Deterministic Finite Automata

Lecture 3

CS 374 Tips

This course moves pretty fast

CS 374 expects you to already be *completely comfortable* with the basics: formal notation, definition and proofs (including induction)

Be prepared to refresh CS 173 material at home!

There will often be more slides posted online than would actually be covered in class (e.g., worked out examples). Do review them.

Review the notes posted online (before and/or after the lecture)

Complexity of Languages

Central Question: How complex an algorithm is needed to compute (aka decide) a language?

Today: a simple class of algorithms, that are fast and can be implemented using minimal hardware

Deterministic Finite Automata (DFA)

DFAs around us: Vending machines, Elevators, Digital watch logic, Calculators, Lexical analyzers (part of program compilation), …

Today

DFA : what are they?

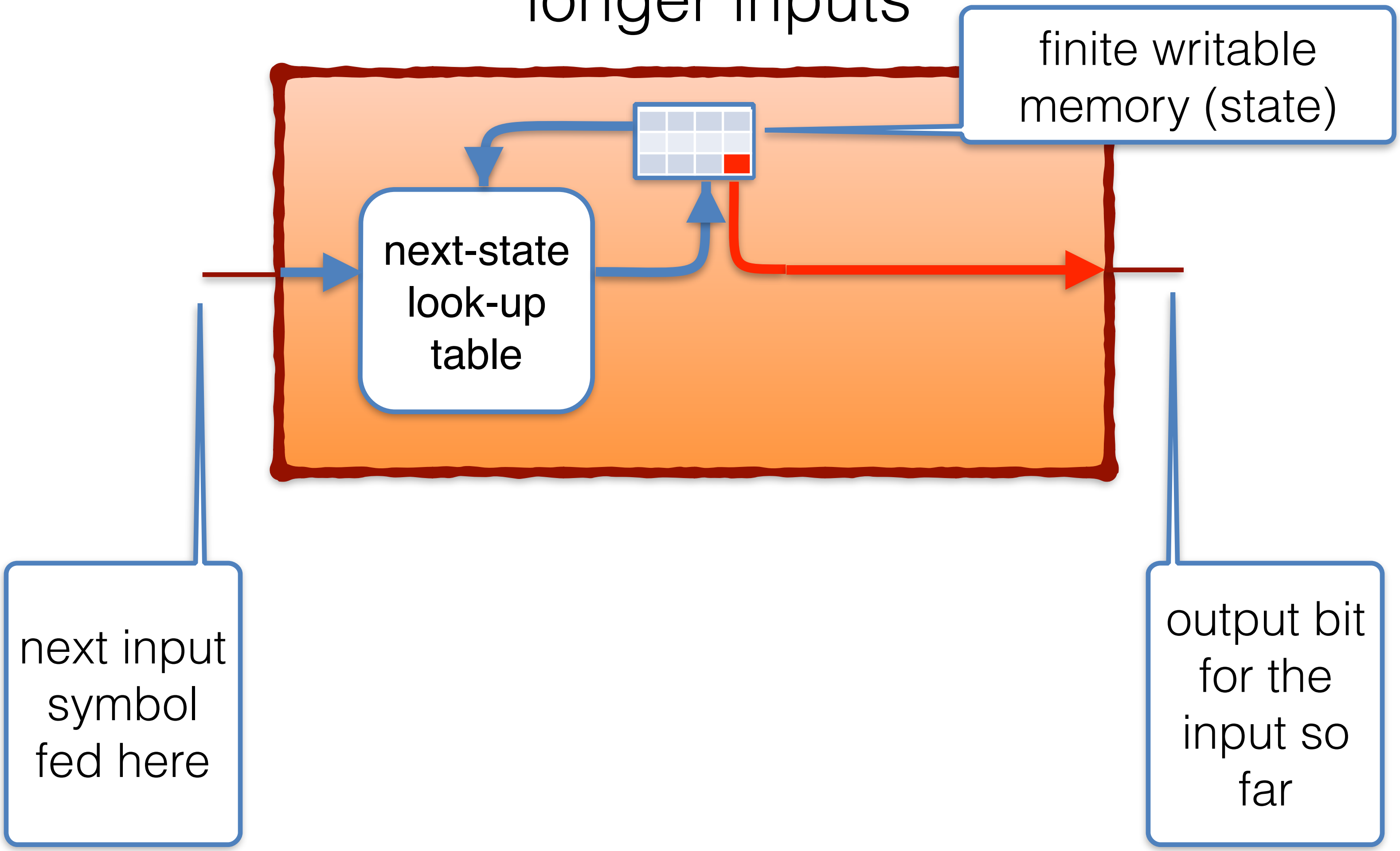
What kind of languages can be decided using them?

How to build DFAs

How to build DFAs from simpler DFAs

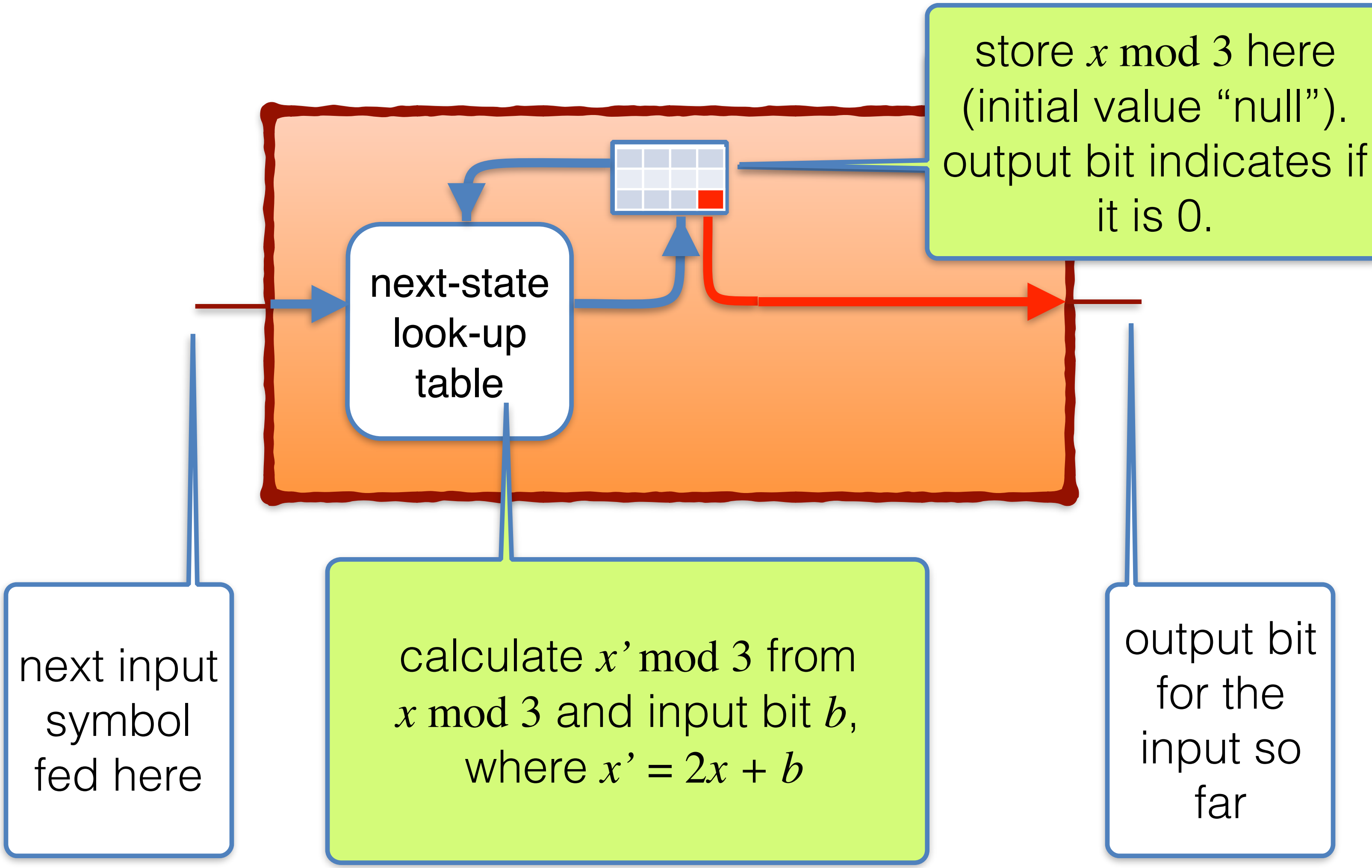
DFA (a.k.a. FSM)

Finite: cannot use more memory to work on longer inputs



DFA (a.k.a. FSM)

Example: check if binary input is a multiple of 3



DFA (a.k.a. FSM)

Example: check if binary input is a multiple of 3

	state (x mod 3)	output
start	null	0
	0	1
	1	0
	2	0

input bit	current state	next state
0	null	0
1	null	1
0	0	0
1	0	1
0	1	2
1	1	0
0	2	1
1	2	2

next inp
symbo
fed here

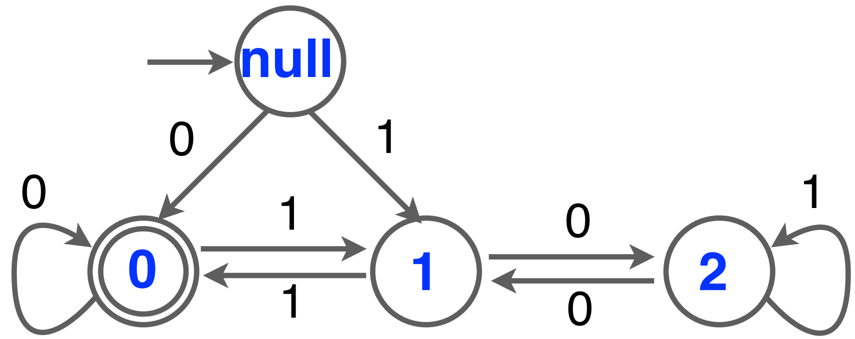
mod 3 from
input bit b,
$= 2x + b$

output bit
for the
input so
far

DFA (a.k.a. FSM)

Example: check if input (MSB first) is a multiple of 3

state (x mod 3)	output
start null	0
0	1
1	0
2	0



input bit	current state	next state
0	null	0
1	null	1
0	0	0
1	0	1
0	1	2
1	1	0
0	2	1
1	2	2

How to fully specify a DFA:

Alphabet: Σ

Set of States: Q

Start state: $s \in Q$

Set of Final states: $F \subseteq Q$

Transition Function: $\delta : Q \times \Sigma \to Q$

Behavior of a DFA on an input

$$M = (\Sigma, Q, \delta, s, F)$$

$$\begin{aligned} \delta^*(q,\varepsilon) &= q \\ \delta^*(q,au) &= \delta^*(\delta(q,a), u) \end{aligned}$$

$\delta^*(q,w)$ be the state M reaches on input $w \in \Sigma^*$, starting from a state $q \in Q$

Formally, $\delta^*(q,\varepsilon) = q$, and $\delta^*(q,au) = \delta^*(\delta(q,a), u)$

We write $q \overset{w}{\rightsquigarrow} p$ to denote $\delta^*(q,w) = p$

We write $q \xrightarrow{a} p$ to denote $\delta(q,a) = p$

Behavior of a DFA on an input

$\delta^*(q,\varepsilon) = q$
$\delta^*(q,au) = \delta^*(\delta(q,a), u)$

$\delta^*($**null**$,01001) =$ **0**

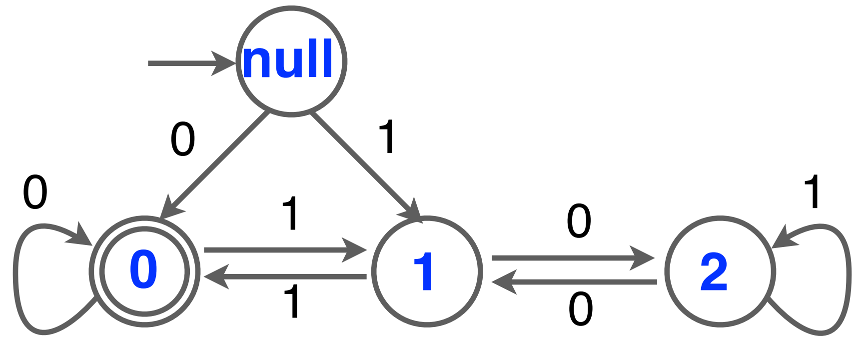
$\delta^*($**0**$,01001) =$ **0**

$\delta^*($**null**$,\varepsilon) =$ **null**

$\delta^*($**0**$,\varepsilon) =$ **0**

$\delta^*($**null**$,010) =$ **2**

$\delta^*($**2**$,01) =$ **0**



Behavior of a DFA on an input

Theorem: $\delta^*(q,uv) = \delta^*(\delta^*(q,u),v)$

$\delta^*(q,\varepsilon) = q$
$\delta^*(q,au) = \delta^*(\delta(q,a), u)$

Proof: By induction on $|u|$

Base case: $|u|=0$. $\delta^*(q,uv) = \delta^*(q,v)$ and
$\delta^*(\delta^*(q,u),v) = \delta^*(\delta^*(q,\varepsilon),v) = \delta^*(q,v)$

Induction: Let $n > 0$.
Assume that the claim holds for all u, $|u| < n$ (and all v, q).
Consider any u, v, q, s.t. $|u|=n$. Let $u=aw$, where $|w| = n\text{-}1 < n$.

$\delta^*(q,uv) = \delta^*(q,awv) = \delta^*(q',wv)$ where $q'=\delta(q,a)$ [by def.]
$= \delta^*(\delta^*(q',w),v)$ [by IH]
But $\delta^*(q',w) = \delta^*(\delta(q,a),w) = \delta^*(q,aw)$ [by def.]
Hence $\delta^*(q,uv) = \delta^*(\delta^*(q,u), v)$ [QED]

Input Accepted by a DFA

We say that M **accepts** $w \in \Sigma^*$ if M, on input w, starting from the start state s, reaches a final state

i.e., $\delta^*(s,w) \in F$

$L(M)$ is the set of all strings accepted by M

i.e., $L(M) = \{ w \mid \delta^*(s,w) \in F \}$

Called the language accepted by M

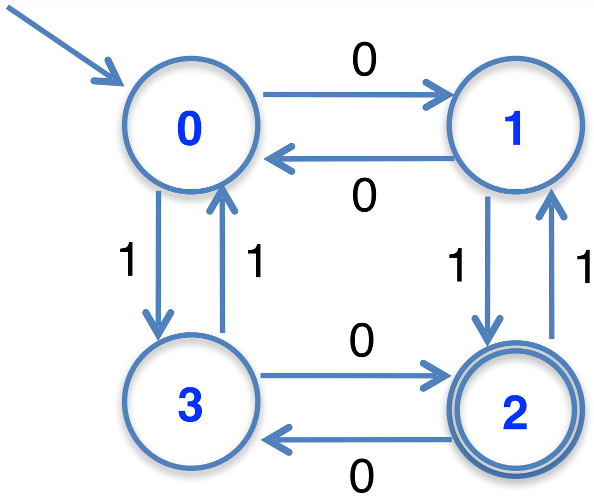
Warning

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

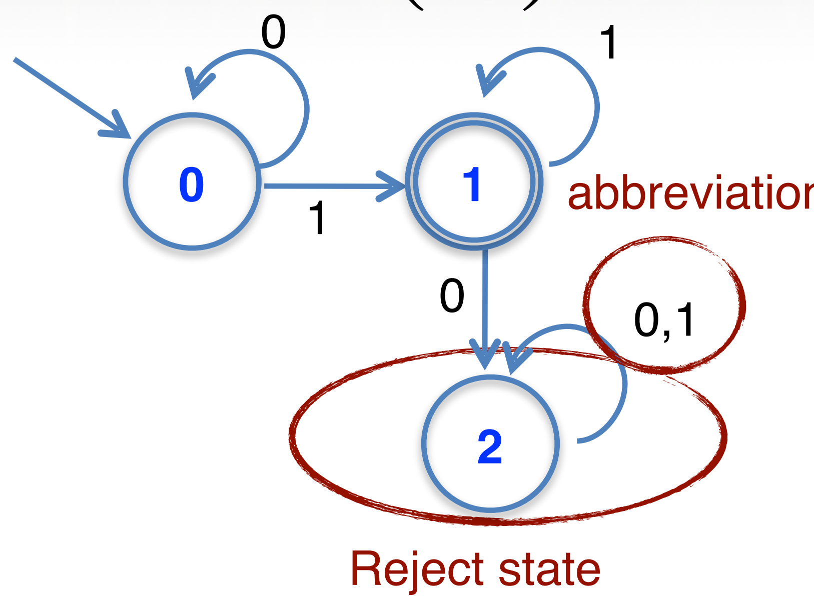
"M accepts language L" means M accepts each string in L and no others!

$$L(M) = L$$

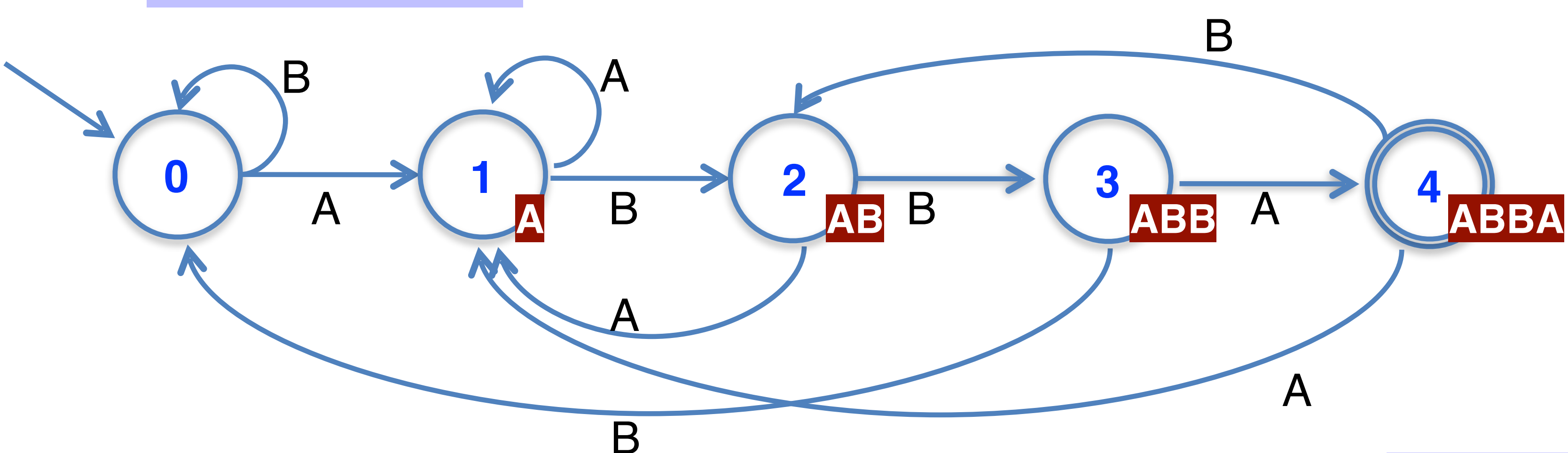
Examples: What is $L(M)$?



odd #0 and odd #1



0*11*



(A+B)*ABBA

Recall Regular Languages

Any regular language has a regular expression
and vice versa

Atomic expressions (Base cases)	
$\emptyset$	$L(\emptyset) = \emptyset$
ε	$L(\varepsilon) = \{ \varepsilon \}$
a for $a \in \Sigma$	$L(a) = \{ a \}$

Inductively defined expressions	
(r_1+r_2)	$L(r_1+r_2) = L(r_1) \cup L(r_2)$
$(r_1 r_2)$	$L(r_1 r_2) = L(r_1)L(r_2)$
$(r)^*$	$L(r^*) = L(r)^*$

Any regular language is accepted by a DFA and vice versa (*to be proven later*)

Building DFAs

State = Memory

First, decide on Q

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

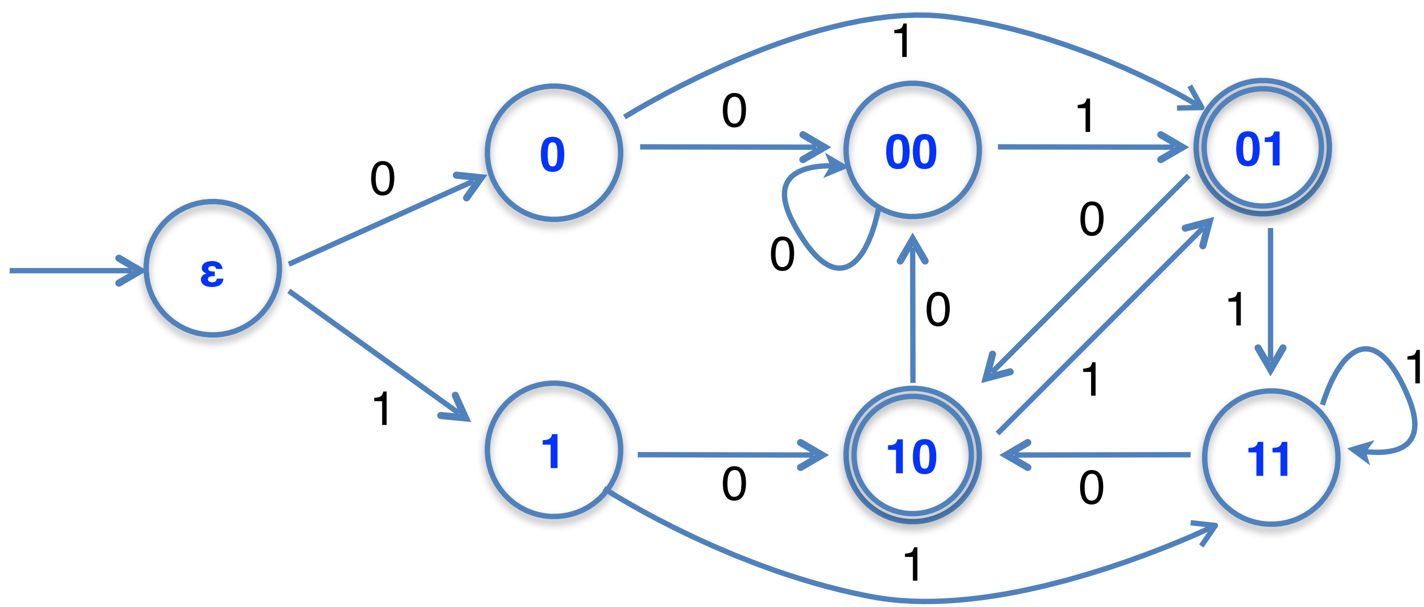
When designing a DFA, think “what do I need to know at this moment?” That is your state.

Construction Exercise

$L(M) = \{w \mid w$ ends in 01 or 10 $\}$

What should be in the memory?

Last two bits seen.
Possible values: **ε**, **0**, **1**, **00**, **01**, **10**, **11**

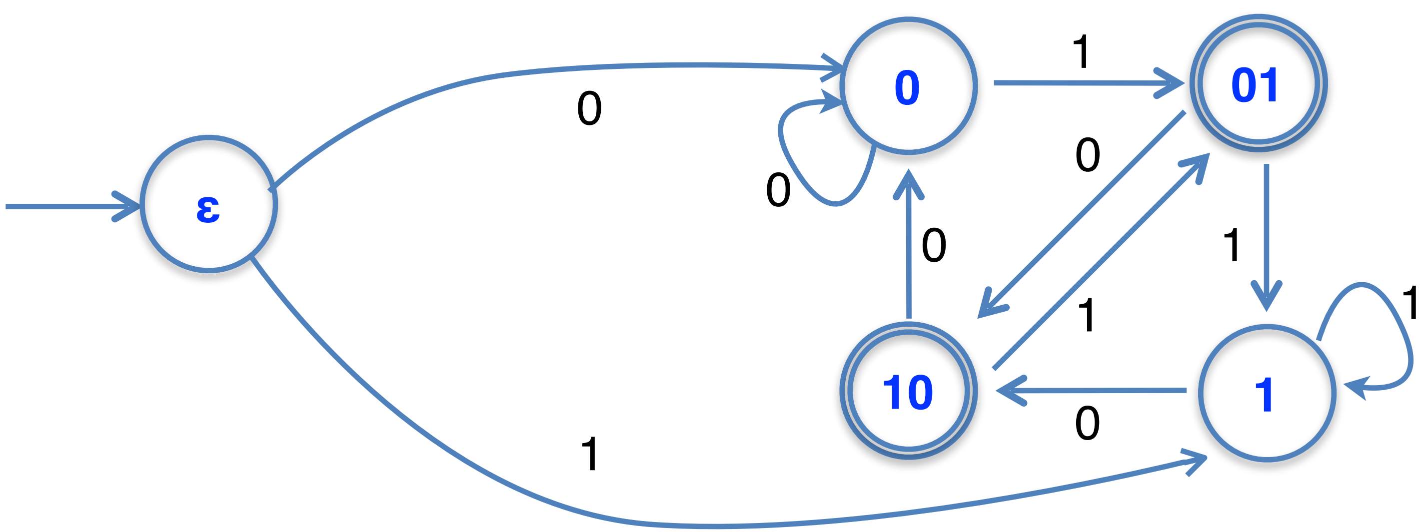


Construction Exercise

$L(M) = \{w \mid w$ ends in 01 or 10 $\}$

What should be in the memory?

Last two bits seen.
Possible values: **ε**, **(0+00)**, **(1+11)**, **01**, **10**

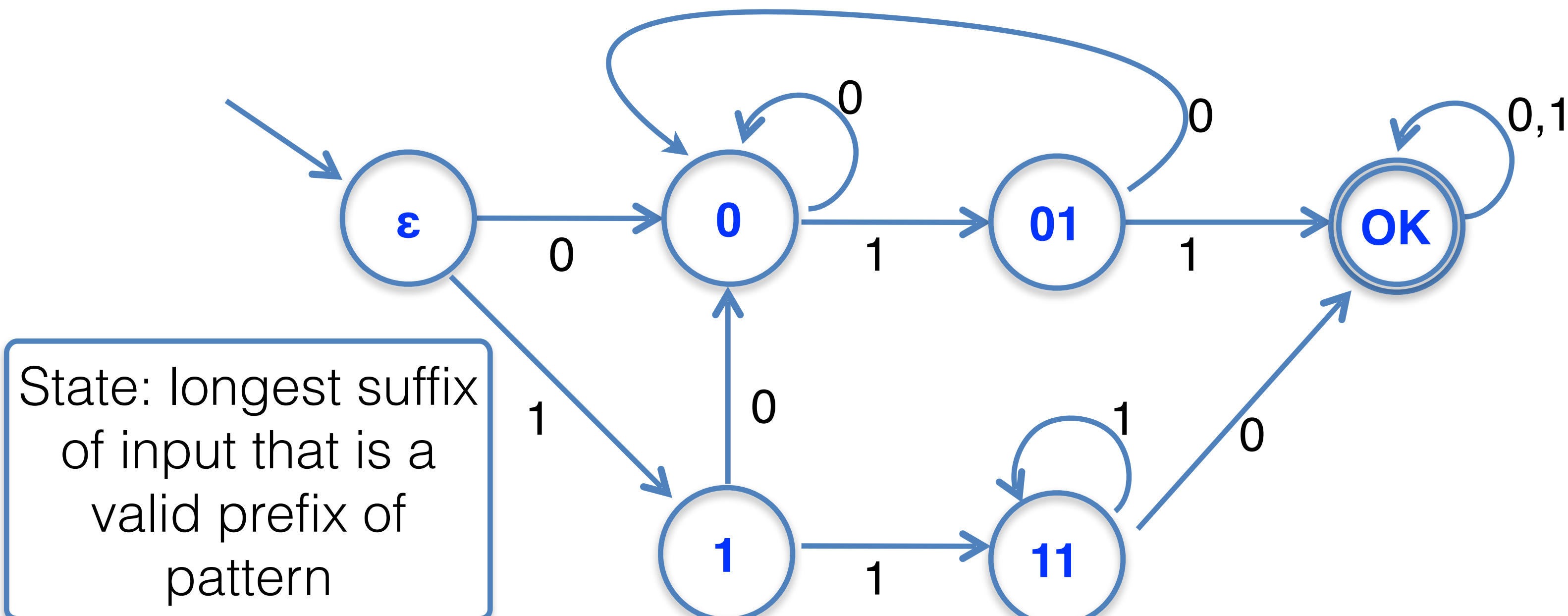


Construction Exercise

$L(M) = \{w \mid w \text{ contains } 011 \text{ or } 110 \}$

Brute force: Enough to remember last 3 symbols (8+4+2+1=15 states). Stay at accepting states if reached.

“Clever” construction: Enough to remember valid prefixes.
States: **ε**, **0**, **1**, **01**, **11**, **OK** (can forget everything else)



Building DFAs from DFAs

Complement: M' s.t. $L(M') = \Sigma^* - L(M)$

$$F' = Q - F$$

Concatenation: M_{12} s.t. $L(M_{12}) = L(M_1)\, L(M_2)$

Kleene Star: M' s.t. $L(M') = L(M)^*$

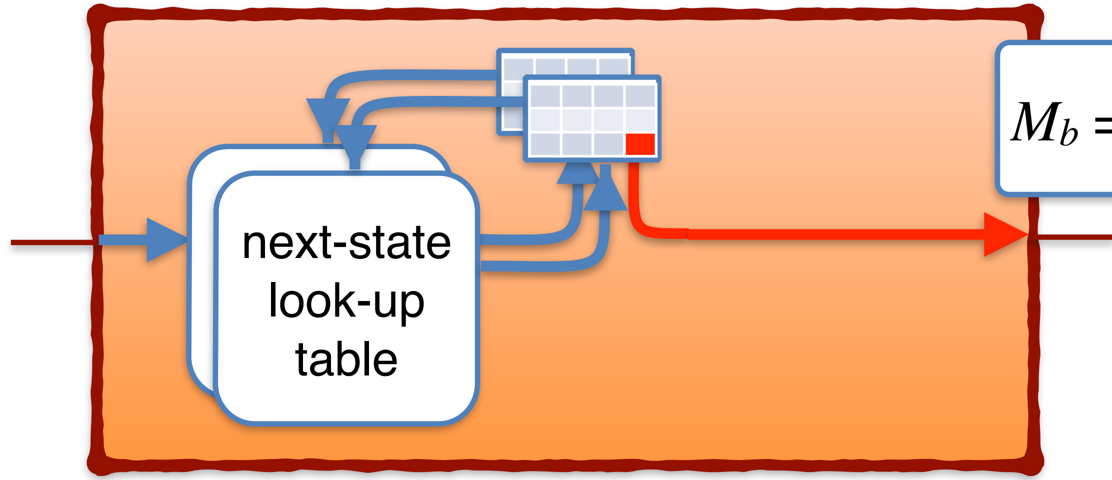
Later

Intersection and Union

DFA simulating two DFAs concurrently

Cross Product of DFAs

DFA M_{12} simulating the execution of 2 DFAs M_1 & M_2



$M_b = (\Sigma, Q_b, s_b, F_b, \delta_b)$

$$Q_{12} = Q_1 \times Q_2 \qquad s_{12} = (s_1, s_2)$$

$$\delta_{12}(\ (q_1,q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)\ )$$

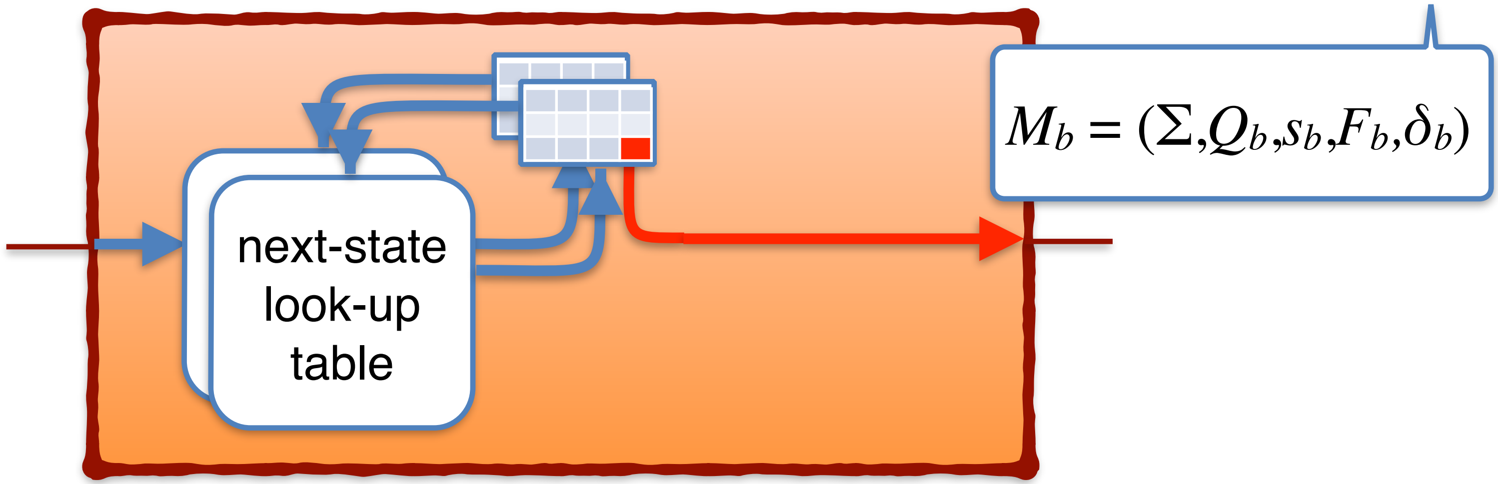
- **Theorem**: $\forall\ w \in \Sigma^*$,
  $\delta^*_{12}(s_{12}, w) = (q_1,q_2) \iff \delta^*_1(s_1,w) = q_1\ \ \&\ \ \delta^*_2(s_2,w) = q_2$

Proof by induction on $|w|$

- F_{12} = ? Depends on what we want

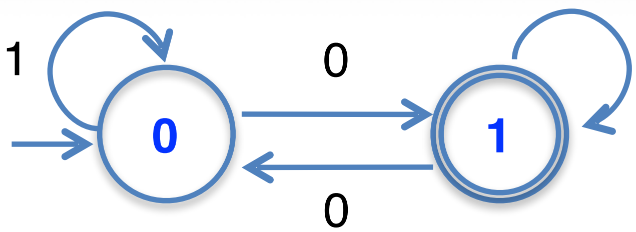
Cross Product of DFAs

DFA M_{12} simulating the execution of 2 DFAs M_1 & M_2

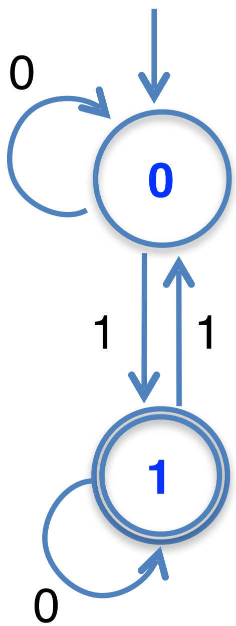


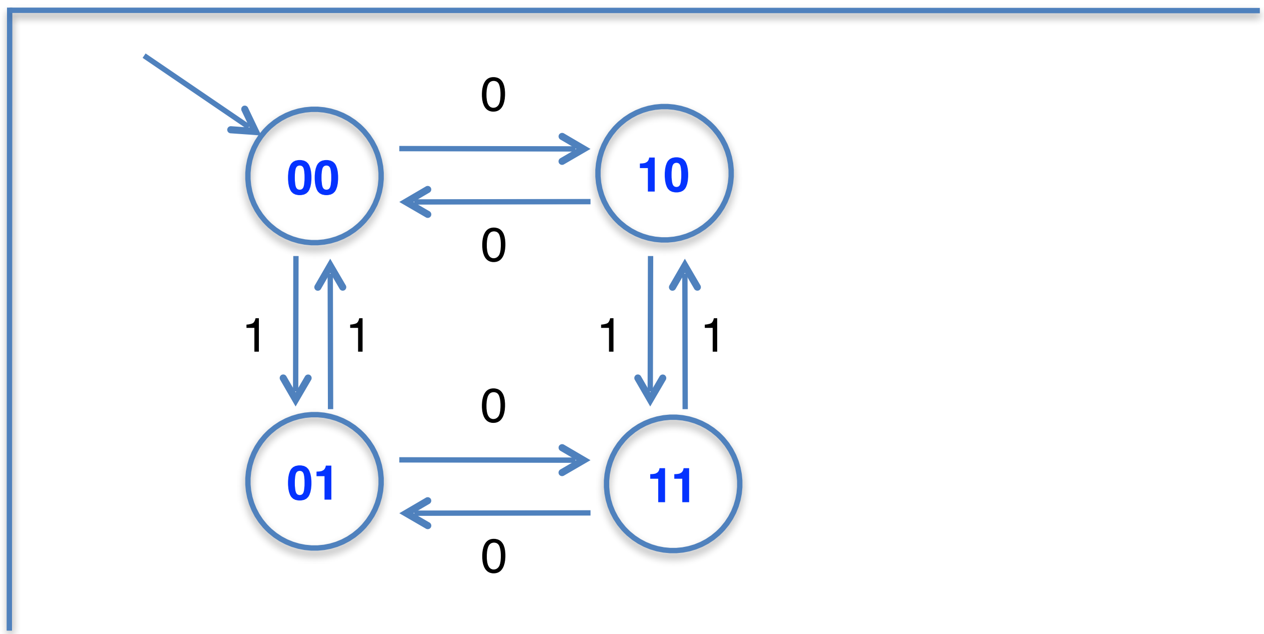
- $F_{12} = F_1 \times F_2 \Rightarrow L(M_{12}) = L(M_1) \cap L(M_2)$
- $F_{12} = (F_1 \times Q_2) \cup (Q_1 \times F_2) \Rightarrow L(M_{12}) = L(M_1) \cup L(M_2)$
- $F_{12} = F_1 \times (Q_2 - F_2) \Rightarrow L(M_{12}) = L(M_1) - L(M_2)$

Cross Product of DFAs



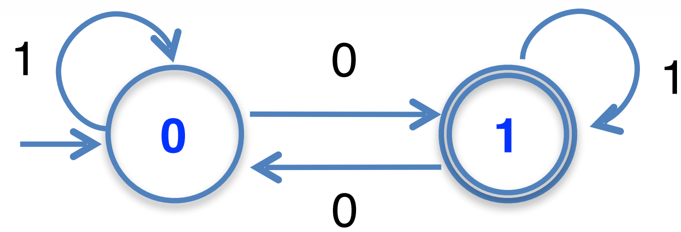
$L(M_1)$: odd #0



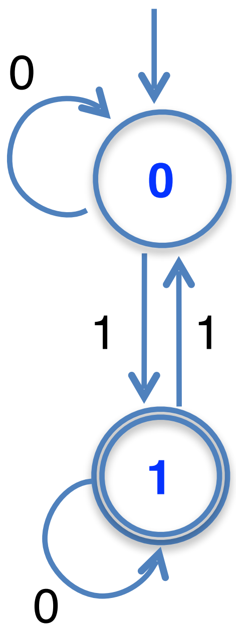


$L(M_2)$: odd #1

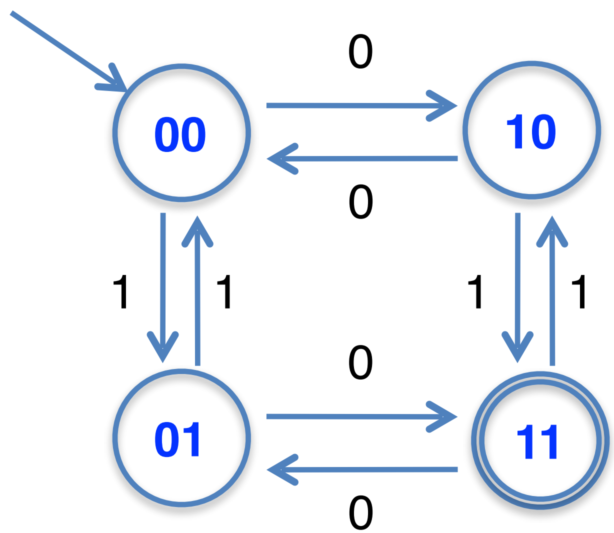
Cross Product of DFAs



$L(M_1)$: odd #0

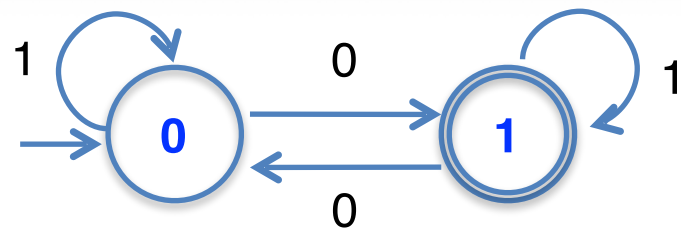


$L(M_2)$: odd #1

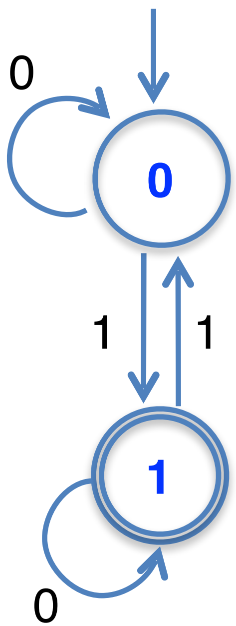


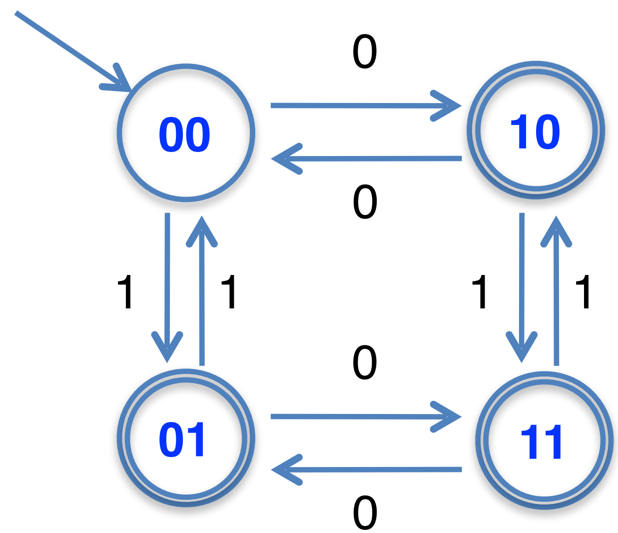
$L(M_{12}) = L(M_1) \cap L(M_2)$

Cross Product of DFAs



$L(M_1)$: odd #0





$L(M_{12}) = L(M_1) \cup L(M_2)$

$L(M_2)$: odd #1

Summary

DFA : $M = (\Sigma, Q, \delta, s, F)$, $L(M) = \{ w \mid \delta^*(s,w) \in F \}$

M provides a linear time algorithm to decide $L(M)$
(*later*: $L(M)$ is a regular language)

How to build DFAs :
Ask what should be in the state

How to build DFAs from simpler DFAs :
Complement, Product Construction.
More later!