Deterministic Finite Automata (DFA)

Reading

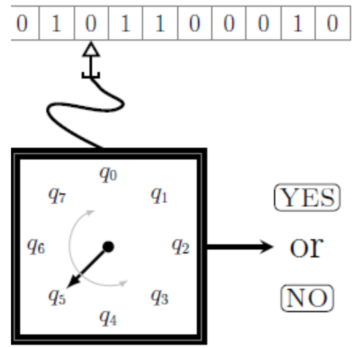
- Malik Magdon-Ismail. Discrete Mathematics and Computing.
  - Chapter 24

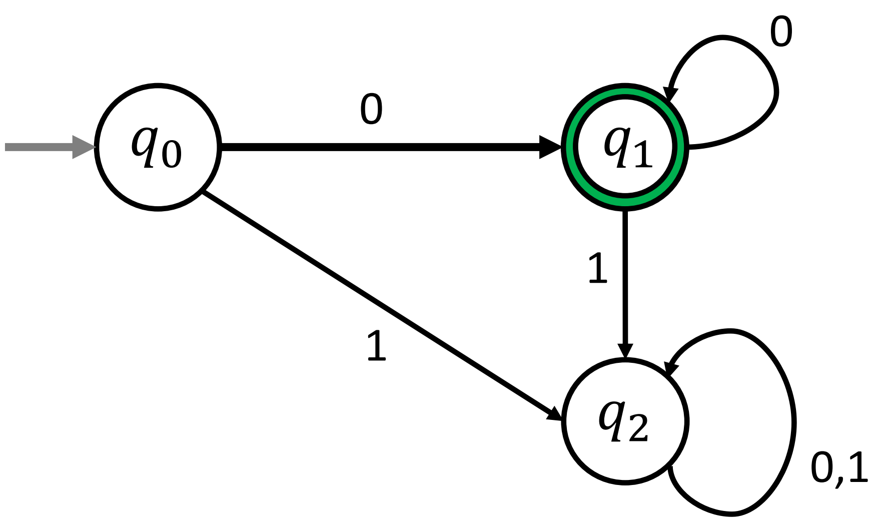
Overview

Rensselaer

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

A Simple Computing Machine



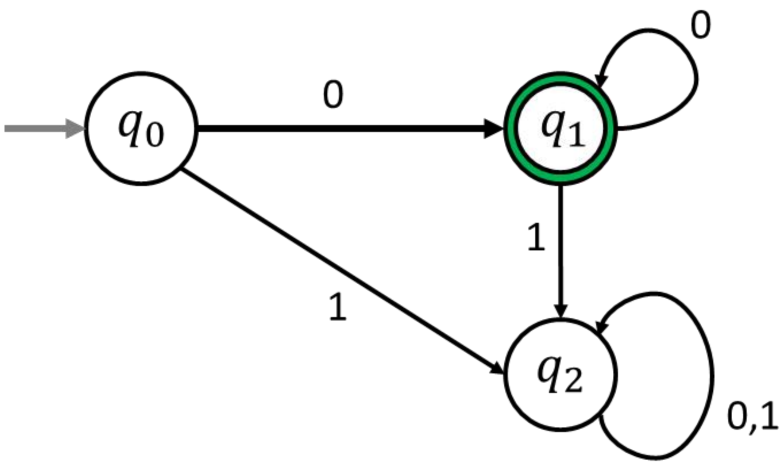


states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

- Transitions

1. q_0 0 q_1 — In state q_0, if you read 0, Transition to 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

A Simple Computing Machine, cont'd



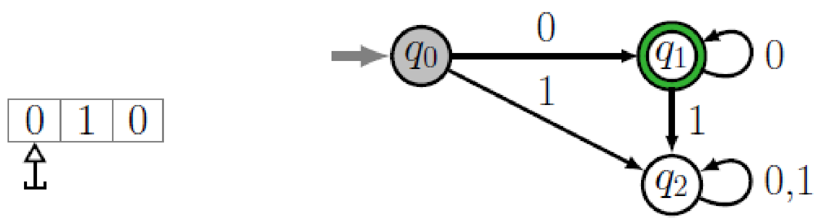
states	
$\rightarrow q_0$	NO
q_1	YES
q_2	NO

- Transitions

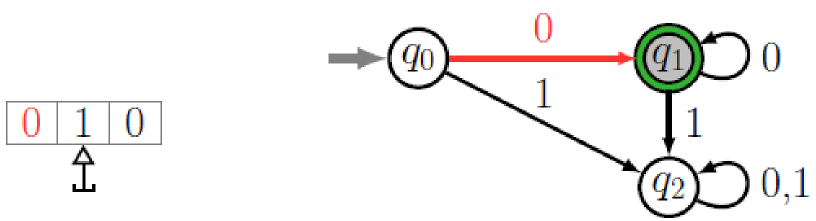
1. $q_0 \;\; 0 \;\; q_1$ In state q_0, if you read 0, Transition to q_1
2. $q_0 \;\; 1 \;\; q_2$
3. $q_1 \;\; 0 \;\; q_1$
4. $q_1 \;\; 1 \;\; q_2$
5. $q_2 \;\; 0 \;\; q_2$
6. $q_2 \;\; 1 \;\; q_2$

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

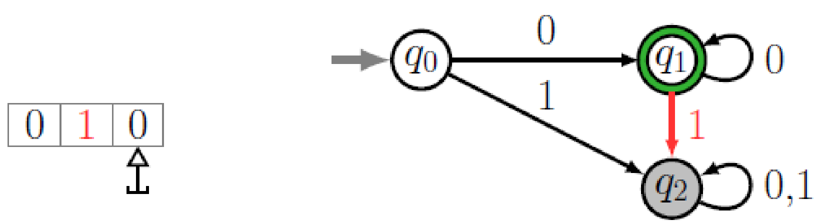


$$q_0|\triangleright 010$$

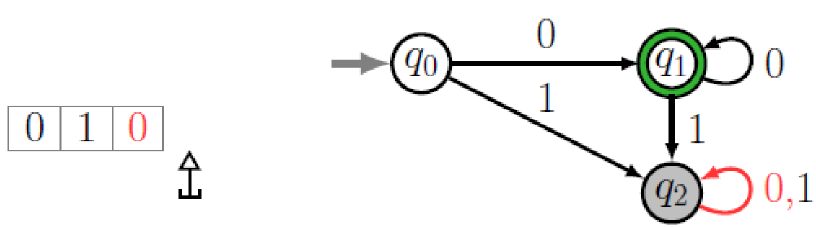


$$q_0|\triangleright 010 \overset{M}{\mapsto} q_1|0\triangleright 10$$

(M is the name of our "Machine")



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



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

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

Computing Problem Solved by a DFA

- The computing problem solved by M is the language
$$\mathcal{L}(M) = \{w | M(w) = YES\}$$

- $\mathcal{L}(M)$ is the automaton's YES-set. For our automaton M:
$$\mathcal{L}(M) = \{0,00,000,0000, \ldots\} = \{0^{\bullet n} | n > 0\}$$

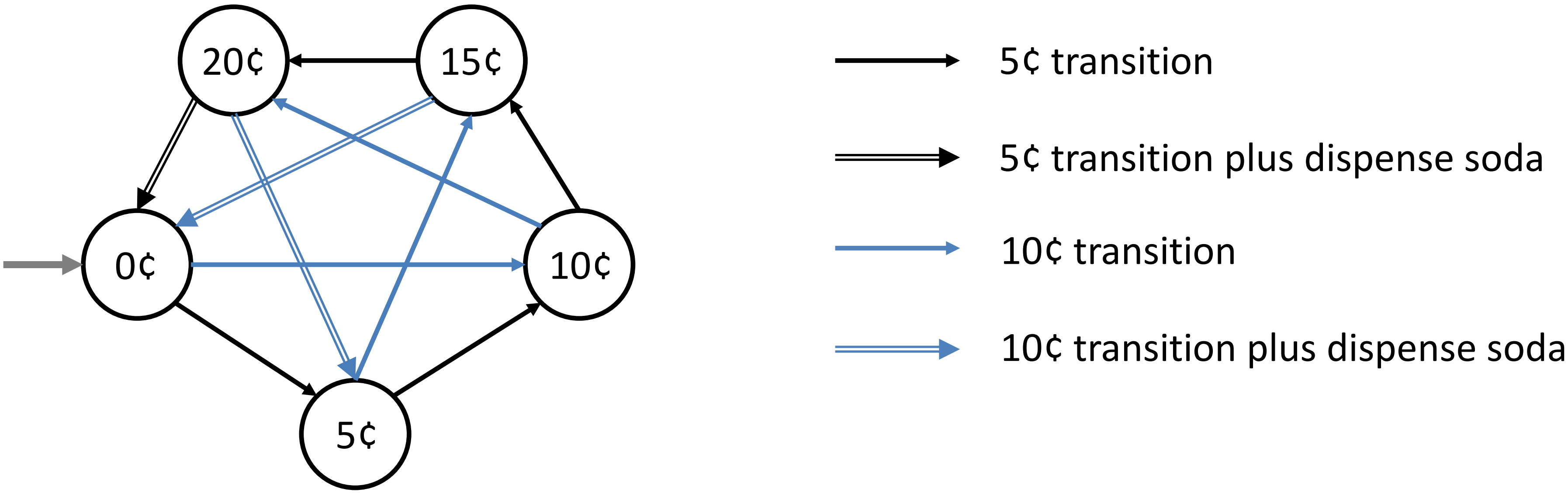
1. For an automaton M, what is the computing problem $\mathcal{L}(M)$ solved by M?

2. For a computing problem $\mathcal{L}$, what automaton M solves $\mathcal{L}$, i.e., $\mathcal{L}(M) = \mathcal{L}$?

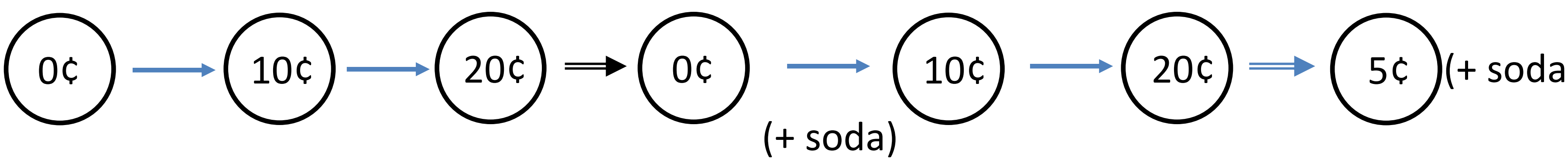
- **Practice.** Exercise 24.2 gives you lots of training in question 1.

The Vending Machine

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

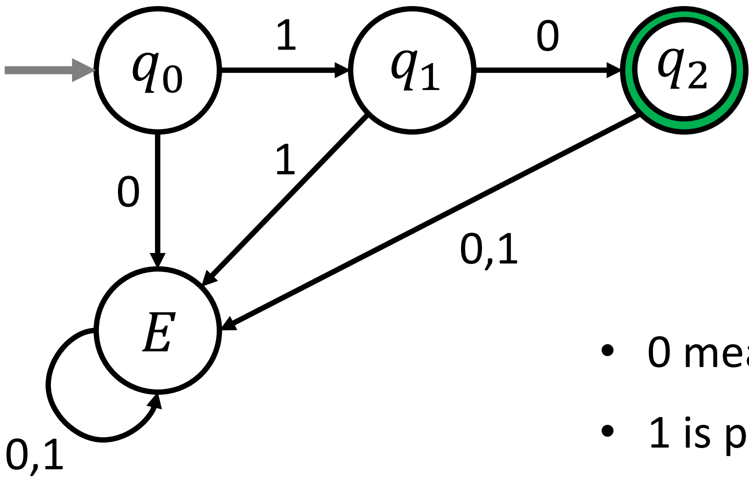


- Input sequence: 10¢, 10¢, 5¢, 10¢, 10¢, 10¢.



DFA for a Finite Language

What's the DFA for $\mathcal{L} = \{10\}$



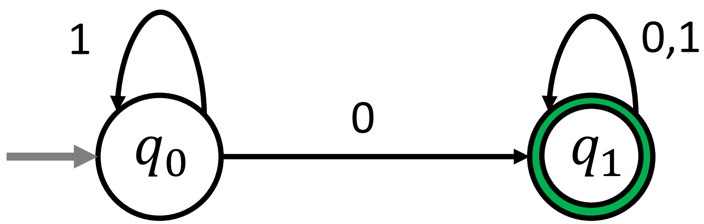
- 0 means move to a rejecting ERROR state and stay there
- 1 is partial success.
- Another 1 puts you into ERROR since you want 0;
- 0 from q_1 and you are ready to accept . . . unless . . .
- More bits arrive, in which case move to ERROR
- **Practice.** Try random strings other than 01 and make sure our DFA rejects them.

DFAs for Infinite Languages

- What is this language:

$$\begin{aligned}\mathcal{L}_1 &= * \ 0 \ * \\ &= \{strings\ with\ a\ 0\} \\ &= \{0,00,01,10,000,001,010,011,110, \dots\}\end{aligned}$$

- (wilrdcard $* = \Sigma^*$)



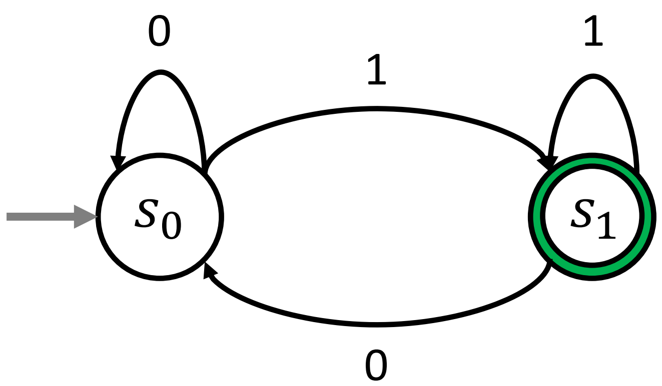
M_1

DFAs for Infinite Languages, cont'

- What is this language:

$$\begin{aligned}\mathcal{L}_1 &= * \, 1 \\ &= \{strings\ ending\ in\ 1\} \\ &= \{1,01,11,001,011,101,111,\ldots\}\end{aligned}$$

- (wilrdcard $* = \Sigma^*$)

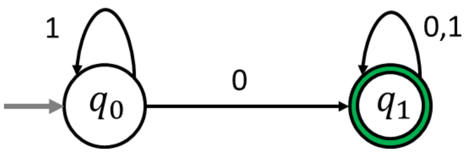


M_2

DFAs for Infinite Languages

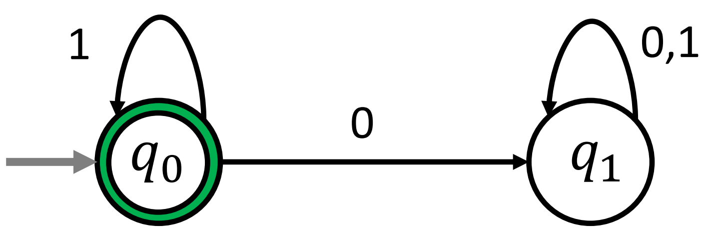
- What is this language:

$$\mathcal{L}_1 = *\,0\,* \\ = \{strings\ with\ a\ 0\} \\ = \{0,00,01,10,000,001,010,011,110, \ldots\}$$



M_1

- **Complement**. Consider $\bar{\mathcal{L}}_1$
  - Must accept strings M_1 rejects
  - Flip YES and NO states

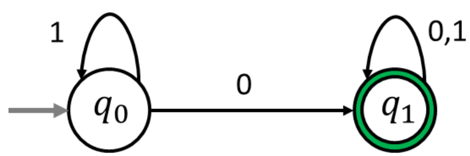


$\overline{M_1}$

Two DFAs in One: Union and Intersection

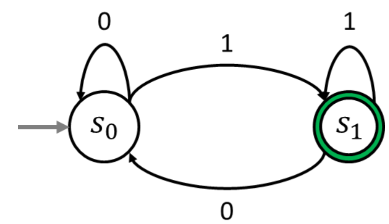
- Let's look at our two small languages:

$\mathcal{L}_1$



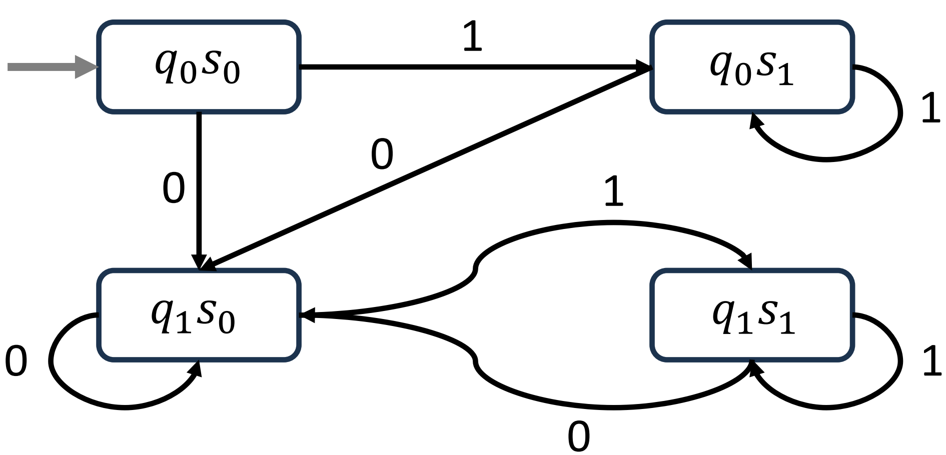
M_1

$\mathcal{L}_2$



M_2

- What's the DFA for $\mathcal{L}_1 \cup \mathcal{L}_2$?
  - The Joint-DFA has product states $\{q_0s_0, q_0s_1, q_1s_0, q_1s_1\}$

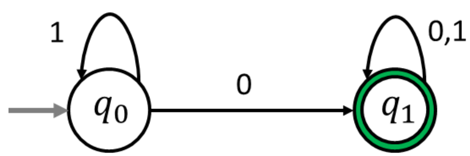


q_0s_0: M_1 is in state q_0 and M_2 is in state s_0
q_0s_1: M_1 is in state q_0 and M_2 is in state s_1
q_1s_0: M_1 is in state q_1 and M_2 is in state s_0
q_1s_1: M_1 is in state q_1 and M_2 is in state s_1

Two DFAs in One: Union and Intersection

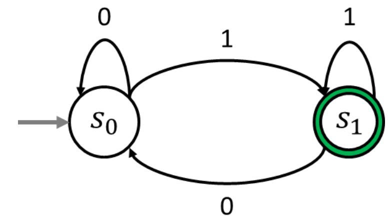
- Let's look at our two small languages:

$\mathcal{L}_1$



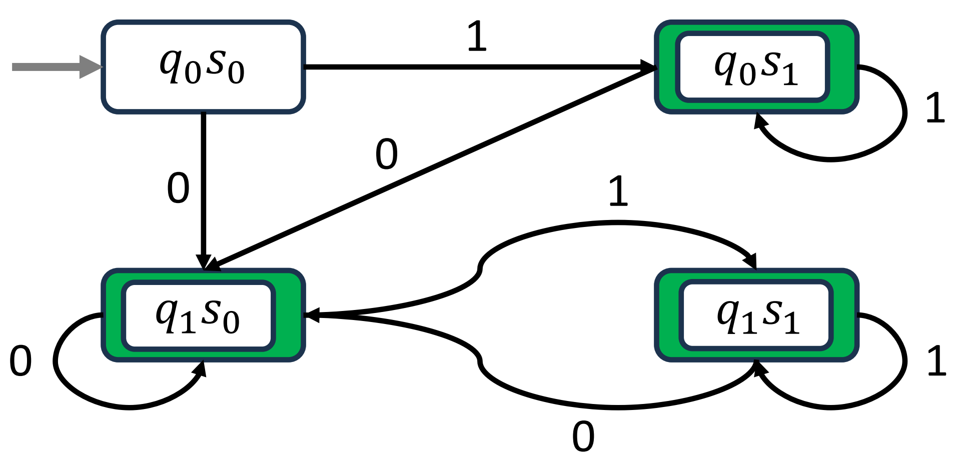
M_1

$\mathcal{L}_2$



M_2

- What's the DFA for $\mathcal{L}_1 \cup \mathcal{L}_2$?
  - The Joint-DFA has product states $\{q_0 s_0, q_0 s_1, q_1 s_0, q_1 s_1\}$



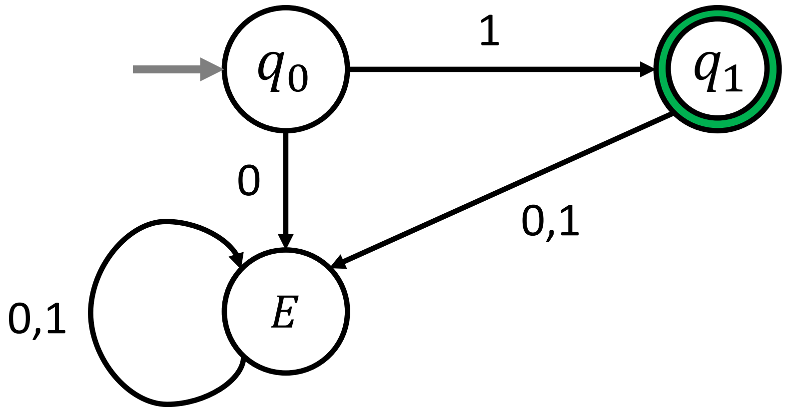
$q_0 s_0$: M_1 is in state q_0 and M_2 is in state s_0
$q_0 s_1$: M_1 is in state q_0 and M_2 is in state s_1
$q_1 s_0$: M_1 is in state q_1 and M_2 is in state s_0
$q_1 s_1$: M_1 is in state q_1 and M_2 is in state s_1

$M_1 \cup M_2$

Concatenation and Kleene Star

- What is the DFA for $\mathcal{L}_1 = \{1\}$

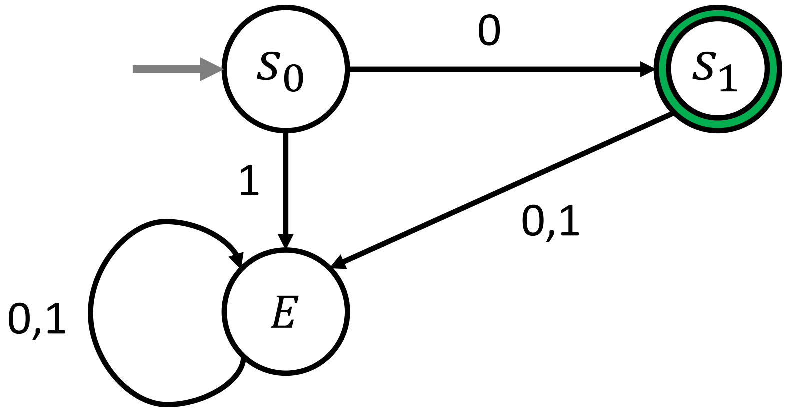
M_1



Concatenation and Kleene Star, cont'd

- What is the DFA for $\mathcal{L}_2 = \{0\}$

M_2

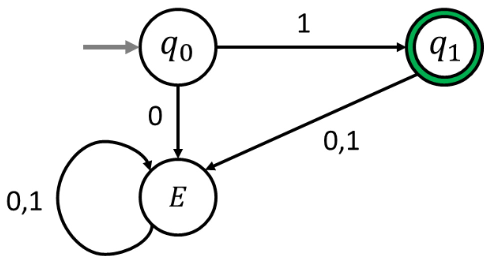


Concatenation and Kleene Star, cont'd

- Let's look at our two small languages:

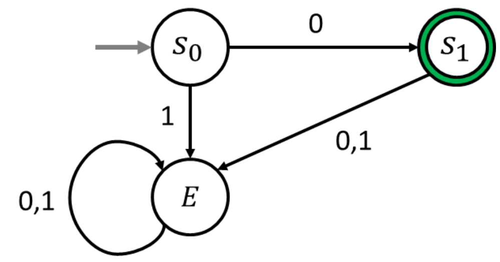
$\mathcal{L}_1$

M_1

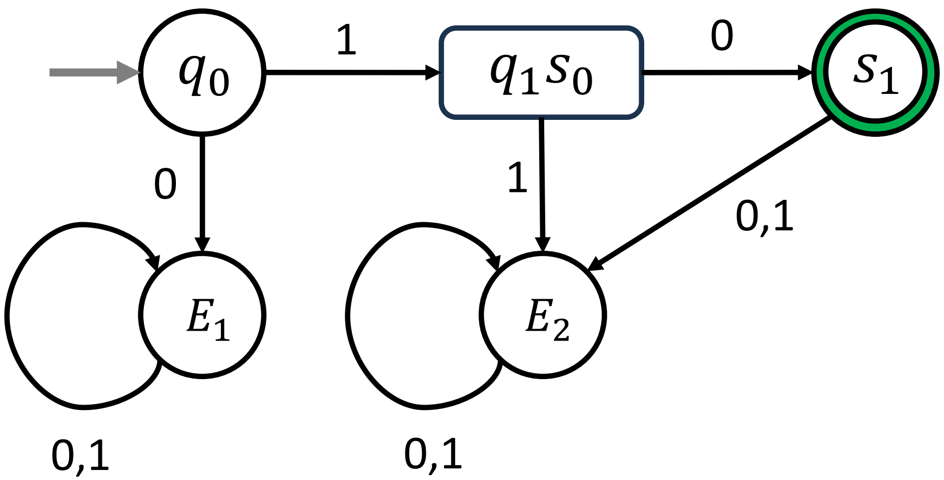


$\mathcal{L}_2$

M_2



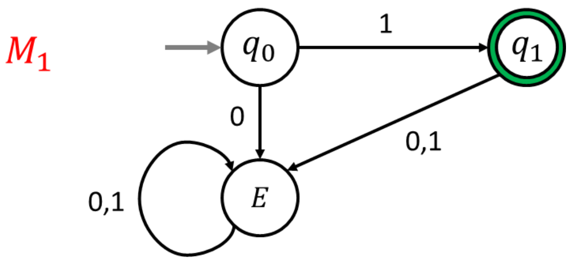
- What is the DFA for $\mathcal{L}_1 \bullet \mathcal{L}_2$?



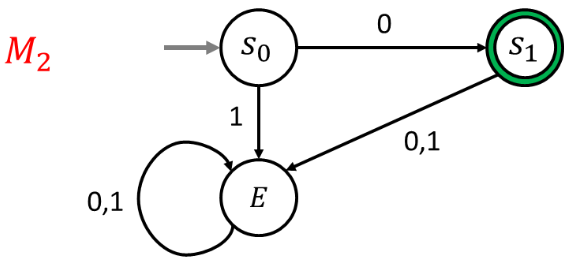
Concatenation and Kleene Star, cont'd

- Let's look at our two small languages:

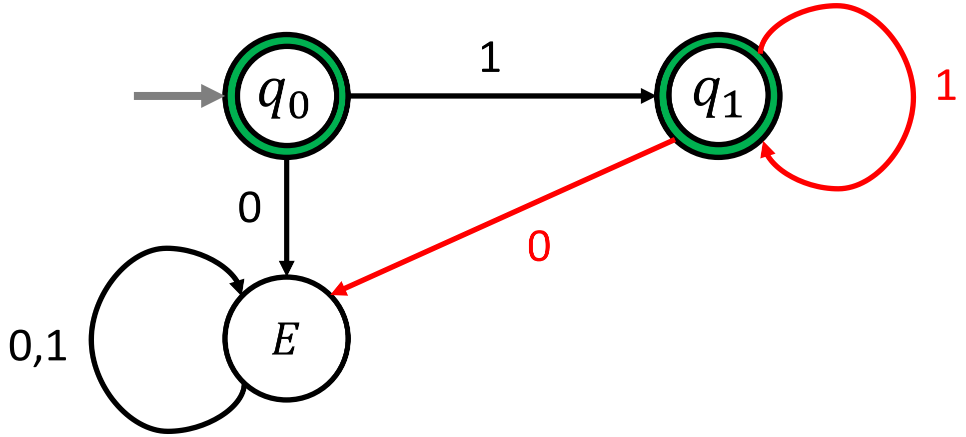
$\mathcal{L}_1$



$\mathcal{L}_2$



- What is the DFA for $\mathcal{L}_1^*$?



The Power of DFAs: What can they Solve?

Rensselaer

- Finite languages.
  - (building blocks of regular expressions)
- Complement, intersection, union.
  - (operations to form complex regular expressions)
- Concatenation and Kleene-star (a 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?

Rensselaer

- **Exercise**. Give a DFA to solve $\{0\}^* \bullet \{1\}^* = \{0^{\bullet n} 1^{\bullet k} | n \geq 0, k \geq 0\}$
- What about "equality":

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

- *Theorem*. There is no DFA that solves $\mathcal{L}_{0^n 1^n}$.
- *Proof*.
  - By Contradiction. Suppose a DFA with k states solves $\{0^n 1^n\}$
  - What happens to this DFA when you keep feeding it 0's?

$$q_0 = state\left(0^{\bullet 0}\right) \xrightarrow{M} state\left(0^{\bullet 1}\right) \xrightarrow{M} \cdots \xrightarrow{M} state(0^{\bullet k-1}) \xrightarrow{M} state(0^{\bullet k})$$

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

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

  - 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 0's in both strings, it is in state q
    - What then?
    - 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.

<u>Computing Model</u>
Rules to
1. Construct machine
2. Solve problems

→

<u>Analyze Model</u>
1. Capabilities: what can be solved
2. Limitations: what can't be solved?

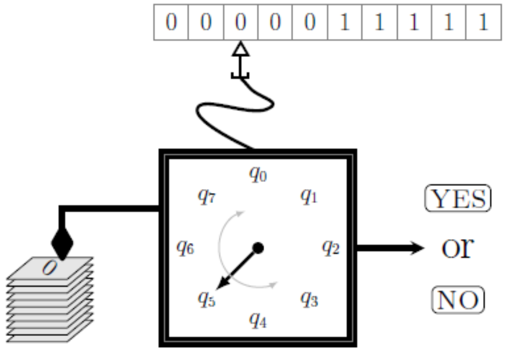
→

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.
- **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 $\mathcal{L}_{0^n1^n} = \{0^{\bullet n}1^{\bullet n} | n \geq 0\}$?
  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.