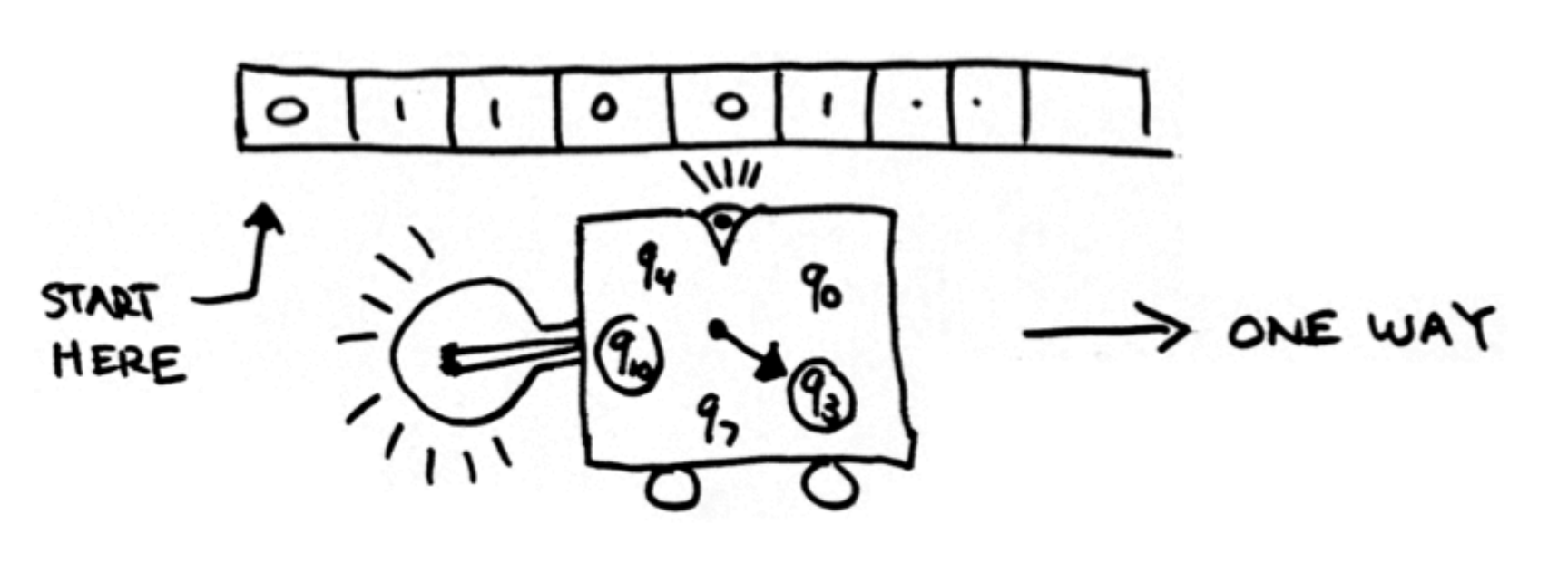
Deterministic Finite Automata

DFAs *(also called FSMs)*

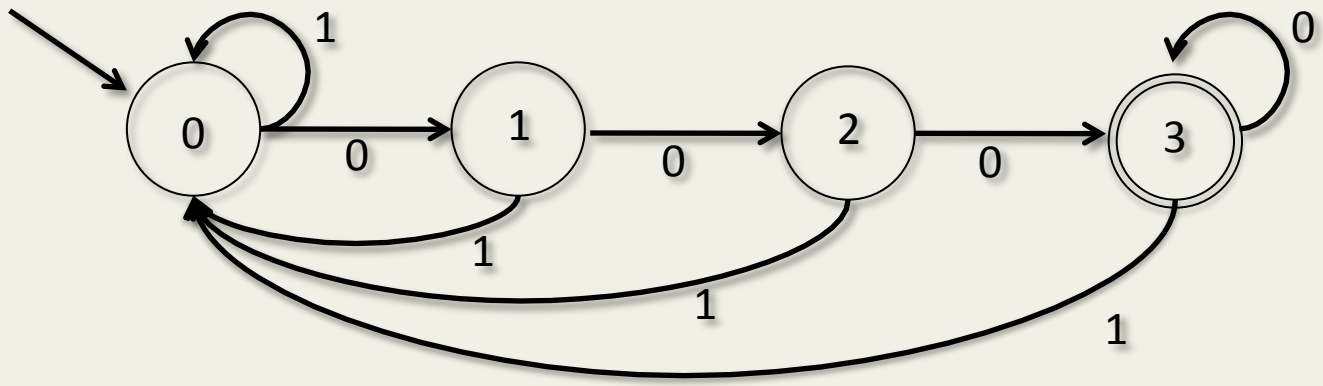
- A simple(st?) model of what a computer is
- Many devices modeled, programmed as DFAs
  - Vending machines
  - Elevators
  - Digital watch logic
  - Calculators
  - Lexical analysis part of program compilation
- Very limited, but observable universe is finite...

Typical DFA



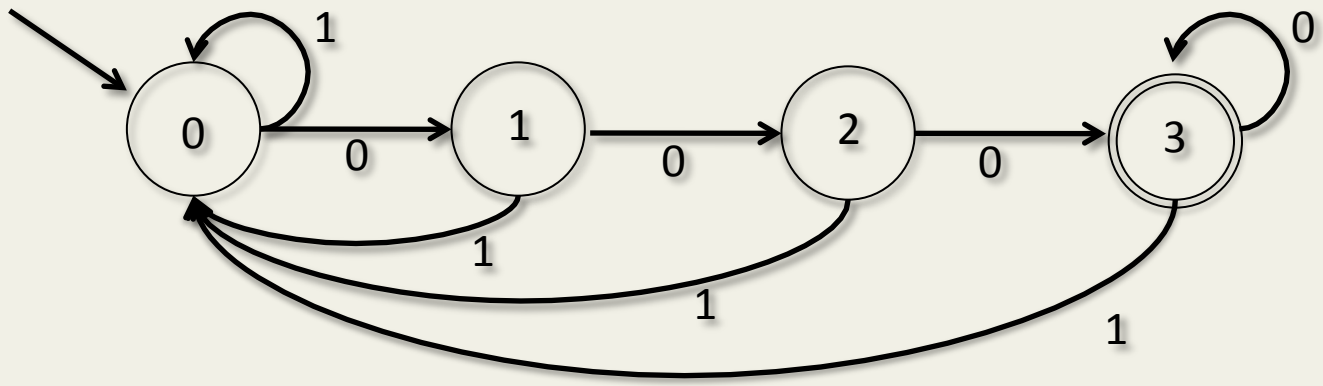
- Start state q_0
- Start at left, scan symbol, change state, move right.
- Rules of form "if in state *q* scanning symbol *s* then go to state *p* and move right."
- Some states (circled) are *accepting.*
- *M accepts* the input string if a circled state is reached after scanning the last symbol.

Graphical Representation



- Directed graph with edges labeled with chars in Σ
- For each state (vertex) *q* and symbol *a* in Σ there is *exactly* one edge leaving *q* labeled with *a*. $q \xrightarrow{a} p$
- Accepting state(s) are double-circled
- Initial state has pointer, or is obviously labeled (0, q_0, *"start"*...)

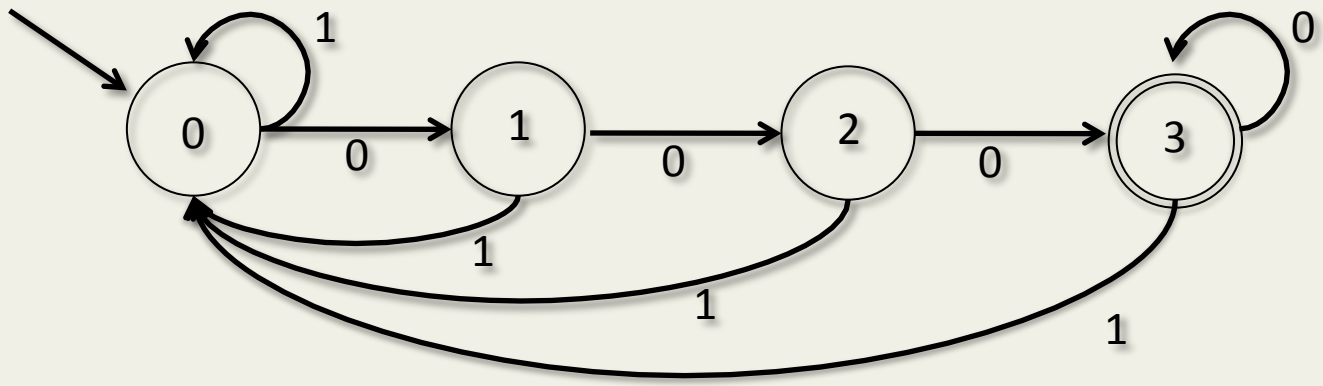
Graphical Representation



- Where does 001 lead? 10010?
- Which strings end up in accepting state?
- Prove it
- *Every string* has *one path* that it follows

$q \xrightarrow{a} p$ versus $q \overset{w}{\rightsquigarrow} p$

Graphical Representation



Definition

- A DFA *M* *accepts a string* *w* iff the unique path starting at the initial state and spelling out *w* ends at an accepting state.

- The *language accepted* *(or "recognized")* by a DFA *M* is denoted *L(M)* and defined by

$$L(M) = \{ w \mid M \text{ accepts } w \}$$

Warning

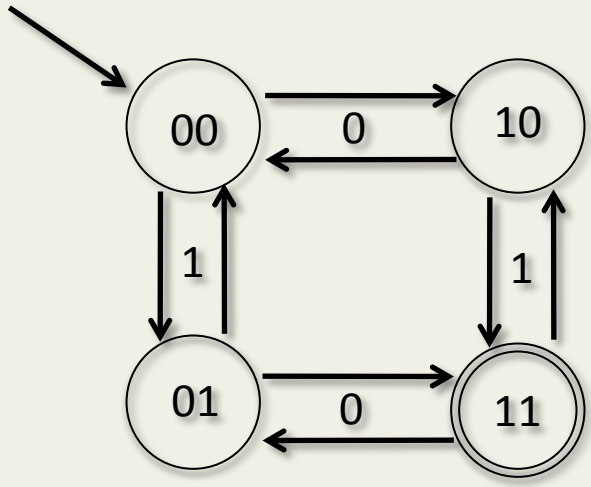
- "*M* accepts language *L*" does not mean simply that *M* accepts each string in *L*.

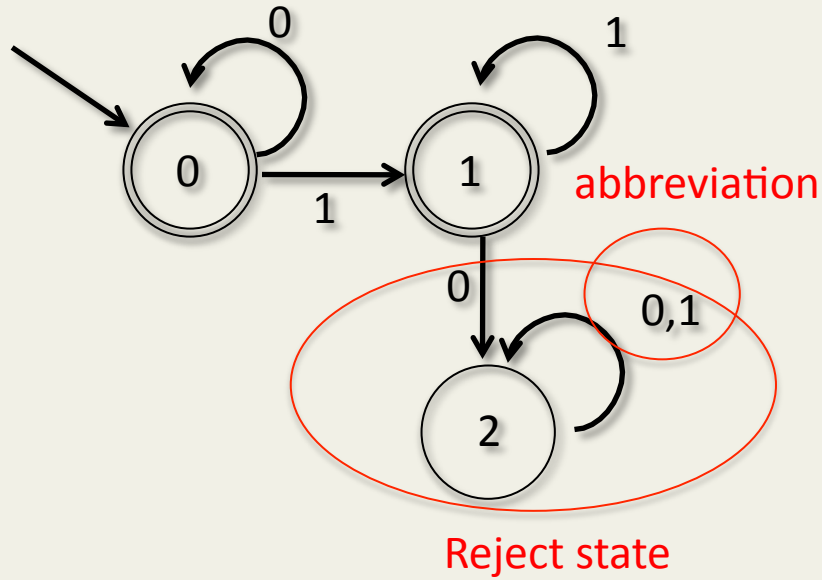
- "*M* accepts language *L*" means

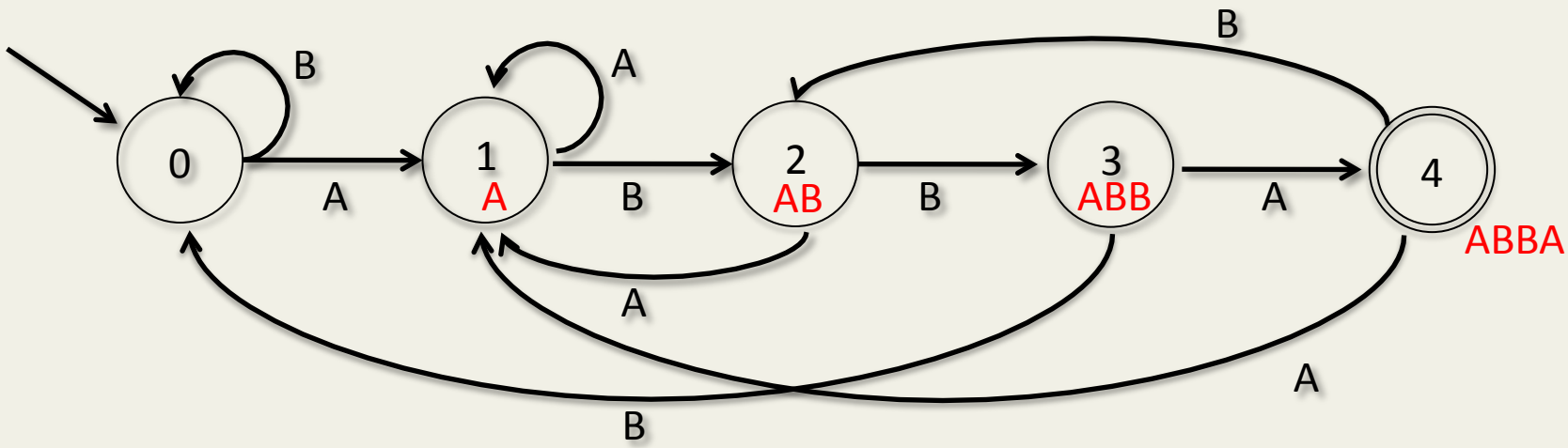
  M accepts each string in *L* *and no others!*

- *M* "recognizes" *L* is a better term, but "accepts" is widely accepted (and recognized).

Examples: What is L(M) ?





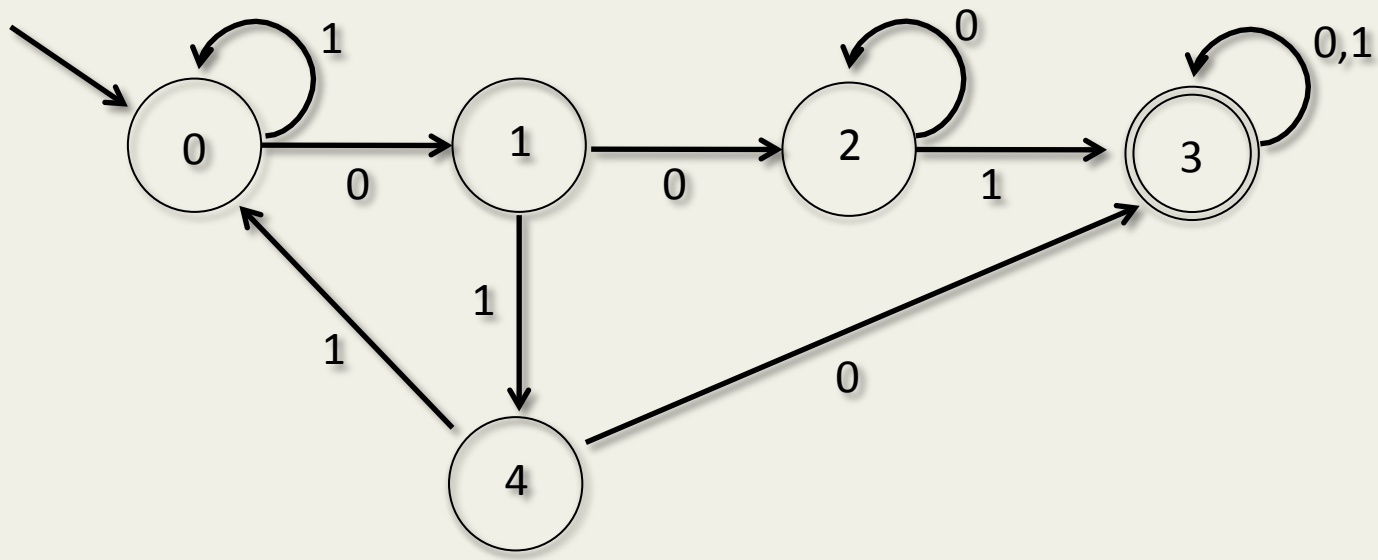


State = Memory

- The state of a DFA is its entire memory of what has come before
- The state must capture enough information to complete the computation on the suffix to come
- When designing a DFA, think "what do I need to know at this moment?" *That* is your state.

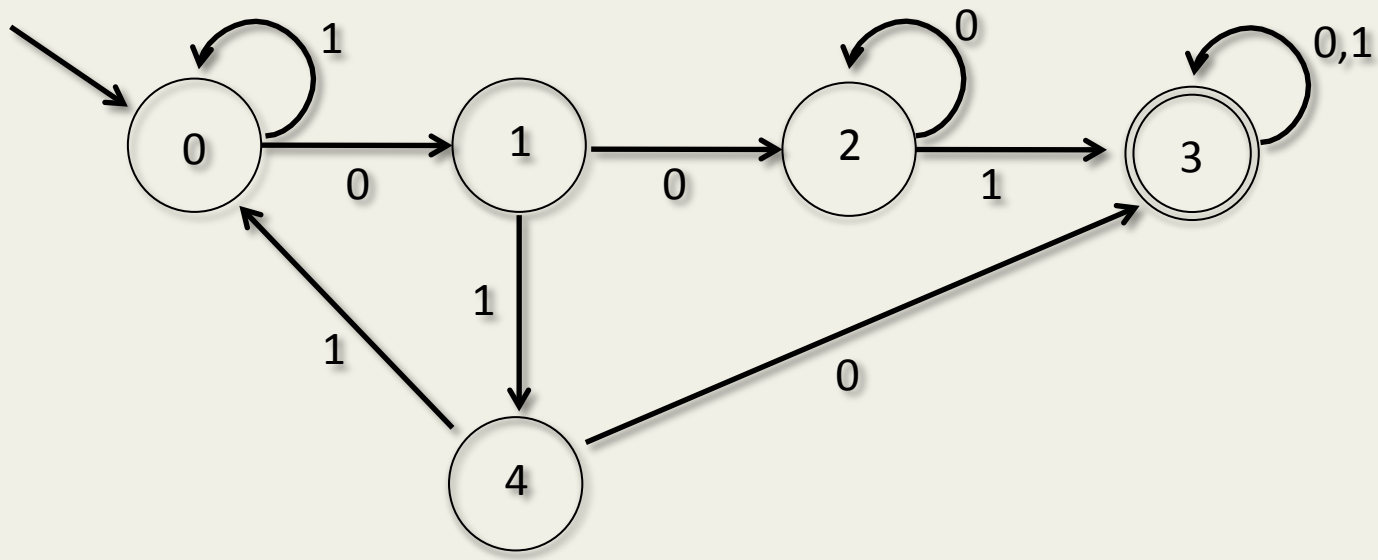
Construction Challenge

- $L(M) = \{w \mid w \text{ contains } 001 \text{ or } 010\}$



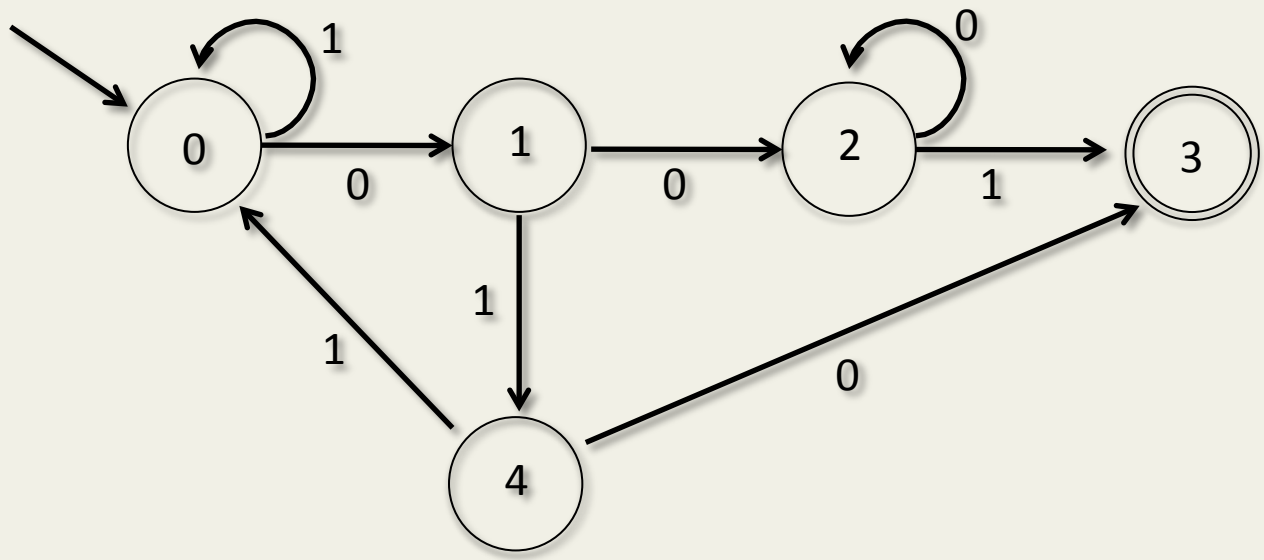
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



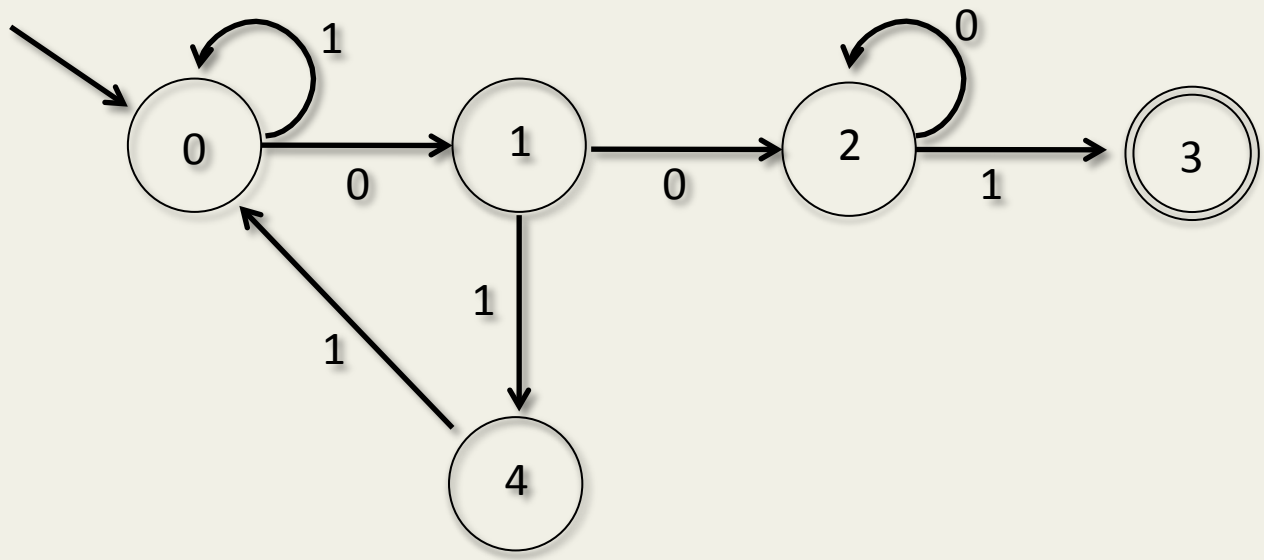
Construction Challenge

- *L(M)* = {*w* | *w* ~~contains~~ ends with 001 or 010}



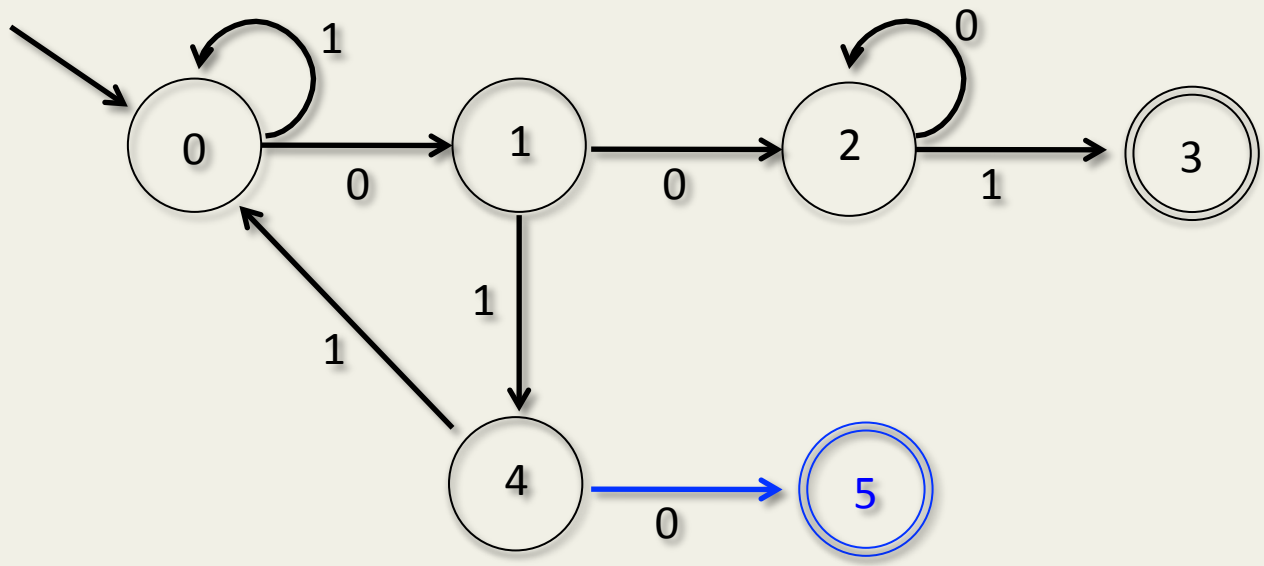
Construction Challenge

- *L(M)* = {*w* | *w* ~~contains~~ ends with 001 or 010}



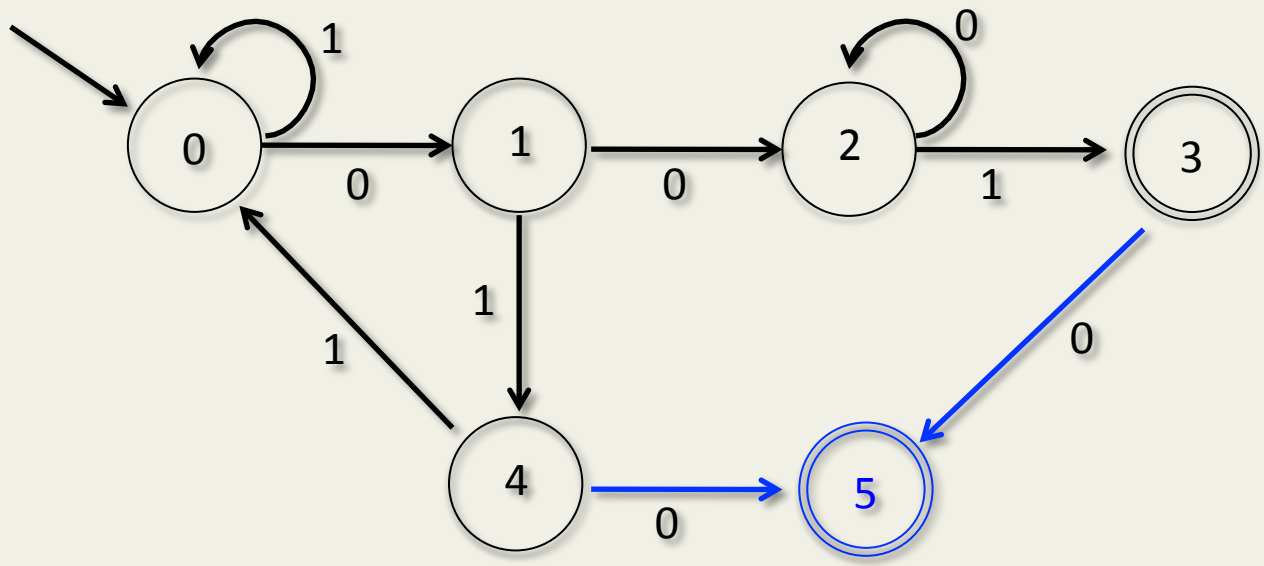
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



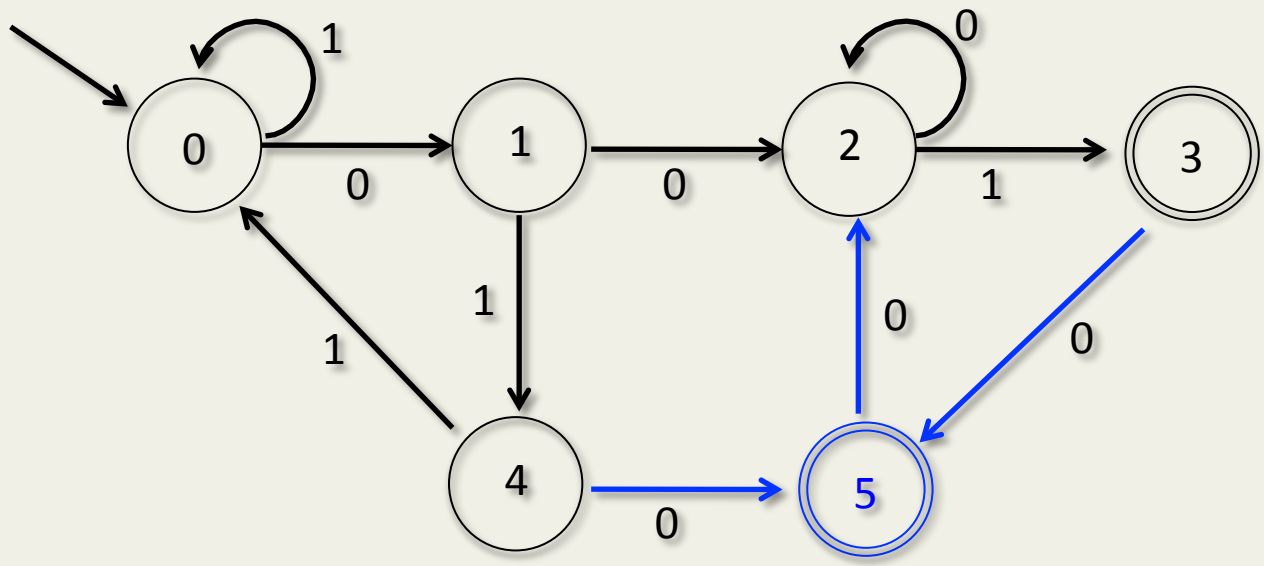
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



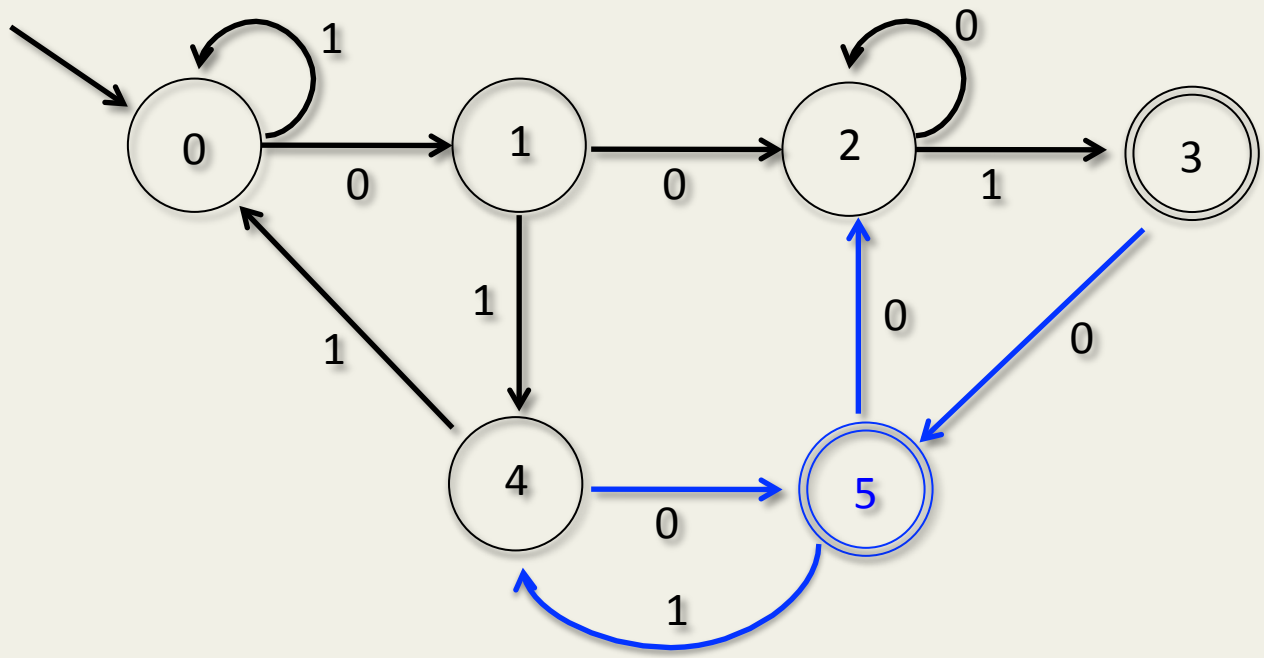
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



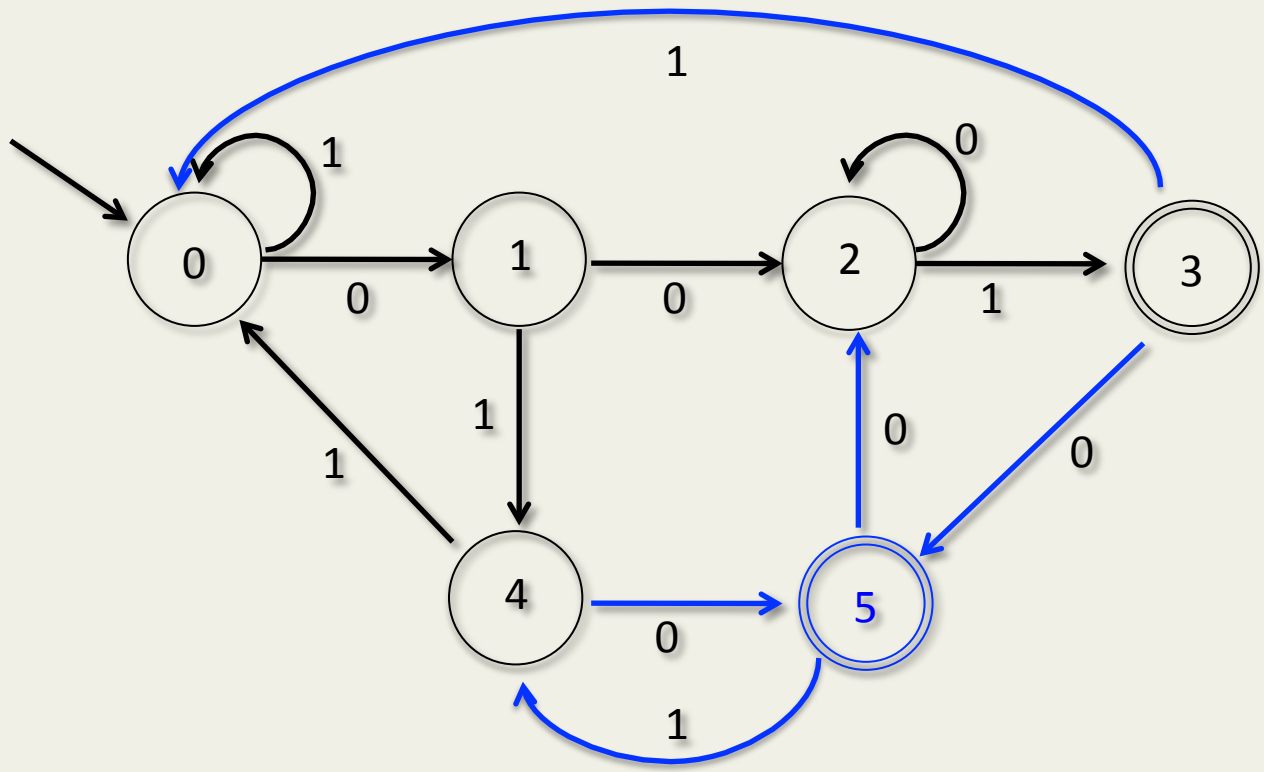
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



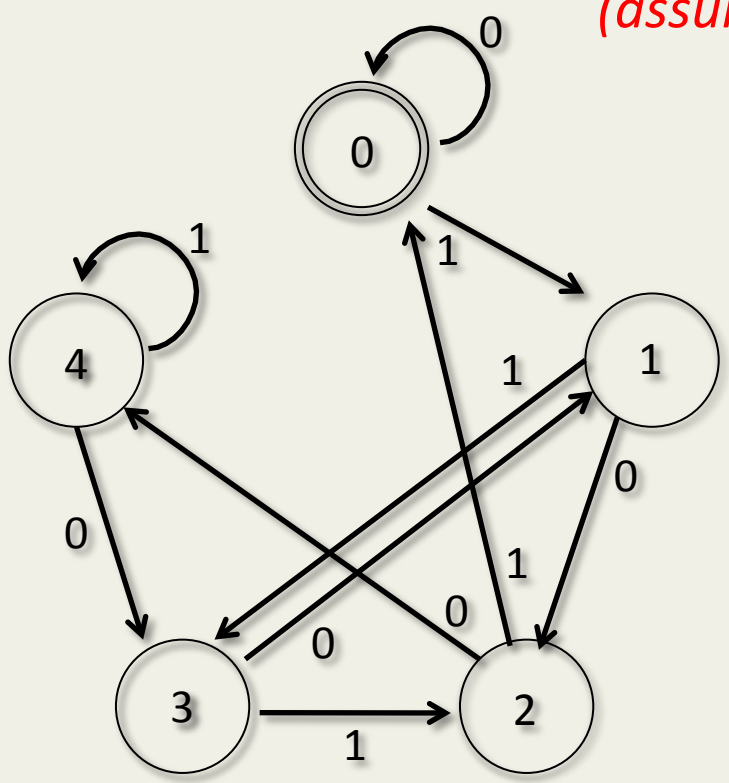
Construction Challenge

- $L(M) = \{w \mid w$ ~~contains~~ ends with 001 or 010$\}$



Binary #s congruent to 0 mod 5

(assume no leading 0s)



Key Idea

If *w mod 5 = a, then:*

- *w0 mod 5 = 2a mod 5*
- *w1 mod 5 = 2a+1 mod 5*

Test: 1101011 = 107 = 2 mod 5

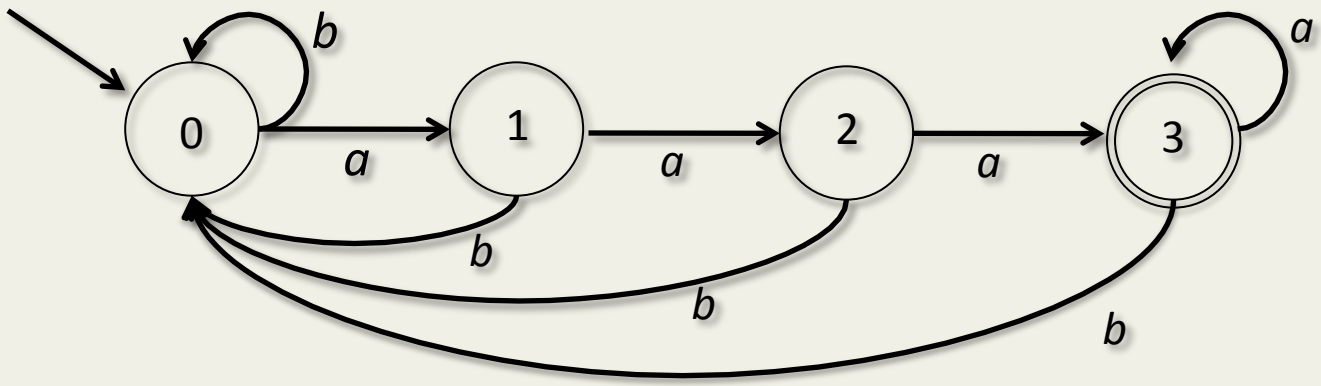
Formal (tuple) Representation

Sometimes, it is easier to specify the DFA using this formalism, instead of drawing a graph

A DFA is a quintuple $M=(Q,\Sigma,\delta,q_0,F)$, *where:*

- Q is a finite set of *states*
- Σ is a finite *alphabet* of symbols
- $\delta: Q \times \Sigma \rightarrow Q$ is a *transition function*
- q_0 is the *initial state*
- $F \subseteq Q$ is the set of *accepting states*

Example



- $Q = \{0,1,2,3\}$
- $\Sigma = \{a,b\}$
- δ specified at right
- $q_0 = 0$
- $F = \{3\}$

state \ input	a	b
0	1	0
1	2	0
2	3	0
3	3	0

Extending δ

- $\delta(q,a) = p$ means in graph that $q \xrightarrow{a} p$
- But how can we define $\delta(q,w)$ to express $q \overset{w}{\rightsquigarrow} p$
- Must extend $\delta: Q \times \Sigma^* \to Q$
  - $\delta(q,\varepsilon) = q$ for every q; $\delta(q,a)$ already defined
  - $\delta(q,au) = \delta(\underbrace{\delta(q,a)}_{\text{take first step according to } \delta},u)$ for $|u| \geq 1$, all q, a

take rest of steps inductively according to δ

$\delta(q,w) = p$ corresponds to $q \overset{w}{\rightsquigarrow} p$

Formal definition of L(M)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

Then $L(M) = \{w \mid \delta(q_0, w) \in F\}$

We will show later that:

Theorem

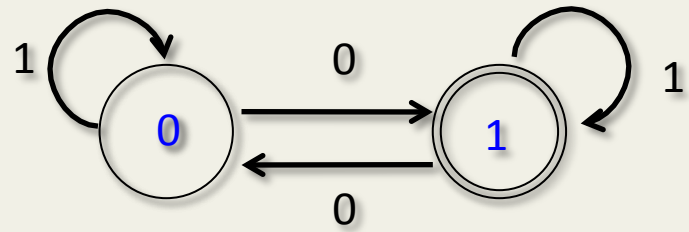
L is regular if and only if L = L(M) for some DFA M

Example use

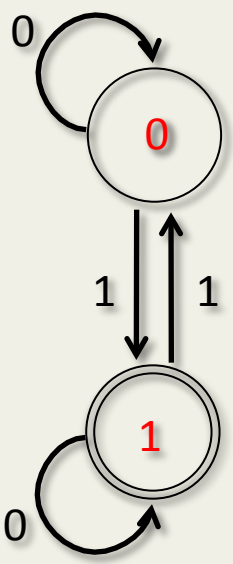
$L(M) = \{w \mid w$ in base b is congruent to k mod m

- $Q = \{0,1,\ldots,m\text{-}1\}$
- $\Sigma = \{0,1,\ldots,b\text{-}1\}$
- $q_0 = 0$
- $\delta\,(n,a) = bn+a \bmod m$
- $F = \{k\}$

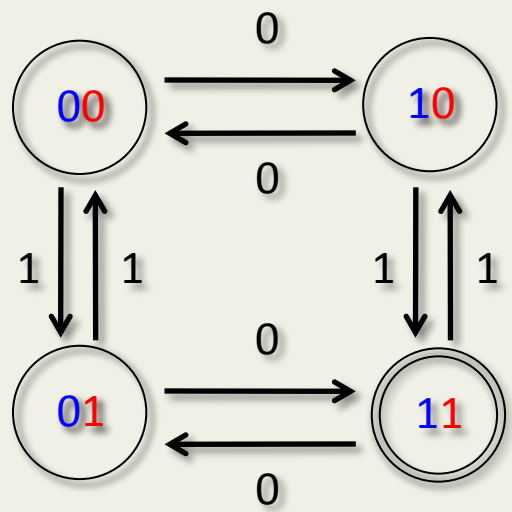
M simulating both M_1 and M_2



M_1 accepts #0 = odd



M_2 accepts #1 = odd



Cross-product machine

M accepting $L(M_1) \cap L(M_2)$

$Q = Q_1 \times Q_2$

$q_0 = (q_0^{(1)}, q_0^{(2)})$

$F = F_1 \times F_2 = \{ (q_1, q_2) \mid q_1 \text{ in } F_1 \text{ and } q_2 \text{ in } F_2 \}$

Transition function:

$\delta((q_1, q_2), a) = (\delta_1(q_1, a) , \delta_2(q_2, a))$

$(q_1, q_2) \xrightarrow{a} (p_1, p_2)$ if and only if

- $q_1 \xrightarrow{a}_1 p_1$ in M_1
- $q_2 \xrightarrow{a}_2 p_2$ in M_2

Proof that simulation is correct

- Induction *on what?* that *what?*
- Will need to prove that action of machine is correct starting from any states.
- We know that:

$(q_1, q_2) \xrightarrow{a} (p_1, p_2)$ iff

- $q_1 \xrightarrow{a}_1 p_1$ *in* M_1
- $q_2 \xrightarrow{a}_2 p_2$ *in* M_2

By definition

Show that:

$(q_1, q_2) \overset{w}{\rightsquigarrow} (p_1, p_2)$ iff

- $q_1 \overset{w}{\rightsquigarrow}_1 p_1$ *in* M_1
- $q_2 \overset{w}{\rightsquigarrow}_2 p_2$ *in* M_2

Just like definition of δ,
but with *w* instead of *a*

DEF for M:

$(q_1, q_2) \xrightarrow{a} (p_1, p_2)$

means

$q_1 \xrightarrow{a}_1 p_1$ and $q_2 \xrightarrow{a}_2 p_2$

BY INDUCTION on $|w|$

$(q_1, q_2) \overset{w}{\rightsquigarrow} (p_1, p_2)$

iff

$q_1 \overset{w}{\rightsquigarrow}_1 p_1$ and $q_2 \overset{w}{\rightsquigarrow}_2 p_2$

$(q_1, q_2) \overset{w}{\rightsquigarrow} (p_1, p_2)$

$a \mid u$

(r_1, r_2)

pull apart the computation

$(q_1, q_2) \xrightarrow{a} (r_1, r_2)$

$(r_1, r_2) \overset{u}{\rightsquigarrow} (p_1, p_2)$

apply definition

apply inductive hypothesis since |u| < |w|

$q_1 \xrightarrow{a}_1 r_1$ and $q_2 \xrightarrow{a}_2 r_2$

$r_1 \overset{u}{\rightsquigarrow}_1 p_1$ and $r_2 \overset{u}{\rightsquigarrow}_2 p_2$

$q_1 \overset{w}{\rightsquigarrow}_1 p_1$

paste the computations back together

$q_2 \overset{w}{\rightsquigarrow}_2 p_2$

Finishing up...

- We proved:

$$(q_1, q_2) \overset{w}{\rightsquigarrow} (p_1, p_2)$$

iff

$$q_1 \overset{w}{\rightsquigarrow}_1 p_1 \quad \text{and} \quad q_2 \overset{w}{\rightsquigarrow}_2 p_2$$

By definition, *w* accepted by *M*

iff $(q_0^{(1)}, q_0^{(2)}) \overset{w}{\rightsquigarrow} (f_1, f_2)$ in $F_1 \times F_2$

iff $q_0^{(1)} \overset{w}{\rightsquigarrow} f_1$ *in* F_1 *AND* $q_0^{(2)} \overset{w}{\rightsquigarrow} f_2$ *in* F_2

iff *w in* $L(M_1)$ *AND* *w in* $L(M_2)$

Formal proof that simulation is correct

- We know *by* definition that:
  - for all q_1 *in* Q_1, for all q_2 *in* Q_2
  - for all characters a

  $\delta((q_1, q_2), a) = (\delta_1(q_1,a),\delta_2(q_2,a))$

- We prove by induction on $|w|$ that:
  - for all q_1 *in* Q_1, for all q_2 *in* Q_2
  - for all strings w

  $\delta((q_1, q_2), w) = (\delta_1(q_1,w),\delta_2(q_2,w))$

Looks just like definition of δ, but with *w* instead of *a*

To prove: $\delta((q_1,q_2), w) = (\delta_1(q_1,w),\delta_2(q_2,w))$

Induction on $|w|$

- Base Case: $|w| = 0$, so $w = \varepsilon$.

  $\delta((q_1, q_2), \varepsilon) = (q_1, q_2) = (\delta_1(q_1,\varepsilon),\delta_2(q_2,\varepsilon))$
- Assume true for strings u of length $< n$.
- Let $w = au$ be an arbitrary string of length n.
- $\delta((q_1, q_2), au)$

$= \delta(\delta((q_1, q_2),a),u)$	defn of δ extension
$= \delta((\delta_1(q_1,a),\delta_2(q_2,a)), u)$	by defn of δ
$= \delta((r_1,r_2), u)$	define r's to simplify
$= (\delta_1(r_1,u), \delta_2(r_2,u))$	by induction ($\|u\| < n$)
$= (\delta_1(\delta_1(q_1,a),u), \delta_2(\delta_2(q_2,a),u)))$	get rid of r's
$= (\delta_1(q_1,au), \delta_2(q_2,au))$	unsplitting
$= (\delta_1(q_1,w), \delta_2(q_2,w))$	

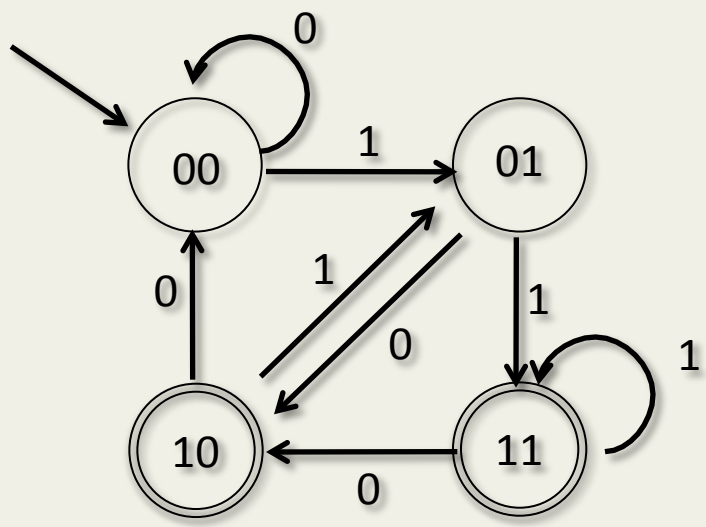
Properties of Regular languages

- We've shown how to accept intersection of two regular languages
- What about union?
- If L is accepted by a DFA, what about $\overline{L}$ *?*
- What about concatenation, and Kleene * ?
- Is there a DFA for $L_1 - L_2$ given M_1 and M_2 ?

The answer to all of these questions, and more, is "Yes."

DFA for L_n = n^{th} *char from end is 1*

- Example: $L_2 = \{w \mid$ next to last char is "1"$\}$
- What needs to be remembered?
- Previous character?



State name = last two characters seen

DFA for L_n = n^{th} *char from end is 1*

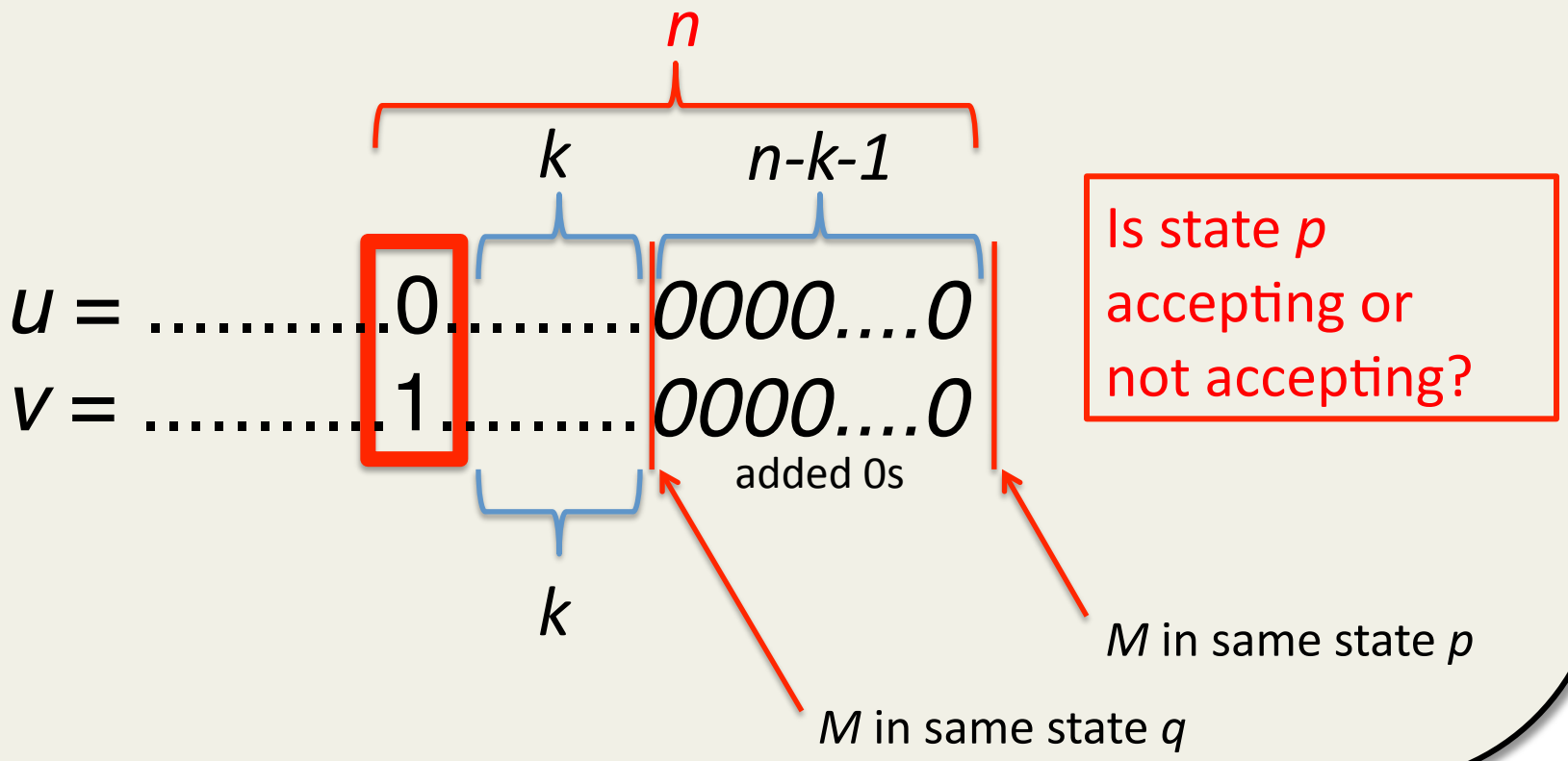
- Q = set of n-bit strings
- $\Sigma = \{0,1\}$
- $q_0 = 000....0$ (why?)
- $\delta\,(b_1 b_2 b_3 ... b_n\,,\ 0) = b_2 b_3 ... b_n\, 0$

  $\delta\,(b_1 b_2 b_3 ... b_n\,,\ 1) = b_2 b_3 ... b_n\, 1$
- $F = \{1 b_2 b_3 ... b_n \mid b_i \text{ in } \{0,1\}\}$

Any DFA for L_n needs $\geq 2^n$ states

Proof by contradiction

- If not, then two *different* n-bit strings u and v must lead to the same state of M.



Things to Know: 2-way DFA

- Why are DFAs required to only move forward?
- Why not allow DFA to scan back and forth?
- "2-way DFA" is such a model.
  - As long as it isn't allowed to *write*, 2-way DFAs accept only the regular languages, thus are no more powerful than regular DFAs.
  - Proof is a very tricky simulation of 2-way DFA by a (1-way) NFA.