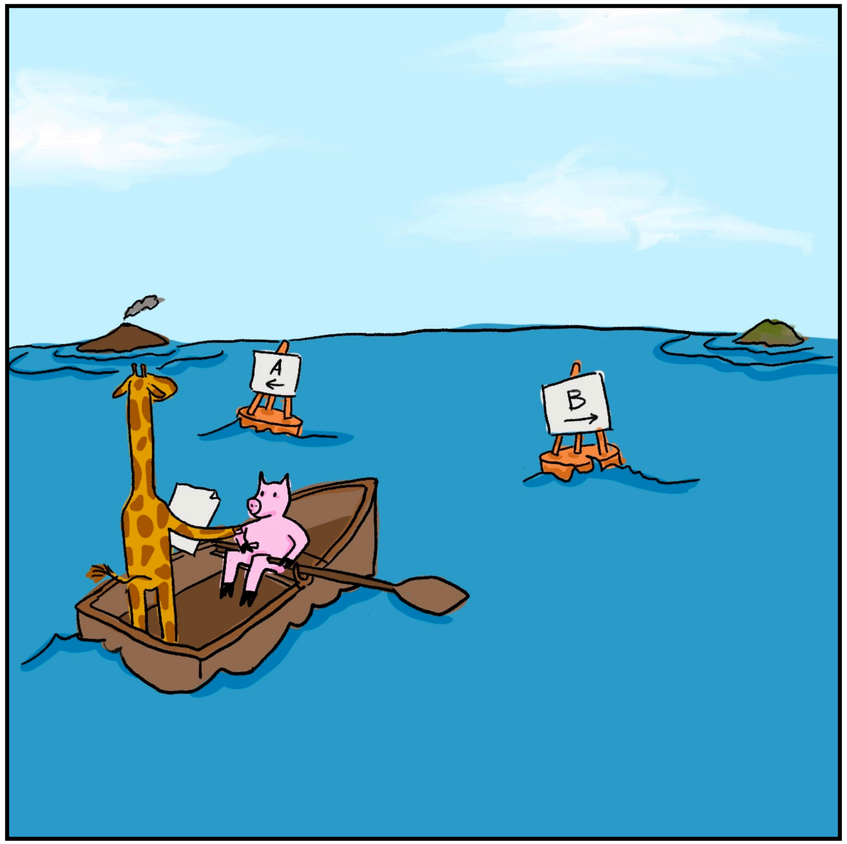
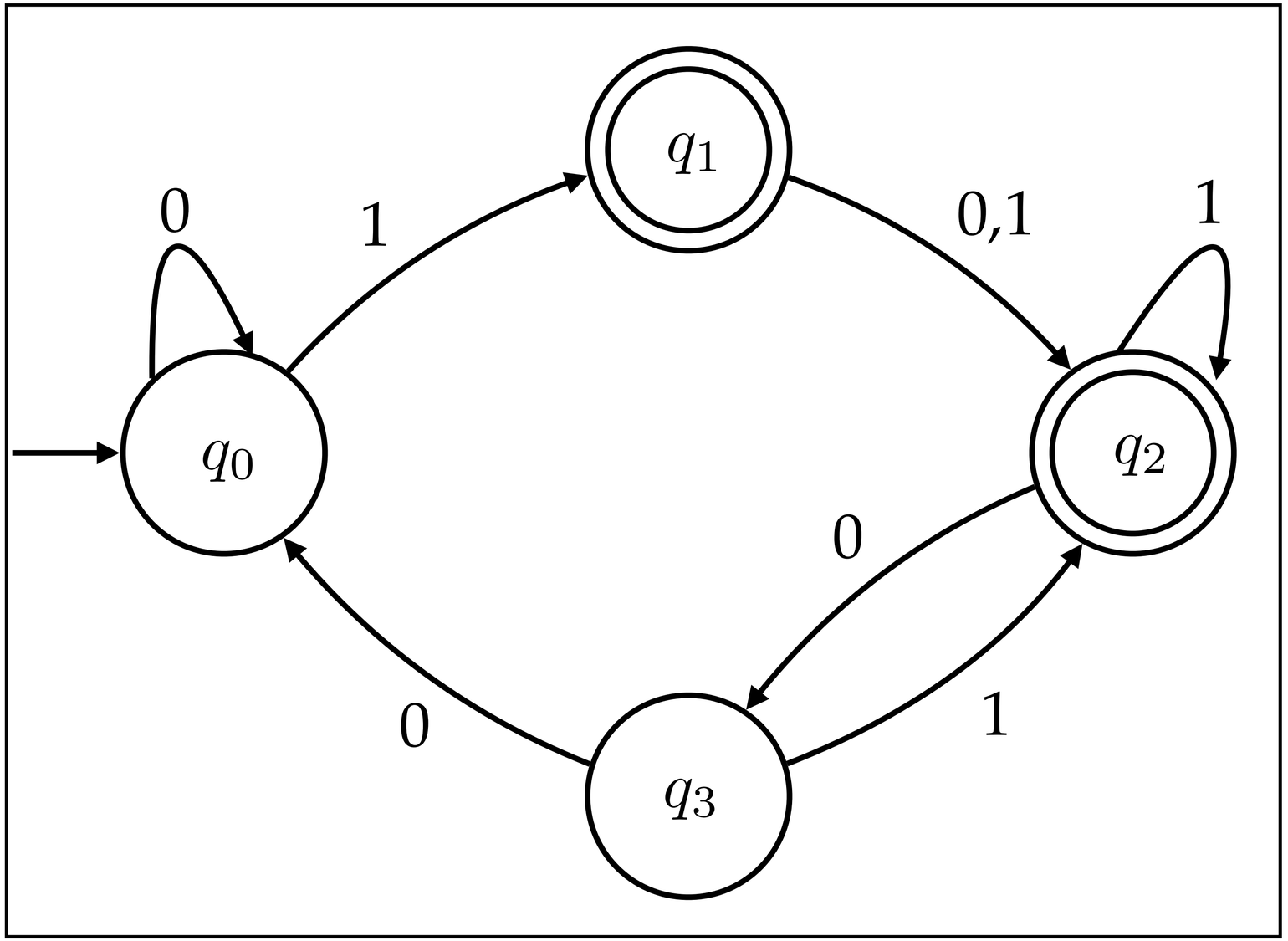
CS251

Great Ideas in *Theoretical* Computer Science



Deterministic Finite Automata 2

Quick review



$$M = (Q, \Sigma, \delta, q_0, F)$$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{0,1\}$$

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

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_3	q_2
q_3	q_0	q_2

q_0 is the start state

$$F = \{q_1, q_2\}$$

$L(M)$ = the set of strings that DFA M accepts.

DFA M **solves** the language $L(M)$.

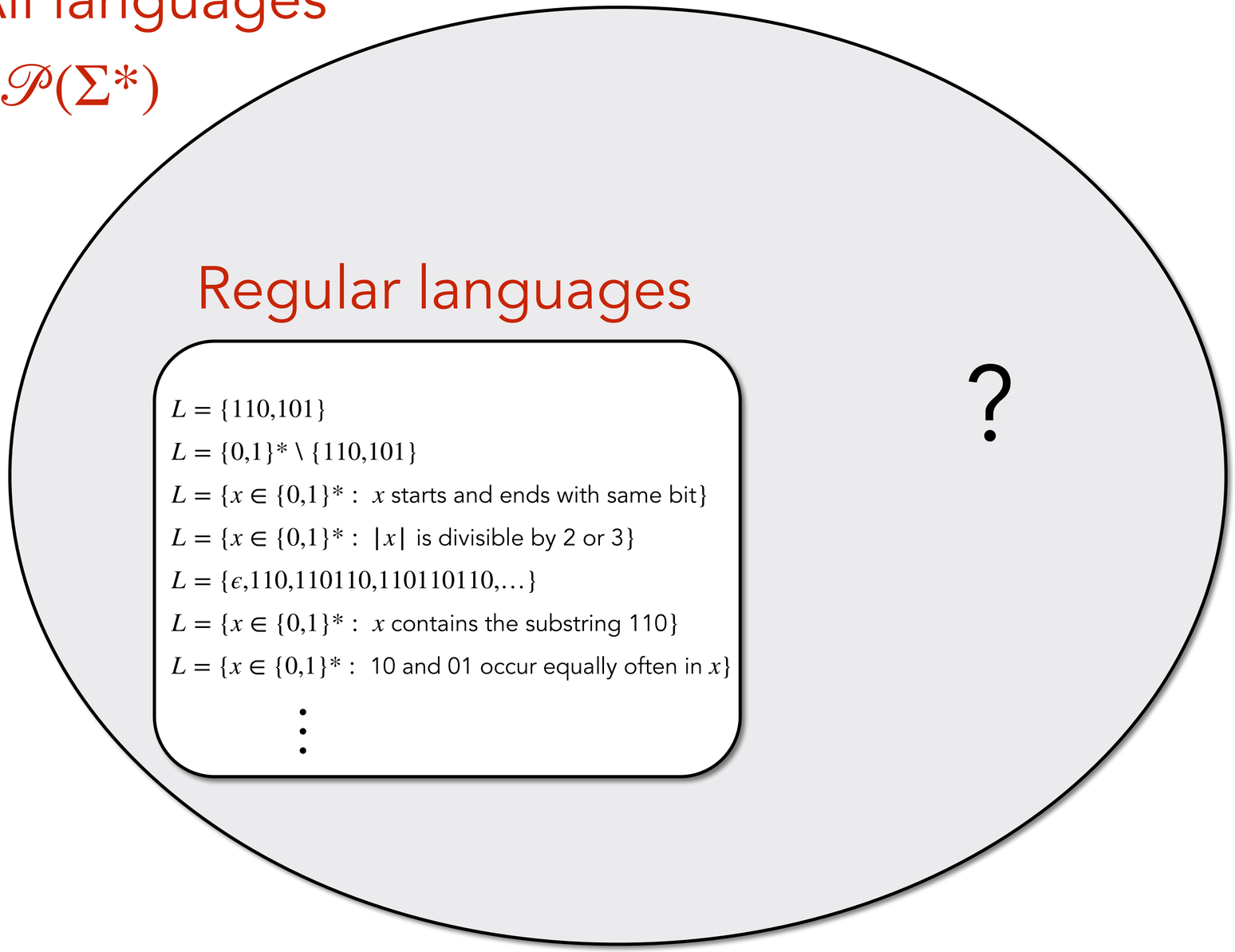
decides

computes

Definition: A language L is called ***regular*** if there is some DFA solving L.

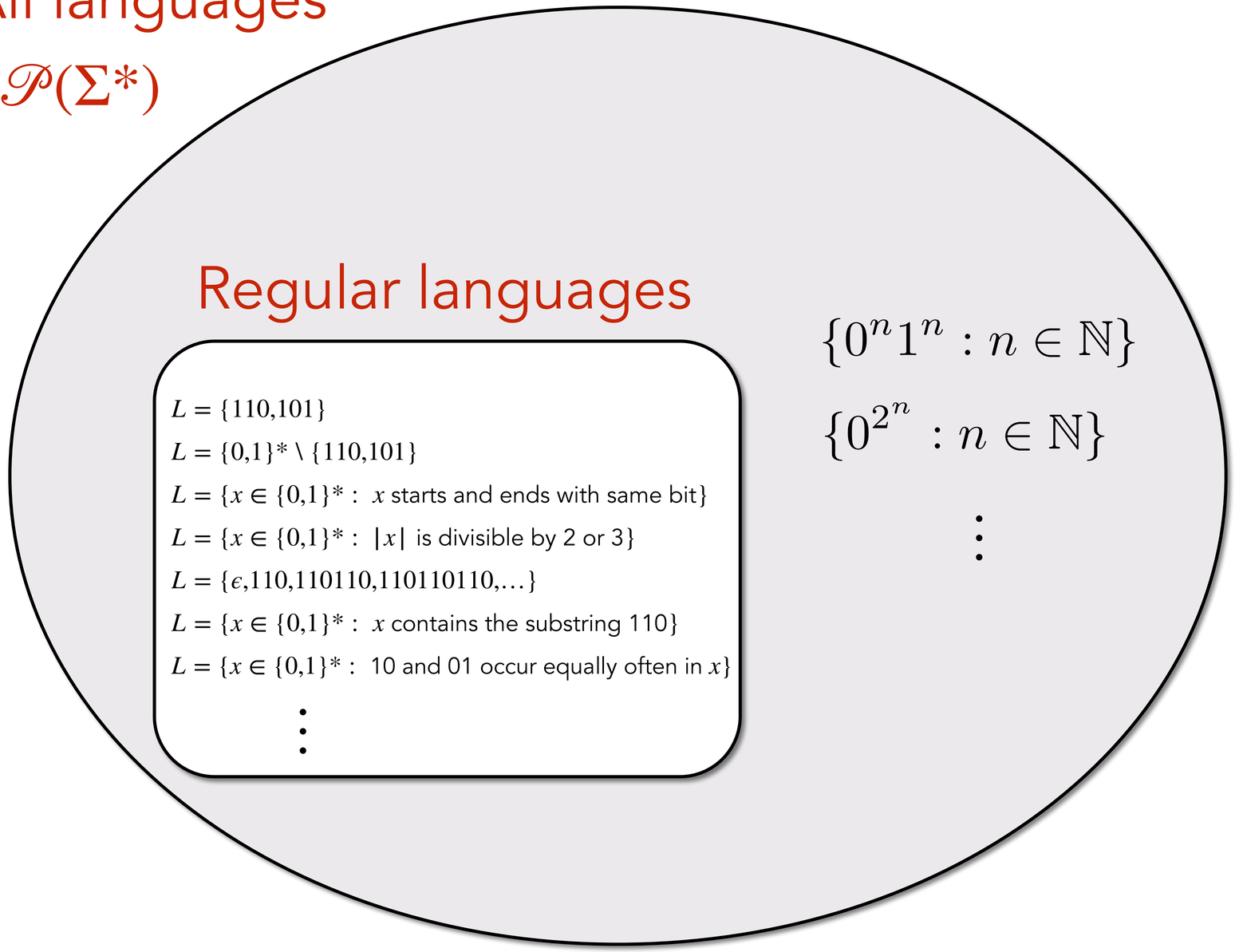
All languages

$\mathscr{P}(\Sigma^*)$



All languages

$\mathscr{P}(\Sigma^*)$



An Application

An application of DFAs

Input: string S of length n; string w of length k.

Output: True if w occurs in S; False otherwise.

$S = acbbabcbbcbaaabaaccccacbaaabcbcbaab$

$w = acba$

Naïve algorithm:

$\sim nk$ steps.

Can we do better?

An application of DFAs

Input: string S of length n; string w of length k.

Output: True if w occurs in S; False otherwise.

DFA solution:

We want to know if $S \in L_w = \Sigma^* \cdot \{w\} \cdot \Sigma^*$.

L_w is regular! So there is a DFA M_w computing it.

Feed S to M_w. $\sim n$ steps.

Time to build M_w? Simple algorithm: $\sim k^3$ steps.

Knuth-Morris-Pratt 1977: $\sim k$ steps.

Closure properties of regular languages

Regular languages are closed under complementation

Proposition: If $L \subseteq \Sigma^*$ is regular, then so is $\overline{L} = \Sigma^* \backslash L$.

Proof: If L is regular, then there is a DFA

$$M = (Q, \Sigma, \delta, q_0, F)$$

solving L. Then the DFA

$$M' = (Q, \Sigma, \delta, q_0, Q \backslash F)$$

solves $\overline{L}$. So $\overline{L}$ is regular.

Regular languages are closed under union

Theorem: If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are regular, then so is the union $L_1 \cup L_2$.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA solving L_1 and let $M' = (Q', \Sigma, \delta', q_0', F')$ be a DFA solving L_2.

We construct a DFA $M'' = (Q'', \Sigma, \delta'', q_0'', F'')$ solving $L_1 \cup L_2$ as follows:

$\vdots$

The Mindset:

STEP 1: Imagine yourself as a DFA.

Rules:

1) Can only scan the input once, from left to right.

2) Can only remember "constant" amount of info.

cannot change
based on input length

STEP 2: Formally define the DFA.

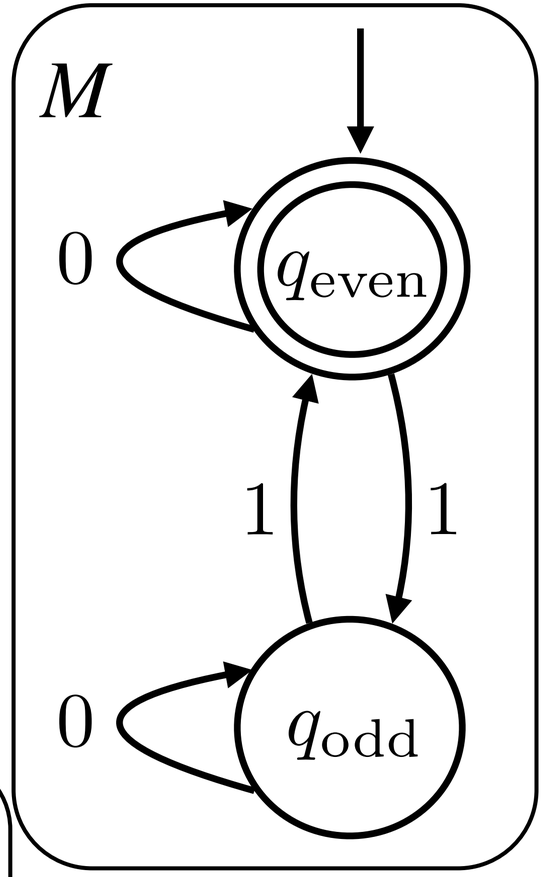
STEP 1: Imagine yourself as a DFA.

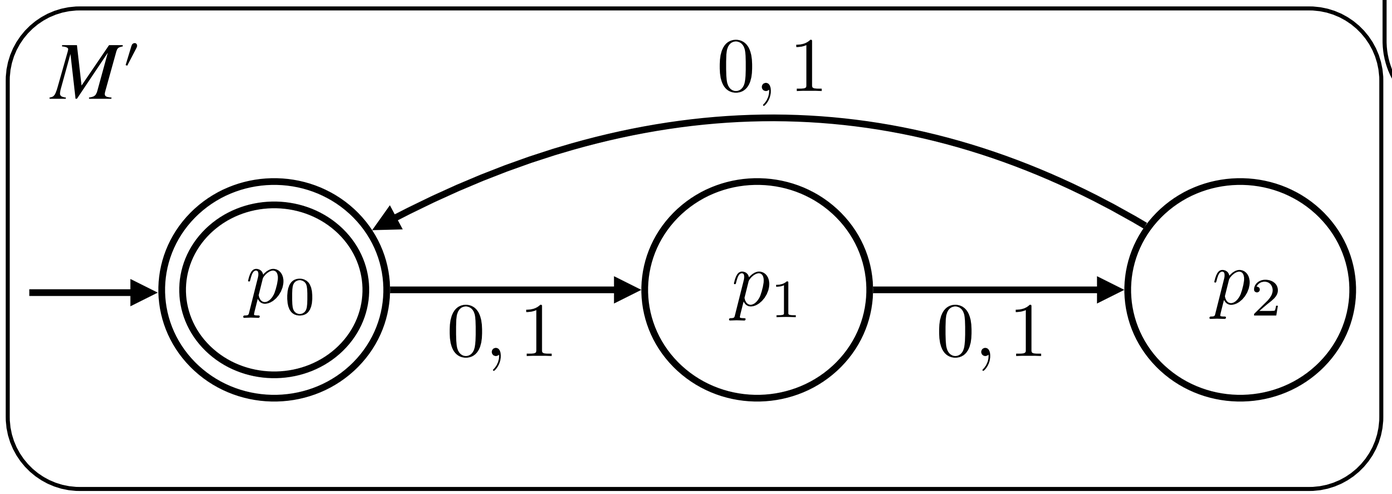
How would you solve the union of regular languages?

Example:

$L_1 =$ strings with even number of 1's

$L_2 =$ strings with length divisible by 3



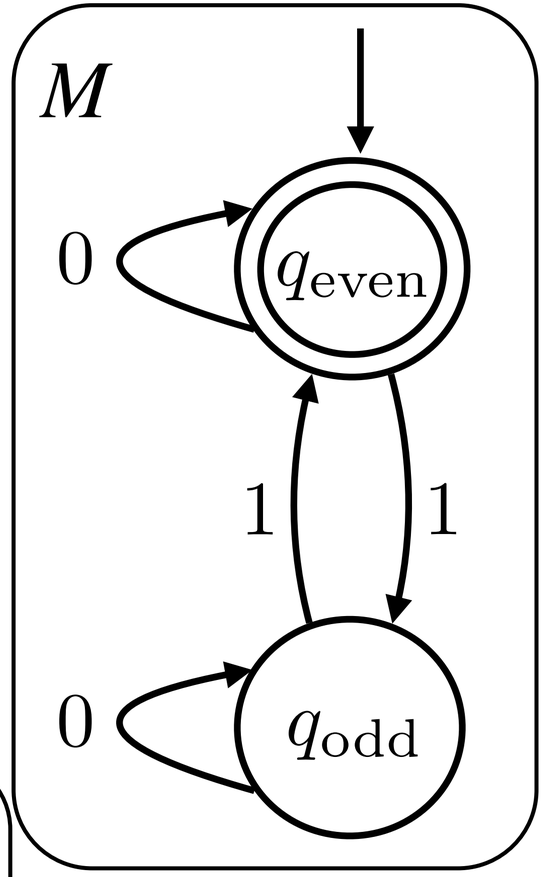


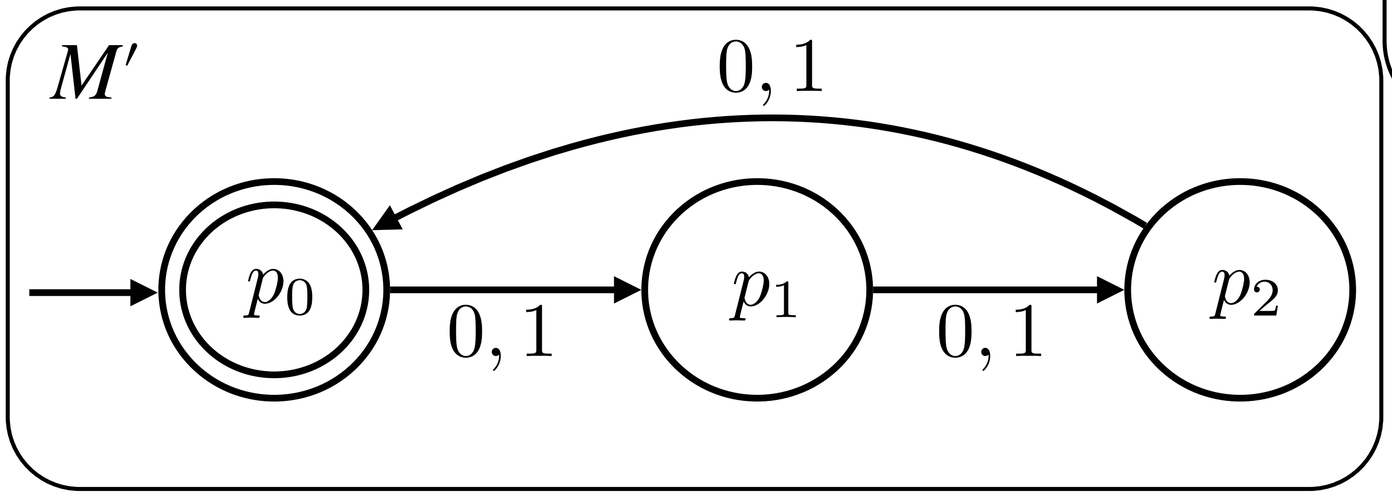
How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

Thread 1:

Thread 2:



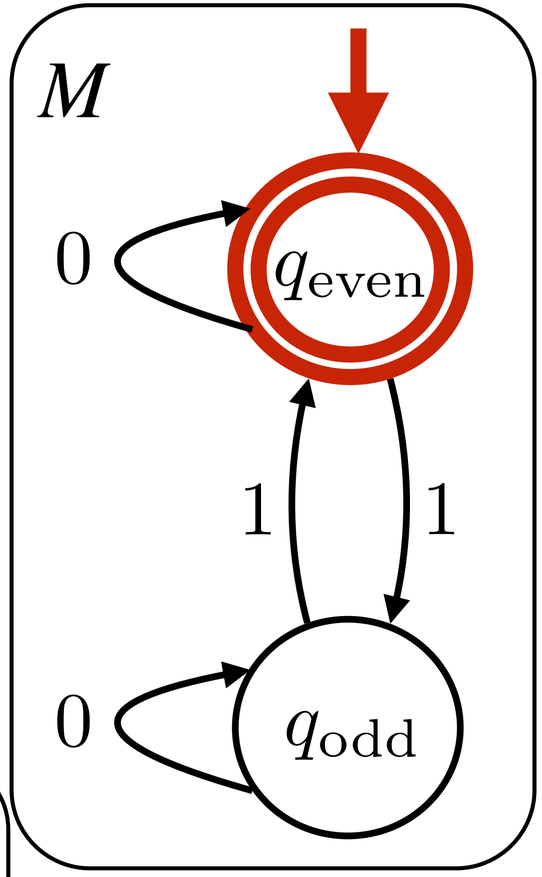


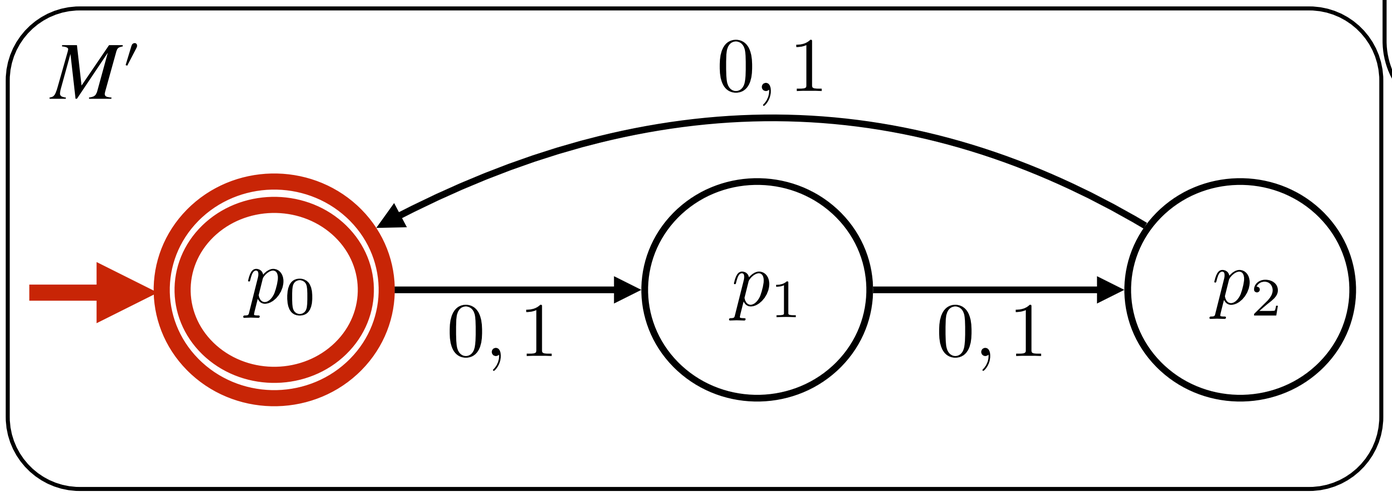
How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

Thread 1: q_{e}

Thread 2: p_0



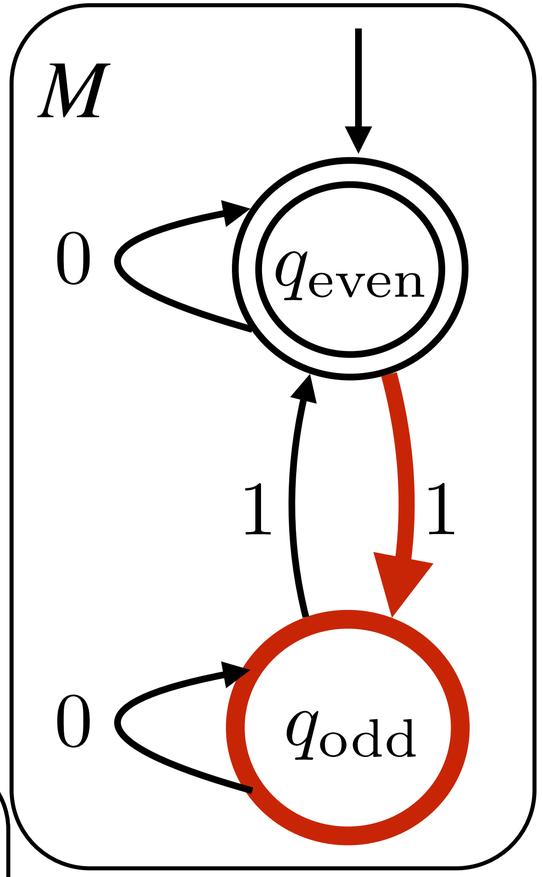


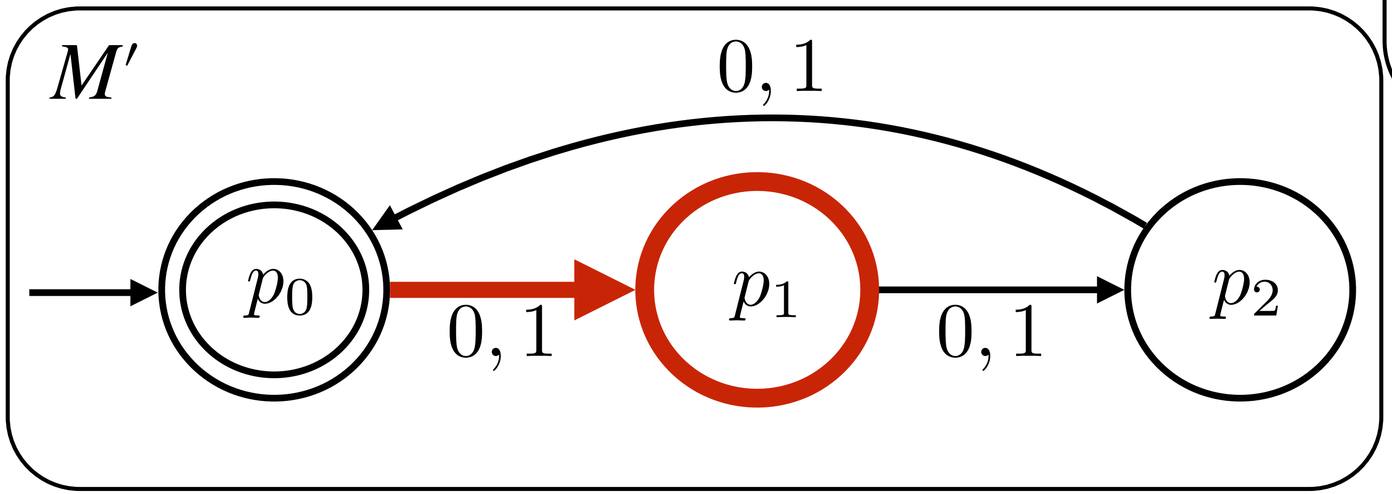
How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

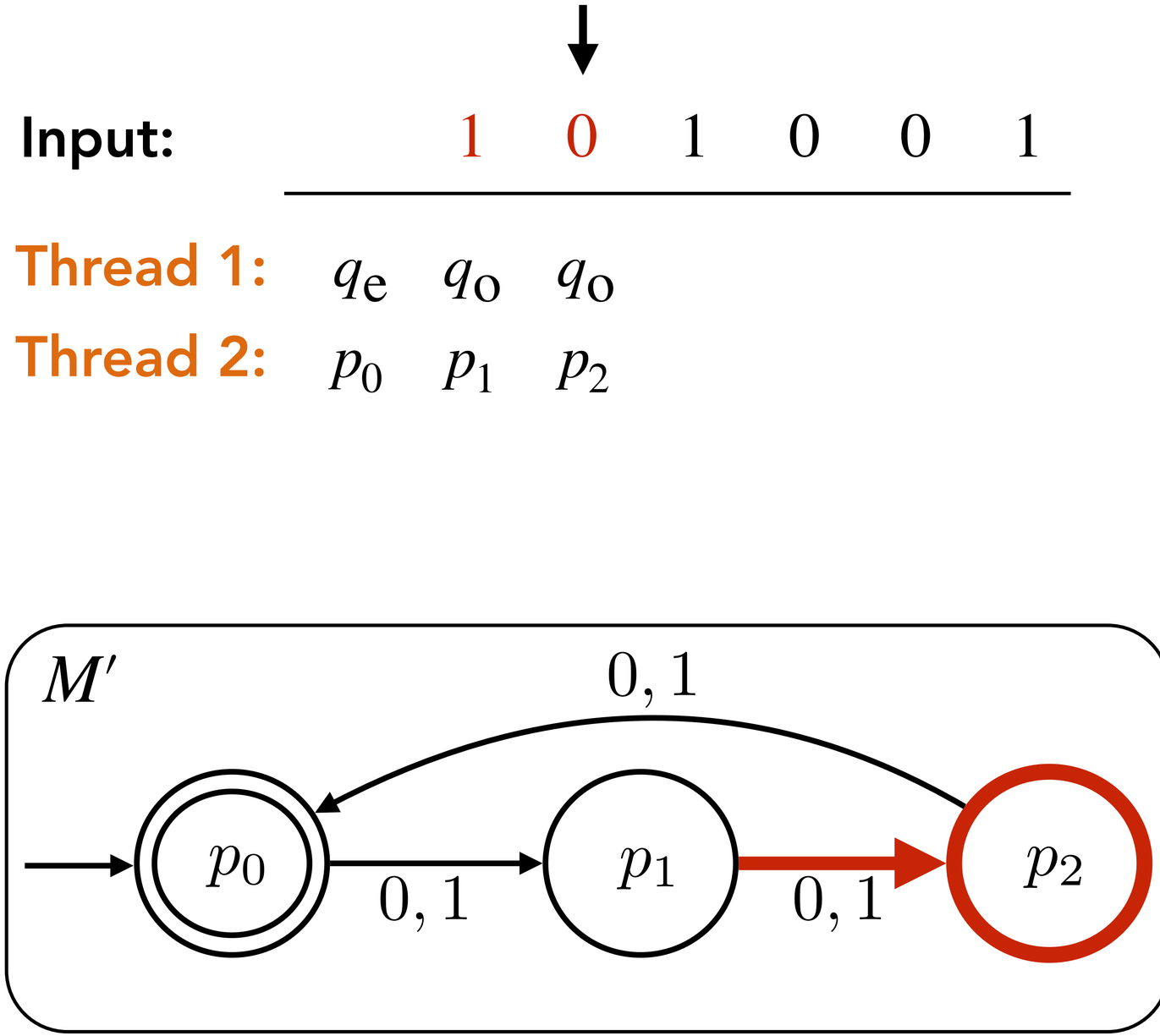
Thread 1: q_e q_o

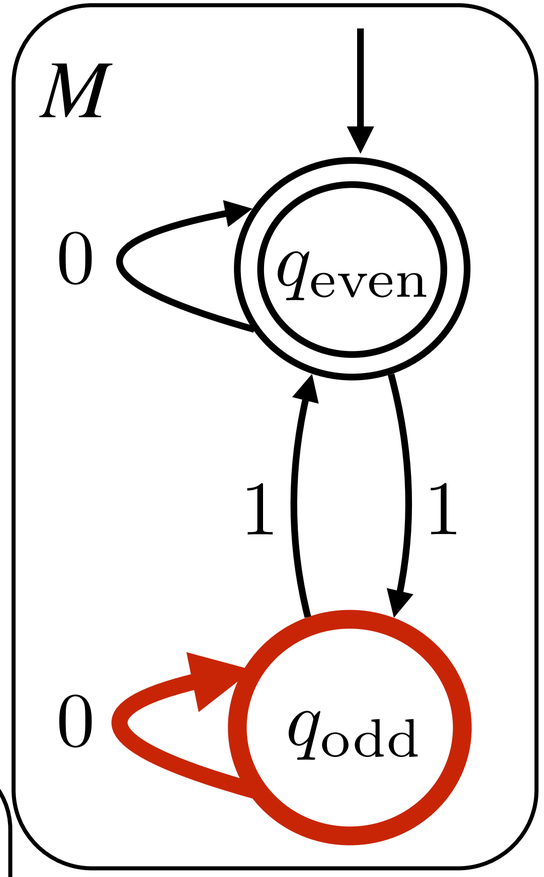
Thread 2: p_0 p_1





How would you solve the union of regular languages?



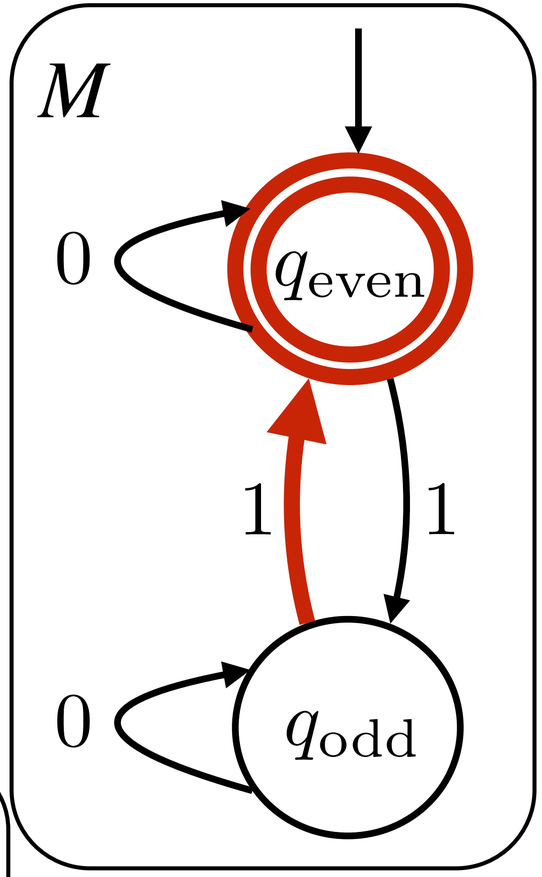


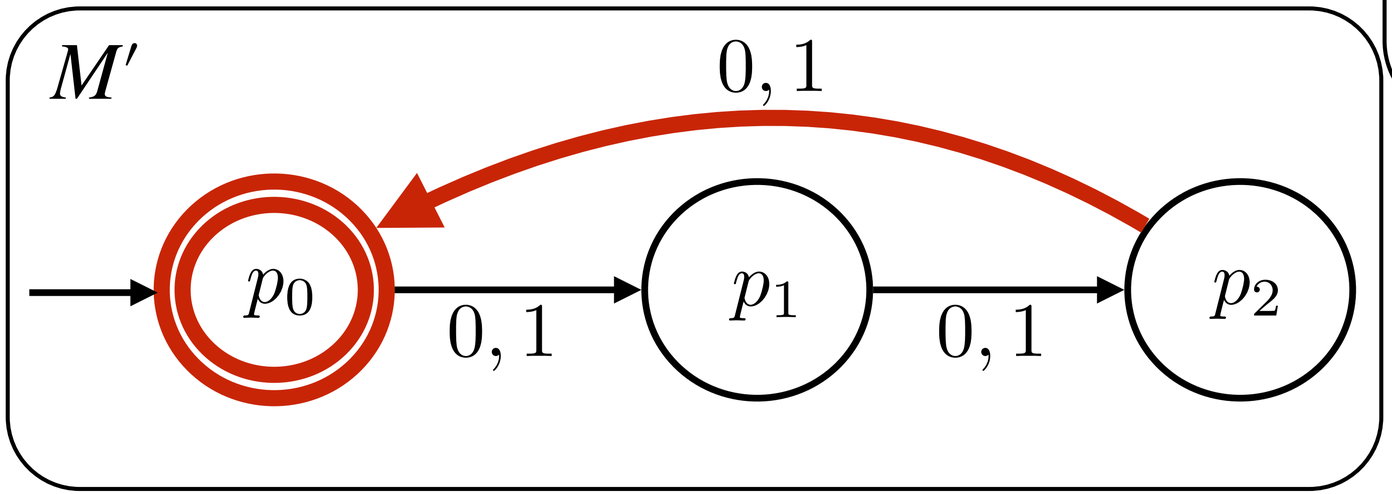
How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

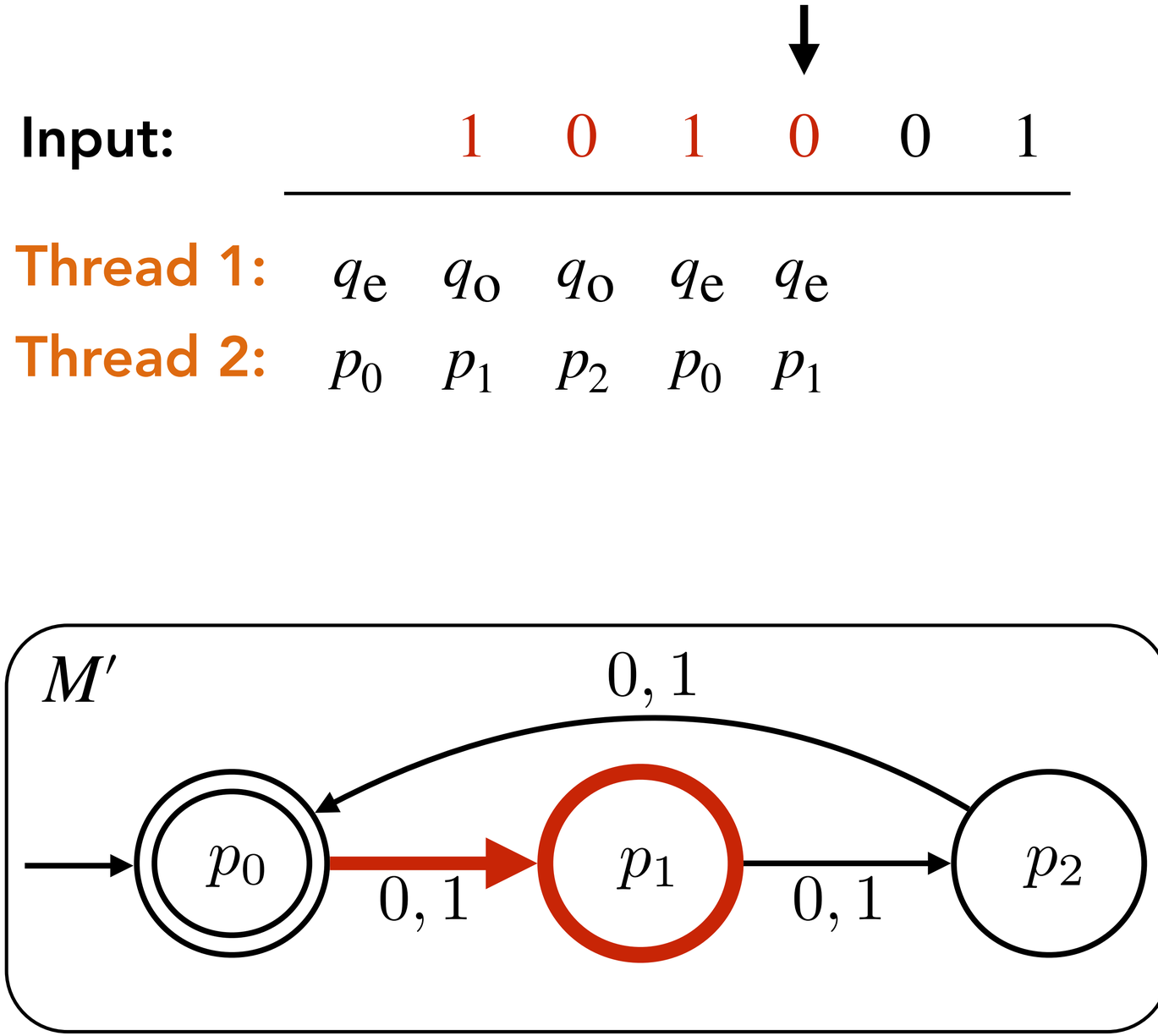
Thread 1: q_e q_o q_o q_e

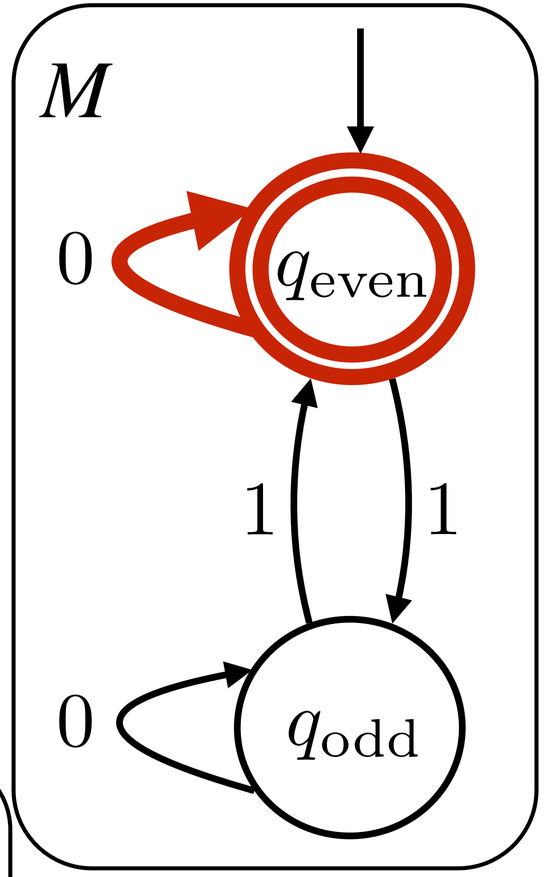
Thread 2: p_0 p_1 p_2 p_0





How would you solve the union of regular languages?

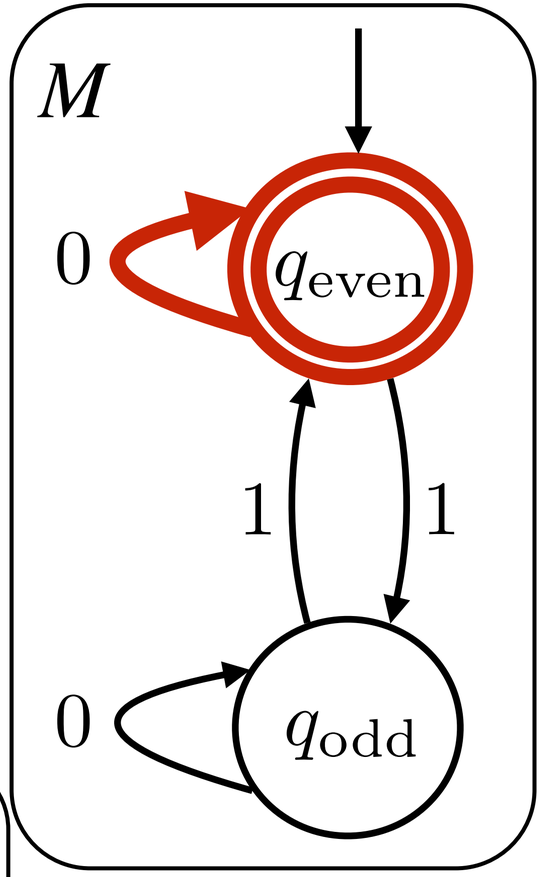


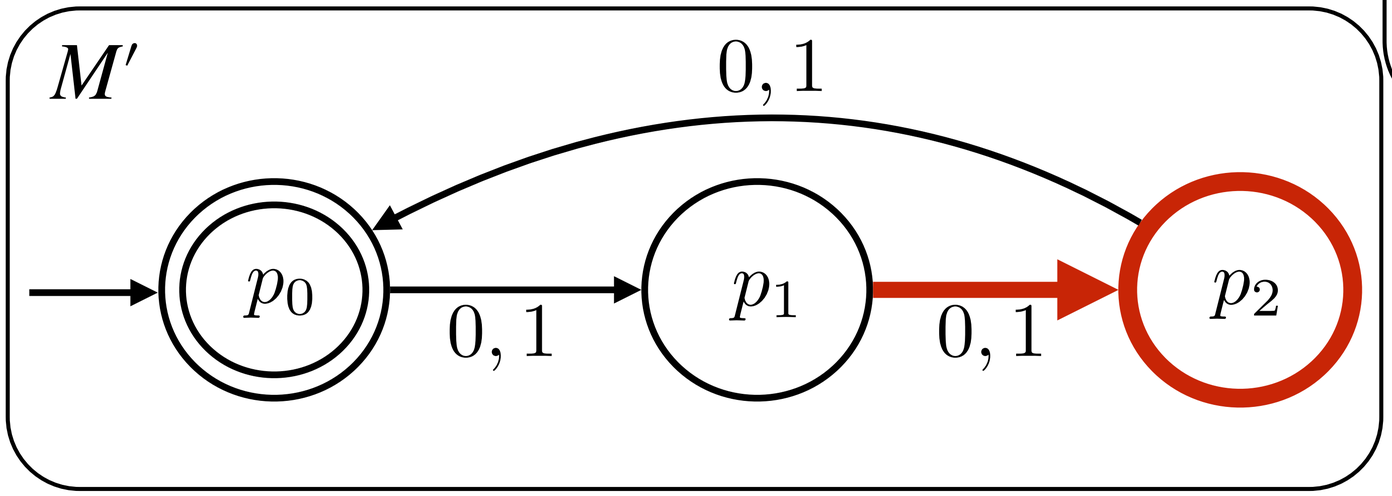


How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

Thread 1:	q_e	q_o	q_o	q_e	q_e	q_e
Thread 2:	p_0	p_1	p_2	p_0	p_1	p_2

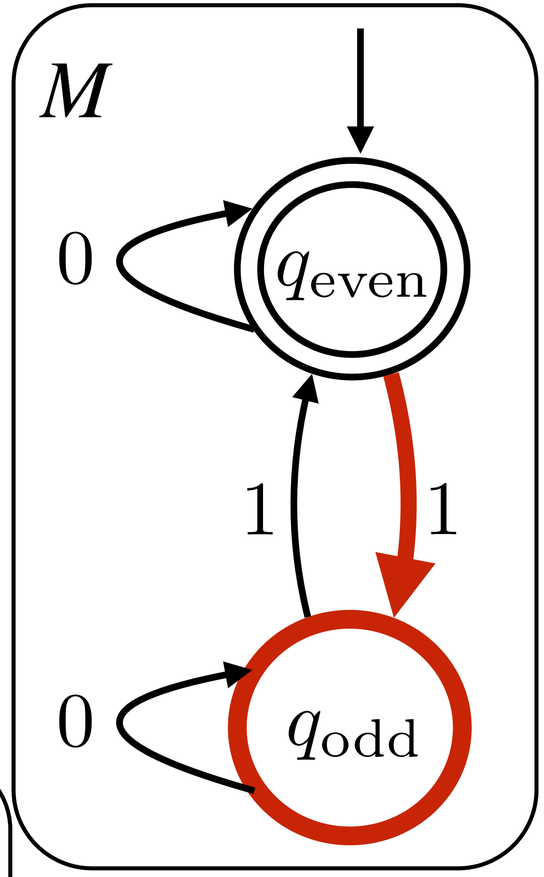


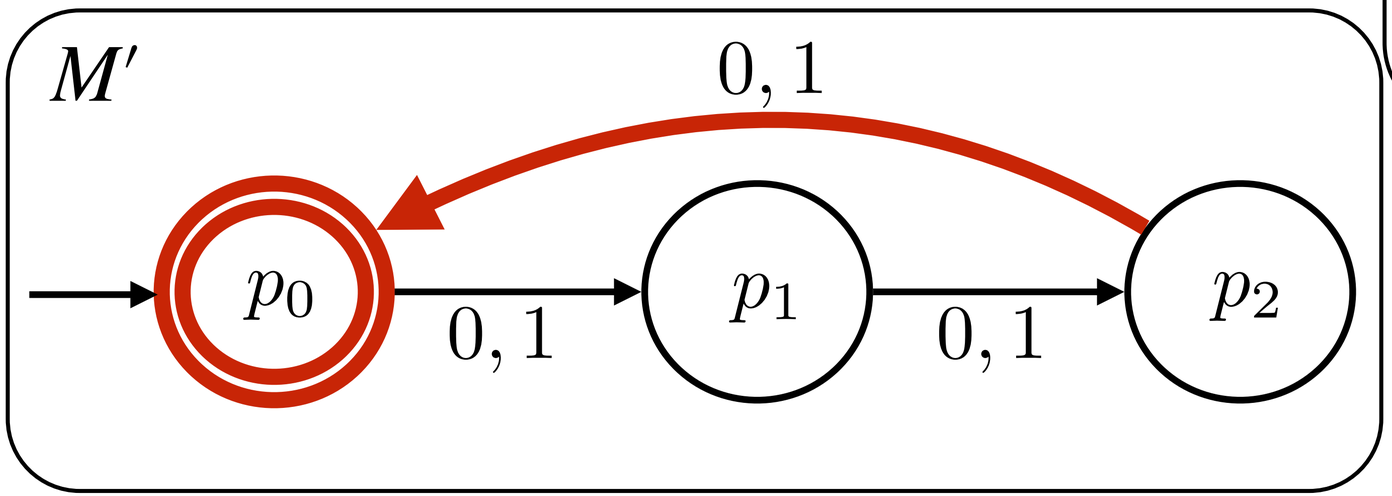


How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

Thread 1:	q_e	q_o	q_o	q_e	q_e	q_e	q_o
Thread 2:	p_0	p_1	p_2	p_0	p_1	p_2	p_0



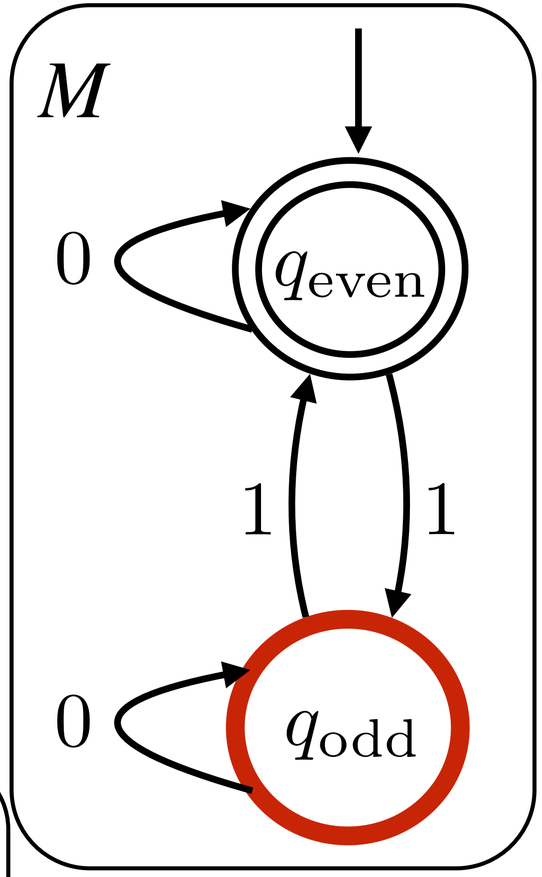


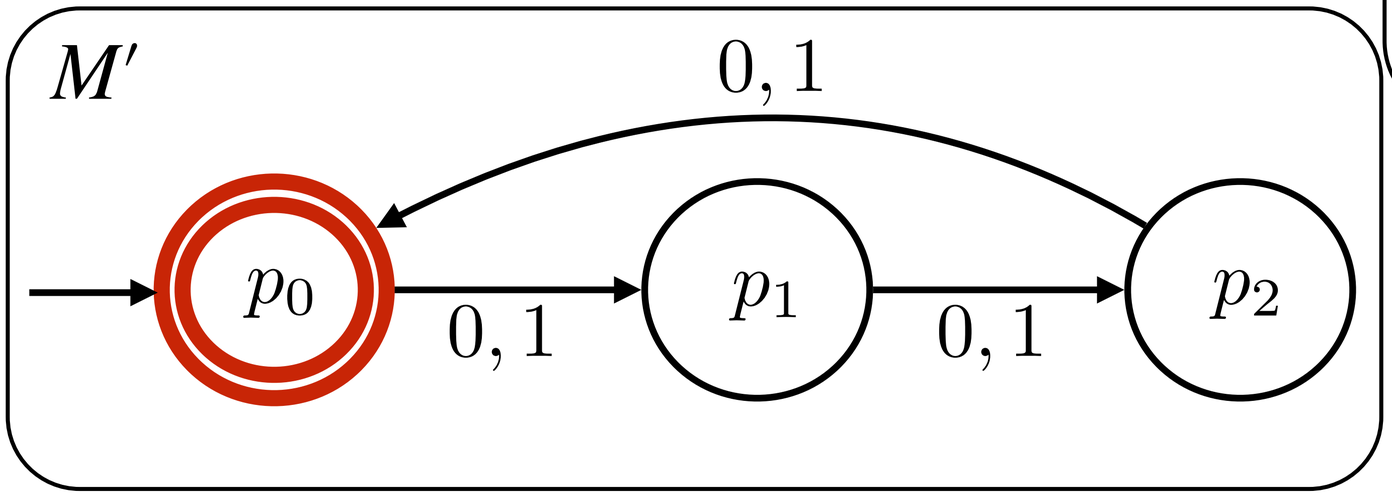
How would you solve the union of regular languages?

Input: 1 0 1 0 0 1

Thread 1:	q_e	q_o	q_o	q_e	q_e	q_e	q_o
Thread 2:	p_0	p_1	p_2	p_0	p_1	p_2	(p_0)

Decision: Accept





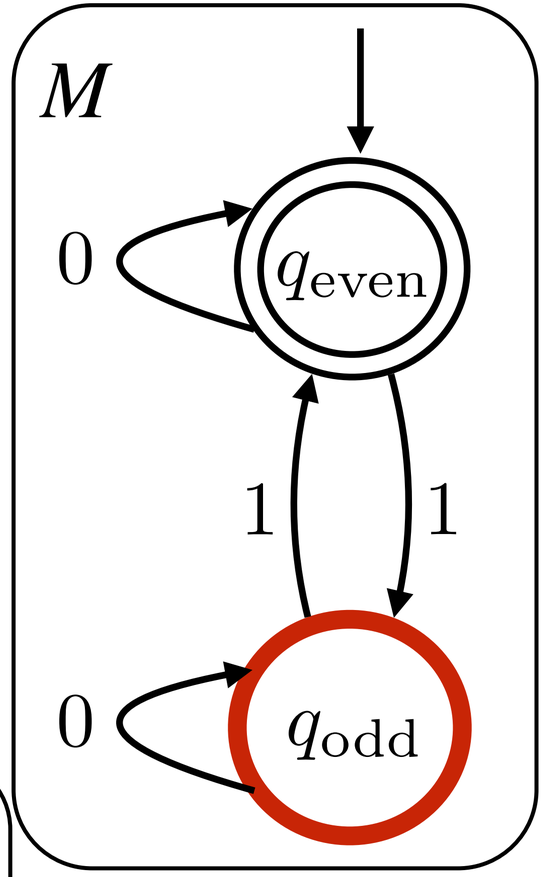
How would you solve the union of regular languages?

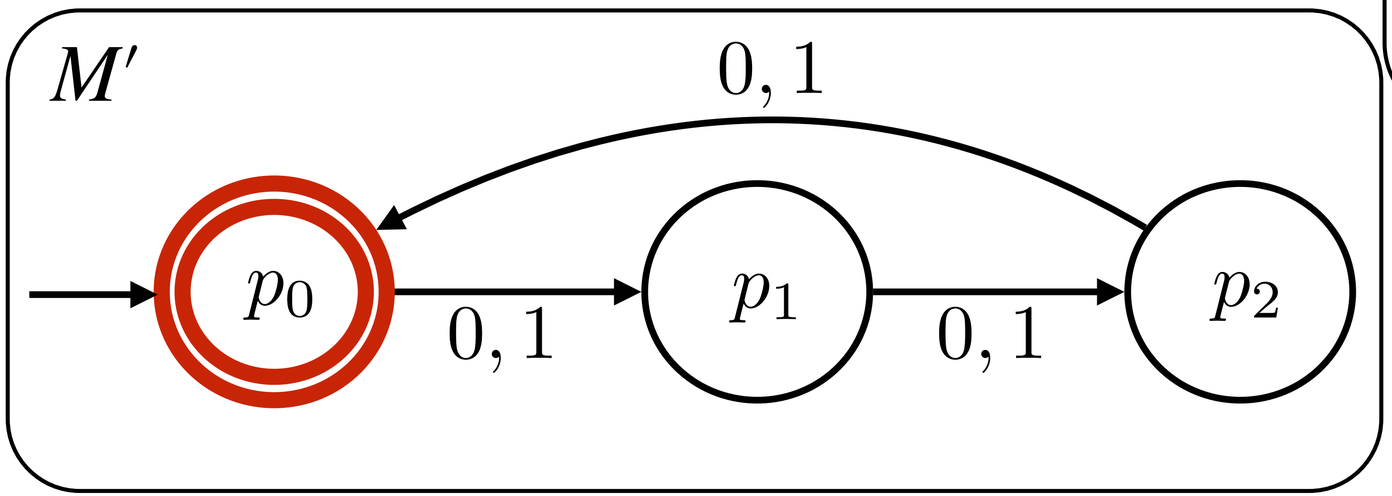
Input: 1 0 1 0 0 1

Thread 1: q_e q_o q_o q_e q_e q_e q_o

Thread 2: p_0 p_1 p_2 p_0 p_1 p_2 (p_0)

1. Are we scanning the input only once?

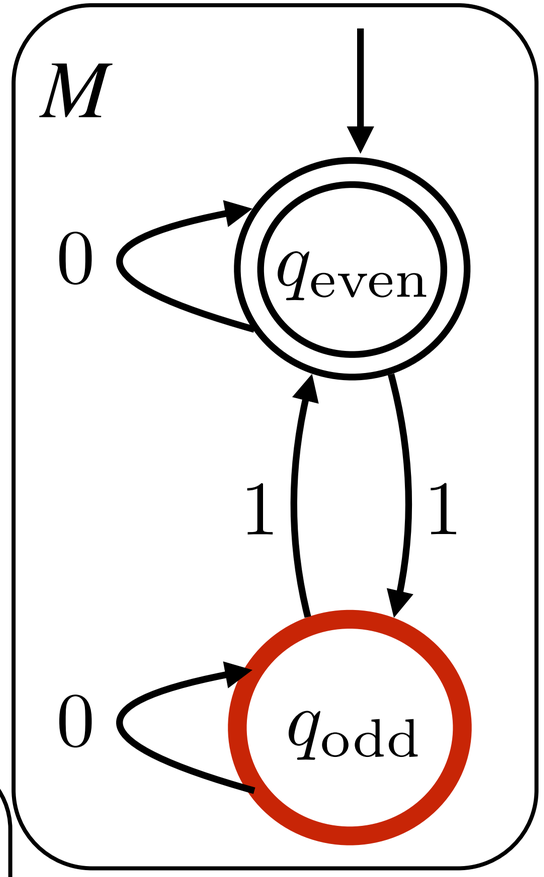


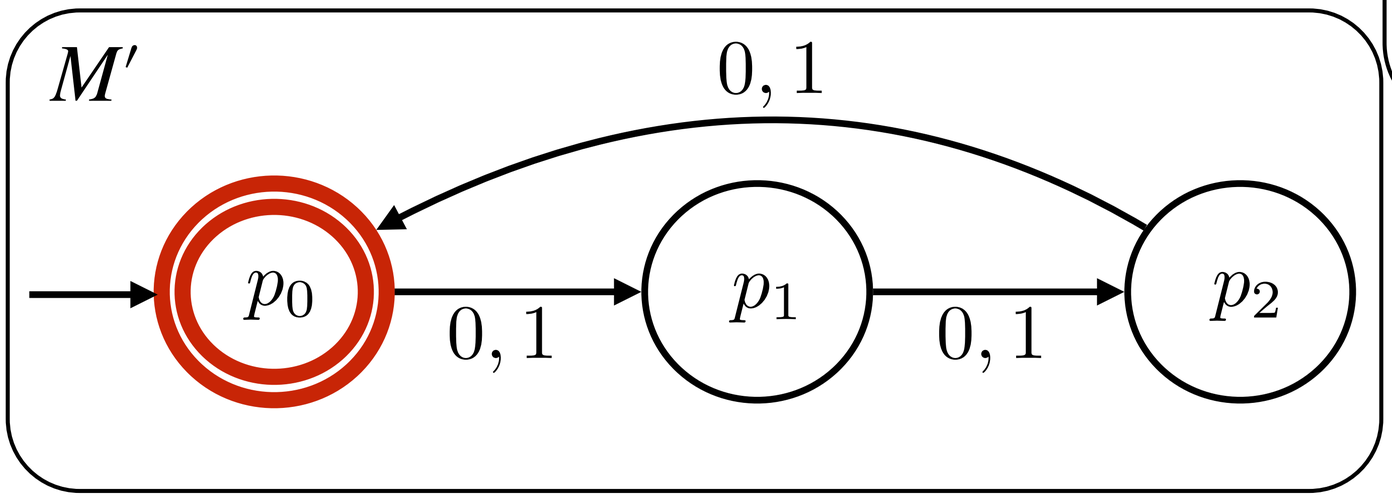


How would you solve the union of regular languages?

Input:		1	0	1	0	0	1
Thread 1:	q_e	q_o	q_o	q_e	q_e	q_e	q_o
Thread 2:	p_0	p_1	p_2	p_0	p_1	p_2	(p_0)

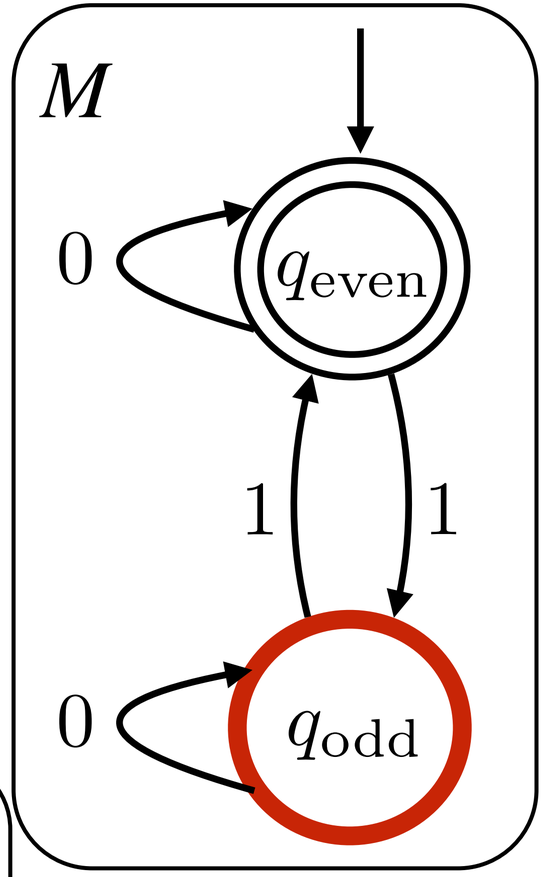
2. Do we need to remember more than constant amount of information?

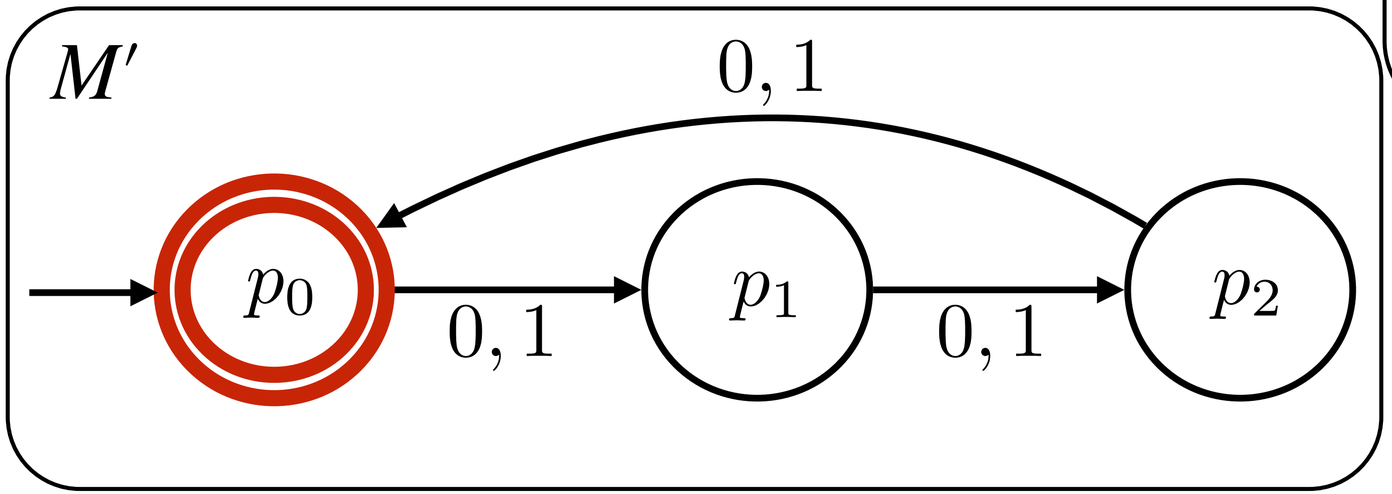




How would you solve the union of regular languages?

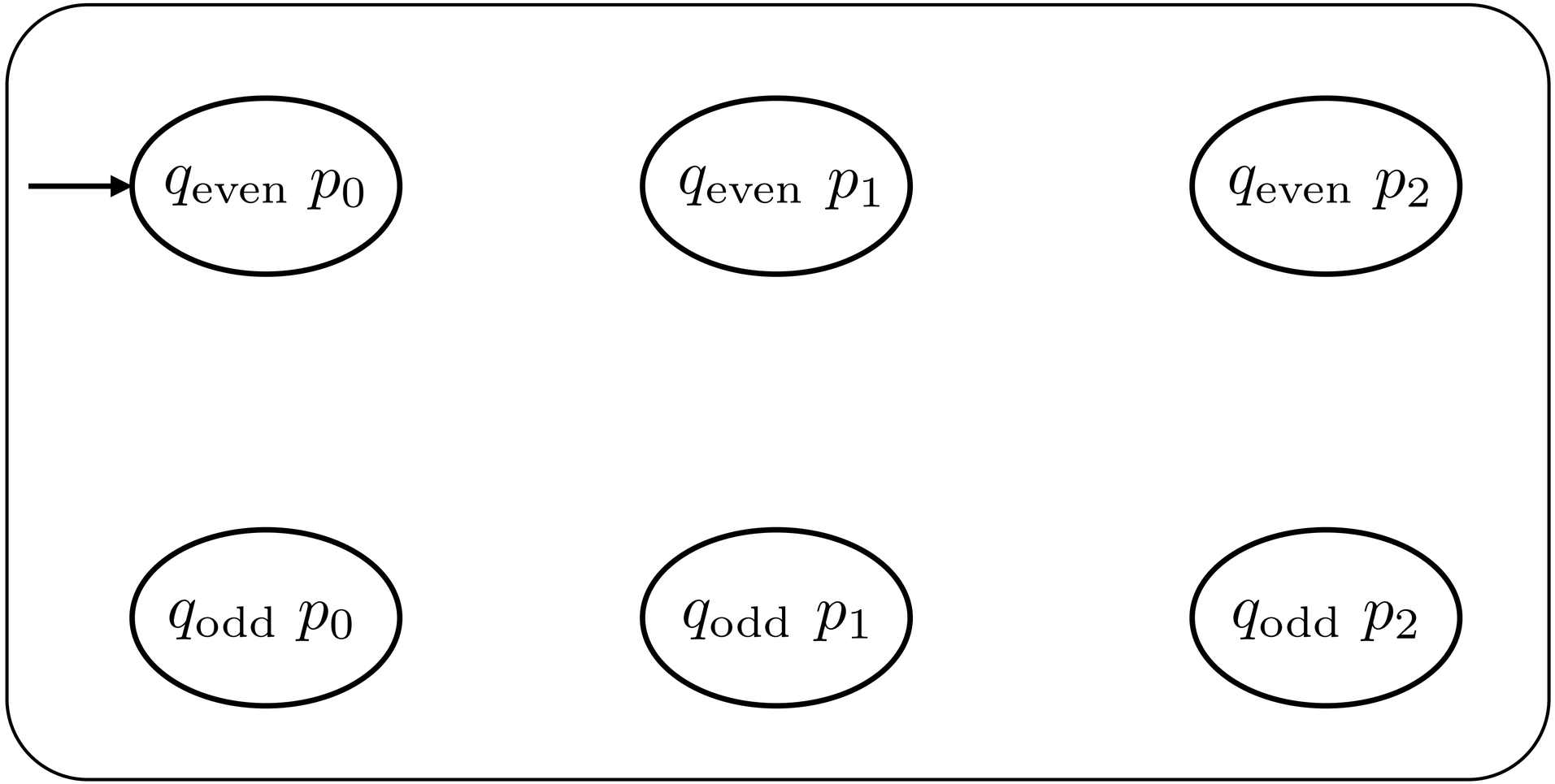
Main Idea: Construct a single DFA that keeps track of both M and M' at once.





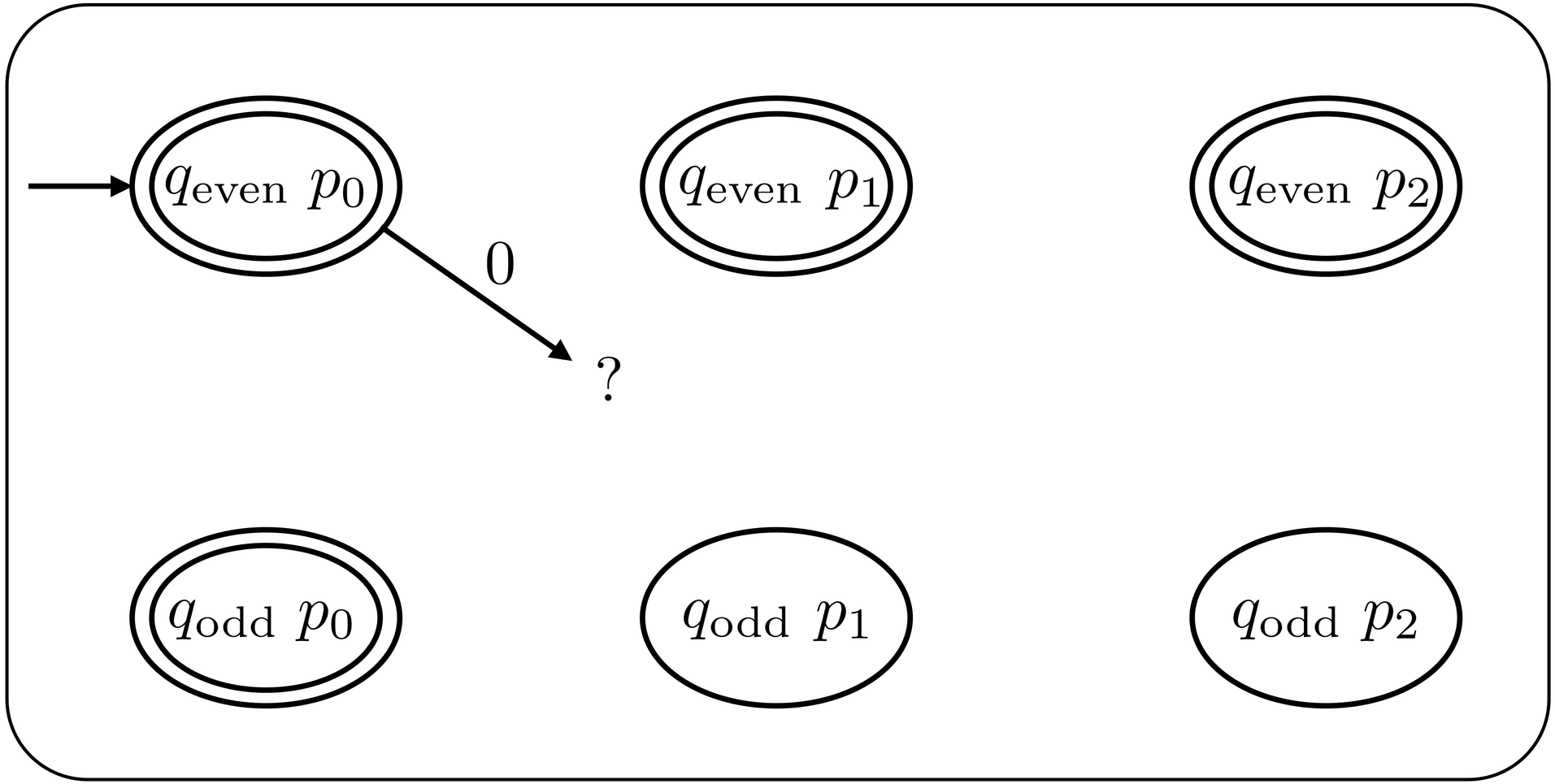
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



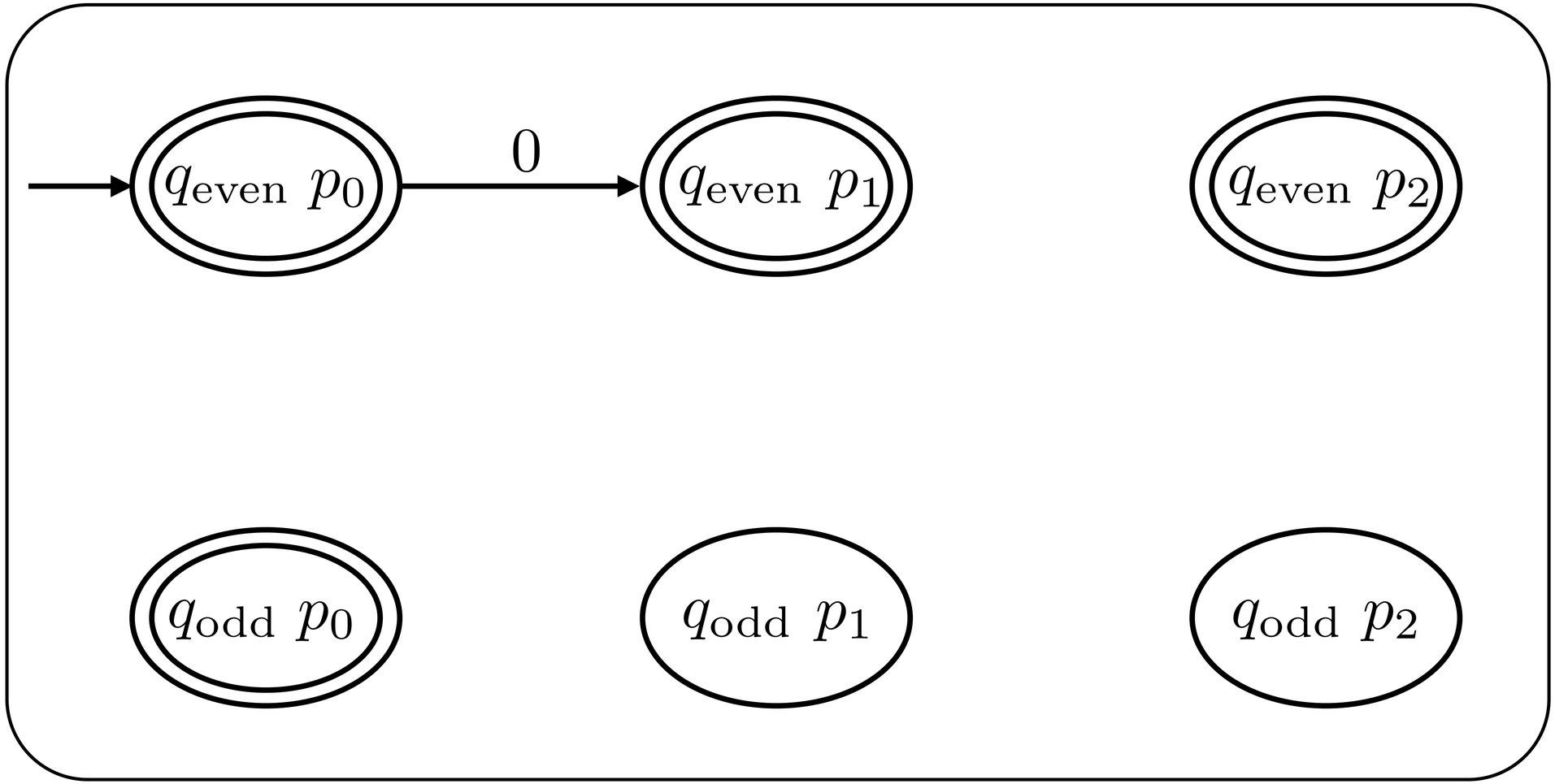
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



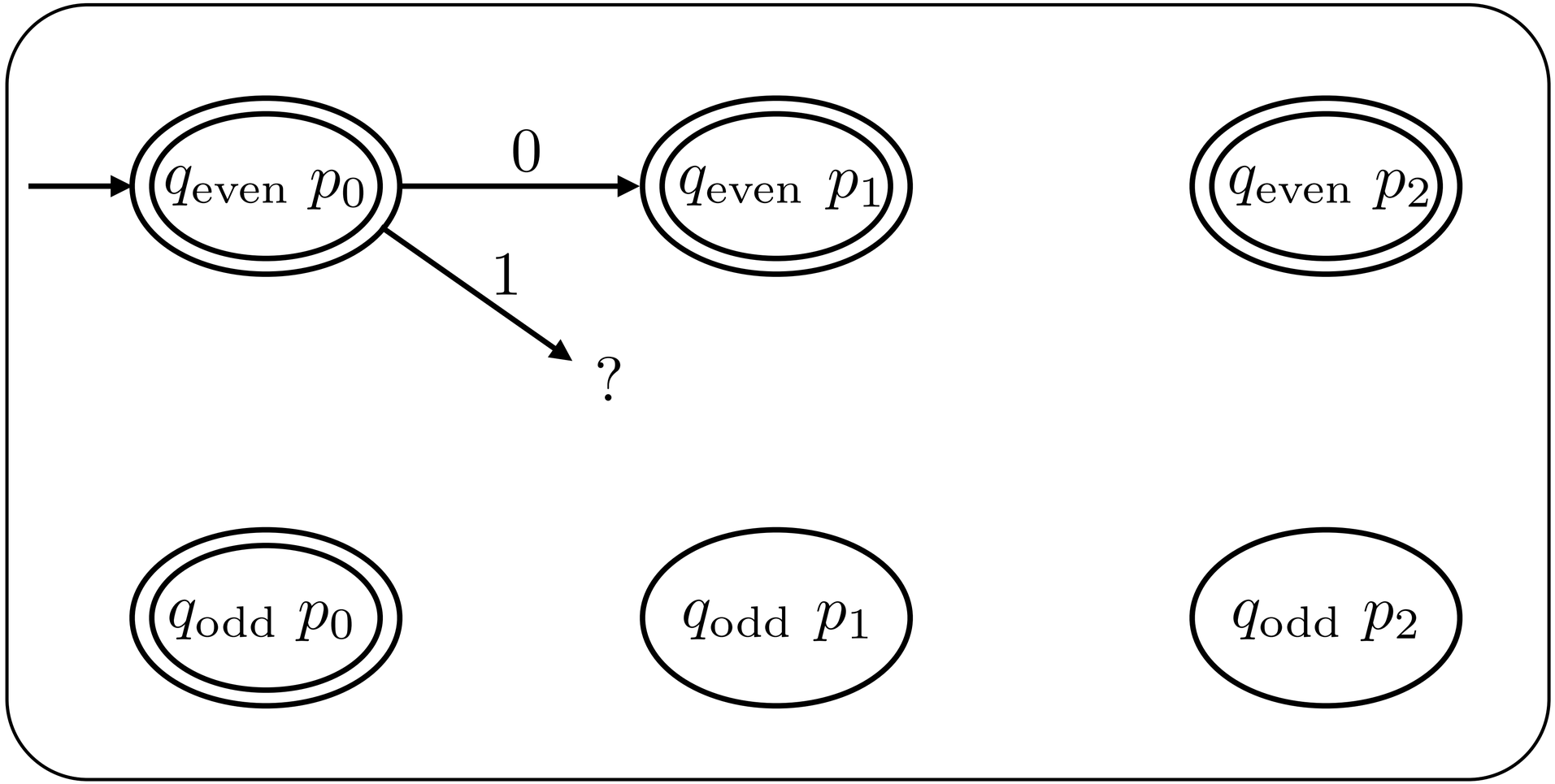
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



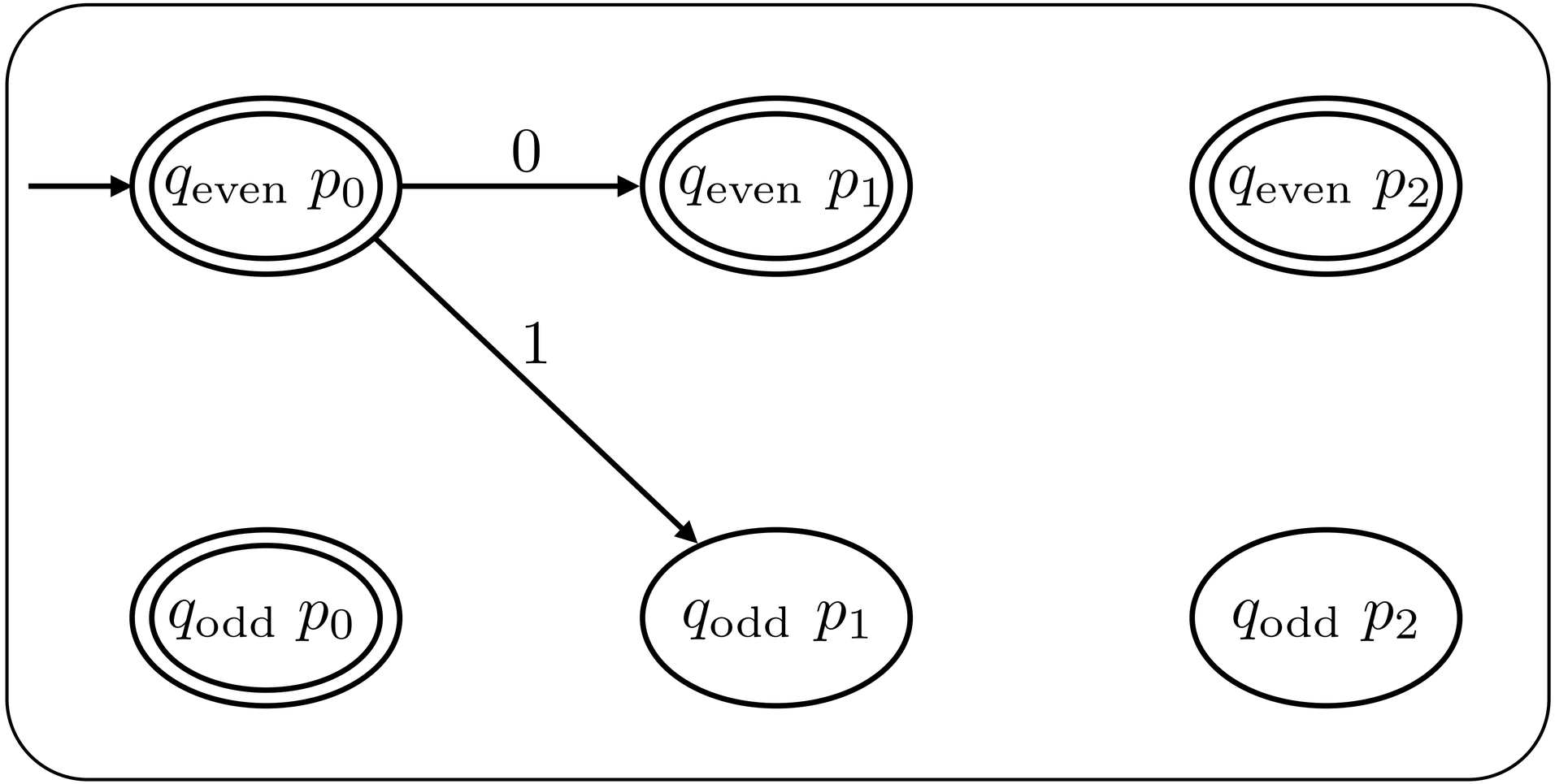
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



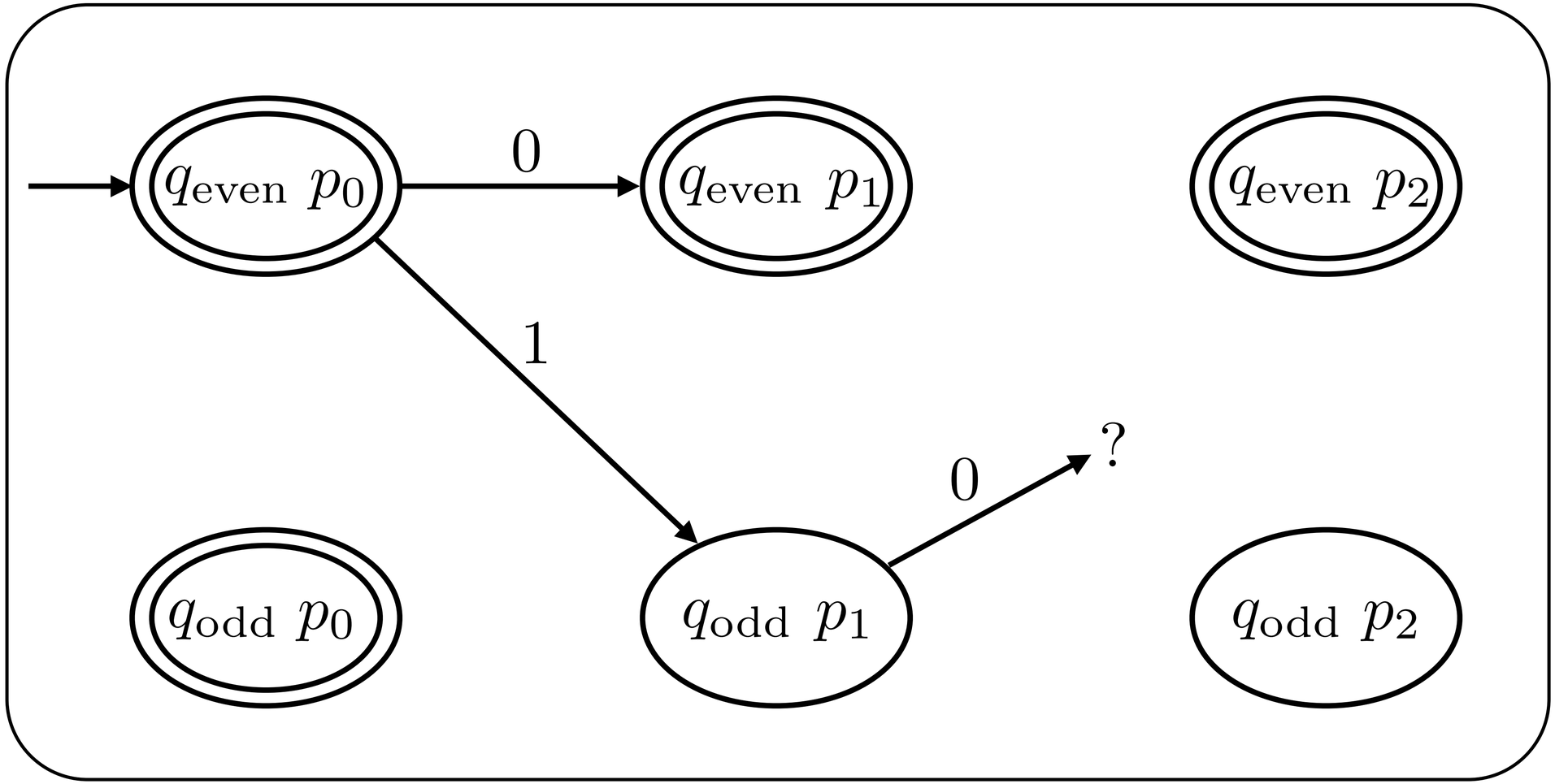
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



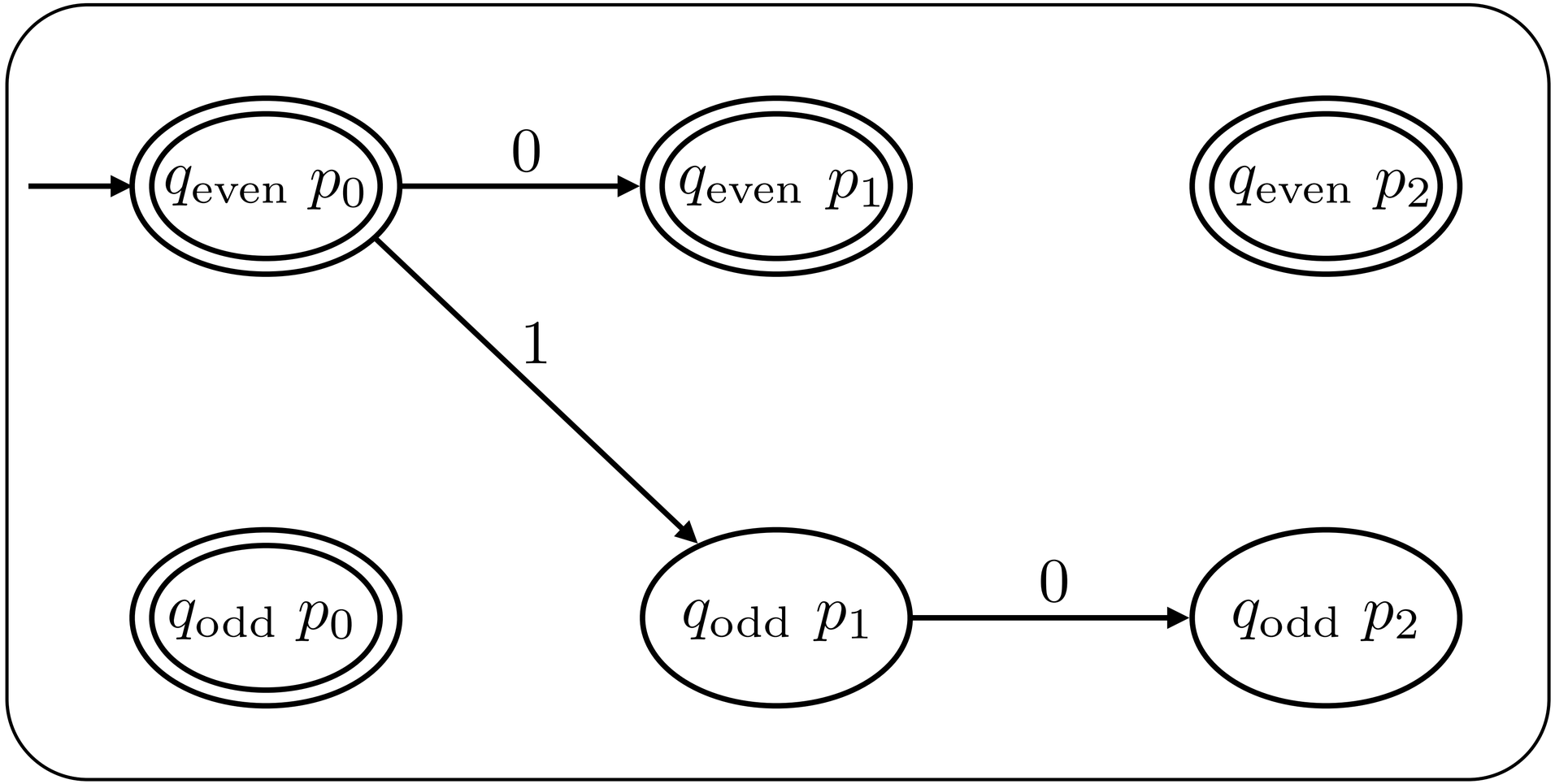
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



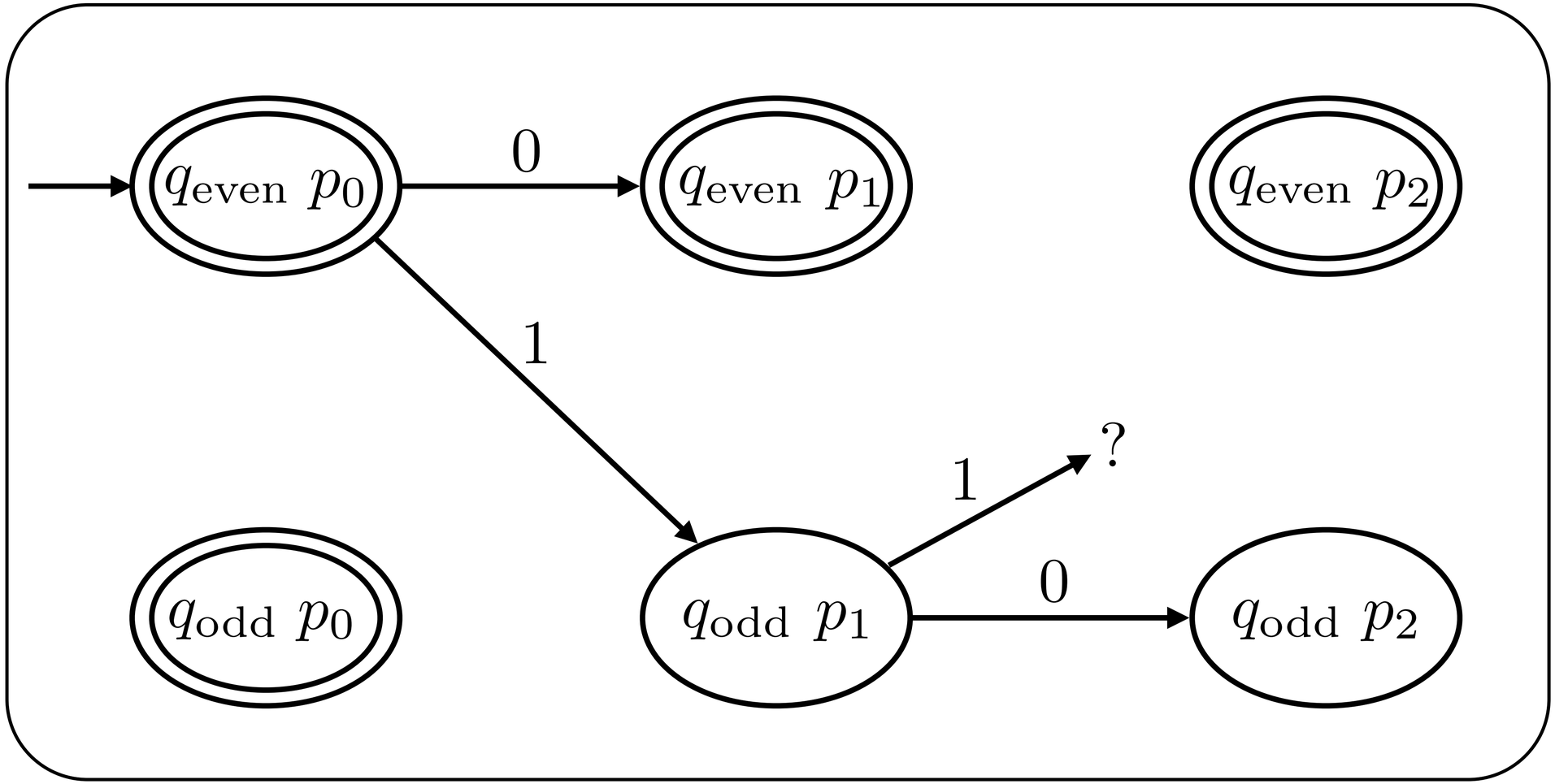
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



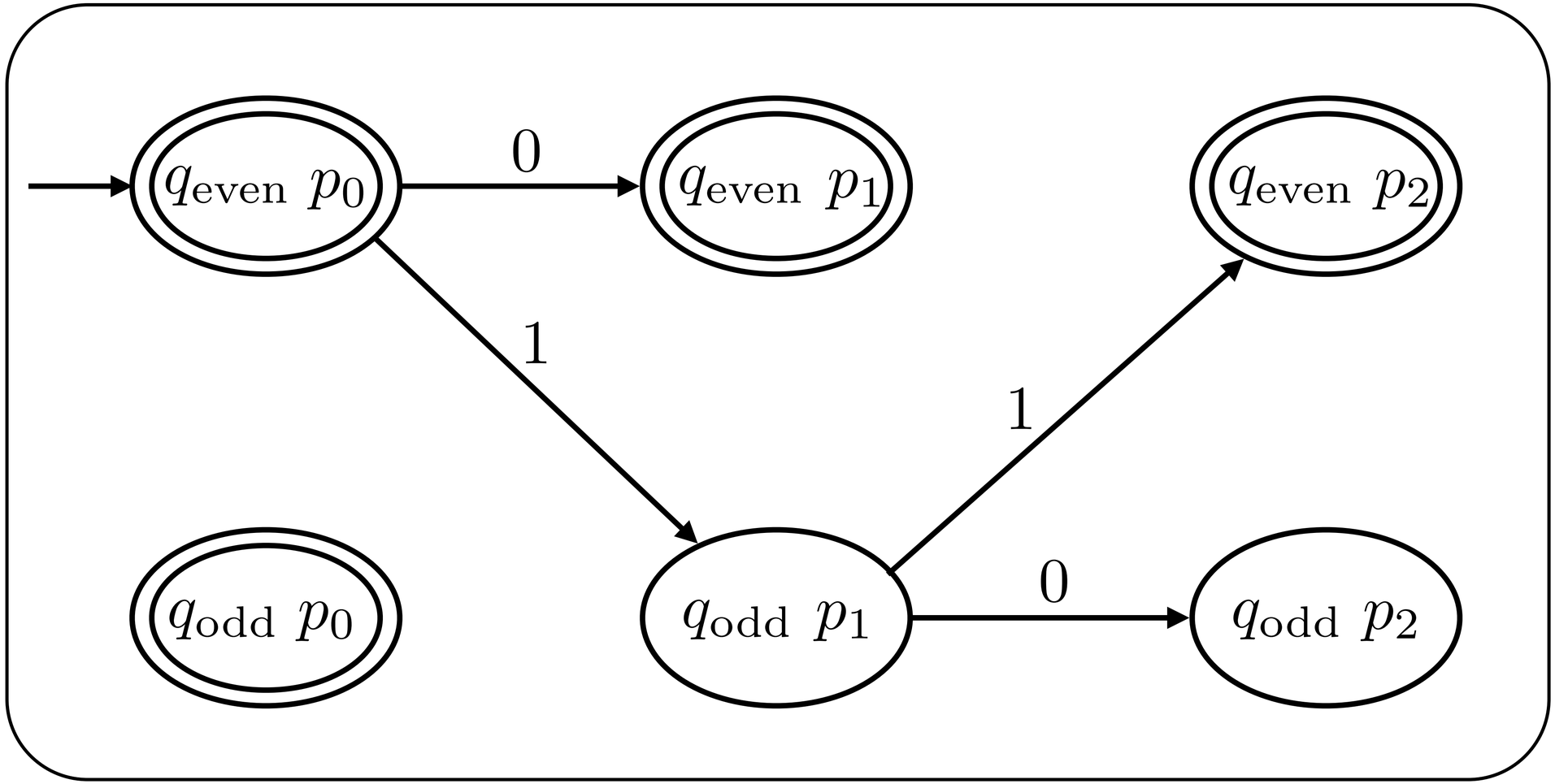
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



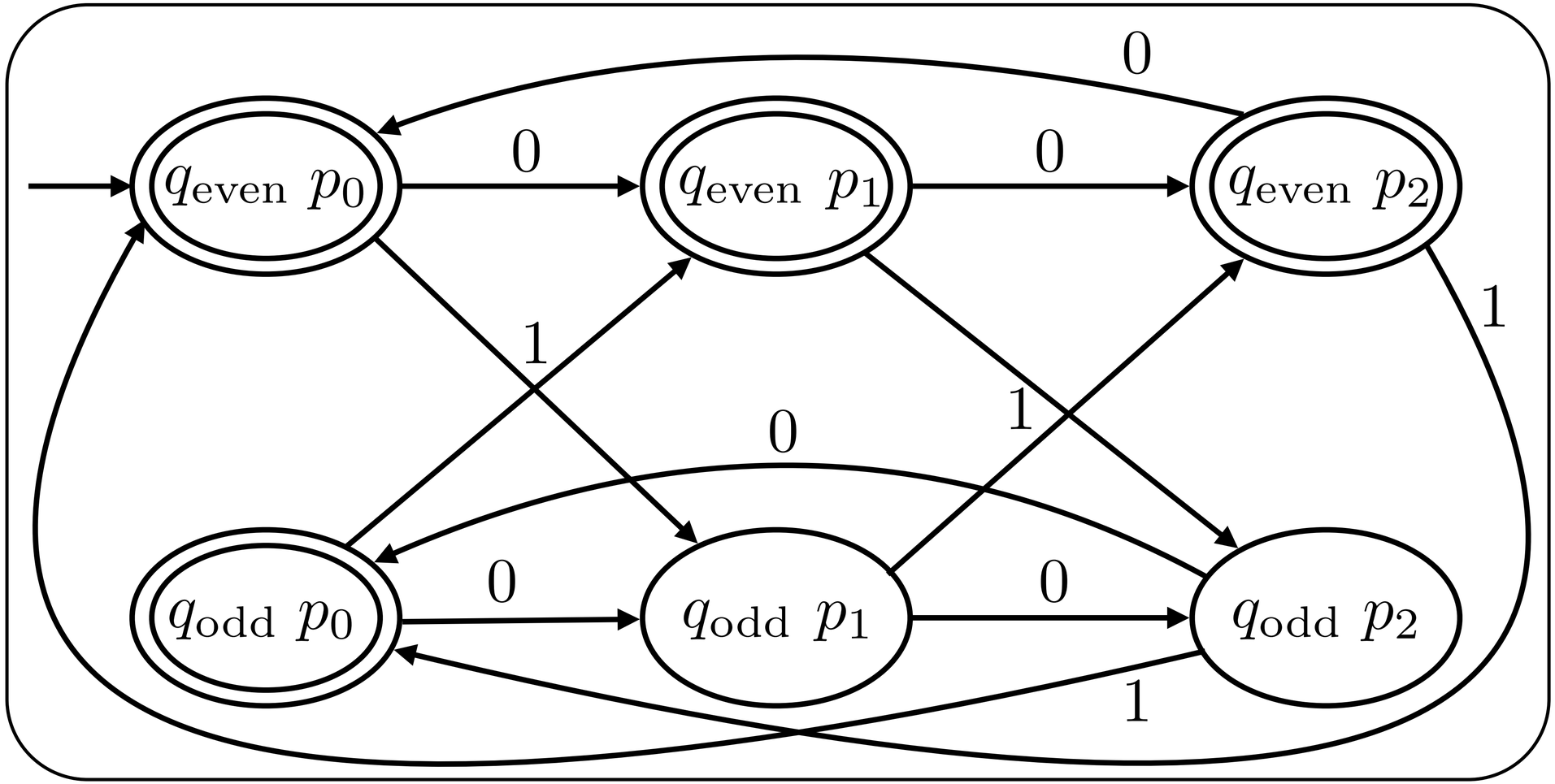
How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



How would you solve the union of regular languages?

Main Idea: Construct a single DFA that keeps track of both M and M' at once.



STEP 2: Formally define the DFA.

Formally defining the union construction

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA solving L_1.

Let $M' = (Q', \Sigma, \delta', q_0', F')$ be a DFA solving L_2.

We construct a DFA $M'' = (Q'', \Sigma, \delta'', q_0'', F'')$ solving $L_1 \cup L_2$:

- $Q'' = Q \times Q' = \{(q, q') : q \in Q, q' \in Q'\}$
- $\delta'' : Q'' \times \Sigma \to Q'', \qquad \delta'' : (Q \times Q') \times \Sigma \to (Q \times Q')$

  for $q \in Q,\ q' \in Q',\ \sigma \in \Sigma :\quad \delta''((q, q'), \sigma) = (\delta(q, \sigma), \delta'(q', \sigma))$
- $q_0'' = (q_0, q_0')$
- $F'' = \{(q, q') : q \in F \text{ or } q' \in F'\}$

It remains to show that $L(M'') = L_1 \cup L_2$.

$L(M'') \subseteq L_1 \cup L_2$: ... $L_1 \cup L_2 \subseteq L(M'')$: ... ■

Regular languages are closed under intersection

Corollary: If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are regular, then so is the intersection $L_1 \cap L_2$.

Proof: Follows from the following 3 facts.

- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

- Regular languages are closed under *complementation*.

- Regular languages are closed under *union*.

Regular languages are closed under intersection

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA solving L_1.

Let $M' = (Q', \Sigma, \delta', q_0', F')$ be a DFA solving L_2.

We construct a DFA $M'' = (Q'', \Sigma, \delta'', q_0'', F'')$ solving $L_1 \cup L_2$:

- $Q'' = Q \times Q' = \{(q, q') : q \in Q, q' \in Q'\}$
- $\delta'' : Q'' \times \Sigma \to Q'', \qquad \delta'' : (Q \times Q') \times \Sigma \to (Q \times Q')$

  for $q \in Q,\ q' \in Q',\ \sigma \in \Sigma :\quad \delta''((q, q'), \sigma) = (\delta(q, \sigma), \delta'(q', \sigma))$
- $q_0'' = (q_0, q_0')$
- $F'' = \{(q, q') : q \in F$ ~~or~~ $q' \in F'\}$

AND

More closure properties

Closed under union:

L_1, L_2 regular $\implies$ $L_1 \cup L_2$ regular.

Closed under concatenation:

L_1, L_2 regular $\implies$ $L_1 L_2$ regular.

Closed under star:

L regular $\implies$ L^* regular.

Simple languages vs regular languages

What is the relationship between
simple languages and **regular** languages?

simple $\subseteq$ regular

In fact:

simple $=$ regular

Simple languages vs regular languages

Theorem: Can define **regular languages** recursively as follows:

- $\emptyset$ is regular.
- For every $a \in \Sigma$, $\{a\}$ is regular.
- L_1, L_2 regular $\Longrightarrow L_1 \cup L_2$ regular.
- L_1, L_2 regular $\Longrightarrow L_1 \cdot L_2$ regular.
- L regular $\Longrightarrow L^*$ regular.

Closed under concatenation

Theorem: If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are regular, then so is the concatenation $L_1 L_2$.

$$L_1 L_2 = \{uv : u \in L_1, v \in L_2\}$$

$w \in L_1 L_2$ iff there is an index i such that

$w_1 \ldots w_i \in L_1$ and $w_{i+1} \ldots w_n \in L_2$.

The Mindset:

STEP 1: Imagine yourself as a DFA.

Rules:

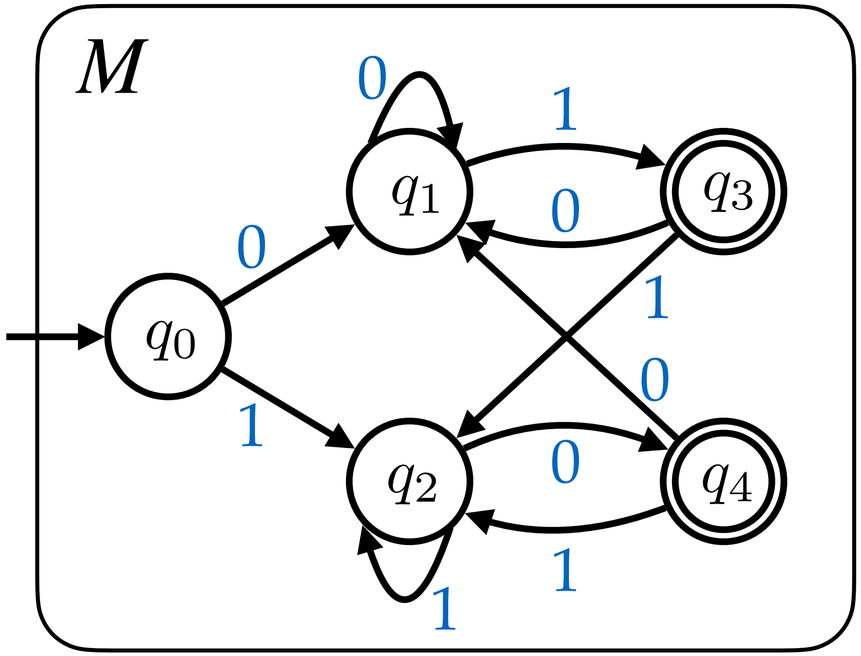
1) Can only scan the input once, from left to right.

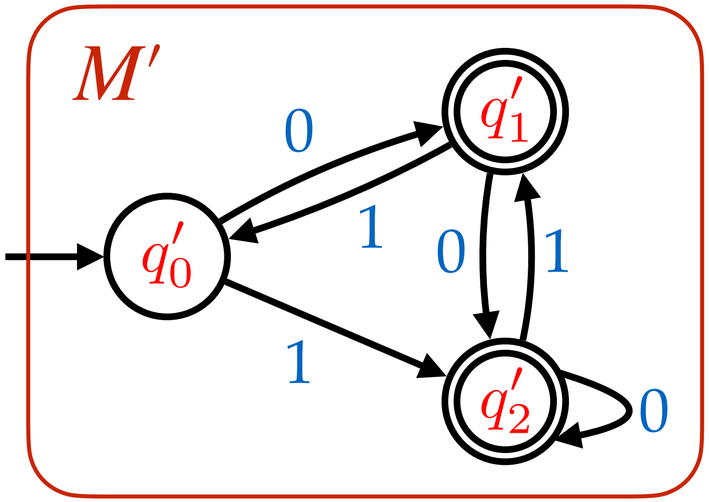
2) Can only remember "constant" amount of info.

cannot change
based on input length

STEP 2: Formally define the DFA.

STEP 1: Imagine yourself as a DFA.



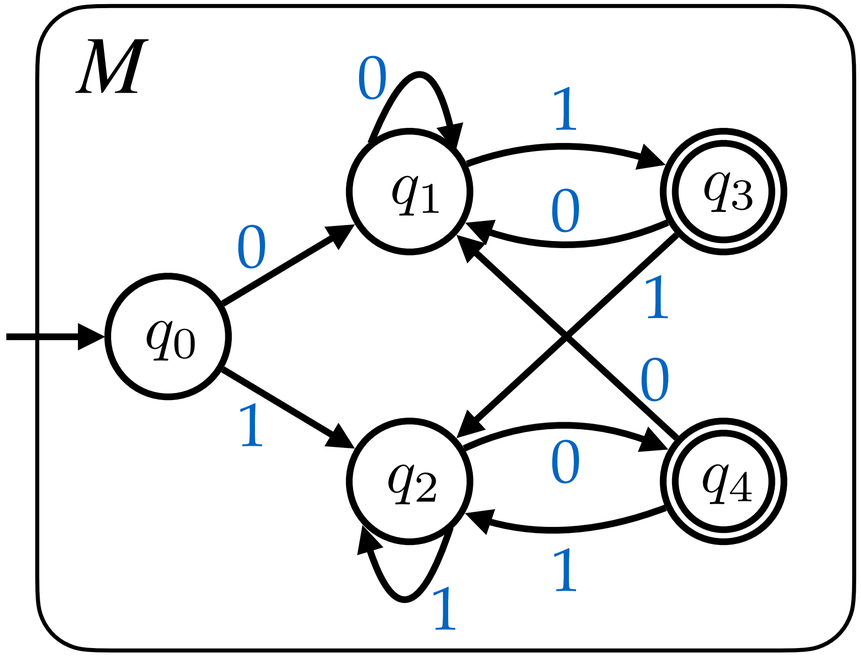


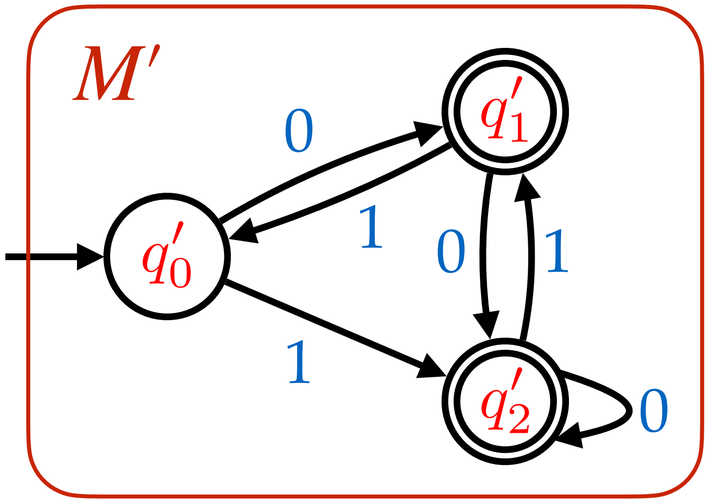
Given $w \in \Sigma^*$, we need to decide if we can write

$w = uv$ such that M accepts u and M' accepts v.

Problem: Don't know where u ends, v begins.

When do you stop simulating M and start simulating M'?

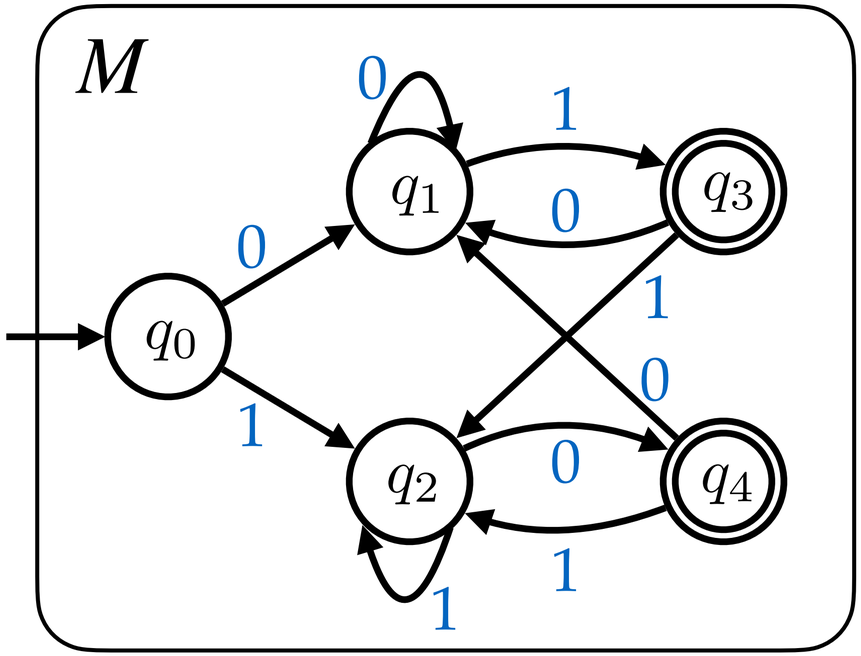


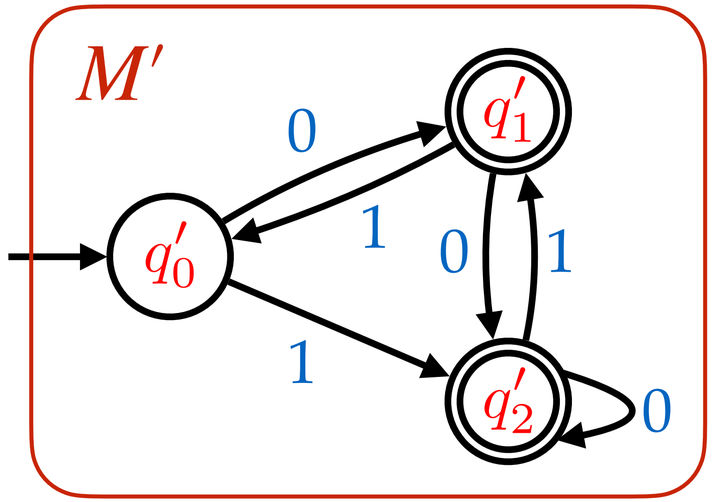


Suppose God tells you u ends at w_3.

	0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
q_0	q_1	q_1	(q_3)							
			q'_0	q'_2	q'_2	q'_2	q'_1	q'_2	q'_2	q'_1

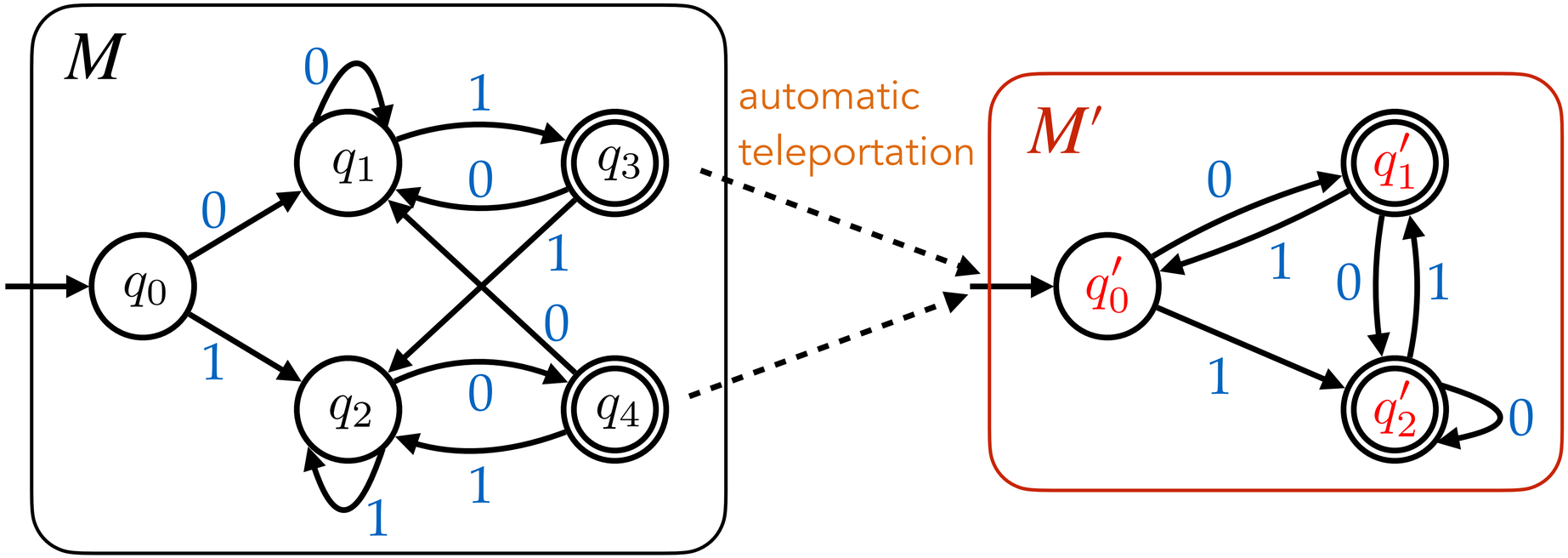
thread: a simulation of M and then M' that corresponds to breaking up input w as uv where $u \in L_1$.



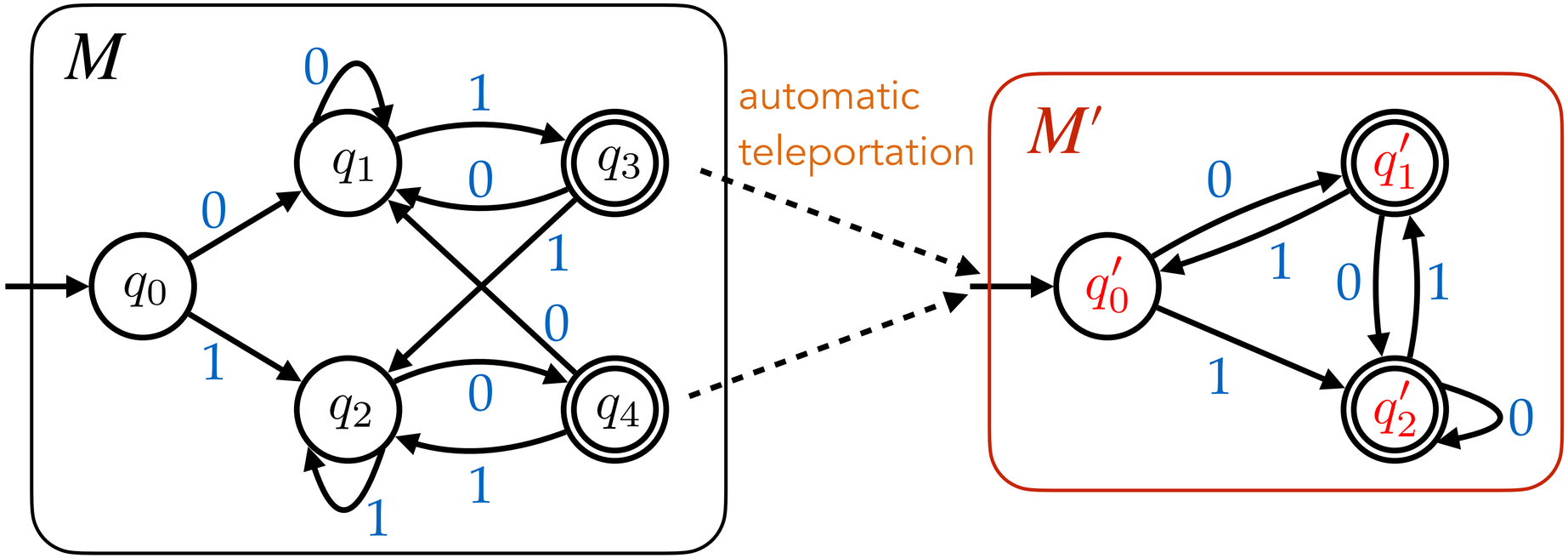


thread: a simulation of M and then M' that corresponds to breaking up input w as uv where $u \in L_1$.

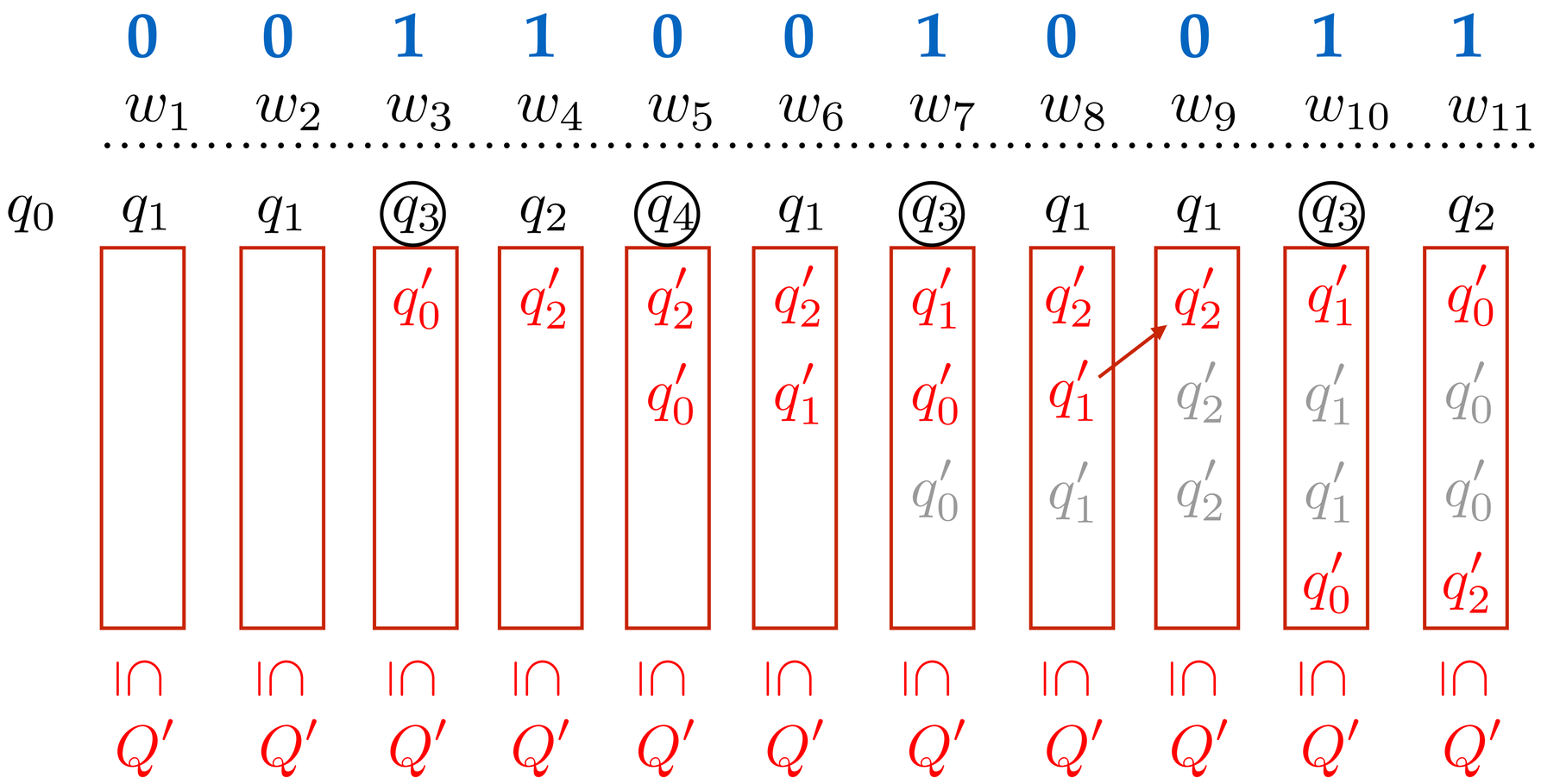
$w \in L_1 L_2$ iff ∃ a thread ending in an accepting state of M'.



	0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
q_0	q_1	q_1	$\textcircled{q_3}$	q_2	$\textcircled{q_4}$	q_1	$\textcircled{q_3}$	q_1	q_1	$\textcircled{q_3}$	q_2
thread1			q'_0	q'_2	q'_2	q'_2	q'_1	q'_2	q'_2	q'_1	q'_0
thread2					q'_0	q'_1	q'_0	q'_1	q'_2	q'_1	q'_0
thread3							q'_0	q'_1	q'_2	q'_1	q'_0
thread4										q'_0	q'_2



	0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
q_0	q_1	q_1	$\textcircled{q_3}$	q_2	$\textcircled{q_4}$	q_1	$\textcircled{q_3}$	q_1	q_1	$\textcircled{q_3}$	q_2
thread1			q'_0	q'_2	q'_2	q'_2	q'_1	q'_2	q'_2	q'_1	q'_0
thread2					q'_0	q'_1	q'_0	q'_1	q'_2	q'_1	q'_0
thread3							q'_0	q'_1	q'_2	q'_1	q'_0
thread4										q'_0	q'_2



At any point, need to remember:

- an element of Q
- a subset of Q'

(an element of $\mathcal{P}(Q')$)

constant amount of information!

STEP 2: Formally define the DFA.

Great Exercise

$M = (Q, \Sigma, \delta, q_0, F)$ $\qquad$ $M' = (Q', \Sigma, \delta', q_0', F')$

$Q'' = \quad Q \times \mathscr{P}(Q') \;\; = \{(q, S) : \; q \in Q, \;\; S \in \mathscr{P}(Q')\}$

$\delta'' :$

$q_0'' =$

$F'' = \; \{(q, S) : \quad q \in Q, \quad S \in \mathscr{P}(Q'), \quad S \cap F' \neq \varnothing\}$

Next Chapter