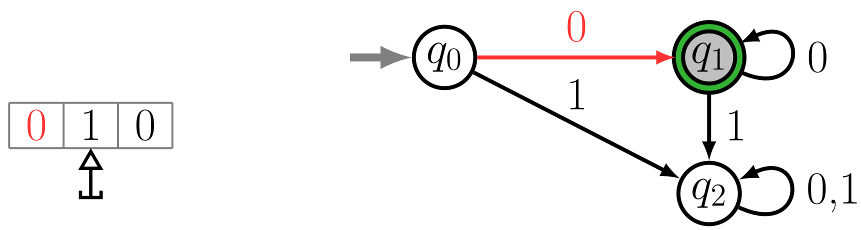


$$\begin{aligned} q_0|\triangleright 010 \ &\overset{M}{\mapsto}\ q_1|0\triangleright 10 \\ &\overset{M}{\mapsto}\ q_2|01\triangleright 0 \end{aligned}$$

Running the Machine on an Input

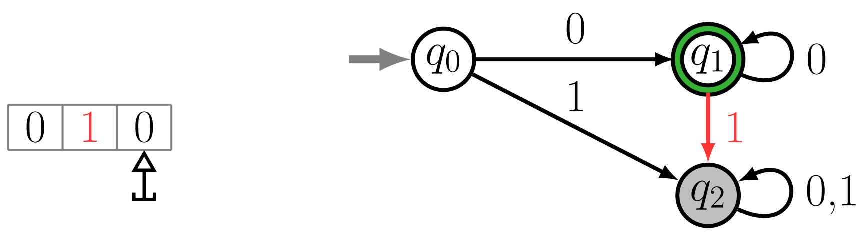


$q_0|\triangleright 010$

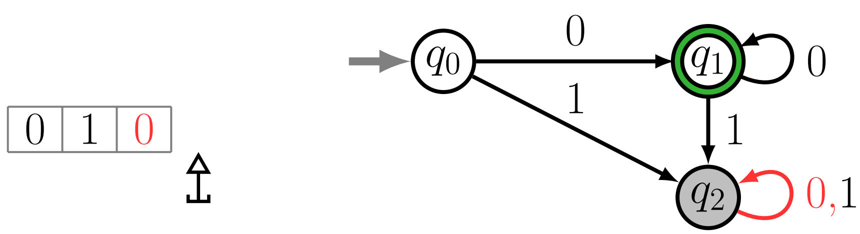


$q_0|\triangleright 010 \overset{M}{\mapsto} q_1|0\triangleright 10$

(M is the name of our "Machine")

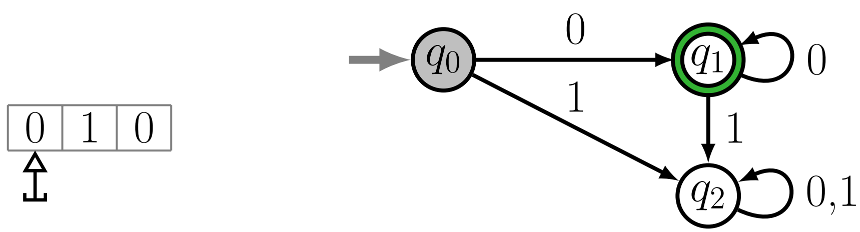


$$
\begin{aligned}
q_0|\triangleright 010 &\overset{M}{\mapsto} q_1|0\triangleright 10\\
&\overset{M}{\mapsto} q_2|01\triangleright 0
\end{aligned}
$$

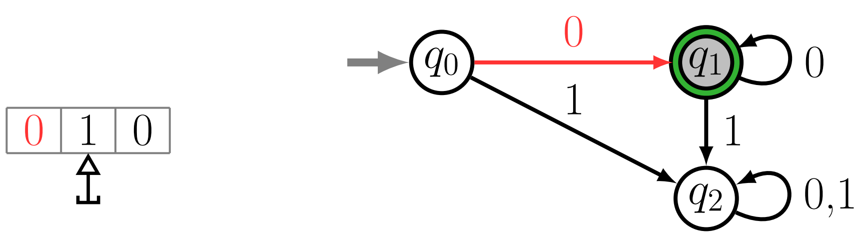


$$
\begin{aligned}
q_0|\triangleright 010 &\overset{M}{\mapsto} q_1|0\triangleright 10\\
&\overset{M}{\mapsto} q_2|01\triangleright 0\\
&\overset{M}{\mapsto} q_2|010\triangleright
\end{aligned}
$$

Running the Machine on an Input

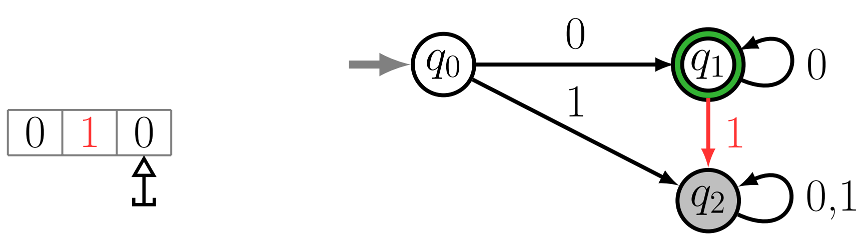


$q_0|\triangleright 010$

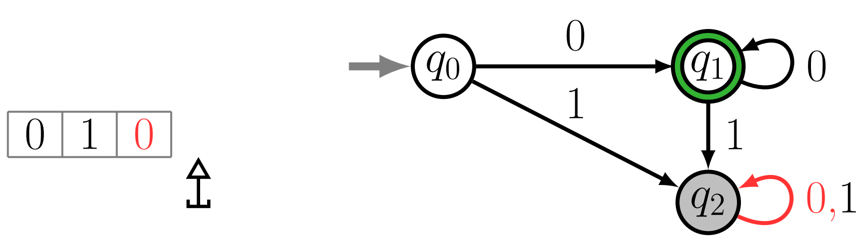


$q_0|\triangleright 010 \;\overset{M}{\mapsto}\; q_1|0\triangleright 10$

(M is the name of our "Machine")



$q_0|\triangleright 010 \;\overset{M}{\mapsto}\; q_1|0\triangleright 10$
$\overset{M}{\mapsto}\; q_2|01\triangleright 0$



$q_0|\triangleright 010 \;\overset{M}{\mapsto}\; q_1|0\triangleright 10$
$\overset{M}{\mapsto}\; q_2|01\triangleright 0$
$\overset{M}{\mapsto}\; q_2|010\triangleright$

$\boxed{\text{NO}}$, REJECT

Pop Quiz. Give computation trace for ε, 010, 000. What strings does the machine ACCEPT and say $\boxed{\text{YES}}$?

Pop Quiz. Determine $\boxed{\text{YES}}$ or $\boxed{\text{NO}}$ if you can from partial traces. $q_0 \mid \boxed{?} \triangleright 0000$; $q_1 \mid \boxed{?} \triangleright 0000$; $q_2 \mid \boxed{?} \triangleright 0000$.

Computing Problem Solved by a DFA

The computing problem solved by M is the language $\mathcal{L}(M) = \{w \mid M(w) = \boxed{\text{YES}}\}$.

$\mathcal{L}(M)$ is the automaton's $\boxed{\text{YES}}$-set. For our automaton M

$$\mathcal{L}(M) = \{0, 00, 000, 0000, \ldots\} = \{0^{\bullet n} \mid n > 0\}.$$

Computing Problem Solved by a DFA

The computing problem solved by M is the language $\mathcal{L}(M) = \{w \mid M(w) = \boxed{\text{YES}}\}$.

$\mathcal{L}(M)$ is the automaton's $\boxed{\text{YES}}$-set. For our automaton M

$$\mathcal{L}(M) = \{0, 00, 000, 0000, \ldots\} = \{0^{\bullet n} \mid n > 0\}.$$

1. For an automaton M, what is the computing problem $\mathcal{L}(M)$ solved by M?
2. For a computing problem $\mathcal{L}$, what automaton M solves $\mathcal{L}$, i.e., $\mathcal{L}(M) = \mathcal{L}$?

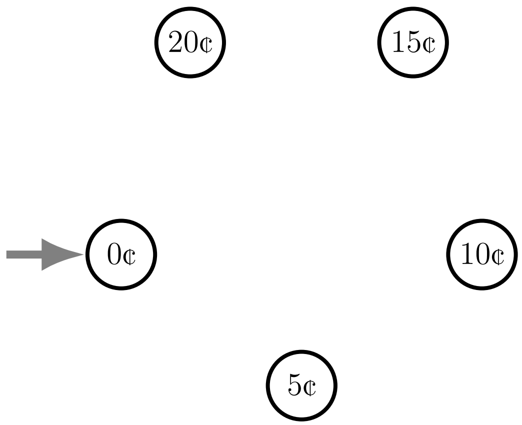
Practice. Exercise 24.2 gives you lots of training in question 1.

The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.

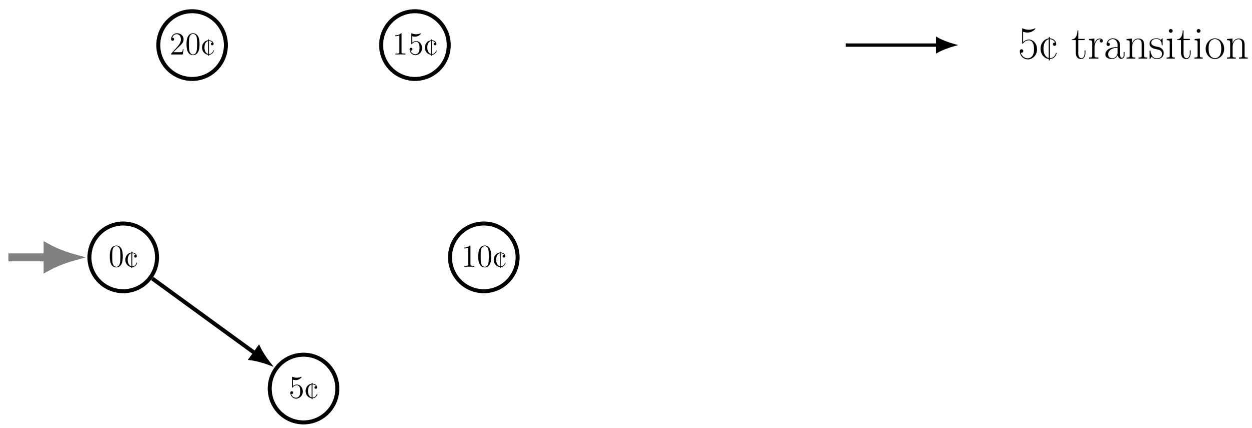
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



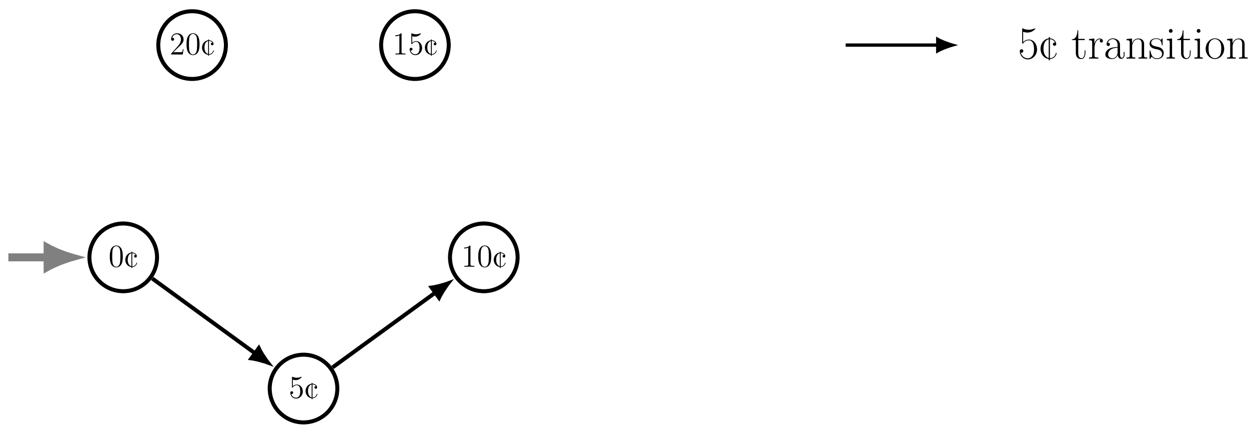
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



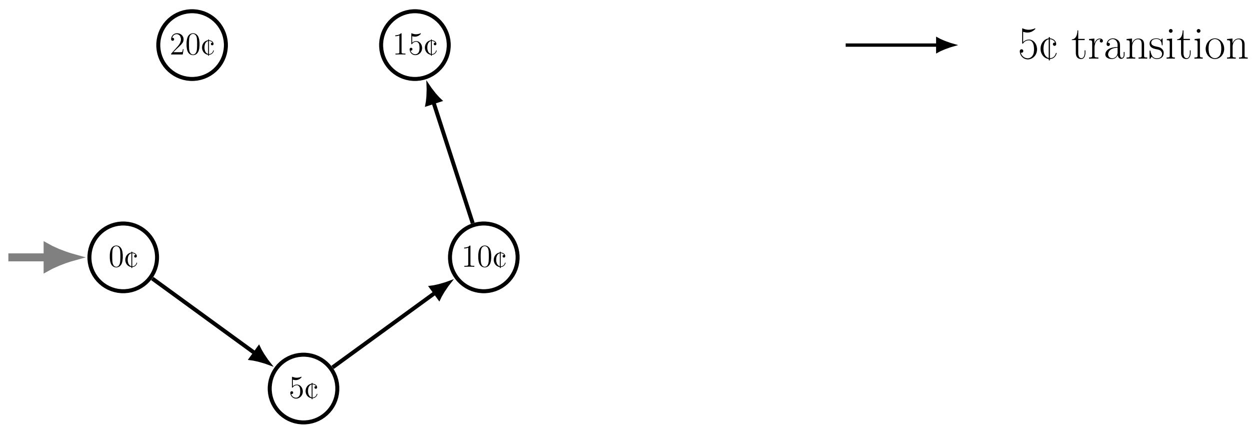
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



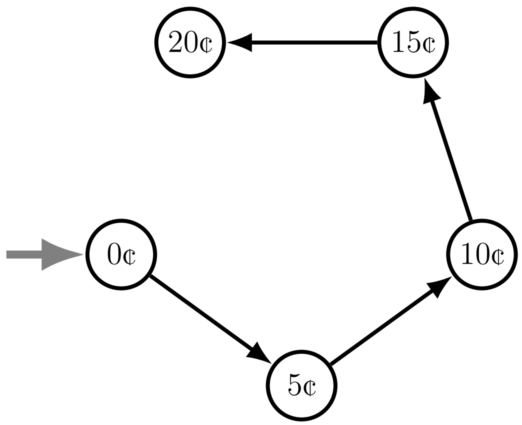
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



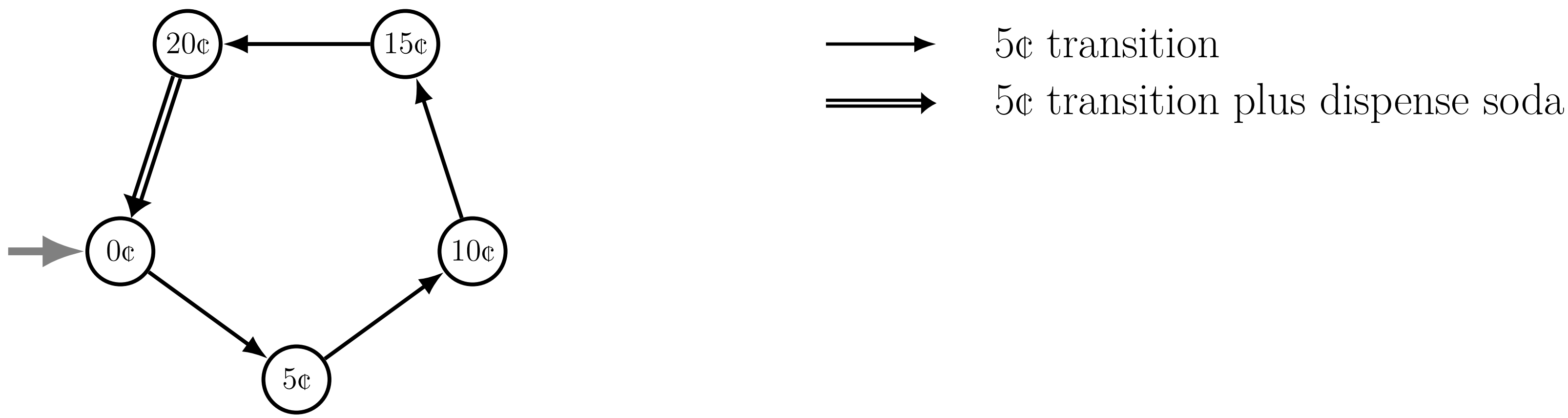
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



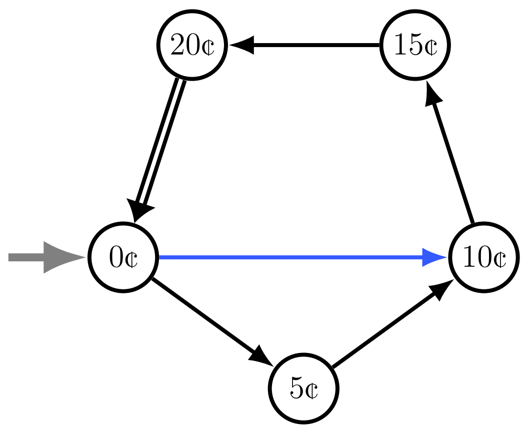
The Vending Machine

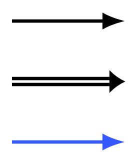
Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



The Vending Machine

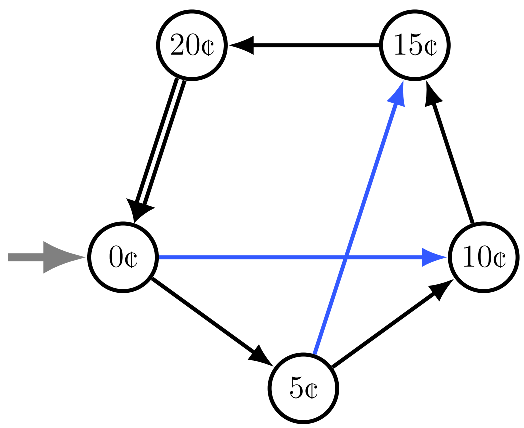
Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.





The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



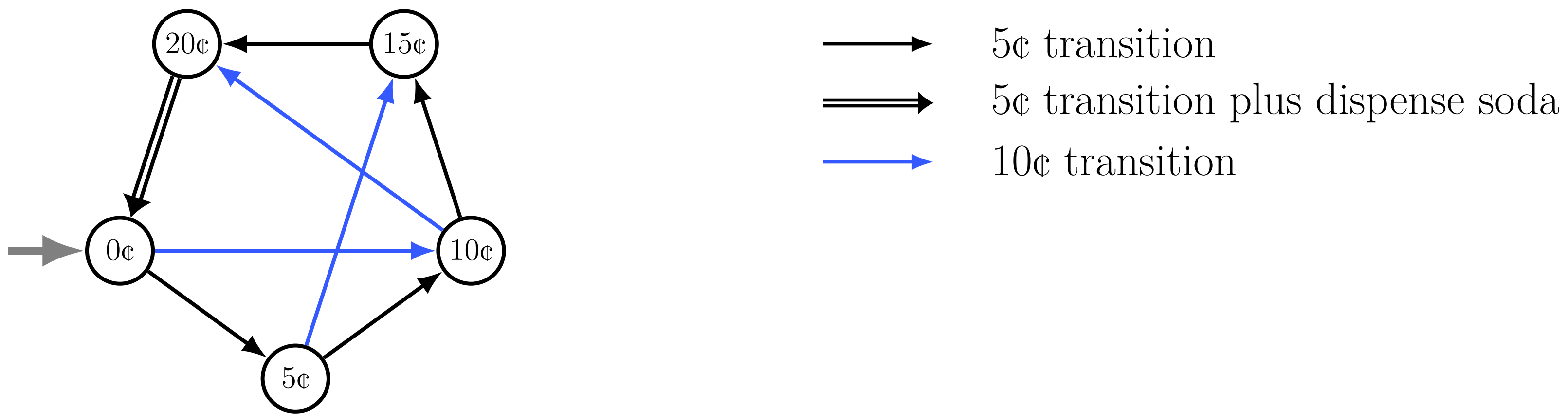
5¢ transition

5¢ transition plus dispense soda

10¢ transition

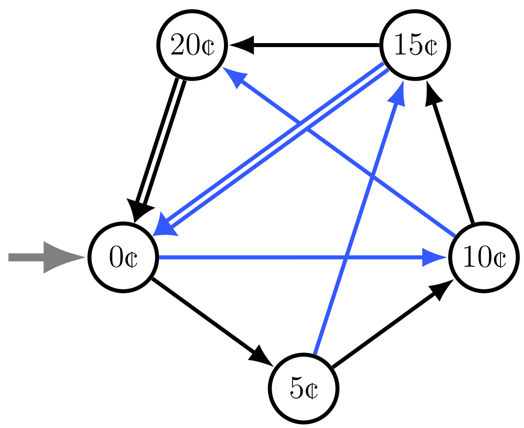
The Vending Machine

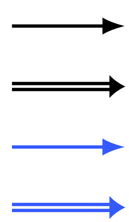
Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



The Vending Machine

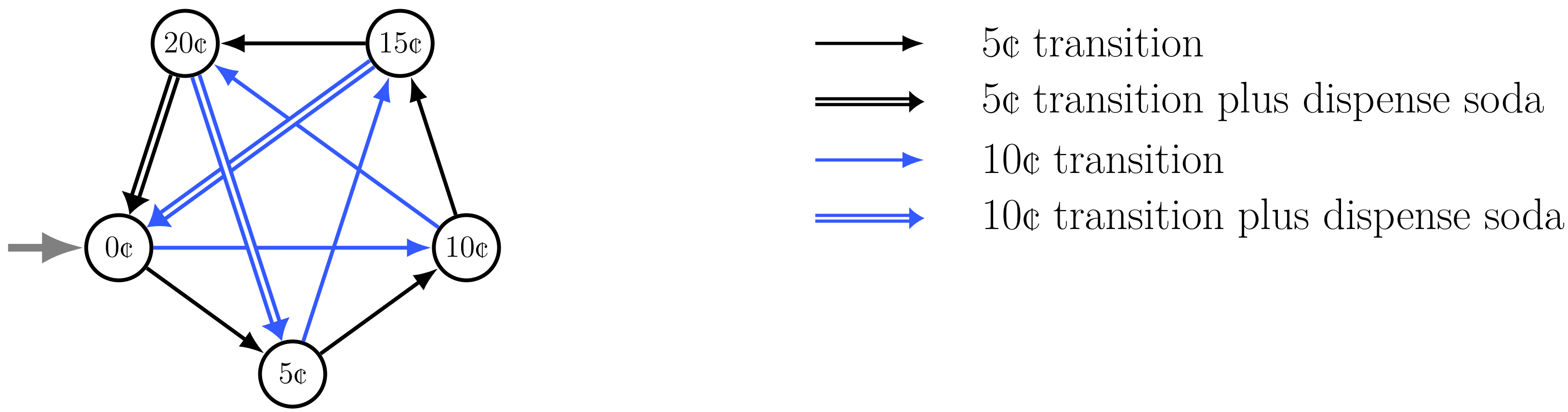
Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.





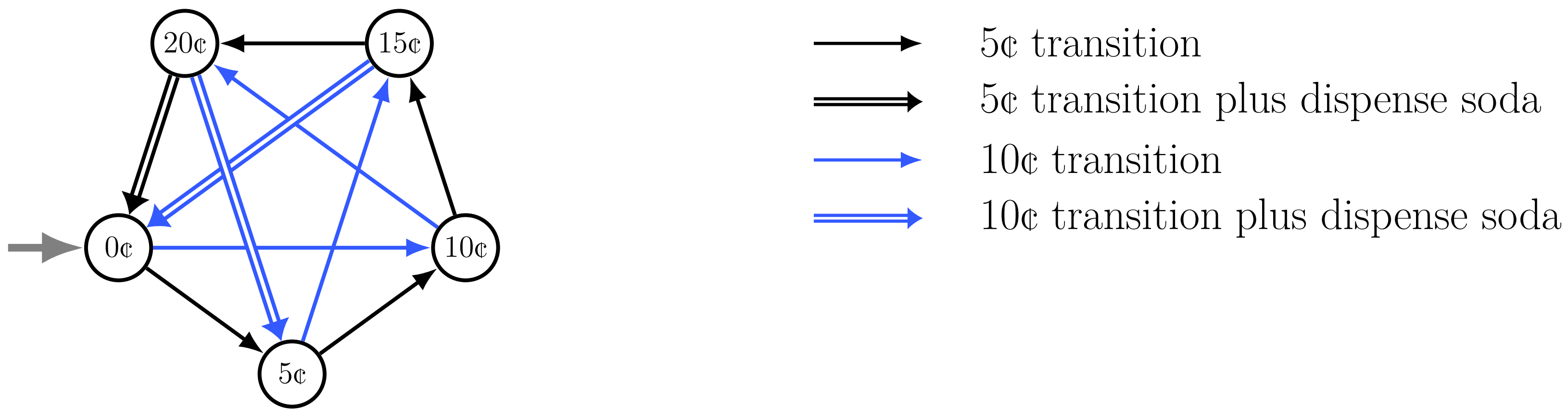
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



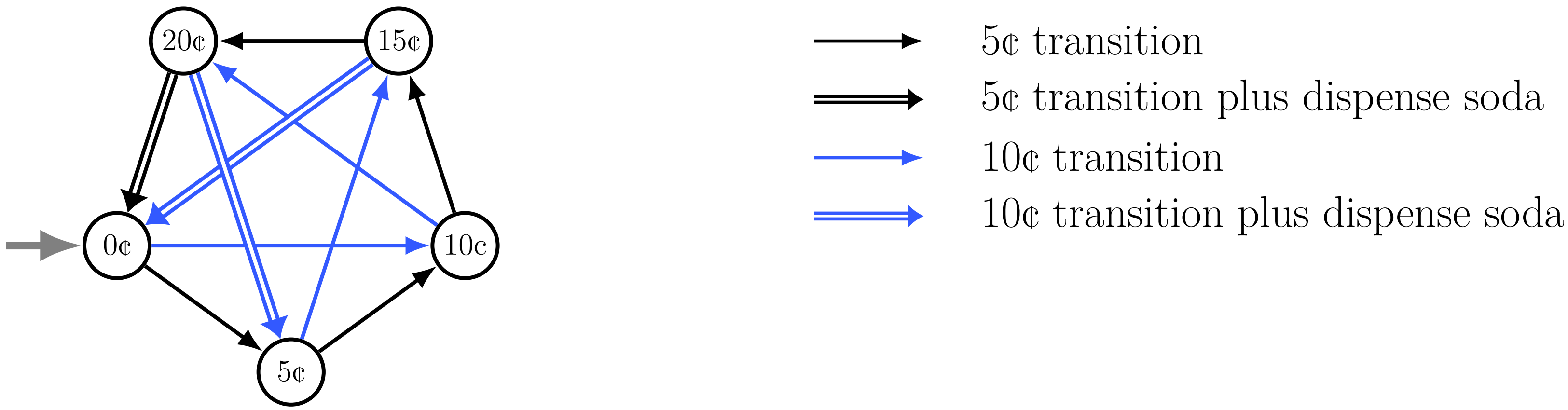
The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.

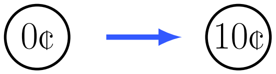


The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.

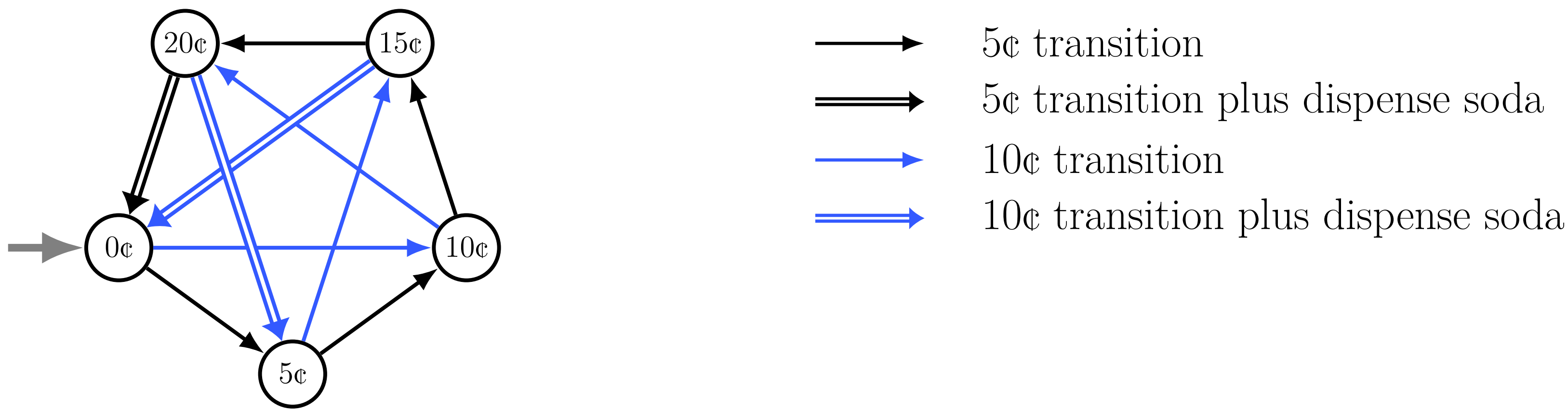


Input sequence: 10¢,

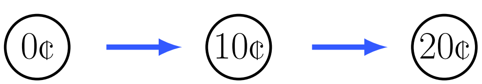


The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.

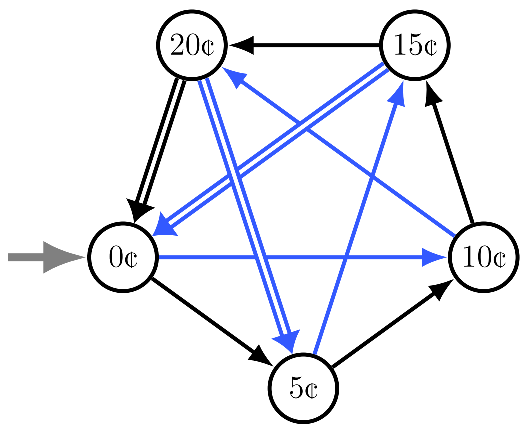


Input sequence: 10¢, 10¢,



The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



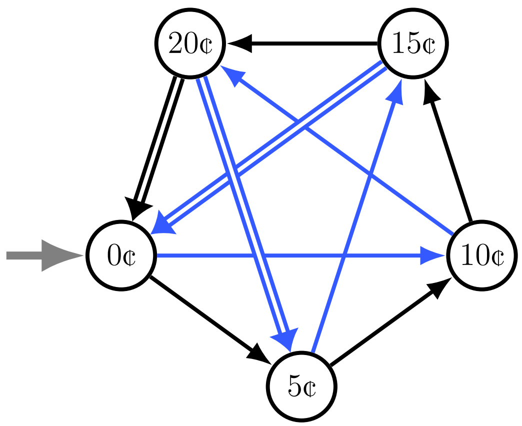
- 5¢ transition
- 5¢ transition plus dispense soda
- 10¢ transition
- 10¢ transition plus dispense soda

Input sequence: 10¢, 10¢, 5¢,

0¢ → 10¢ → 20¢ ⇒ 0¢ (+ soda)

The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



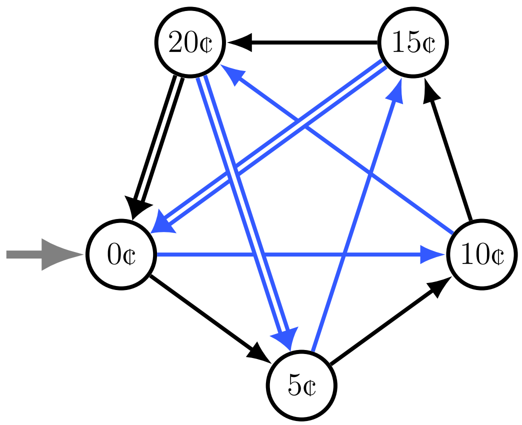
- 5¢ transition
- 5¢ transition plus dispense soda
- 10¢ transition
- 10¢ transition plus dispense soda

Input sequence: 10¢, 10¢, 5¢, 10¢,

0¢ → 10¢ → 20¢ ⇒ 0¢ (+ soda) → 10¢

The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



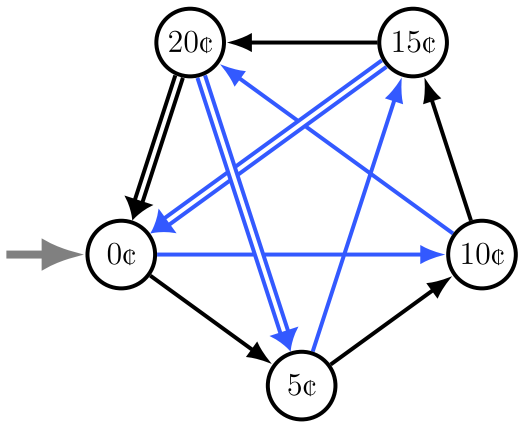
- 5¢ transition
- 5¢ transition plus dispense soda
- 10¢ transition
- 10¢ transition plus dispense soda

Input sequence: 10¢, 10¢, 5¢, 10¢, 10¢,

0¢ → 10¢ → 20¢ ⇒ 0¢ (+ soda) → 10¢ → 20¢

The Vending Machine

Vending machine takes nickels and dimes and dispenses a soda when it has 25¢.



5¢ transition

5¢ transition plus dispense soda

10¢ transition

10¢ transition plus dispense soda

Input sequence: 10¢, 10¢, 5¢, 10¢, 10¢, 10¢.

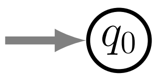
0¢ → 10¢ → 20¢ ⇒ 0¢ (+ soda) → 10¢ → 20¢ ⇒ 5¢ (+ soda).

DFA for a Finite Language

$$\mathcal{L} = \{10\}.$$

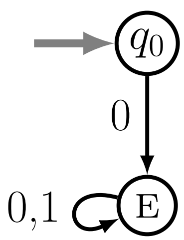
DFA for a Finite Language

$$\mathcal{L} = \{10\}.$$



DFA for a Finite Language

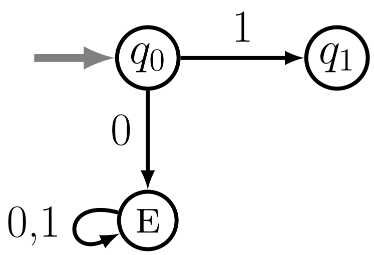
$$\mathcal{L} = \{10\}.$$



- 0 means move to a rejecting ERROR state and stay there

DFA for a Finite Language

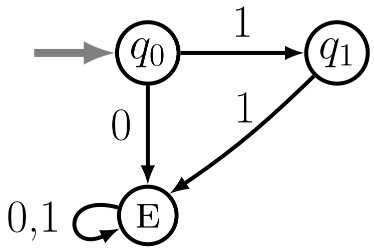
$$\mathcal{L} = \{10\}.$$



- 0 means move to a rejecting ERROR state and stay there
- 1 is partial success.

DFA for a Finite Language

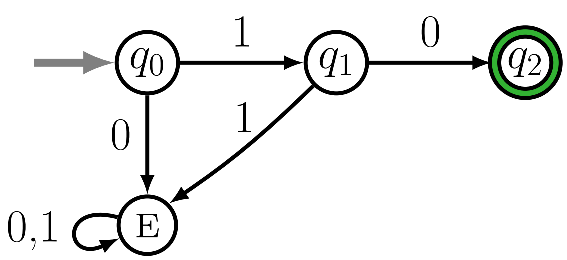
$$\mathcal{L} = \{10\}.$$



- 0 means move to a rejecting ERROR state and stay there
- 1 is partial success.
- Another 1 puts you into ERROR since you want 0;

DFA for a Finite Language

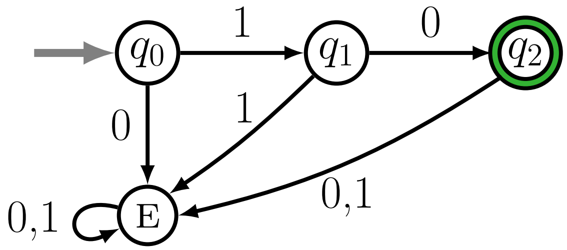
$$\mathcal{L} = \{10\}.$$



- 0 means move to a rejecting ERROR state and stay there
- 1 is partial success.
- Another 1 puts you into ERROR since you want 0;
- 0 from q_1 and you are ready to accept . . . unless . . .

DFA for a Finite Language

$$\mathcal{L} = \{10\}.$$



- 0 means move to a rejecting ERROR state and stay there
- 1 is partial success.
- Another 1 puts you into ERROR since you want 0;
- 0 from q_1 and you are ready to accept . . . unless . . .
- More bits arrive, in which case move to ERROR.

Practice. Try random strings other than 01 and make sure our DFA rejects them.

DFAs for Infinite Languages

$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

DFAs for Infinite Languages

$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

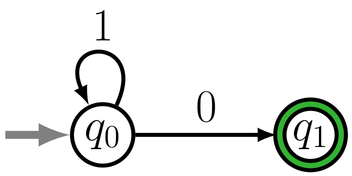
(wildcard $* = \Sigma^*$)

DFAs for Infinite Languages

$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a }0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in }1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

(wildcard $* = \Sigma^*$)



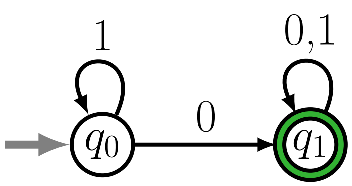
M_1

DFAs for Infinite Languages

$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

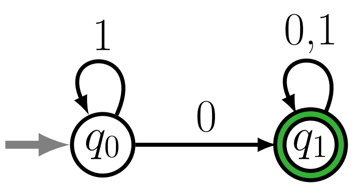
(wildcard $* = \Sigma^*$)



M_1

DFAs for Infinite Languages

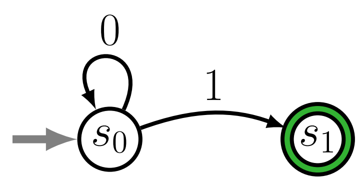
$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$



M_1

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

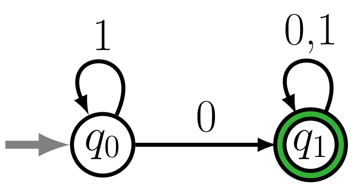
(wildcard $* = \Sigma^*$)



M_2

DFAs for Infinite Languages

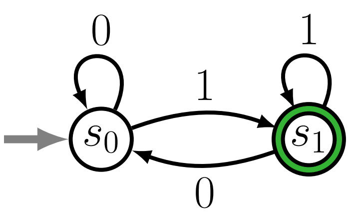
$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$



M_1

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

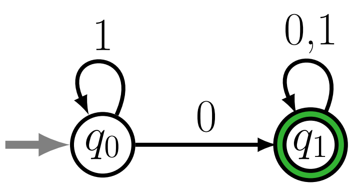
(wildcard $* = \Sigma^*$)



M_2

DFAs for Infinite Languages

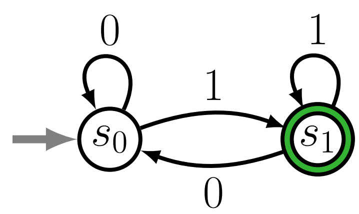
$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$



M_1

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

(wildcard $* = \Sigma^*$)



M_2

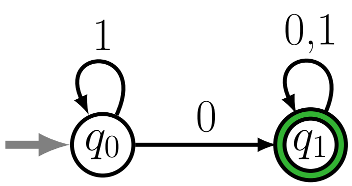
Complement. Consider $\overline{\mathcal{L}_1}$: Must ACCEPT strings M_1 REJECTS.

DFAs for Infinite Languages

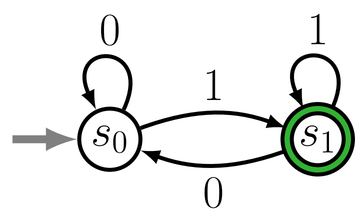
$$\begin{aligned}\mathcal{L}_1 &= *0* \\ &= \{\text{strings with a } 0\} \\ &= \{0, 00, 01, 10, 000, 001, 010, 011, 100, \ldots\}\end{aligned}$$

$$\begin{aligned}\mathcal{L}_2 &= *1 \\ &= \{\text{strings ending in } 1\} \\ &= \{1, 01, 11, 001, 011, 101, 111, \ldots\}\end{aligned}$$

(wildcard $* = \Sigma^*$)

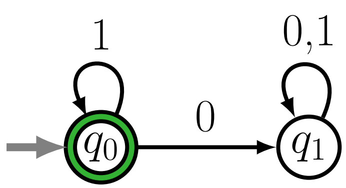


M_1



M_2

Complement. Consider $\overline{\mathcal{L}_1}$: Must ACCEPT strings M_1 REJECTS.

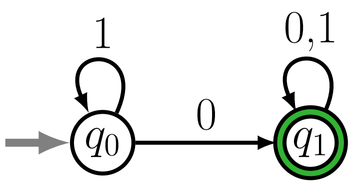


M

$\leftarrow$ flip $\boxed{\text{YES}}$ and $\boxed{\text{NO}}$-states.

Two DFAs in One: Union and Intersection

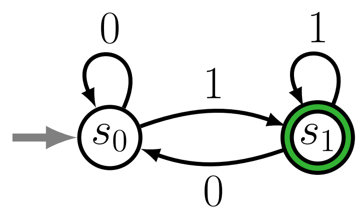
$\mathcal{L}_1 = *0*$



M_1

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

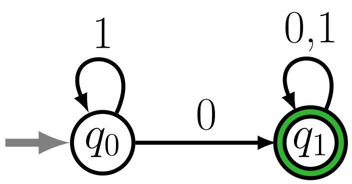


M_2

The Joint-DFA has product states $\{q_0 s_0, q_0 s_1, q_1 s_0, q_1 s_1\}$:

Two DFAs in One: Union and Intersection

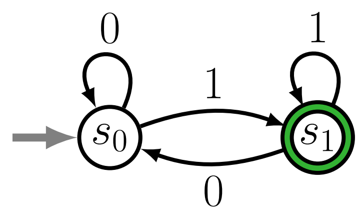
$\mathcal{L}_1 \;=\; *0*$



M_1

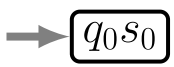
$\mathcal{L}_2 \;=\; *1$

(wildcard $* = \Sigma^*$)



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



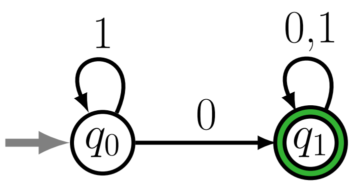
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.

Two DFAs in One: Union and Intersection

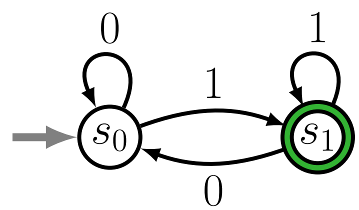
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

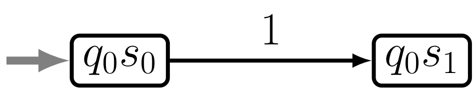


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



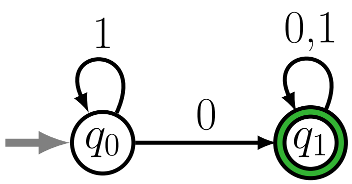
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.

Two DFAs in One: Union and Intersection

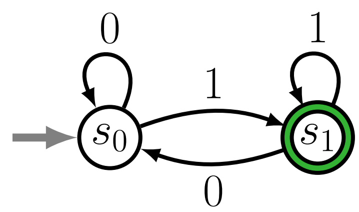
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

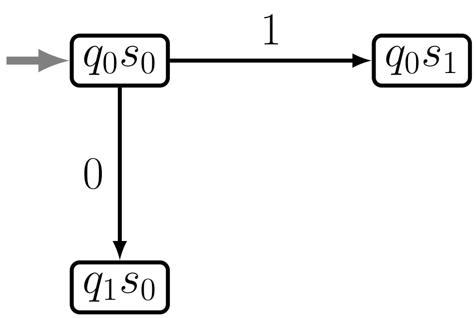


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



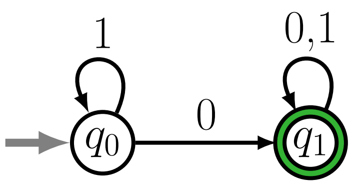
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0.

Two DFAs in One: Union and Intersection

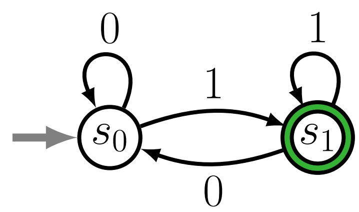
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

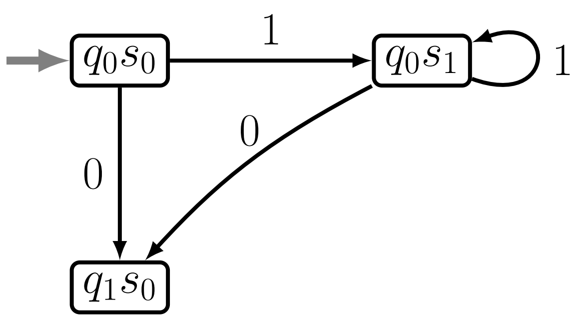


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



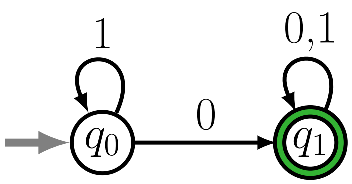
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0.

Two DFAs in One: Union and Intersection

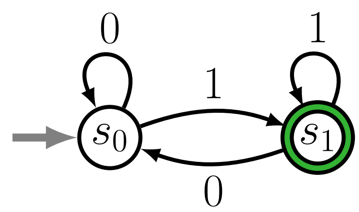
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

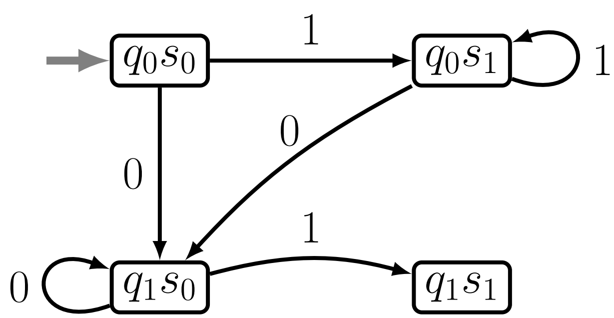


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



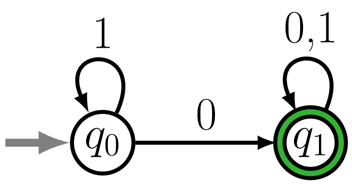
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0.
q_1s_1: M_1 is in state q_1 and M_2 is in state s_1.

Two DFAs in One: Union and Intersection

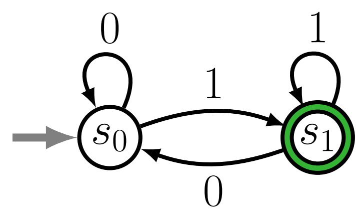
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

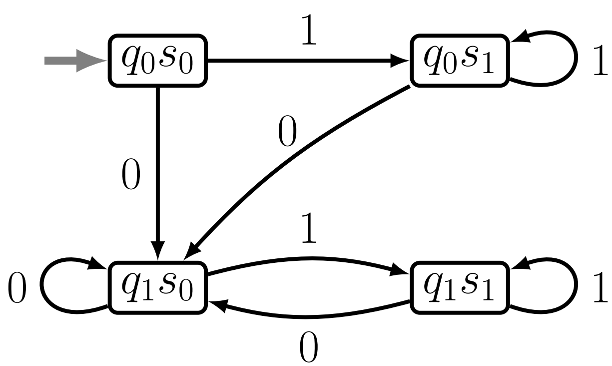


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



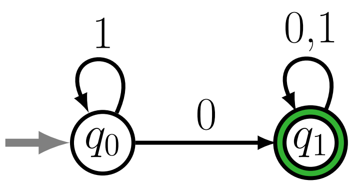
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0.
q_1s_1: M_1 is in state q_1 and M_2 is in state s_1.

Two DFAs in One: Union and Intersection

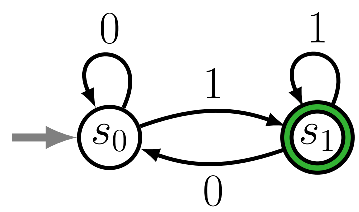
$\mathcal{L}_1 = *0*$

$\mathcal{L}_2 = *1$

(wildcard $* = \Sigma^*$)

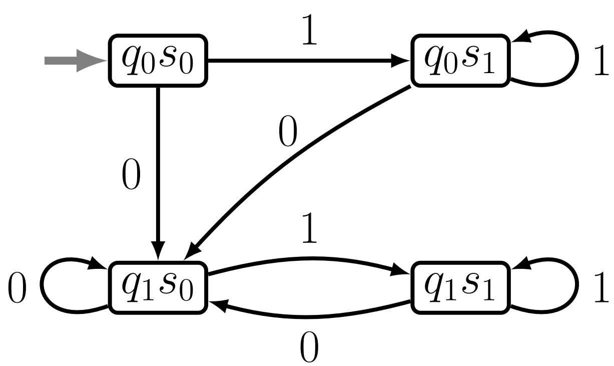


M_1



M_2

The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$:



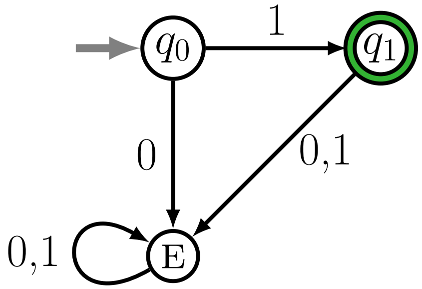
q_0s_0: M_1 is in state q_0 and M_2 is in state s_0.
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1.
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0.
q_1s_1: M_1 is in state q_1 and M_2 is in state s_1.

Pop Quiz.

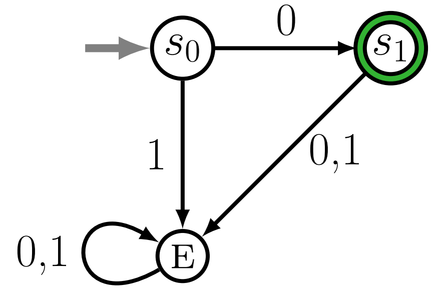
1. Run the joint and individual DFAs for $\varepsilon, 0100, 11, 101$. What are the final states of each DFA?
2. If you want to solve $\mathcal{L}_1 \cup \mathcal{L}_2$, what should the accept states of the joint-DFA be?
3. If you want to solve $\mathcal{L}_1 \cap \mathcal{L}_2$, what should the accept states of the joint-DFA be?

Concatenation and Kleene Star

$\mathcal{L}_1 = \{1\}$
(M_1)

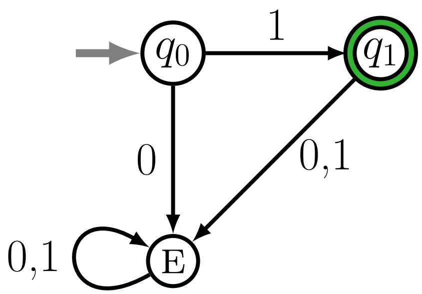


$\mathcal{L}_2 = \{0\}$
(M_2)

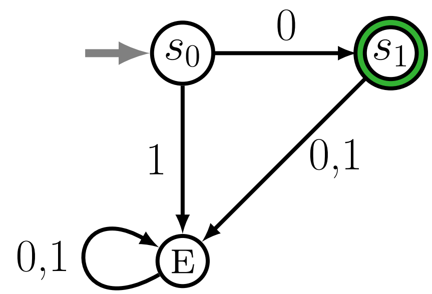


Concatenation and Kleene Star

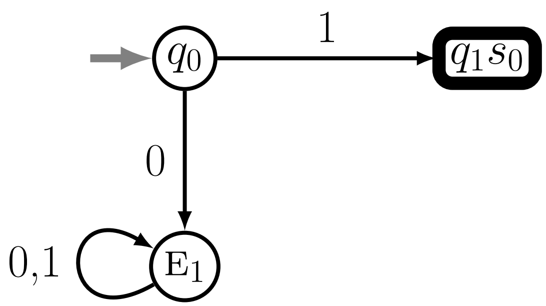
$\mathcal{L}_1 = \{1\}$
(M_1)



$\mathcal{L}_2 = \{0\}$
(M_2)

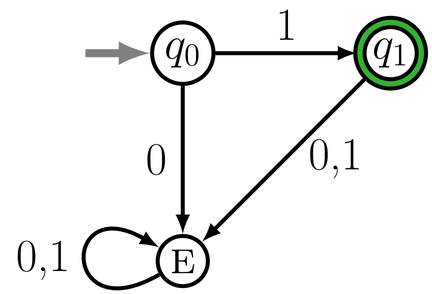


$\mathcal{L}_1 \bullet \mathcal{L}_2$:

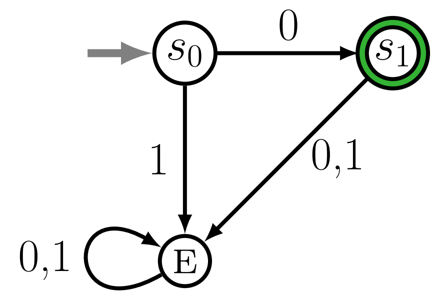


Concatenation and Kleene Star

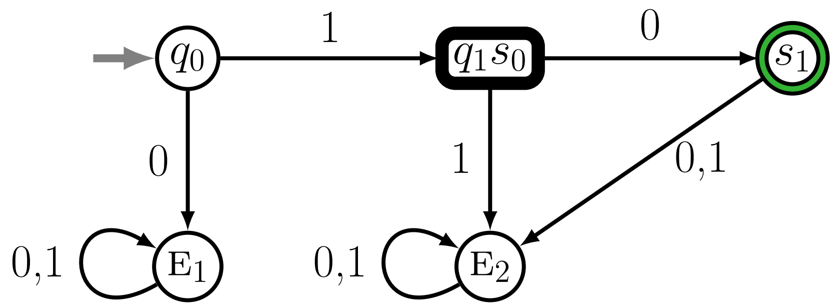
$\mathcal{L}_1 = \{1\}$
(M_1)



$\mathcal{L}_2 = \{0\}$
(M_2)

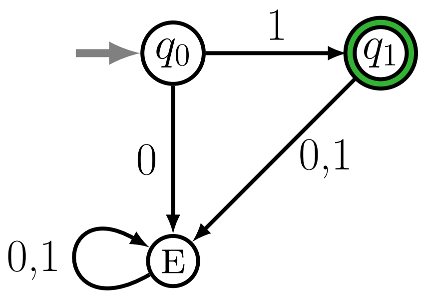


$\mathcal{L}_1 \bullet \mathcal{L}_2$:

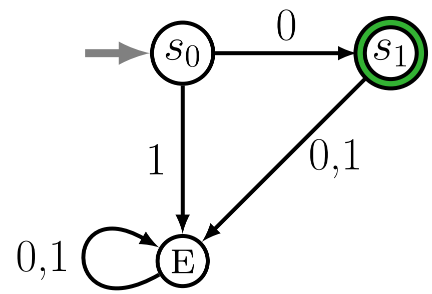


Concatenation and Kleene Star

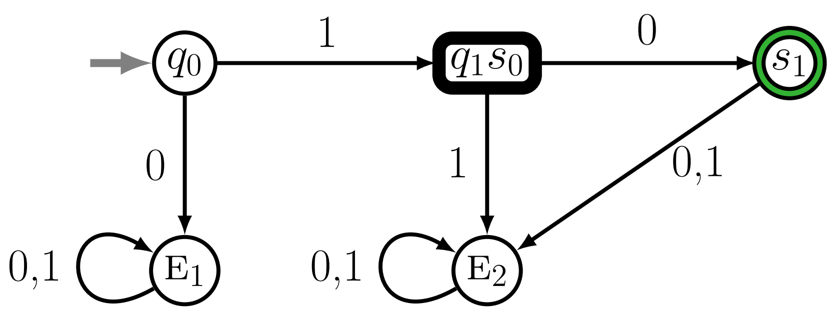
$\mathcal{L}_1 = \{1\}$
(M_1)



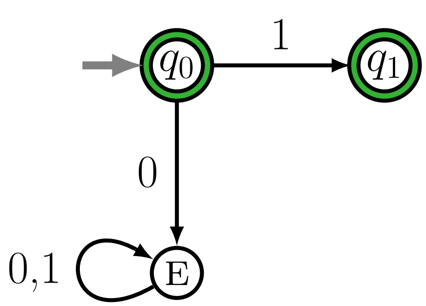
$\mathcal{L}_2 = \{0\}$
(M_2)



$\mathcal{L}_1 \bullet \mathcal{L}_2$:

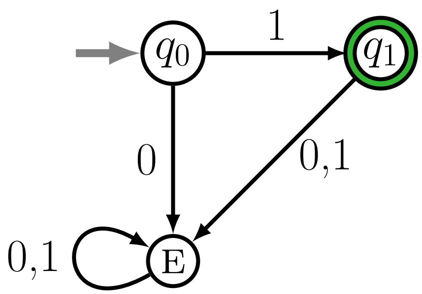


$\mathcal{L}_1^*$:

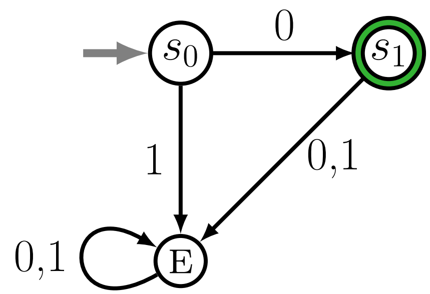


Concatenation and Kleene Star

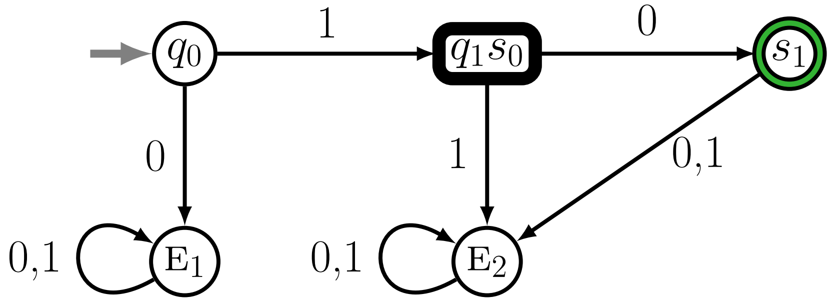
$\mathcal{L}_1 = \{1\}$
(M_1)



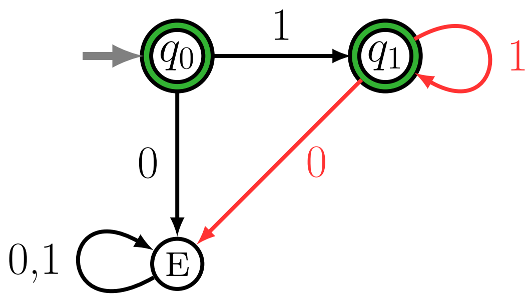
$\mathcal{L}_2 = \{0\}$
(M_2)



$\mathcal{L}_1 \bullet \mathcal{L}_2$:



$\mathcal{L}_1^*$:



The Power of DFAs: What can they Solve?

- Finite languages.
  (building blocks of regular expressions)

- Complement, intersection, union.
  (operations to form complex regular expressions)

- Concatenation and Kleene-star (little more complicated, see text).
  (operations to form complex regular expressions)

That's what we need for regular expressions.

The Power of DFAs: What can they Solve?

- Finite languages.
  (building blocks of regular expressions)

- Complement, intersection, union.
  (operations to form complex regular expressions)

- Concatenation and Kleene-star (little more complicated, see text).
  (operations to form complex regular expressions)

That's what we need for regular expressions.

DFAs solve languages (computing problems) expressed as regular expressions.

(That is why the languages solved by DFAs are called regular languages.)

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n} 1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n} 1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n} 1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n 1^n} = \{0^{\bullet n} 1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n 1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n 1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0})$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet 1})$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n} 1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n 1^n} = \{0^{\bullet n} 1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n 1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n 1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet 1}) \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet 2})$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet 1}) \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet 2}) \quad \overset{M}{\mapsto} \quad \cdots \quad \overset{M}{\mapsto} \quad \text{state}(0^{\bullet k-1})$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Consider the two input strings $0^{\bullet i}1^{\bullet i} \in \mathcal{L}_{0^n1^n}$ and $0^{\bullet j}1^{\bullet i} \notin \mathcal{L}_{0^n1^n}$.

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Consider the two input strings $0^{\bullet i}1^{\bullet i} \in \mathcal{L}_{0^n1^n}$ and $0^{\bullet j}1^{\bullet i} \notin \mathcal{L}_{0^n1^n}$.
After M has processed the 0s in both strings, it is in state q, and the traces of the two computations are

■

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Consider the two input strings $0^{\bullet i}1^{\bullet i} \in \mathcal{L}_{0^n1^n}$ and $0^{\bullet j}1^{\bullet i} \notin \mathcal{L}_{0^n1^n}$.
After M has processed the 0s in both strings, it is in state q, and the traces of the two computations are

$$q \mid 0^{\bullet i} \triangleright 1^{\bullet i} \qquad \text{and} \qquad q \mid 0^{\bullet j} \triangleright 1^{\bullet i}.$$

∎

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n}1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Consider the two input strings $0^{\bullet i}1^{\bullet i} \in \mathcal{L}_{0^n1^n}$ and $0^{\bullet j}1^{\bullet i} \notin \mathcal{L}_{0^n1^n}$.
After M has processed the 0s in both strings, it is in state q, and the traces of the two computations are

$$q \mid \boxed{\;?\;}\rhd 1^{\bullet i} \qquad \text{and} \qquad q \mid \boxed{\;?\;}\rhd 1^{\bullet i}.$$

Same number of 1's remain, from state q. Either both rejected or both accepted. **FISHY!** ■

Is There Anything DFAs Can't Solve?

Pop Quiz. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n} 1^{\bullet k} \mid n \geq 0, k \geq 0\}$.

What about "equality,"

$$\mathcal{L}_{0^n 1^n} = \{0^{\bullet n} 1^{\bullet n} \mid n \geq 0\}.$$

Theorem. There is no DFA that solves $\mathcal{L}_{0^n 1^n}$

Proof. Contradiction. Suppose a DFA M with k states solves $\{0^n 1^n\}$.
What happens to this DFA when you keep feeding it 0's?

$$q_0 = \text{state}(0^{\bullet 0}) \overset{M}{\mapsto} \text{state}(0^{\bullet 1}) \overset{M}{\mapsto} \text{state}(0^{\bullet 2}) \overset{M}{\mapsto} \cdots \overset{M}{\mapsto} \text{state}(0^{\bullet k-1}) \overset{M}{\mapsto} \text{state}(0^{\bullet k})$$

After k 0's, $k+1$ states visited. There must be a repetition (pigeonhole).

$$\text{state}(0^{\bullet i}) = \text{state}(0^{\bullet j}) = q, \qquad i < j \leq k.$$

Consider the two input strings $0^{\bullet i} 1^{\bullet i} \in \mathcal{L}_{0^n 1^n}$ and $0^{\bullet j} 1^{\bullet i} \notin \mathcal{L}_{0^n 1^n}$.
After M has processed the 0s in both strings, it is in state q, and the traces of the computations are

$$q \mid \boxed{\ ?\ } \triangleright 1^{\bullet i} \qquad \text{and} \qquad q \mid \boxed{\ ?\ } \triangleright 1^{\bullet i}.$$

Same number of 1's remain, from state q. Either both rejected or both accepted. **FISHY!** ■

Intuition: The DFA has no "memory" to remember n.

Our First Computing Machine

DFAs can be implemented using basic technology, so practical.

Powerful (regular languages), but also limited.

Computing Model

Rules to:

1. Construct machine;
2. Solve problems.

Our First Computing Machine

DFAs can be implemented using basic technology, so practical.

Powerful (regular languages), but also limited.

Computing Model

Rules to:

1. Construct machine;
2. Solve problems.

→

Analyze Model

1. *Capabilities:* what can be solved?
2. *Limitations:* what can't be solved?

Our First Computing Machine

DFAs can be implemented using basic technology, so practical.

Powerful (regular languages), but also limited.

Computing Model

Rules to:

1. Construct machine;
2. Solve problems.

→

Analyze Model

1. *Capabilities:* what can be solved?
2. *Limitations:* what can't be solved?

→

Do we need a new model?

Our First Computing Machine

DFAs can be implemented using basic technology, so practical.

Powerful (regular languages), but also limited.

Computing Model

Rules to:

1. Construct machine;
2. Solve problems.

Analyze Model

1. *Capabilities:* what can be solved?
2. *Limitations:* what can't be solved?

Do we need a new model?

DFAs fail at so simple a problem as equality.

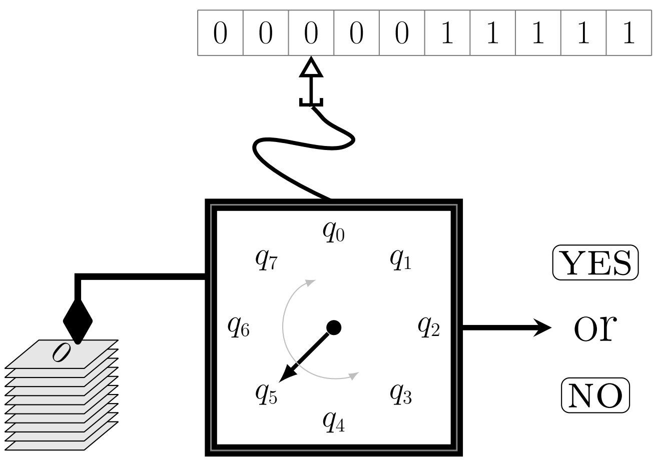
- That's not acceptable.
- We need a more powerful machine.

Adding Memory

DFAs have no scratch paper. It's hard to compute entirely in your head.

Adding Memory

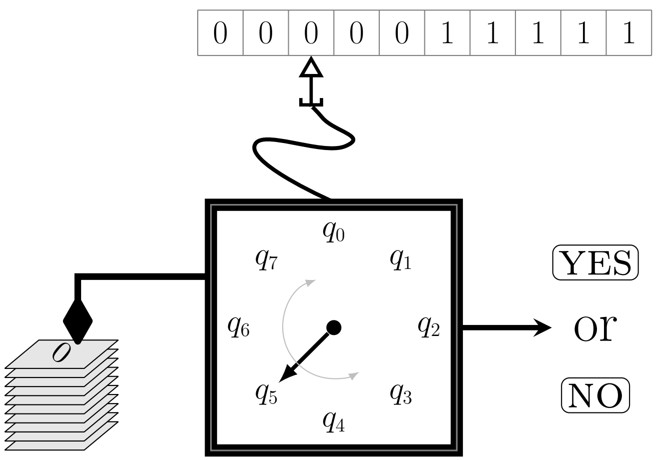
DFAs have no scratch paper. It's hard to compute entirely in your head.



Stack Memory. Think of a file-clerk with a stack of papers.

Adding Memory

DFAs have no scratch paper. It's hard to compute entirely in your head.



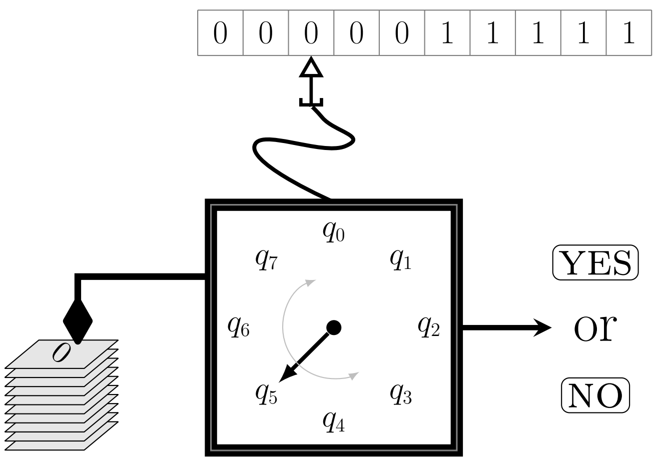
Stack Memory. Think of a file-clerk with a stack of papers. The clerk's capabilities:

- see the top sheet;
- remove the top sheet (*pop*)
- *push* something new onto the top of the stack.
- no access to inner sheets without removing top.

DFA with a stack is a *pushdown automaton (PDA)*

Adding Memory

DFAs have no scratch paper. It's hard to compute entirely in your head.



Stack Memory. Think of a file-clerk with a stack of papers. The clerk's capabilities:

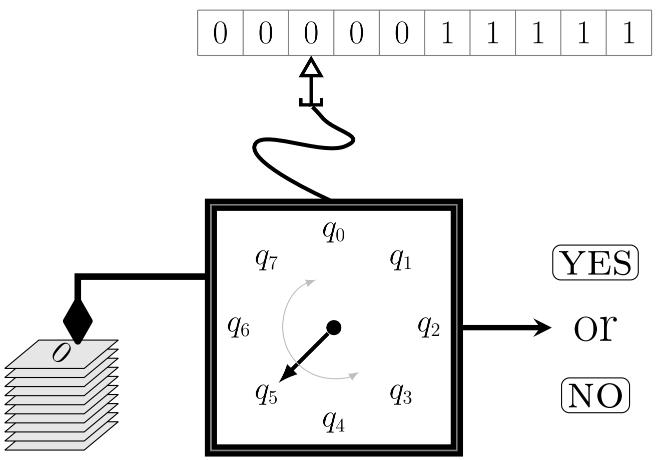
- see the top sheet;
- remove the top sheet (*pop*)
- *push* something new onto the top of the stack.
- no access to inner sheets without removing top.

DFA with a stack is a *pushdown automaton (PDA)*

How does the stack help to solve $\{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}$?

Adding Memory

DFAs have no scratch paper. It's hard to compute entirely in your head.



Stack Memory. Think of a file-clerk with a stack of papers. The clerk's capabilities:

- see the top sheet;
- remove the top sheet (*pop*)
- *push* something new onto the top of the stack.
- no access to inner sheets without removing top.

DFA with a stack is a *pushdown automaton (PDA)*

How does the stack help to solve $\{0^{\bullet n}1^{\bullet n} \mid n \geq 0\}$?

1: When you read in each 0, write it to the stack.
2: For each 1, pop the stack. At the end if the stack is empty, ACCEPT.

The memory allows the automaton to "remember" n.