BU CS 332 – Theory of Computation

Lecture 2:

- Deterministic Finite Automata
- Regular Operations
- Non-deterministic FAs

Reading:

Sipser Ch 1.1-1.2

Mark Bun

January 27, 2020

Deterministic Finite Automata

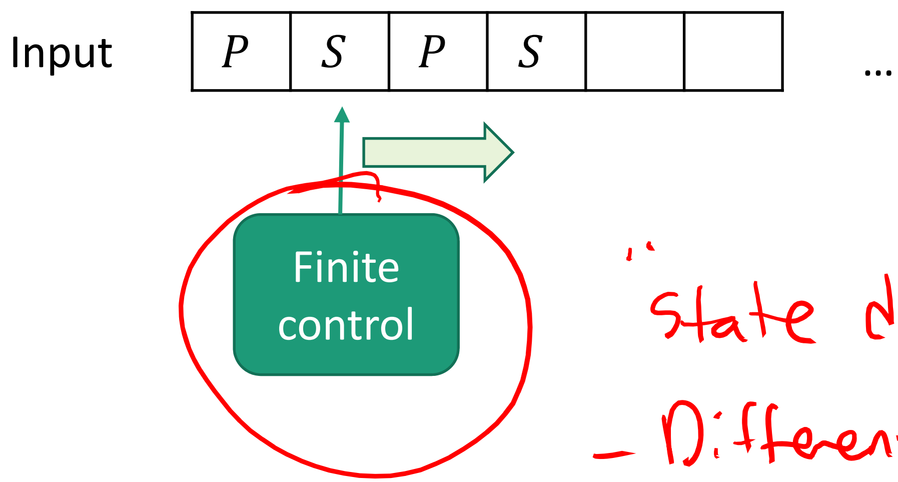
A (Real-Life?) Example

- Example: Car stereo
- P = Power button (ON/OFF)
- S = Source button (cycles through Radio/Bluetooth/USB)
  Only works when stereo is ON, but source remembered when stereo is OFF
- Starts OFF in Radio mode

- A computational problem: Does a sequence of button presses in $\{P, S\}^*$ leave the stereo ON in USB mode?

Machine Models

- Finite Automata (FAs): Machine with a finite amount of unstructured memory

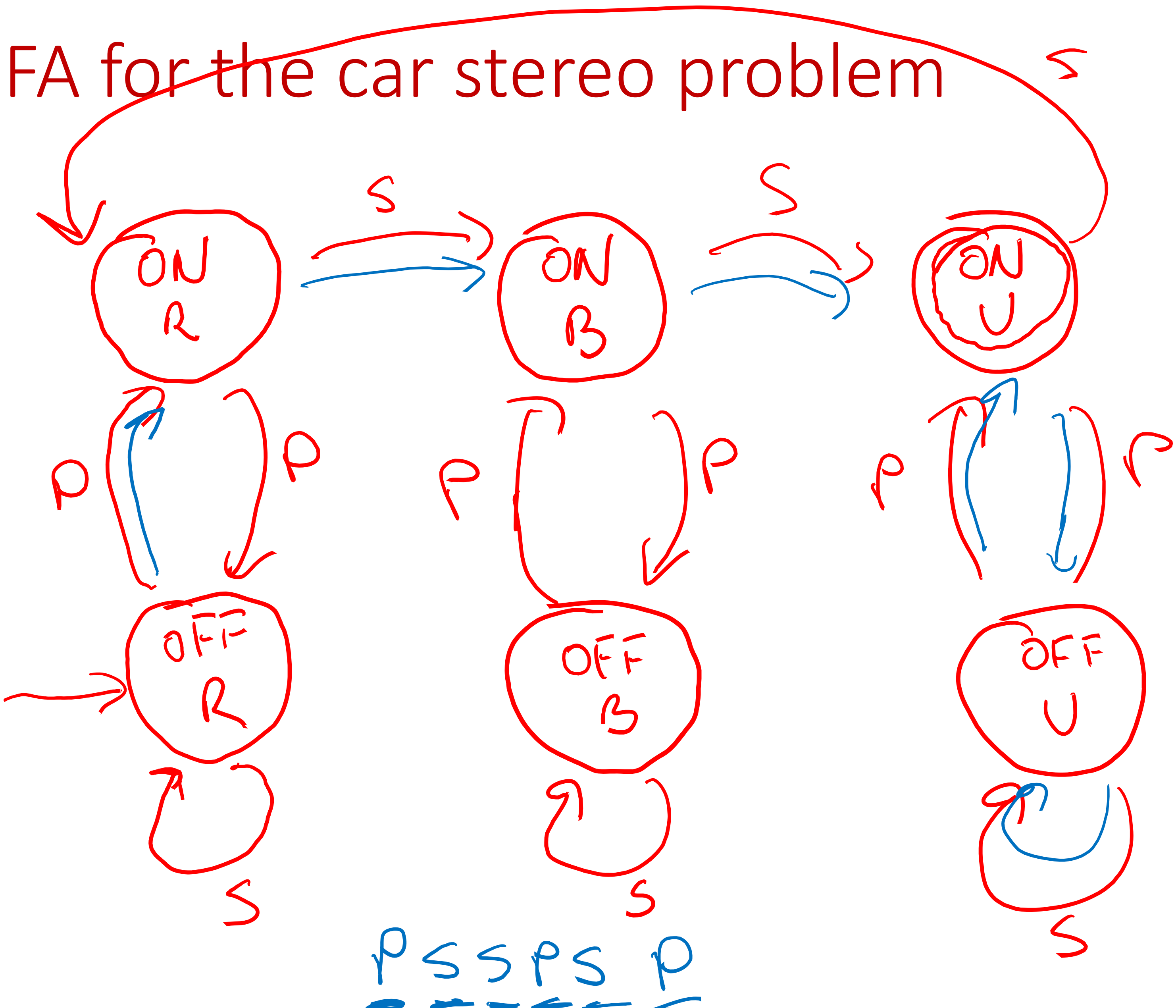


Control scans left-to-right

"state diagram"

- Different states control can be in
- How transitions between states
- How it decides to accept or reject

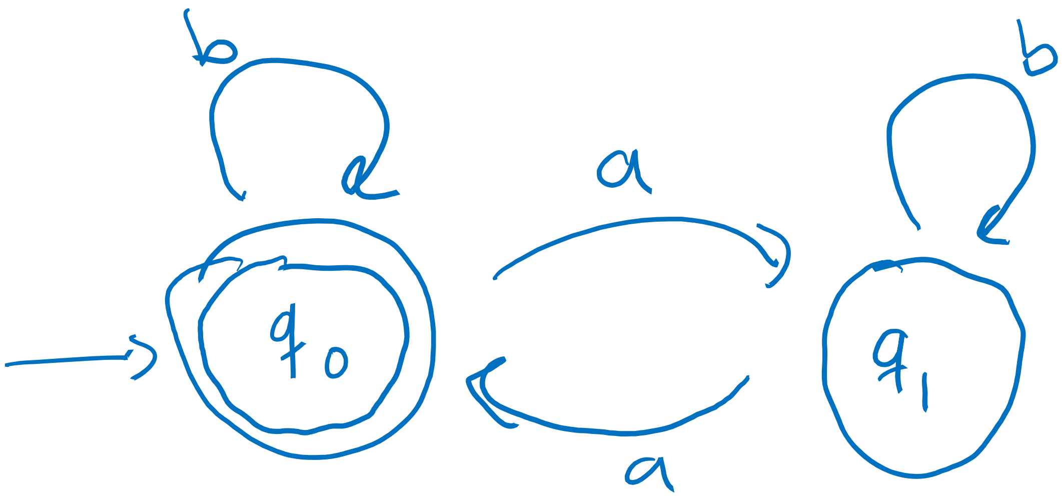
A DFA for the car stereo problem



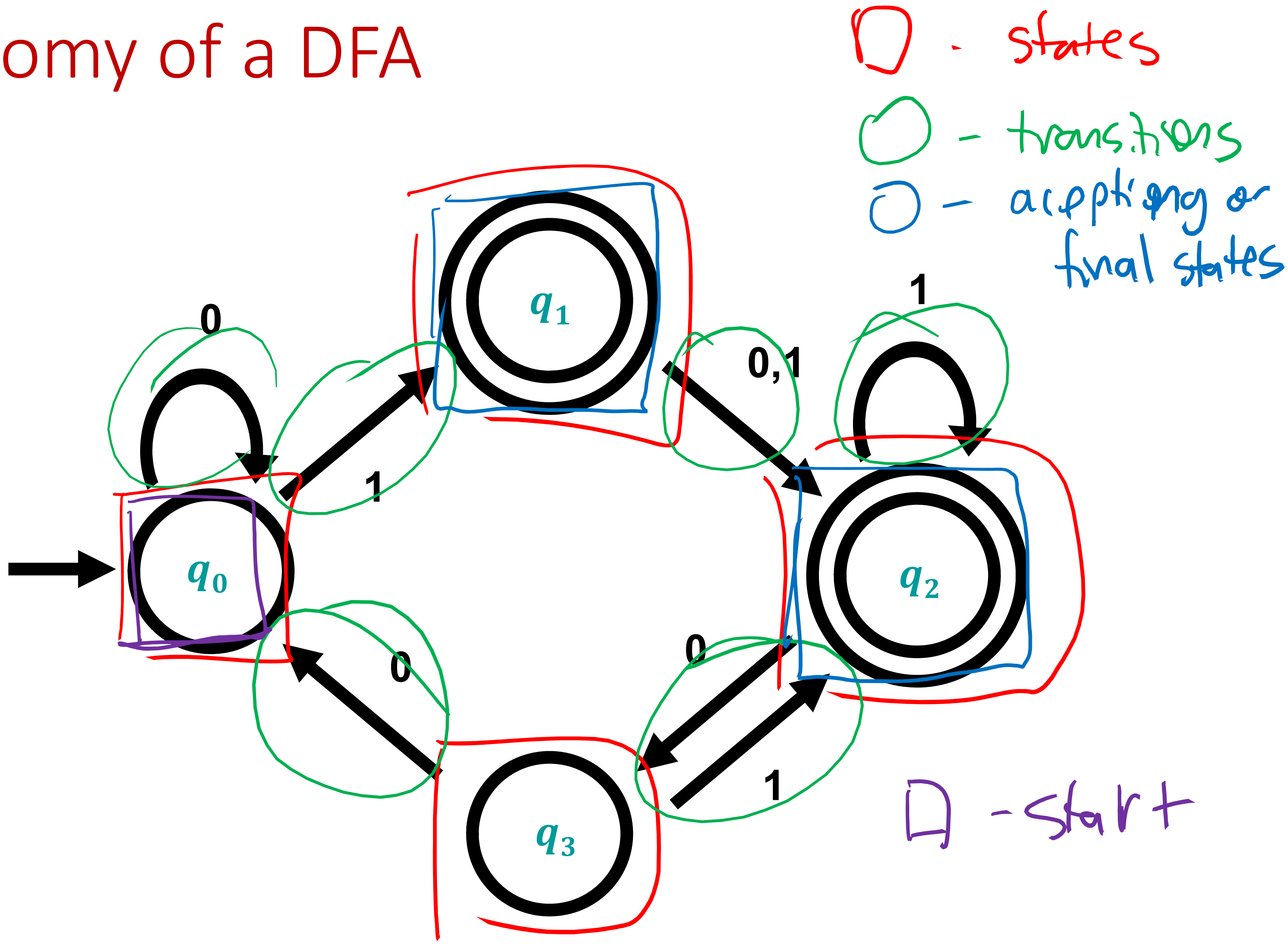
A DFA for Parity

Parity: Given a string consisting of a’s and b’s, does it contain an even number of a’s?

$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$



Anatomy of a DFA



Formal Definition of a DFA

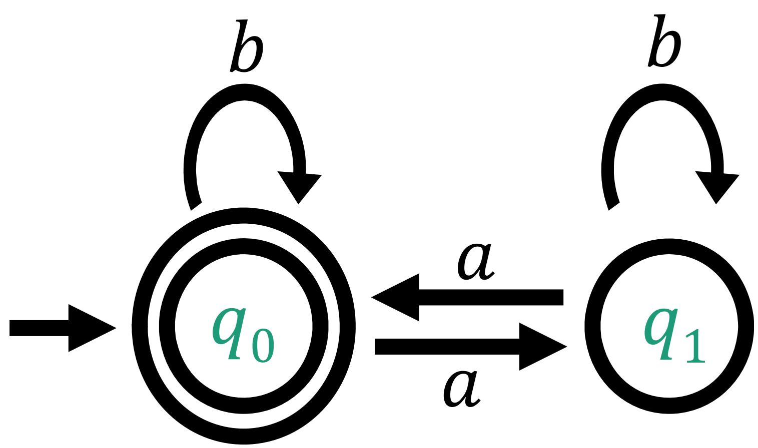
A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

- Q is the set of states
- Σ is the alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

A DFA for Parity

Parity: Given a string consisting of a's and b's, does it contain an even number of a's?

$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$



State set Q = $\{q_0, q_1\}$

Alphabet Σ = $\{a, b\}$

Transition function δ

δ	a	b
q_0	q_1	q_0
q_1	q_0	q_1

Start state q_0

Set of accept states F = $\{q_0\}$

Formal Definition of DFA Computation

Parity: w=ab

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

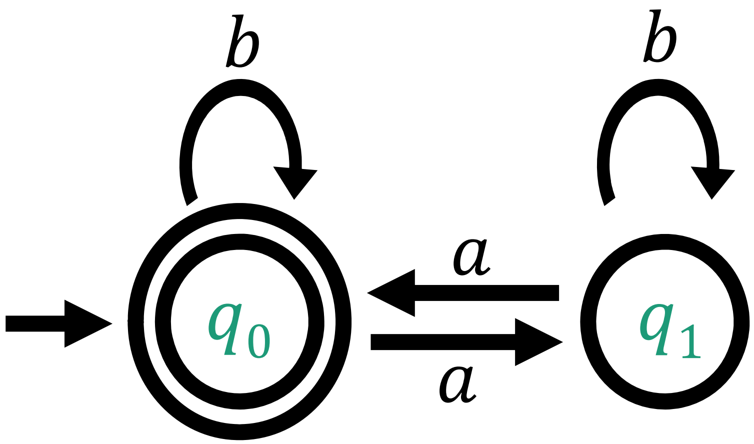
1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n-1$, and
3. $r_n \in F$

$L(M)$ = the language of machine M

= set of all (finite) strings machine M accepts

M recognizes the language $L(M)$

Example: Computing with the Parity DFA



Let $w = abba$
Does M accept w?

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ **accepts** a string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

$r_0 = q_0$
$r_1 = q_1 = \delta(r_0, w_1)$, $w_1 = a$
$r_2 = q_1 = \delta(r_1, w_2)$, $w_2 = b$
$r_3 = q_1 = \delta(r_2, w_3)$, $w_3 = b$
$r_4 = q_0 = \delta(r_3, w_4)$, $w_4 = a$

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n-1$, and
3. $r_n \in F$

Automata Tutor

http://automatatutor.com/

Regular Languages

Definition: A language is regular if it is recognized by a DFA

$L = \{ w \in \{a, b\}^* \mid w$ has an even number of a's $\}$ is regular

$L = \{ w \in \{0, 1\}^* \mid w$ contains 001 $\}$ is regular

Many interesting programs recognize regular languages

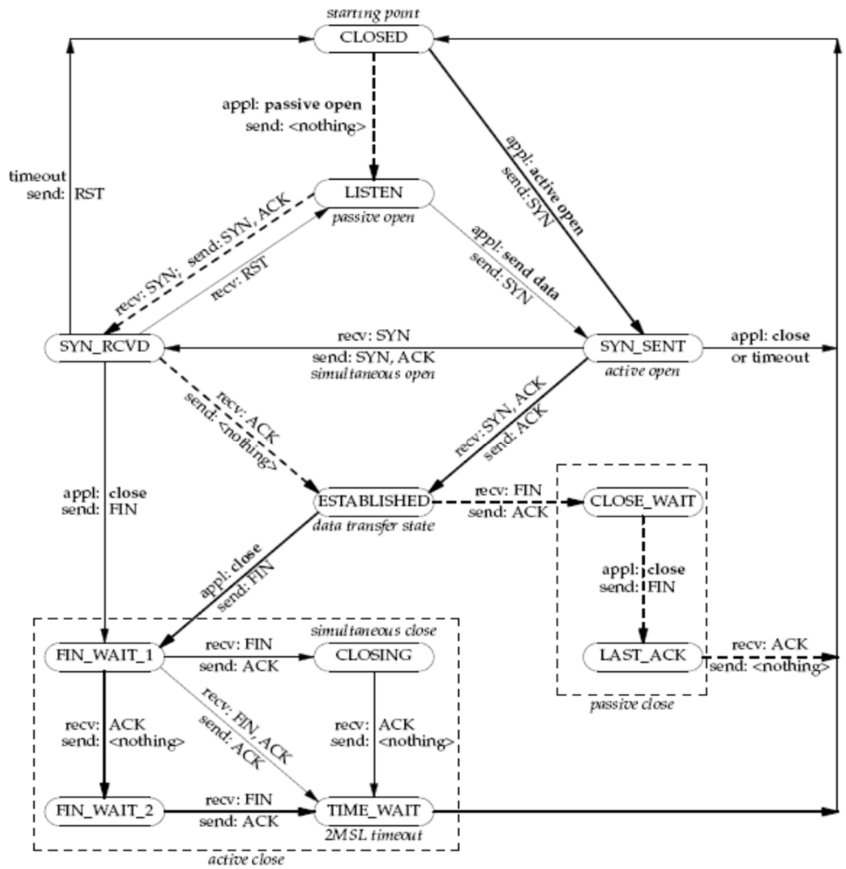
NETWORK PROTOCOLS

COMPILERS

GENETIC TESTING

ARITHMETIC

Internet Transmission Control Protocol



Let TCPS = { w | w is a complete TCP Session}

Theorem. TCPS is regular

Compilers

Comments :

Are delimited by /* */

Cannot have nested /* */

Must be closed by */

***/ is illegal outside a comment**

COMMENTS = {strings over {0,1, /, *} with legal comments}

Theorem. COMMENTS is regular.

Genetic Testing

DNA sequences **are strings over the alphabet** $\{A, C, G, T\}$.

S= CGTACAAAAA

A gene g **is a special substring over this alphabet.**

g = TAC

A genetic test searches a DNA sequence for a gene.

Is TAC a substring of S?

GENETICTEST$_g$ **= {strings over** $\{A, C, G, T\}$ **containing** g **as a substring}**

$S \in \text{GENETICTEST}_g$

Theorem. **GENETICTEST**$_g$ **is regular for every gene** g.

$\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}$ ← Row$_1$ ← Row$_2$ ← Row$_3$

Arithmetic

LET Σ_3 = $\left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$

- **A string over Σ_3 has three ROWS (ROW$_1$, ROW$_2$, ROW$_3$)**
- **Each ROW $b_0 b_1 b_2 \ldots b_N$ represents the integer**

$$b_0 + 2b_1 + \ldots + 2^N b_N.$$

- **Let ADD = $\{S \in \Sigma_3{}^* \mid$ ROW$_1$ + ROW$_2$ = ROW$_3\}$**

Theorem. ADD is regular.

Regular Operations

An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

Example: The integers $\mathbb{Z} = \{\ldots -2, -1, 0, 1, 2, \ldots\}$ are **closed** under

- Addition: $x + y$
- Multiplication: $x \times y$
- Negation: $-x$
- …but NOT Division: $x \,/\, y$ $x = 1,\ y = 2 \quad x/y = 1/2$

We'd like to investigate similar closure properties of the class of regular languages

Regular operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Union: $A \cup B =$ $\{w \in \Sigma^* \mid w \in A \text{ or } w \in B\}$

Concatenation: $A \circ B =$ $\{wv \mid w \in A \text{ and } v \in B\}$

Star: $A^* =$ $\{w_1 w_2 \ldots w_n \mid w_i \in A,\ i = 1, \ldots, n,\ n \in \{0, 1, \ldots\}\}$

$= \{\epsilon\} \cup A \cup A \circ A \cup A \circ A \circ A \cup \ldots$

Other operations

Let $A, B \subseteq \Sigma^*$ be languages. Define

Complement: $\bar{A} = \{ w \in \Sigma^* \mid w \notin A \}$

Intersection: $A \cap B = \{ w \in \Sigma^* \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 w_2 \dots w_n \mid w_n w_{n-1} \dots w_2 w_1 \in A \}$

Closure properties of the regular languages

Theorem: The class of regular languages is closed under all three regular operations (union, concatenation, star), as well as under complement, intersection, and reverse.

i.e., if A and B are regular, applying any of these operations yields a regular language

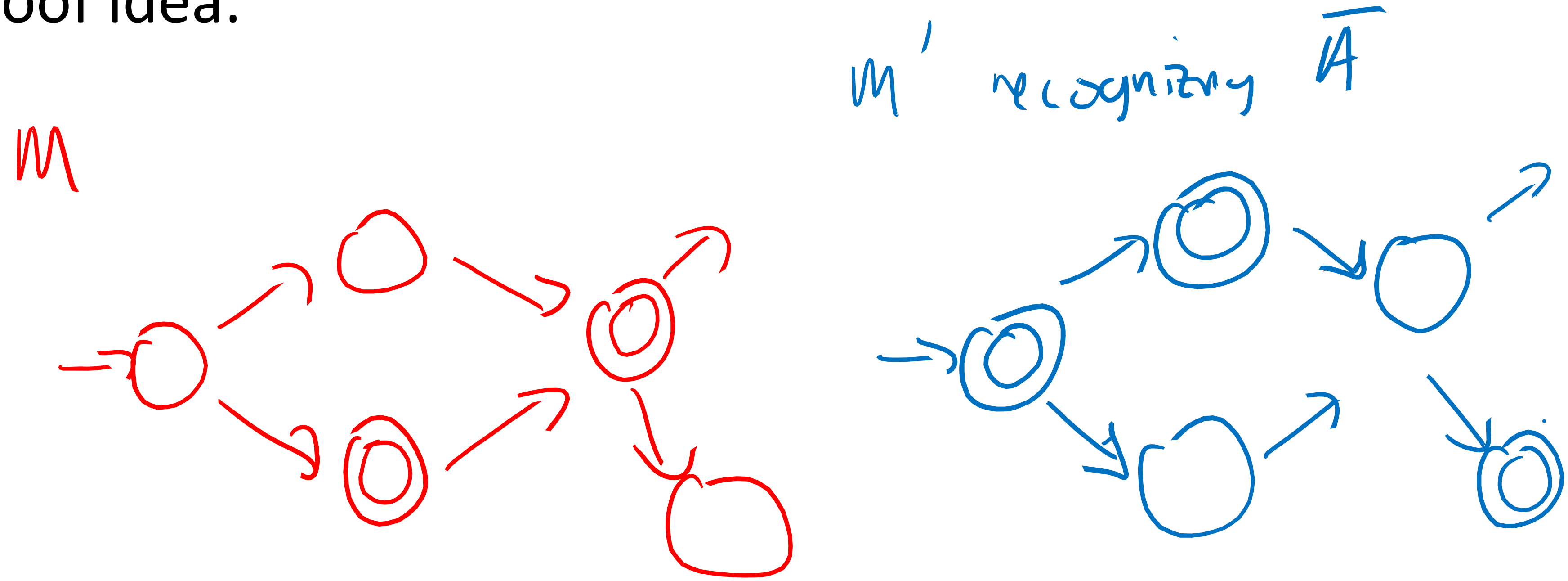
Proving Closure Properties

Complement

Complement: $\bar{A} = \{ w \mid w \notin A\}$

Theorem: If A is regular, then $\bar{A}$ is also regular

Proof idea:



Union

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

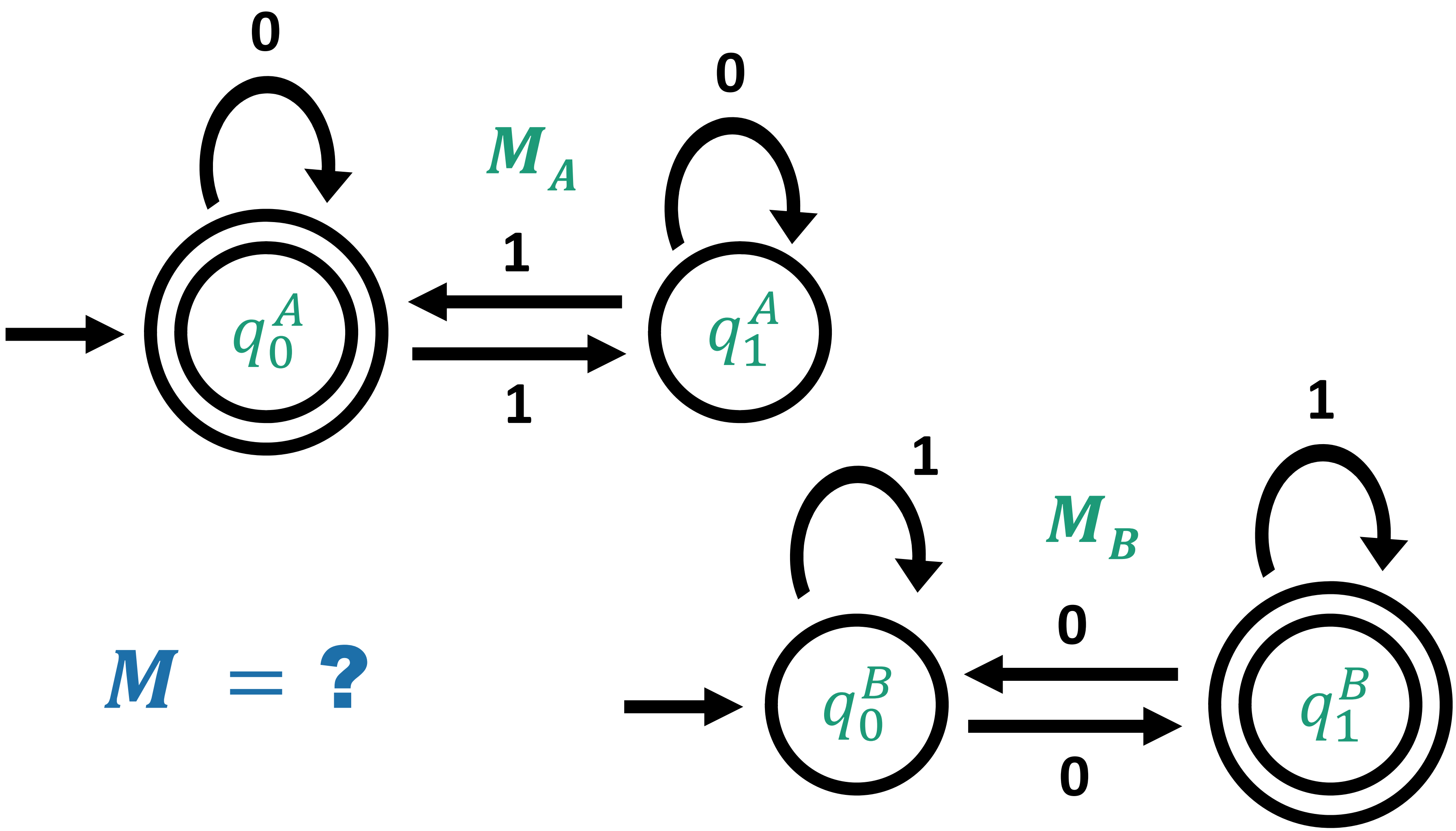
Theorem: If A and B are regular, then so is $A \cup B$

Proof:

Let $M_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A)$ be a DFA recognizing A and
$M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B)$ be a DFA recognizing B

Goal: Construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$
that recognizes $A \cup B$

Example



$M = ?$

Closure under union proof (cont'd)

Idea: Run both M_A and M_B at the same time

"Cross-product construction"

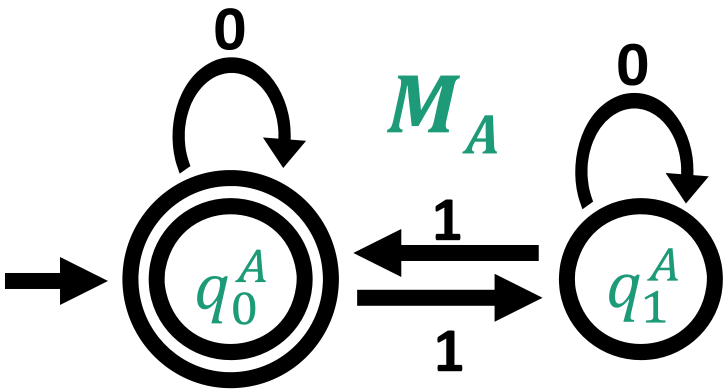
$$Q = Q_A \times Q_B$$
$$= \{(q_A, q_B) \mid q_A \in A \text{ and } q_B \in B\}$$

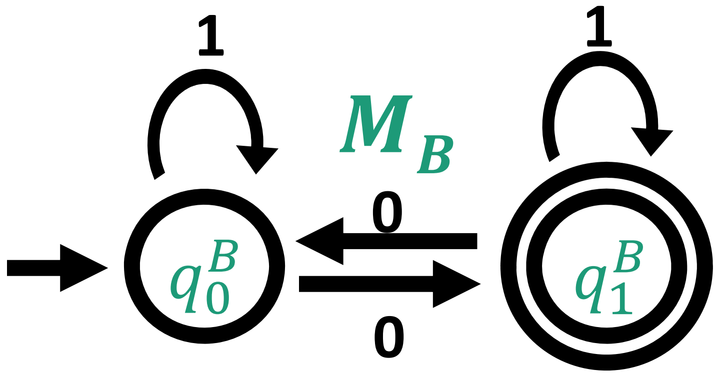
$$\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$$

$$q_0 = (q_0^A, q_0^B)$$

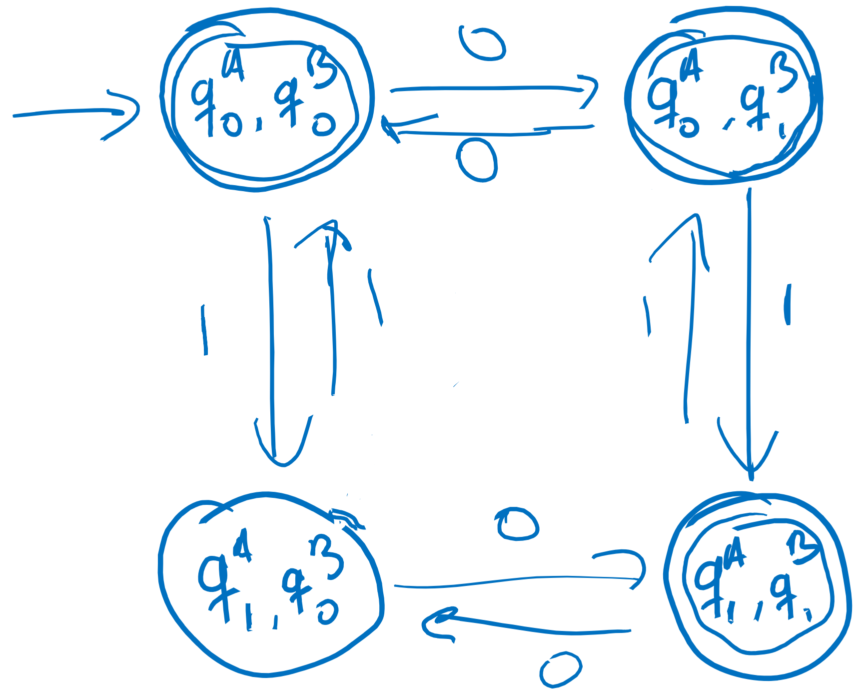
$$F = \{(q_A, q_B) \mid q_A \in F_A \text{ or } q_B \in F_B\}$$
$$= F_A \times Q_B \cup Q_A \times F_B$$

Example (cont'd)





M



Intersection

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B\}$

Theorem: If A and B are regular, then so is $A \cap B$

Proof:

Let $M_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A)$ be a DFA recognizing A and
$M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B)$ be a DFA recognizing B

<u>Goal:</u> Construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$
that recognizes $A \cap B$ Modify union construction

Intersection

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B\}$

Theorem: If A and B are regular, then so is $A \cap B$

Another Proof:

$A \cap B = \overline{\bar{A} \cup \bar{B}}$

Reverse

Reverse: $A^R = \{w_1 w_2 \cdots w_n | w_n \cdots w_1 \in A\}$

Theorem: If A is regular, then A^R is also regular

Proof idea:

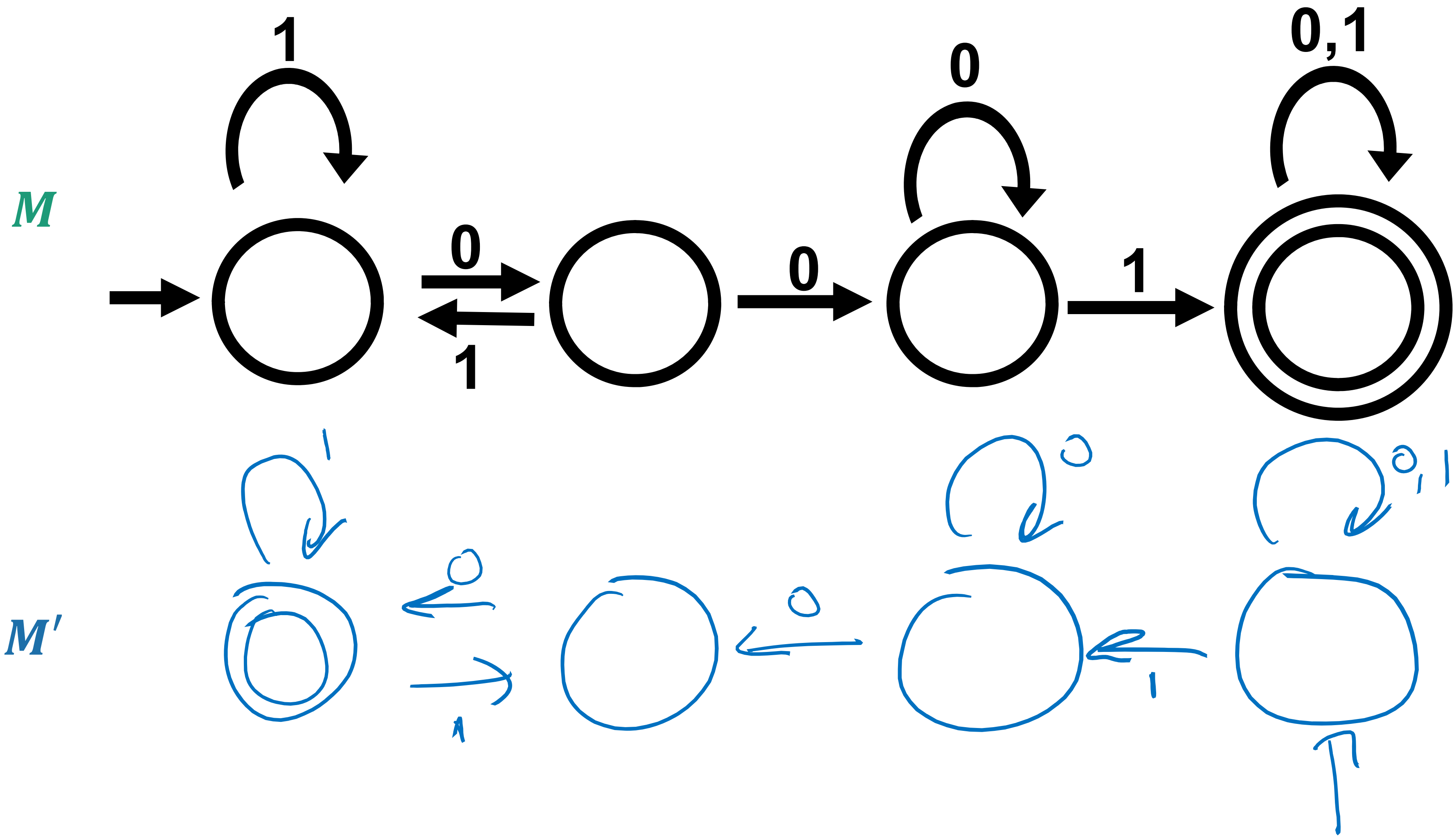
Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A

Goal: Construct a DFA $M' = (Q', \Sigma, \delta', q_0', F')$
that recognizes A^R

Define M' as M but

- With the arrows reversed
- With start and accept states swapped

Example (Reverse)

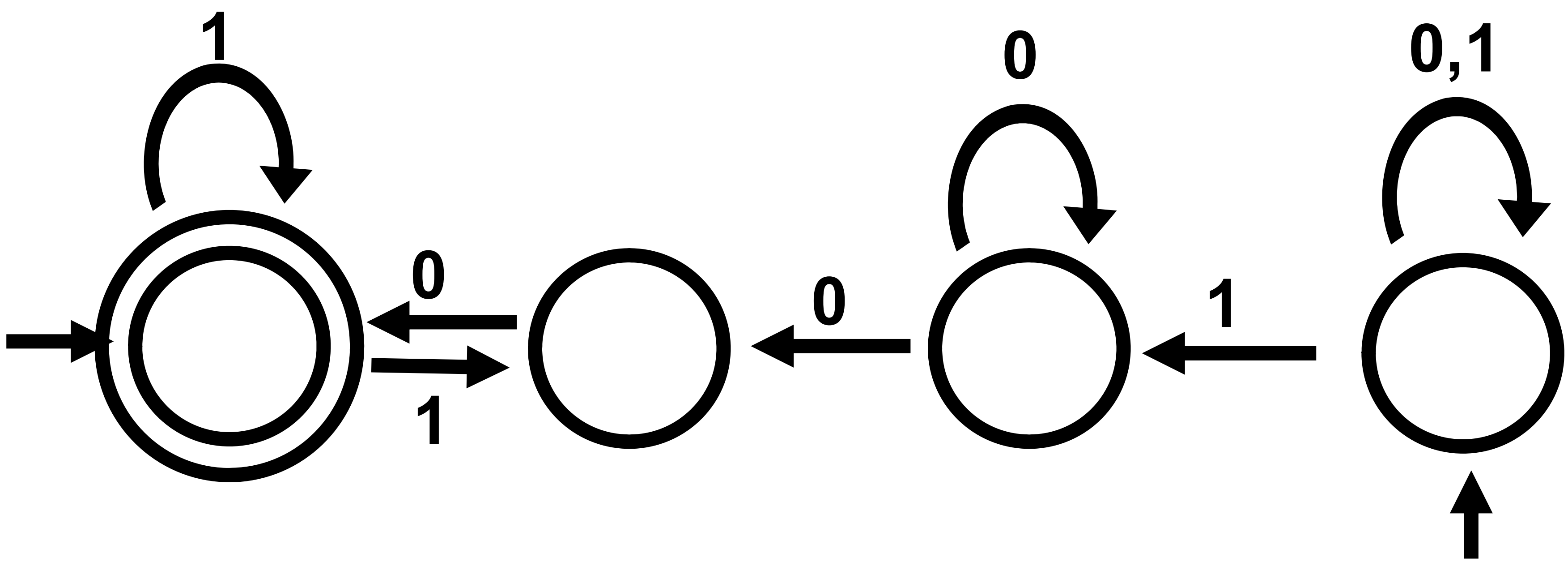


Closure under reverse

M' is not always a DFA!

- It might have many start states
- Some states may have too many outgoing edges, or none at all

Nondeterminism



A Nondeterministic Finite Automaton (NFA) accepts if there is a way to make it reach an accept state.