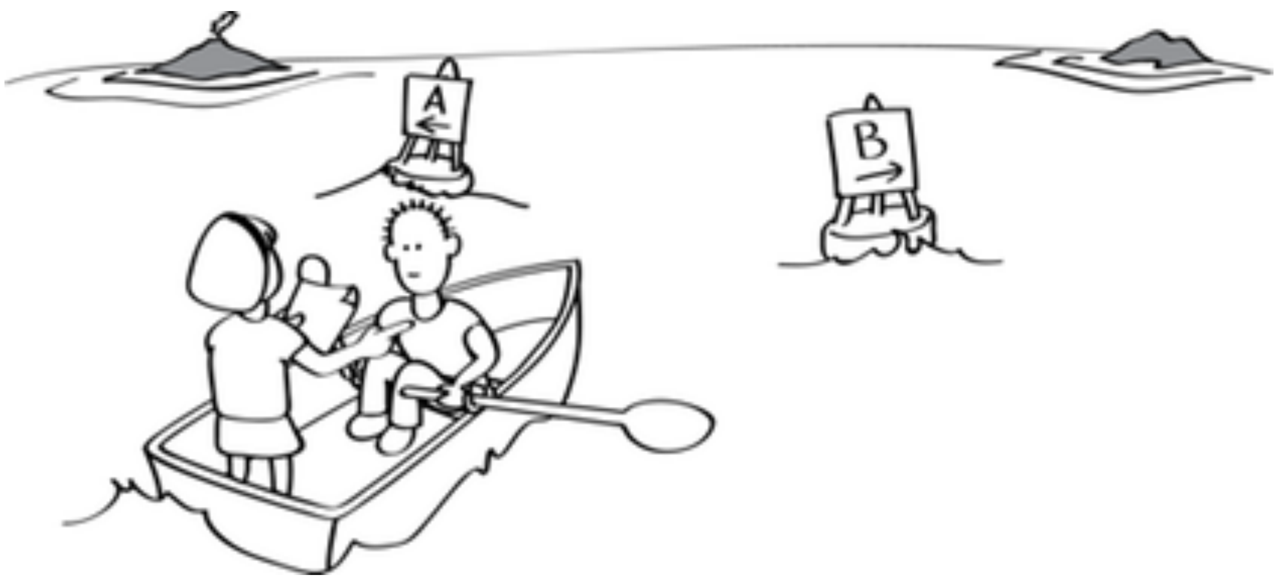
15-251
Great Theoretical Ideas in Computer Science

Lecture 4:
Deterministic Finite Automaton (DFA), Part 2

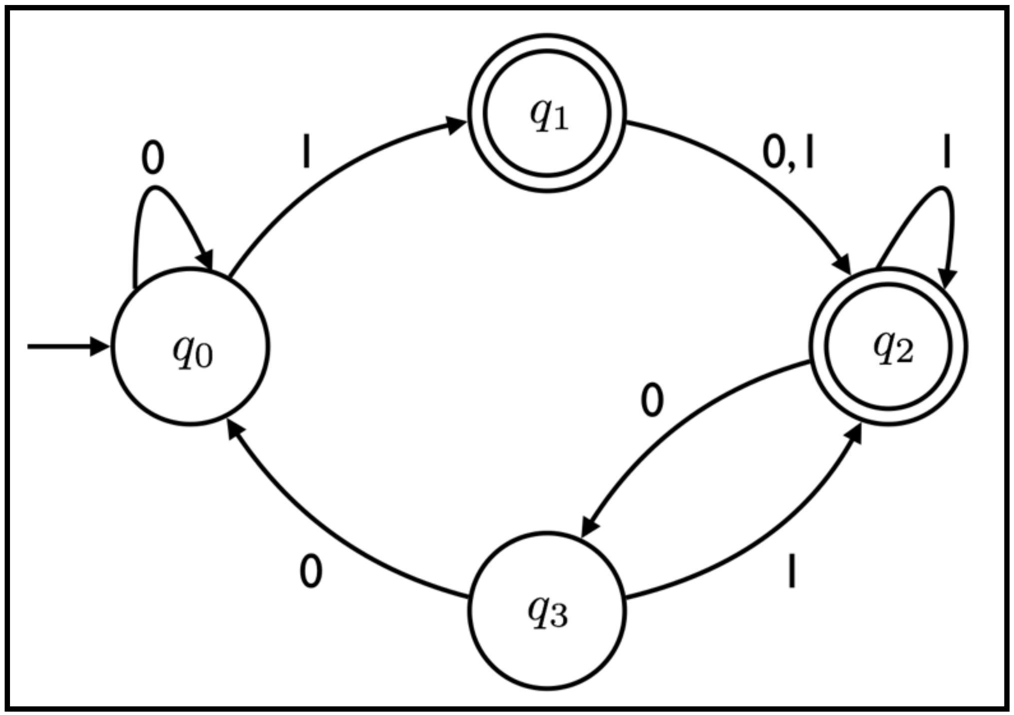


January 26th, 2017

Formal definition: DFA

A deterministic finite automaton (DFA) M is a 5-tuple

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



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

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

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

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

$F = \{q_1, q_2\}$

Formal definition: DFA accepting a string

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

For $q \in Q, w \in \Sigma^*$,

$\delta(q, w) =$ state we end up in when we start at q and read w.

M ***accepts*** w if $\delta(q_0, w) \in F$.

Otherwise M ***rejects*** w.

Definition: Regular languages

Let M be a DFA.

We let $L(M)$ denote the set of strings that M **accepts**.

Definition: A language L is called ***regular*** if
$L = L(M)$ for some DFA M.

Non-regular languages

Theorem:

The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

Theorem:

The language $L = \{a^{2^n} : n \in \mathbb{N}\}$ is **not** regular.

The big picture

All languages
$\mathcal{P}(\Sigma^*)$

Regular languages

$\{110, 101\}$

$\{0,1\}^* \setminus \{110, 101\}$

$\{x \in \{0,1\}^* : x \text{ starts and ends with same bit.}\}$

$\{x \in \{0,1\}^* : |x| \text{ is divisible by 2 or 3.}\}$

$\{\epsilon, 110, 110110, 110110110, \ldots\}$

$\{x \in \{0,1\}^* : x \text{ contains the substring 110.}\}$

$\{x \in \{0,1\}^* : \text{10 and 01 occur equally often in } x.\}$

$\vdots$

$\{0^n 1^n : n \in \mathbb{N}\}$

$\{a^{2^n} : n \in \mathbb{N}\}$

$\vdots$

Regular languages

Questions:

1. Are all languages regular?
   (Are all decision problems computable by a DFA?)

2. Are there other ways to tell if a language is regular?

Closure properties of regular languages

Closed under complementation

Proposition:

Let Σ be some finite alphabet.
If $L \subseteq \Sigma^*$ is regular, then so is $\overline{L} = \Sigma^* \backslash L$.

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

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

recognizing L. Then

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

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

$\square$

Closed under complementation

Closure properties can be used to show languages are not regular.

Corollary:

If $L \subseteq \Sigma^*$ is **non**-regular, then so is $\overline{L}$.

Proof: By contrapositive:

If $\overline{L}$ is regular, then by the previous Proposition

$\overline{\overline{L}} = L$ is regular.

$\square$

Examples:

$\{0,1\}^* \backslash \{0^n 1^n : n \in \mathbb{N}\}$

$\{a\}^* \backslash \{a^{2^n} : n \in \mathbb{N}\}$

are non-regular.

Closed under union

Theorem:

Let Σ be some finite alphabet.
If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are regular, then so is $L_1 \cup L_2$.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA deciding L_1 and $M' = (Q', \Sigma, \delta', q'_0, F')$ be a DFA deciding L_2.
We construct a DFA $M'' = (Q'', \Sigma, \delta'', q''_0, F'')$ that decides $L_1 \cup L_2$, as follows:

$\vdots$

The mindset

Imagine yourself as a DFA.

Rules:

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

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

should not change
based on input length

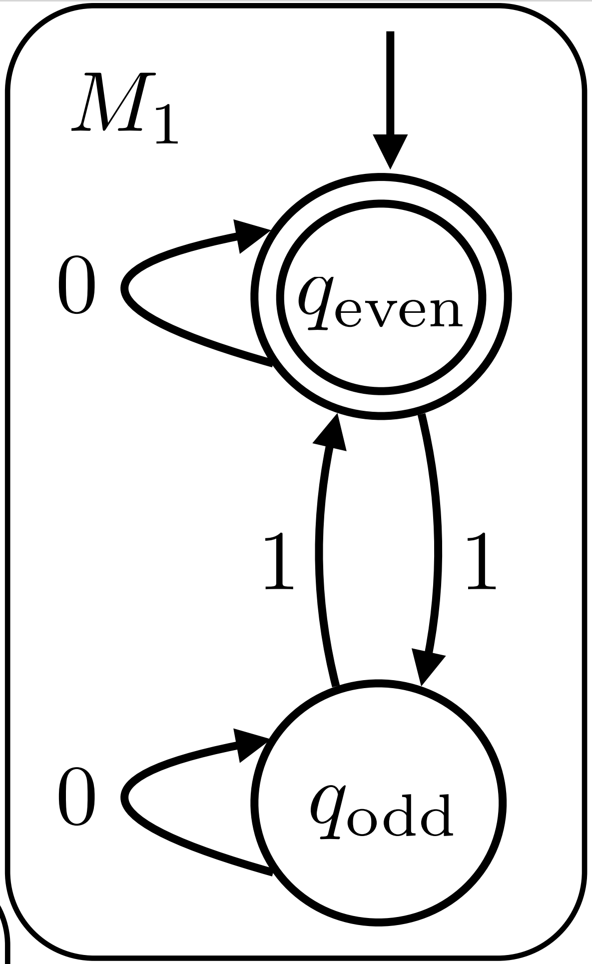
Step 1: Imagining ourselves as a DFA

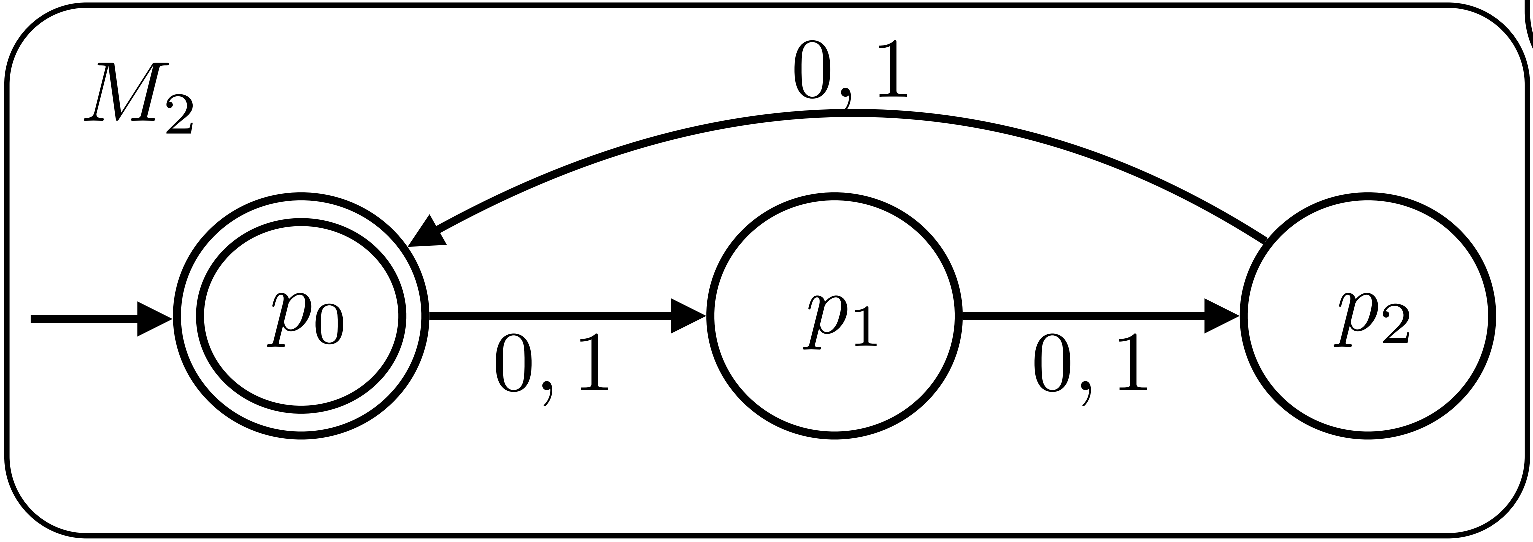
Closed under union

Example

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

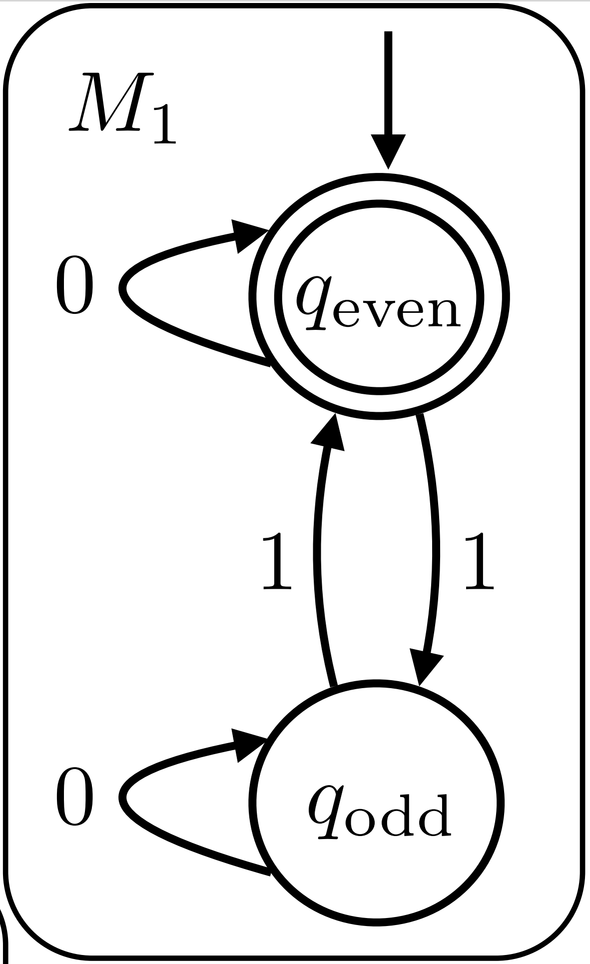
$L_2 =$ strings with length divisible by 3.

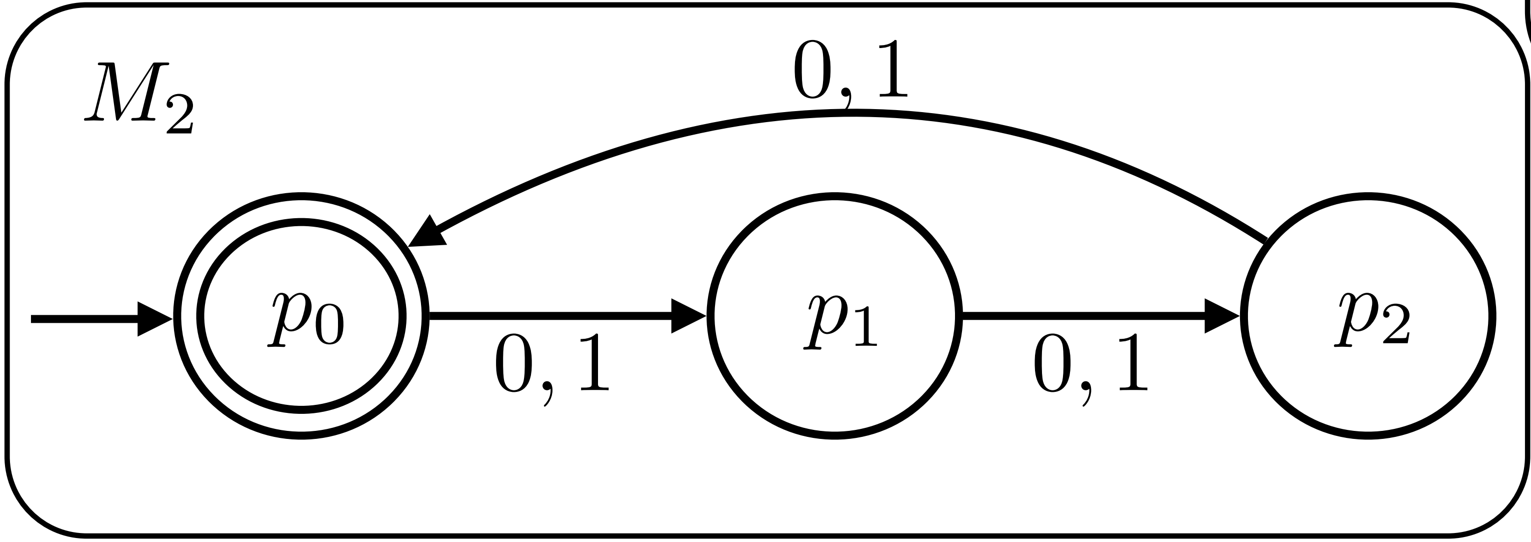




Closed under union

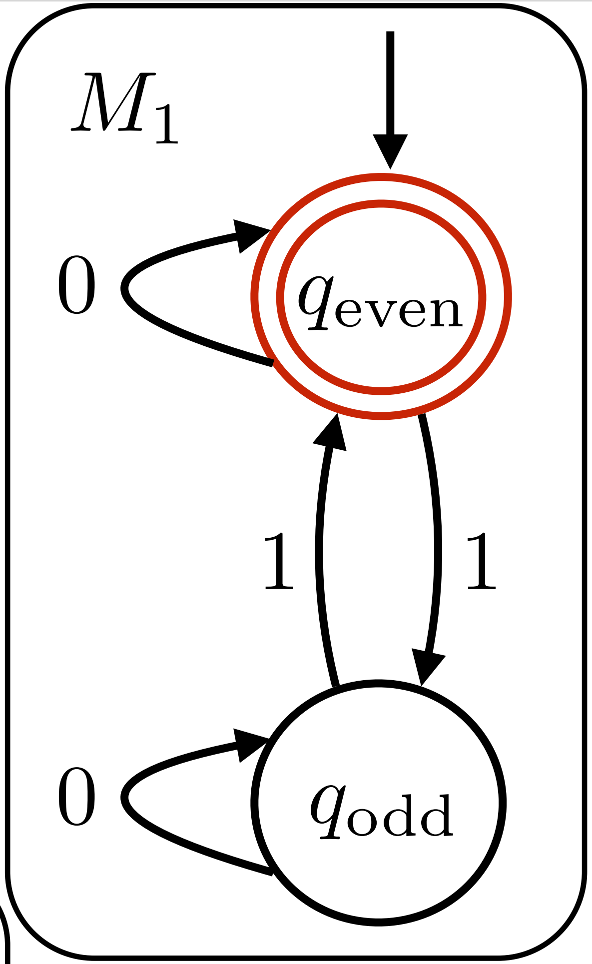
Input: 101001

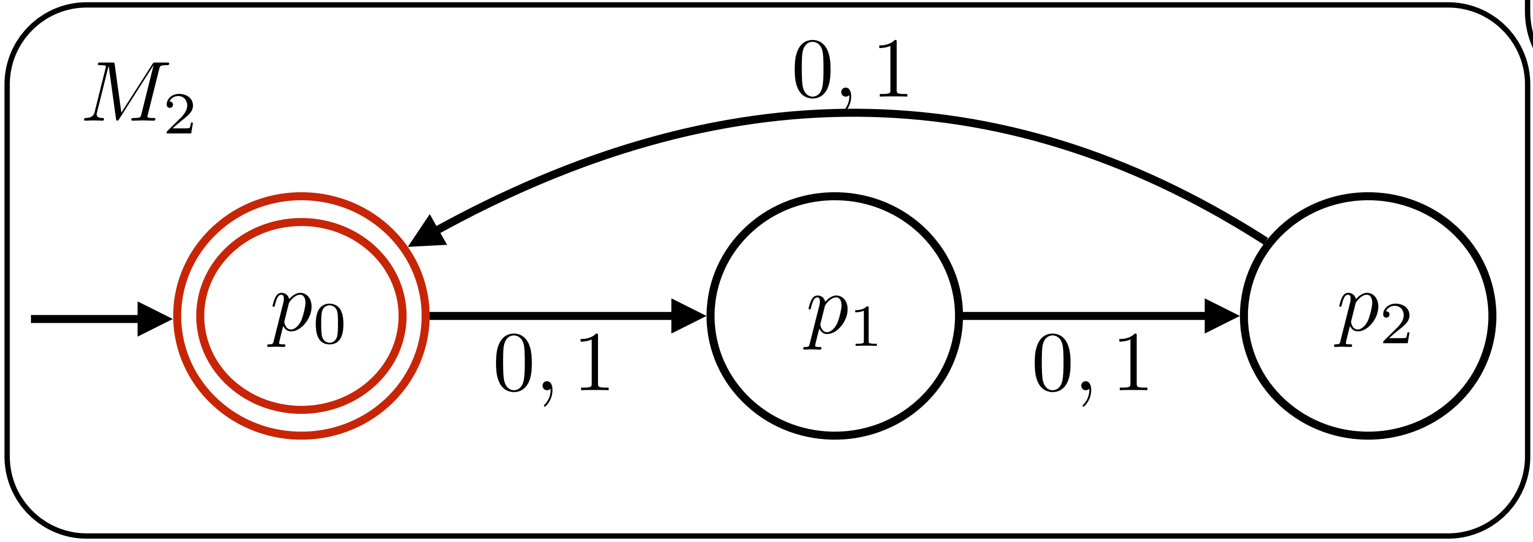




Closed under union

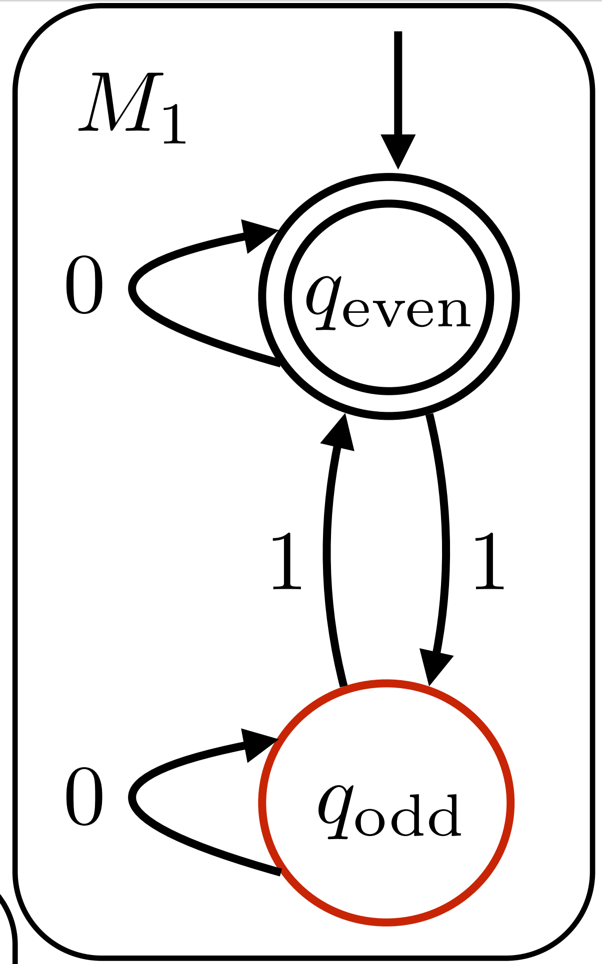
Input: 101001

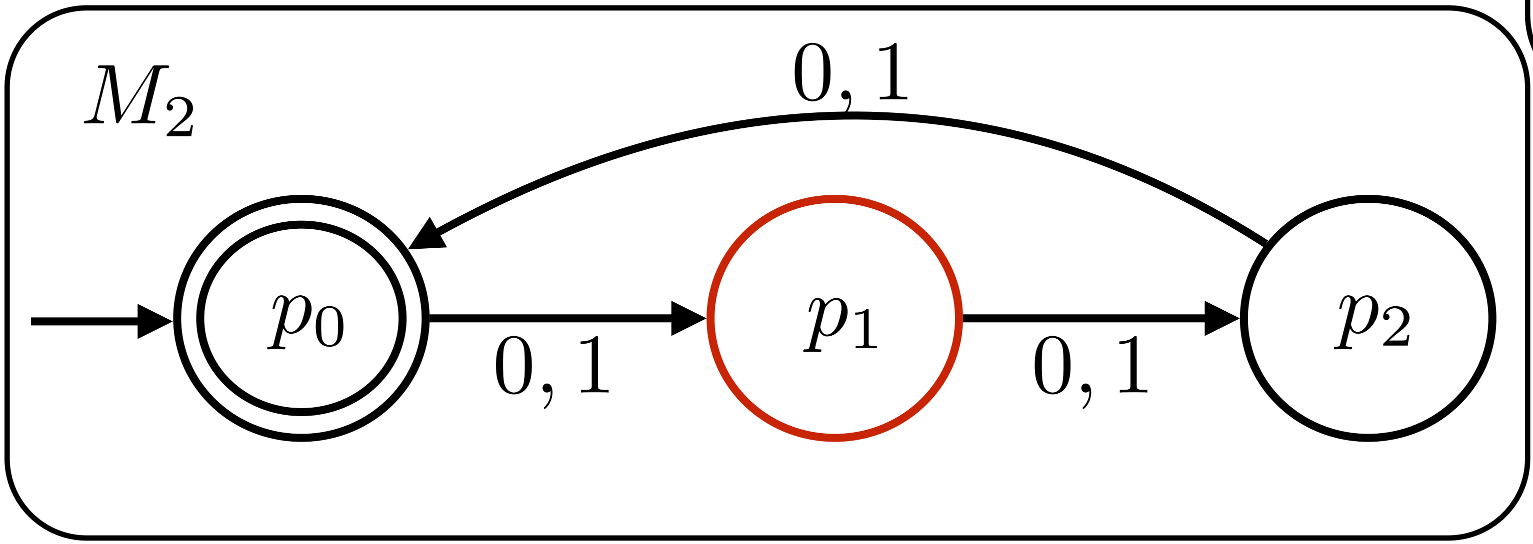




Closed under union

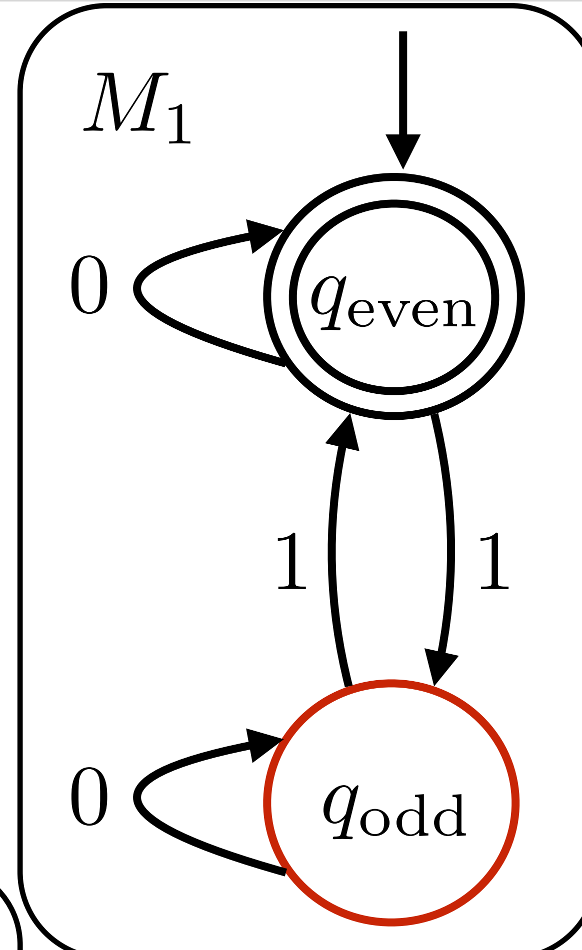
Input: 101001

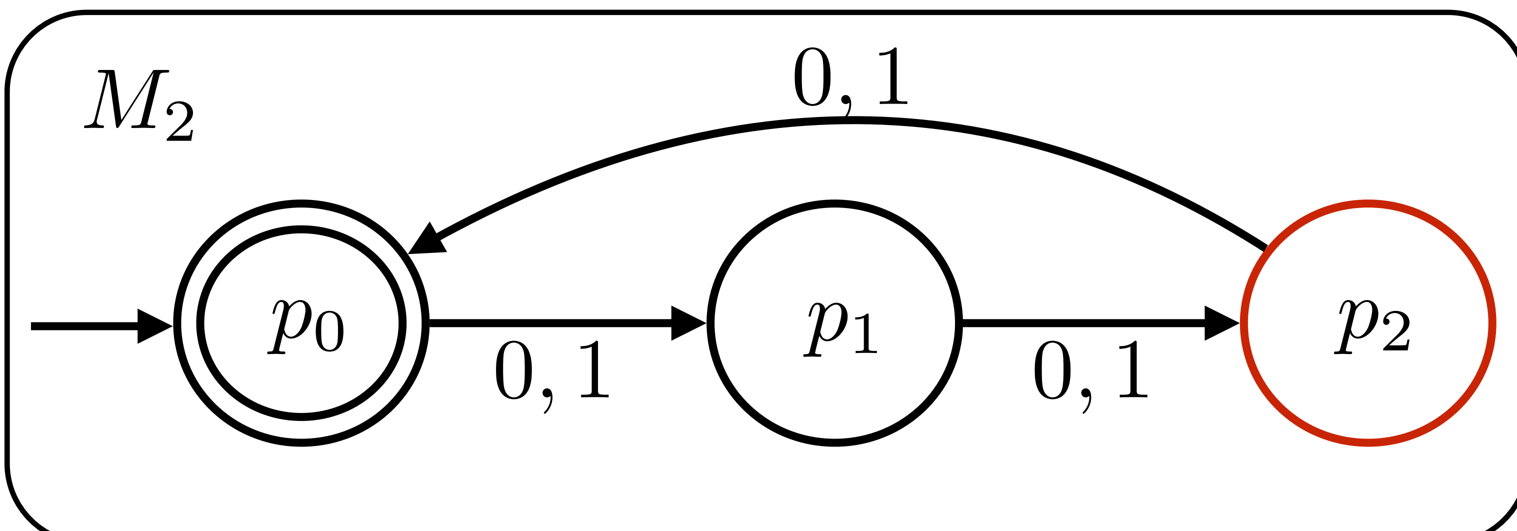




Closed under union

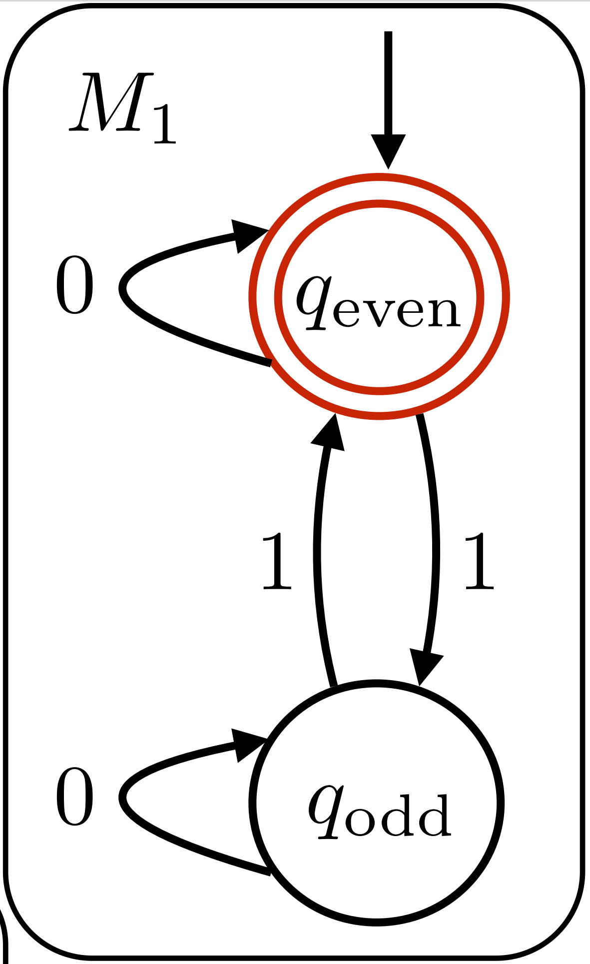
Input: 101001

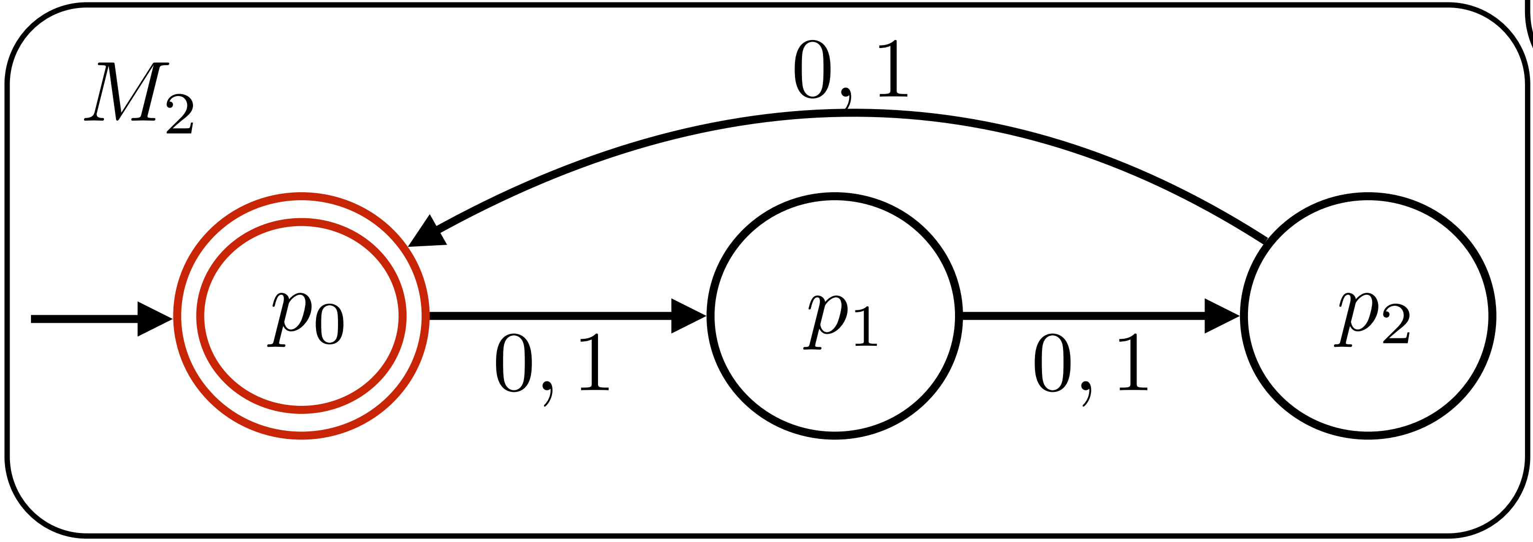




Closed under union

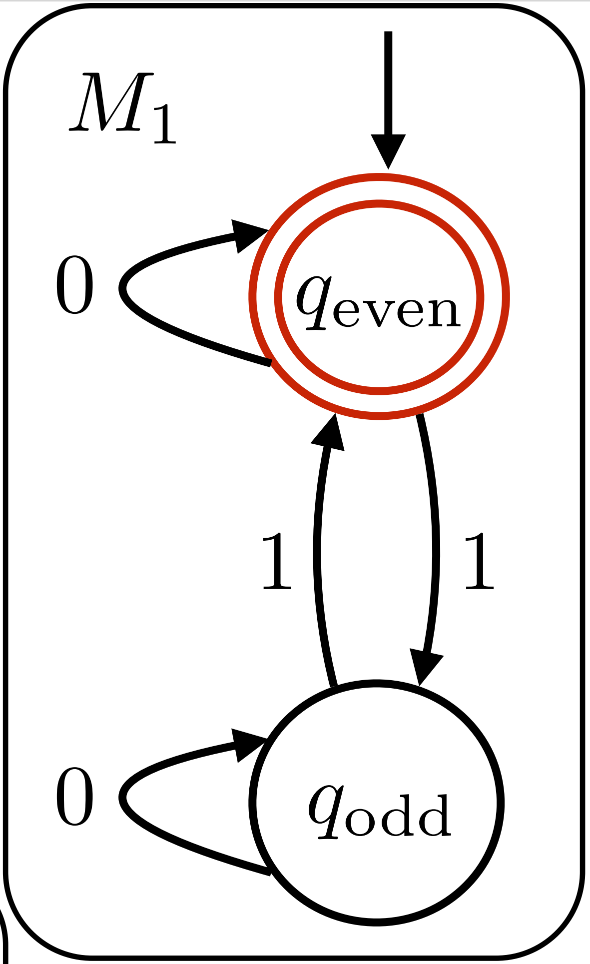
Input: 101001

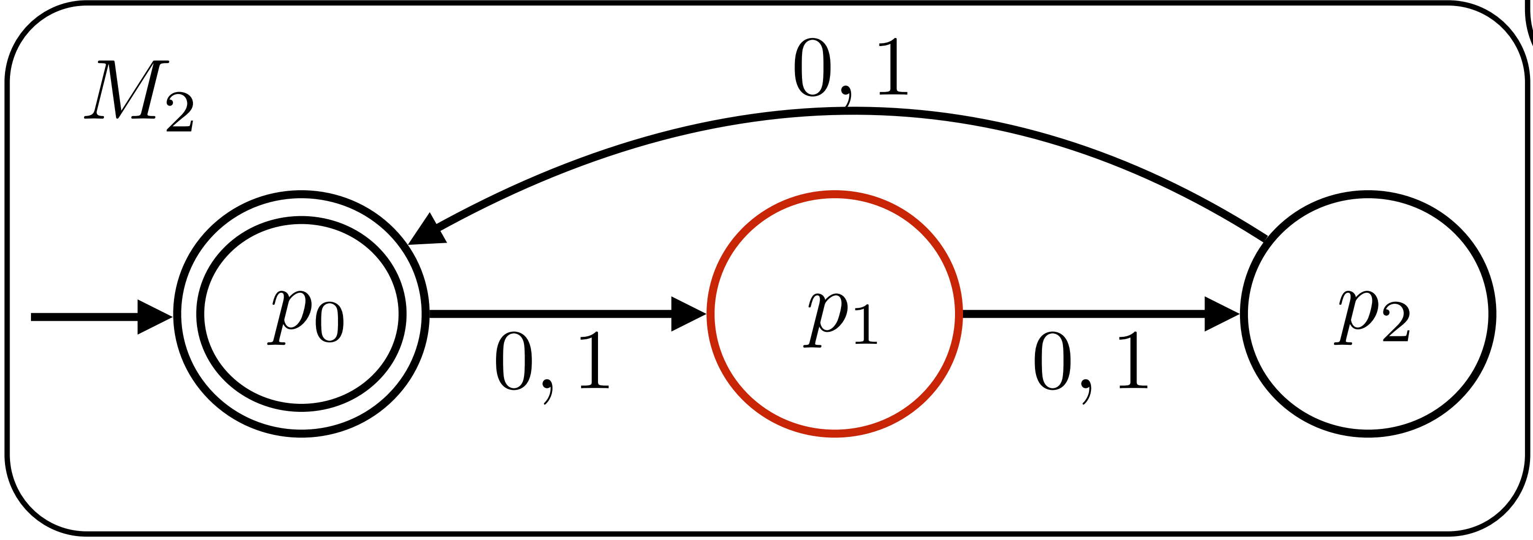




Closed under union

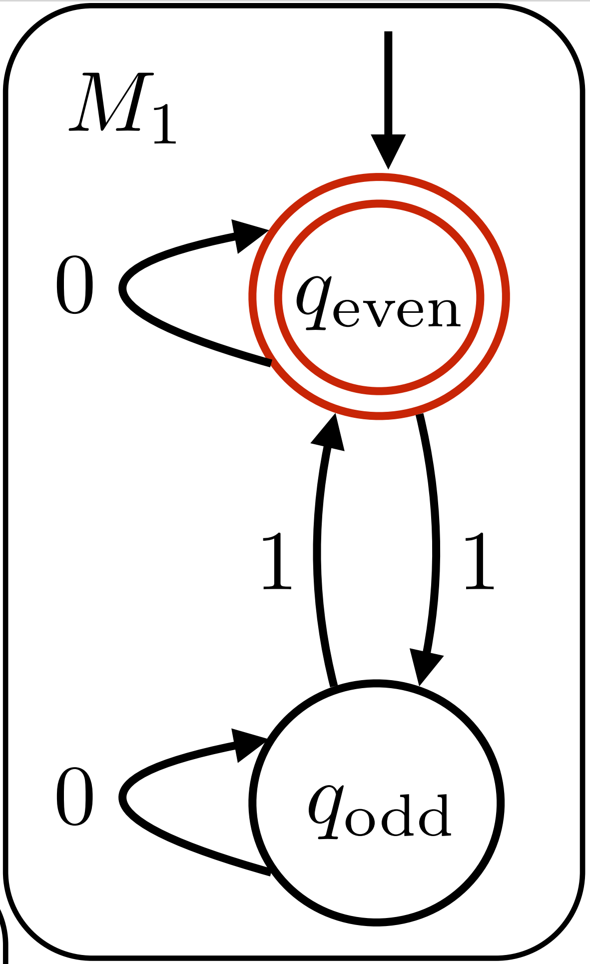
Input: 1010 01

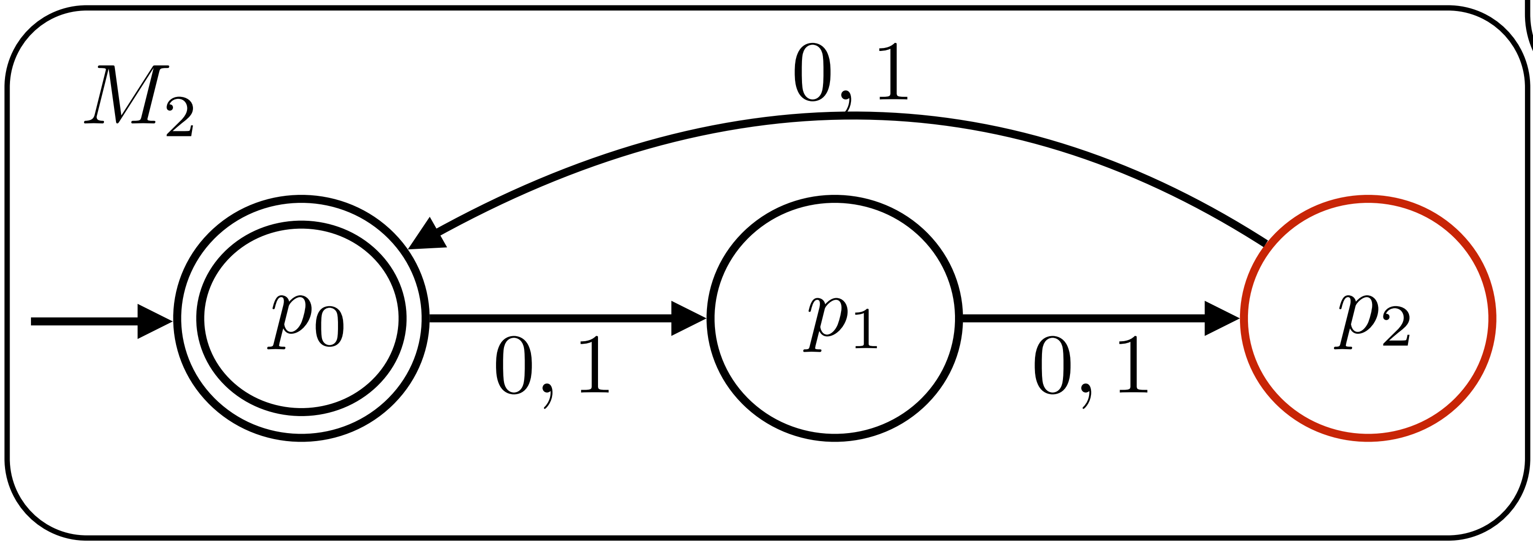




Closed under union

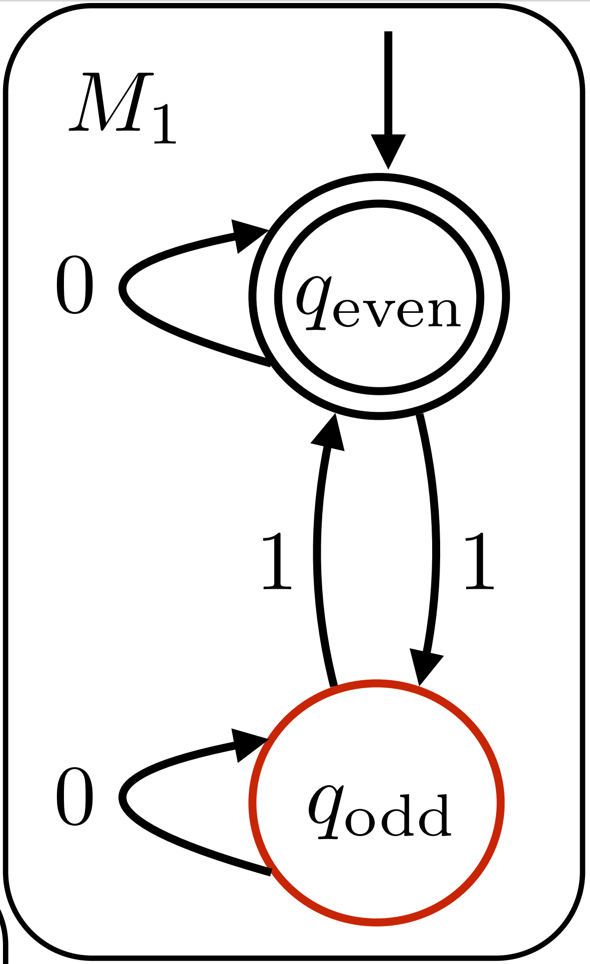
Input: 101001

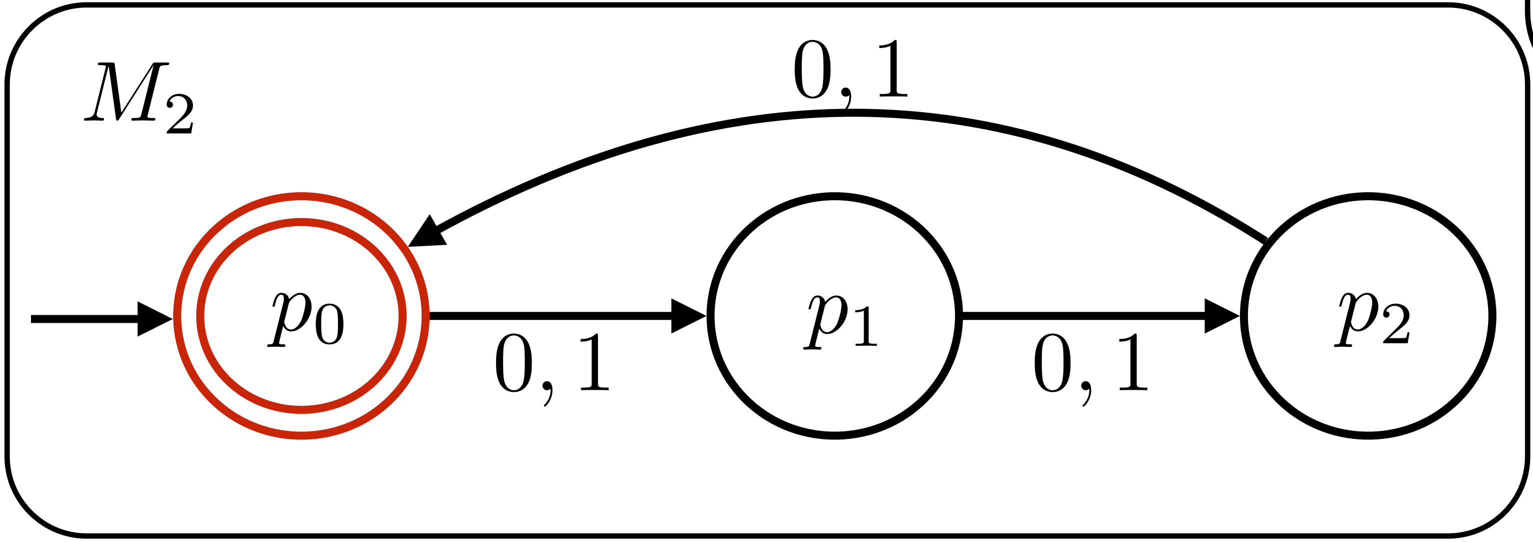




Closed under union

Input: 101001

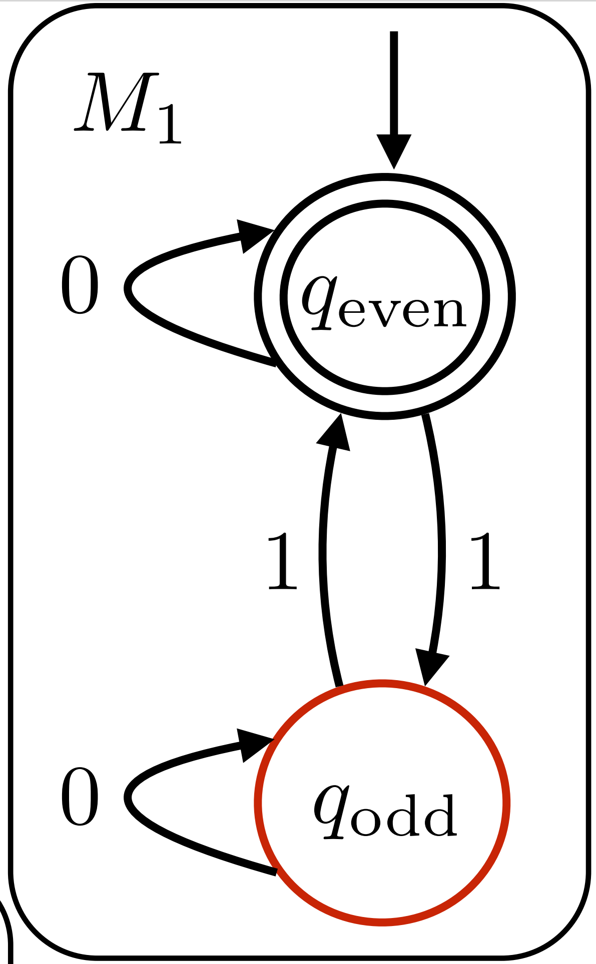


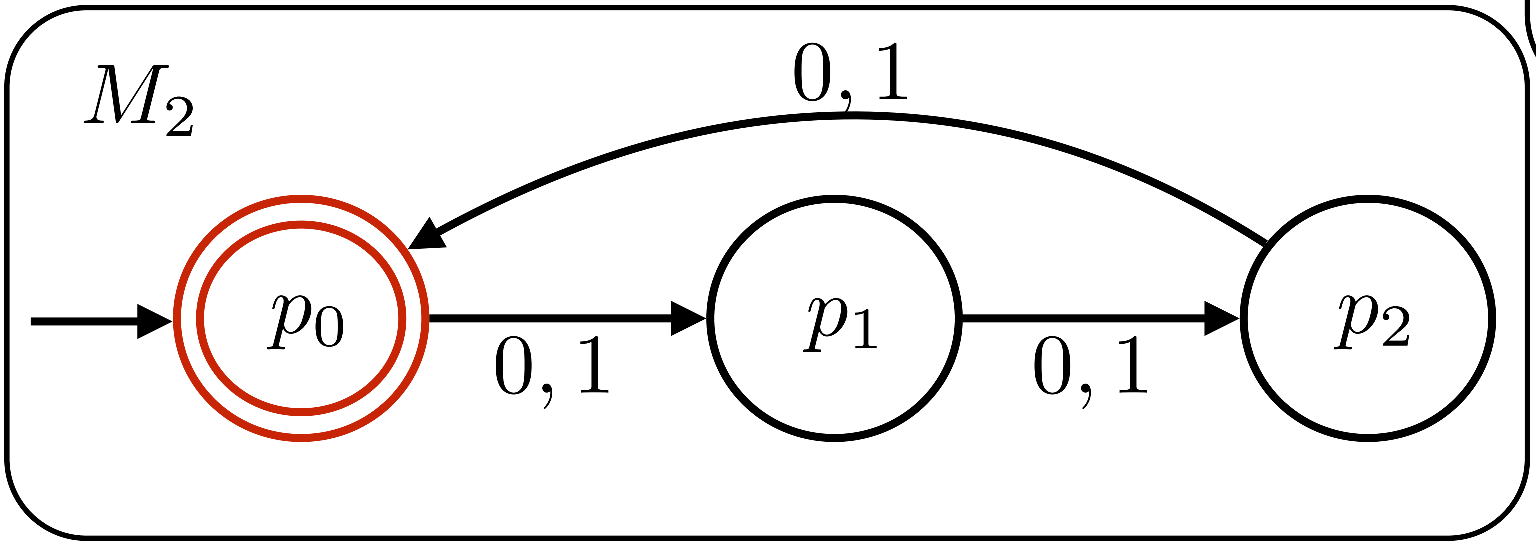


Closed under union

Input: 101001

Accept

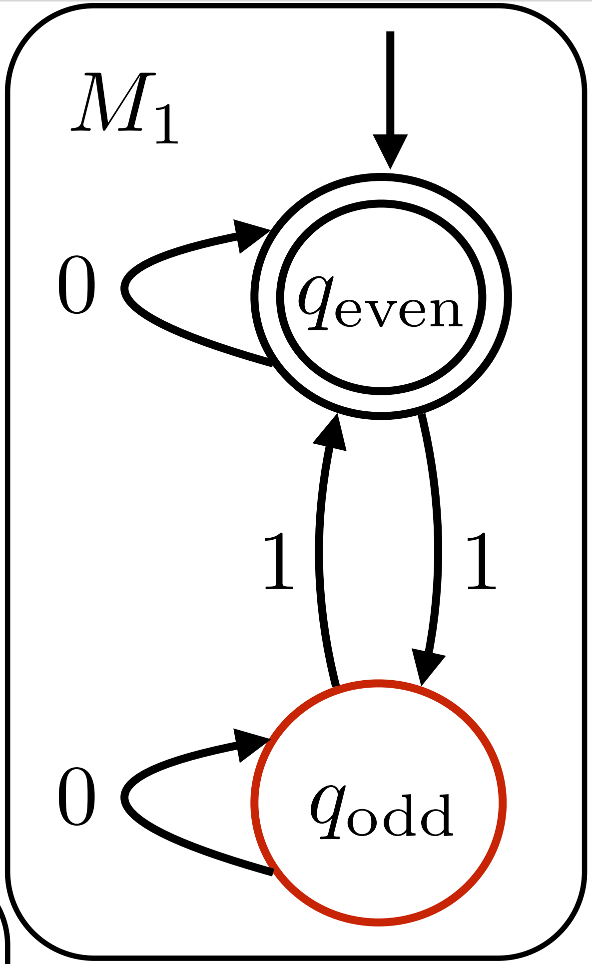


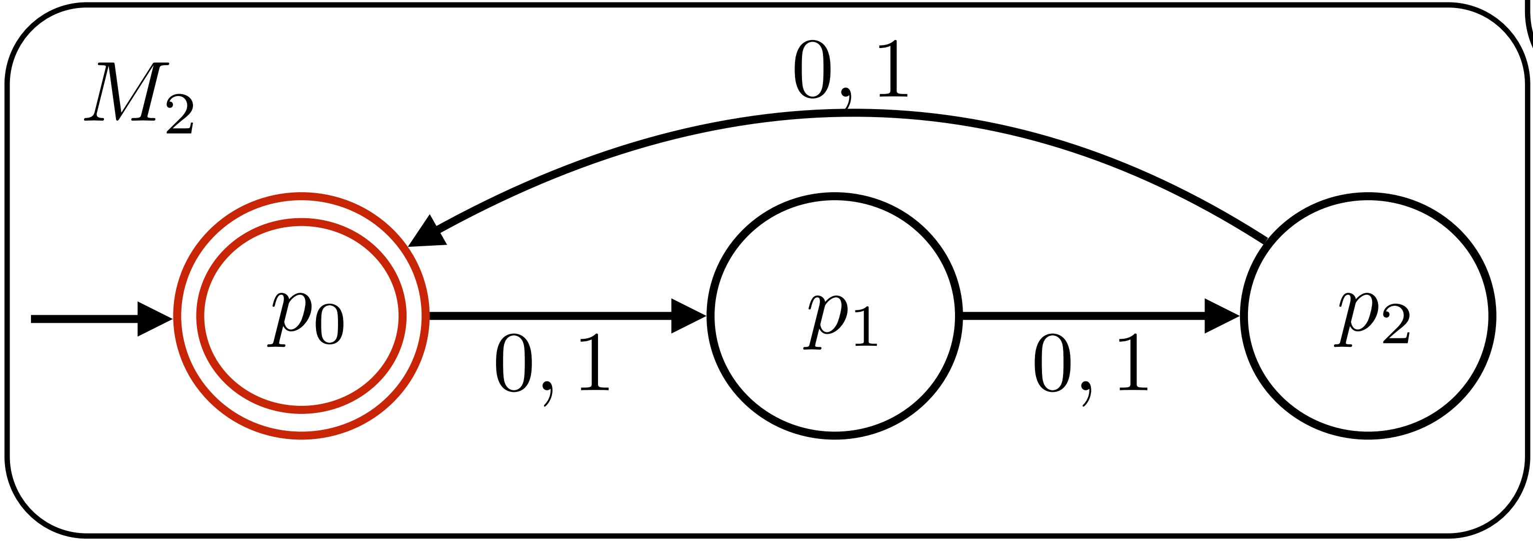


Closed under union

Main idea:

Construct a DFA that keeps track of both at once.

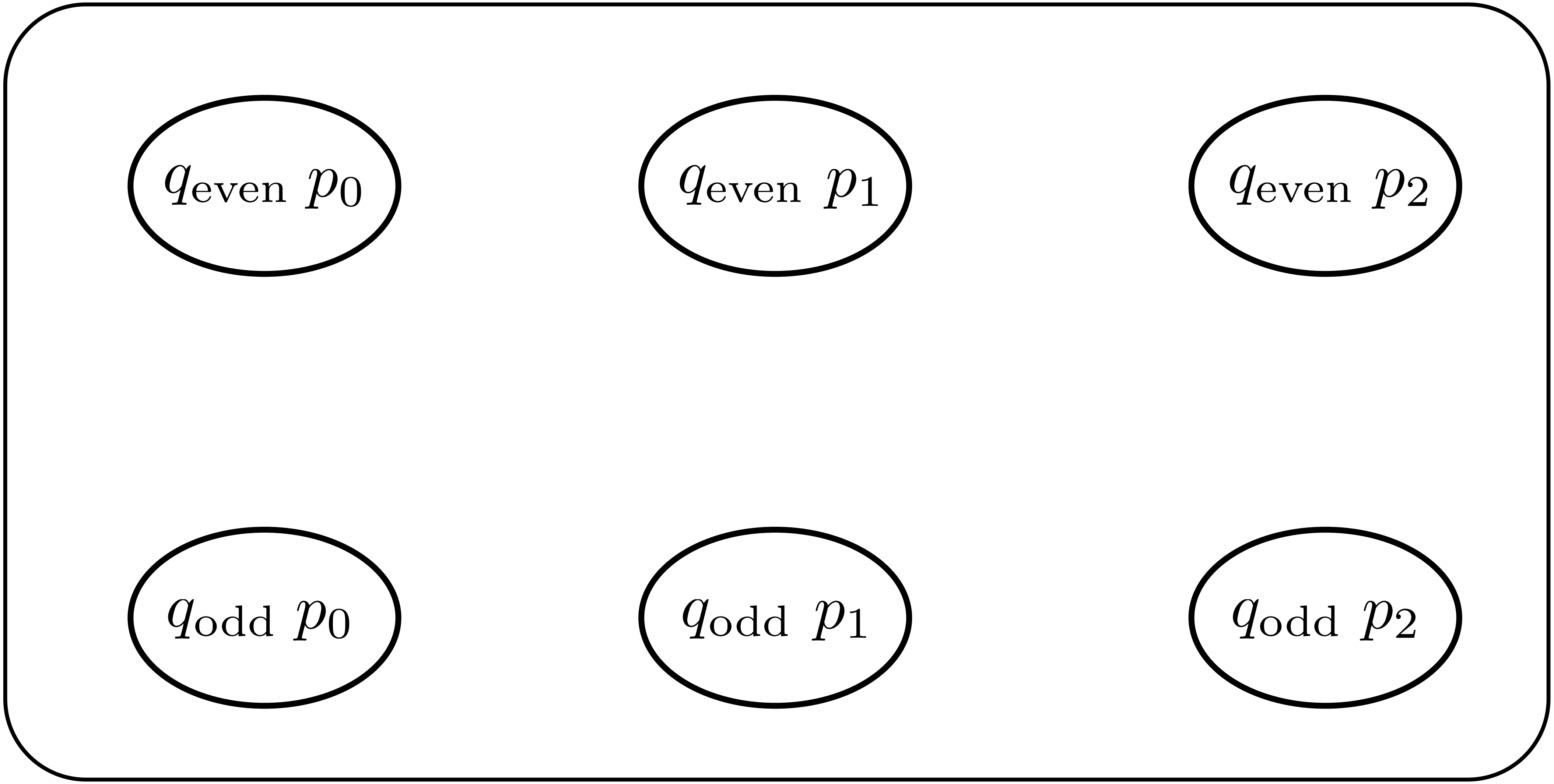




Closed under union

Main idea:

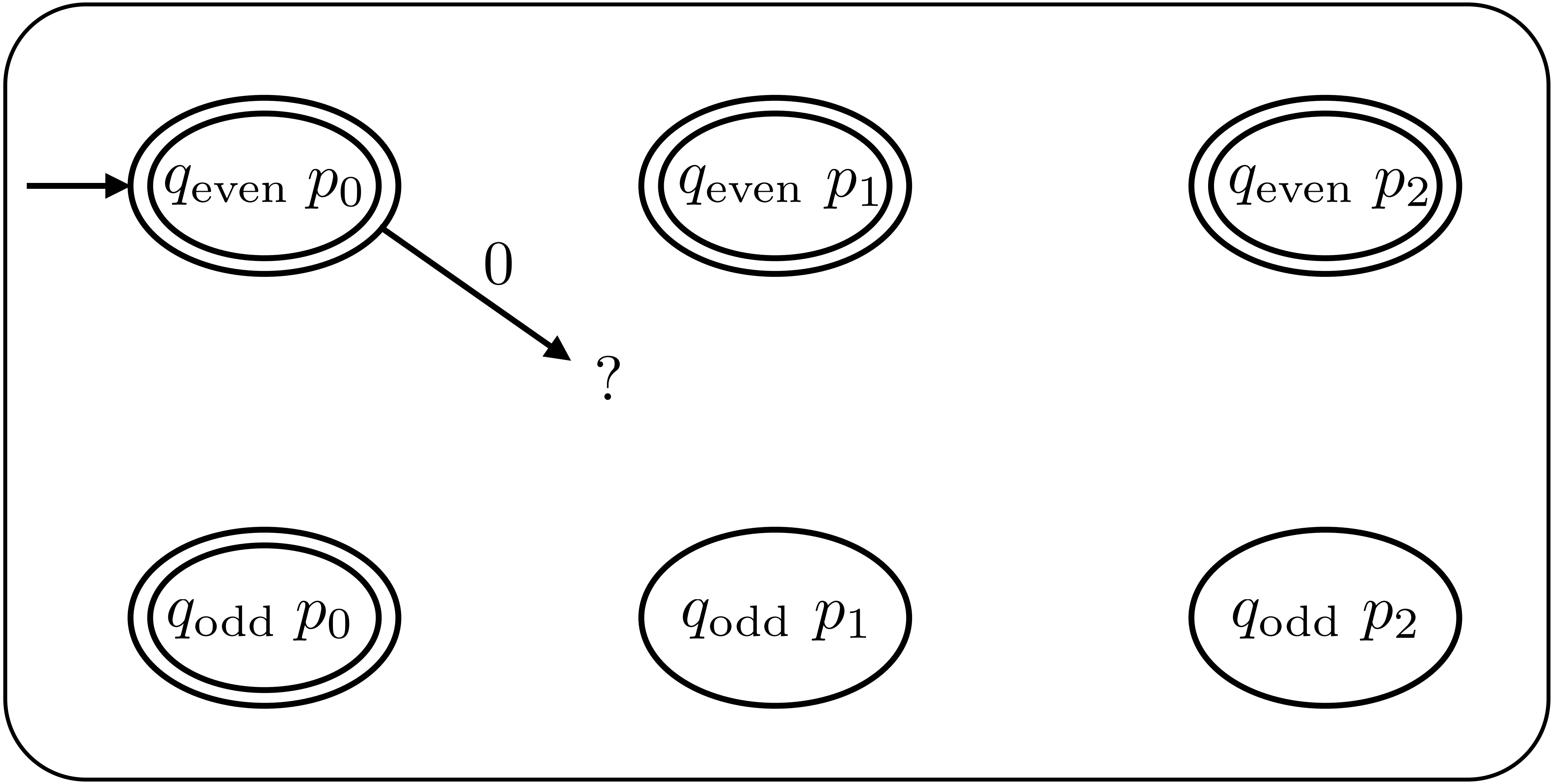
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

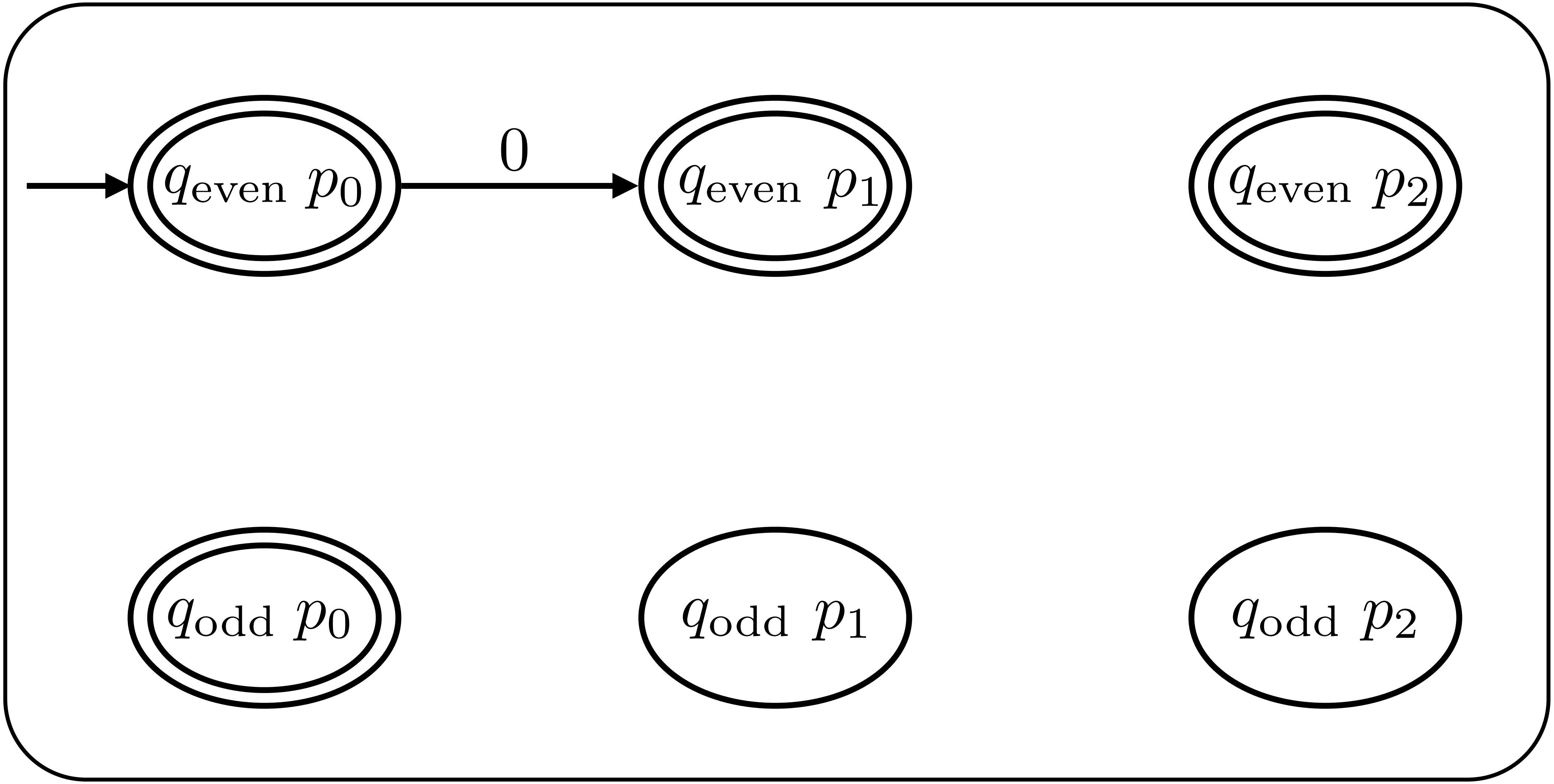
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

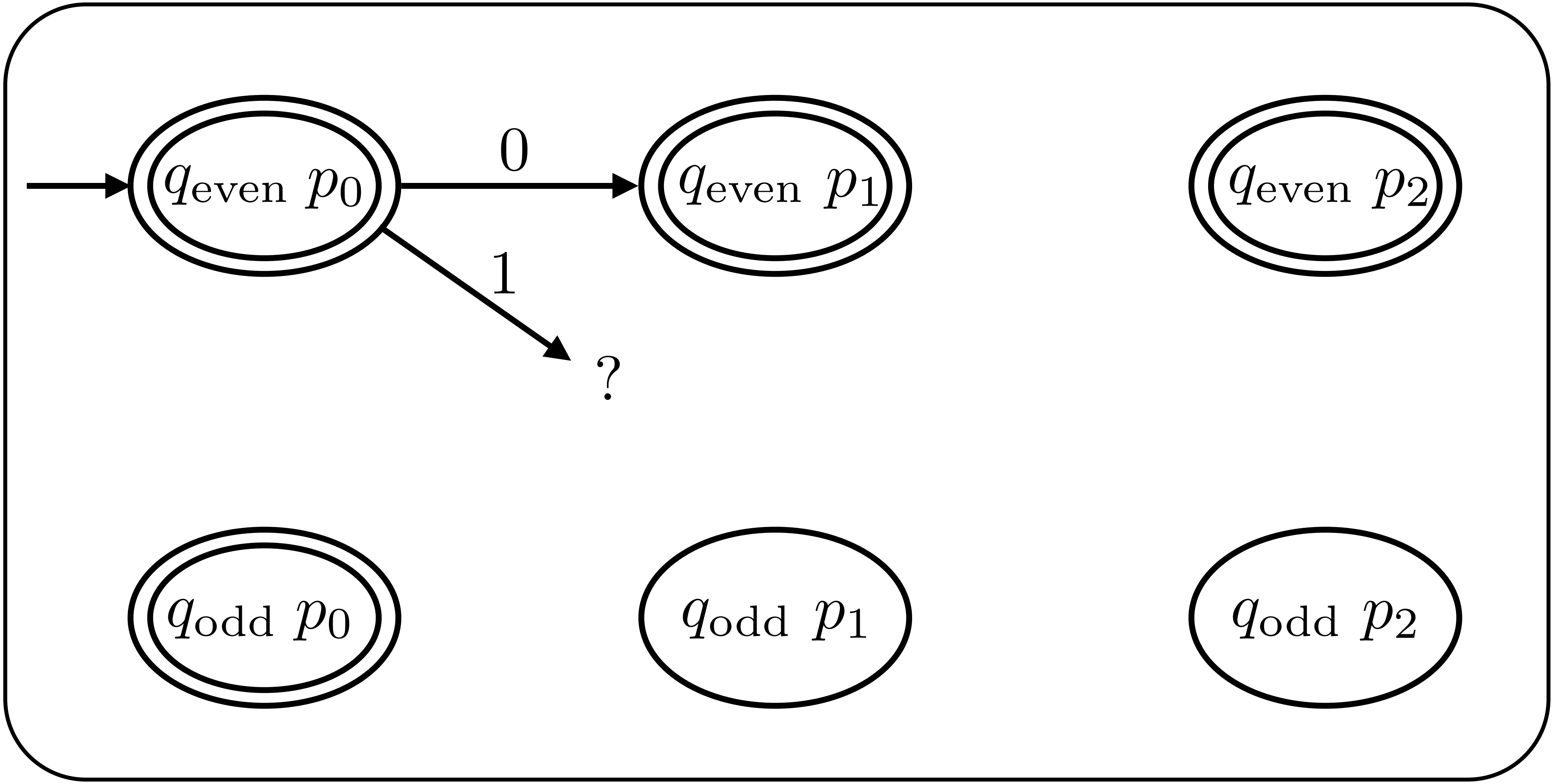
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

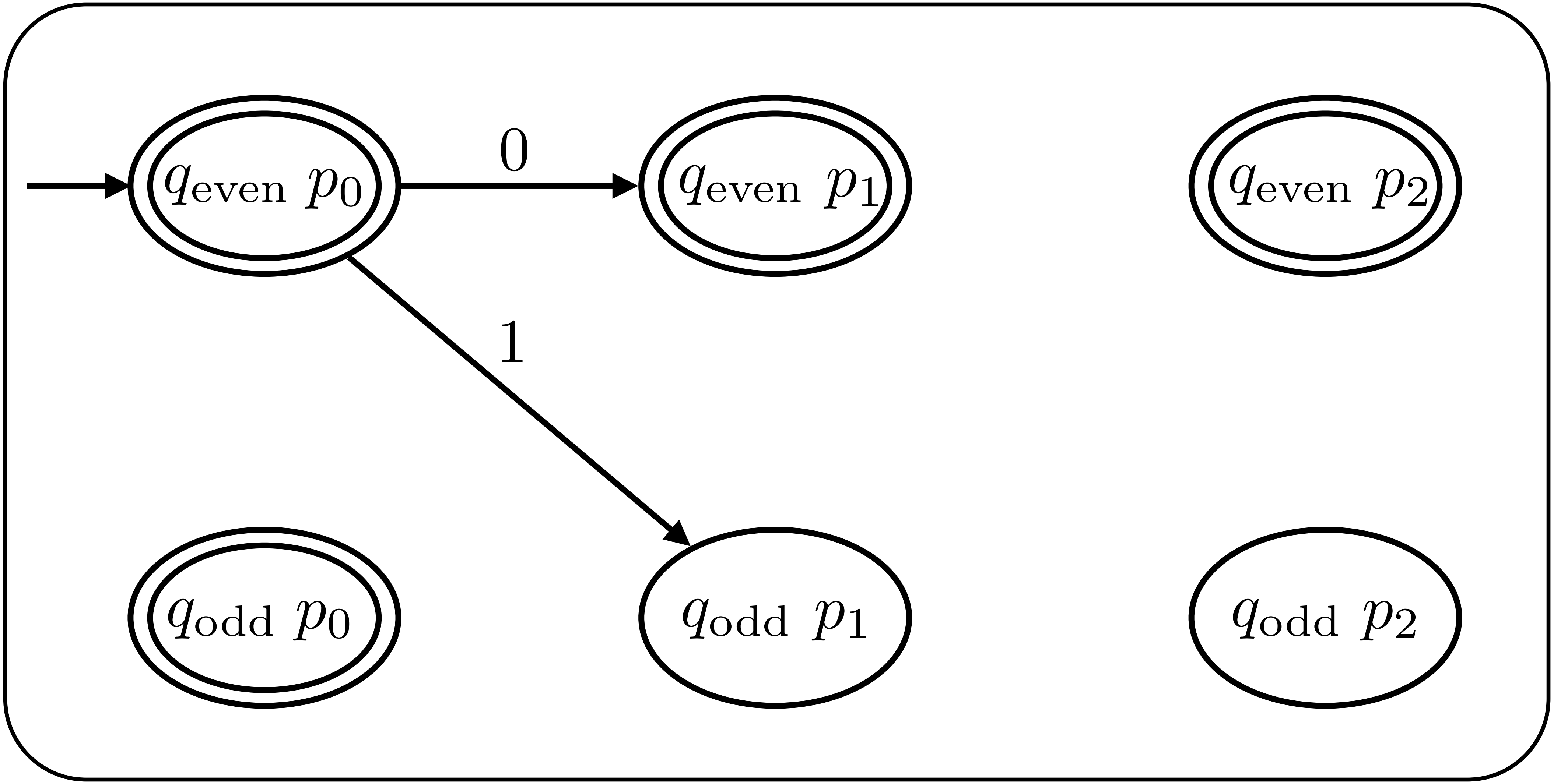
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

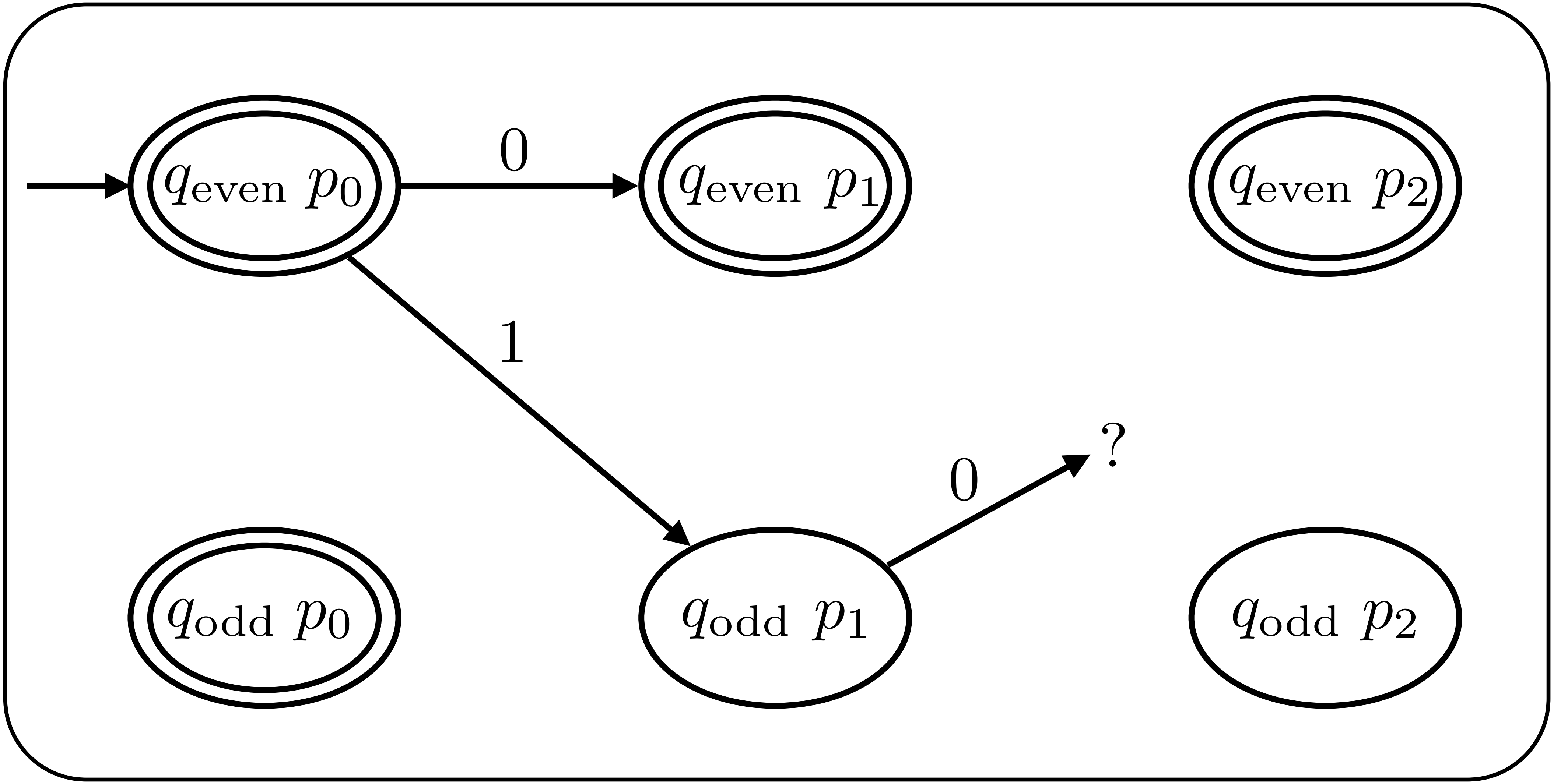
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

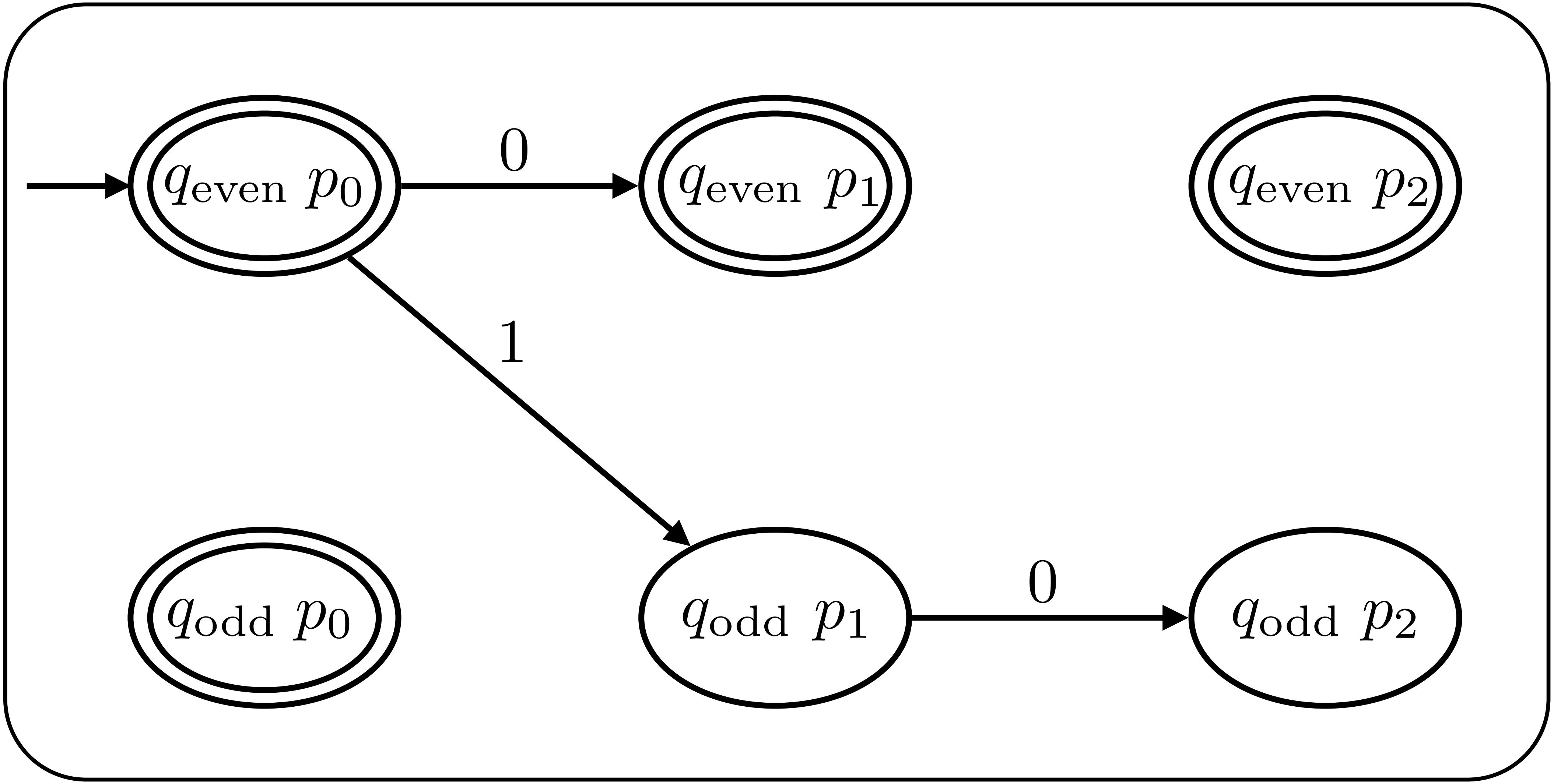
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

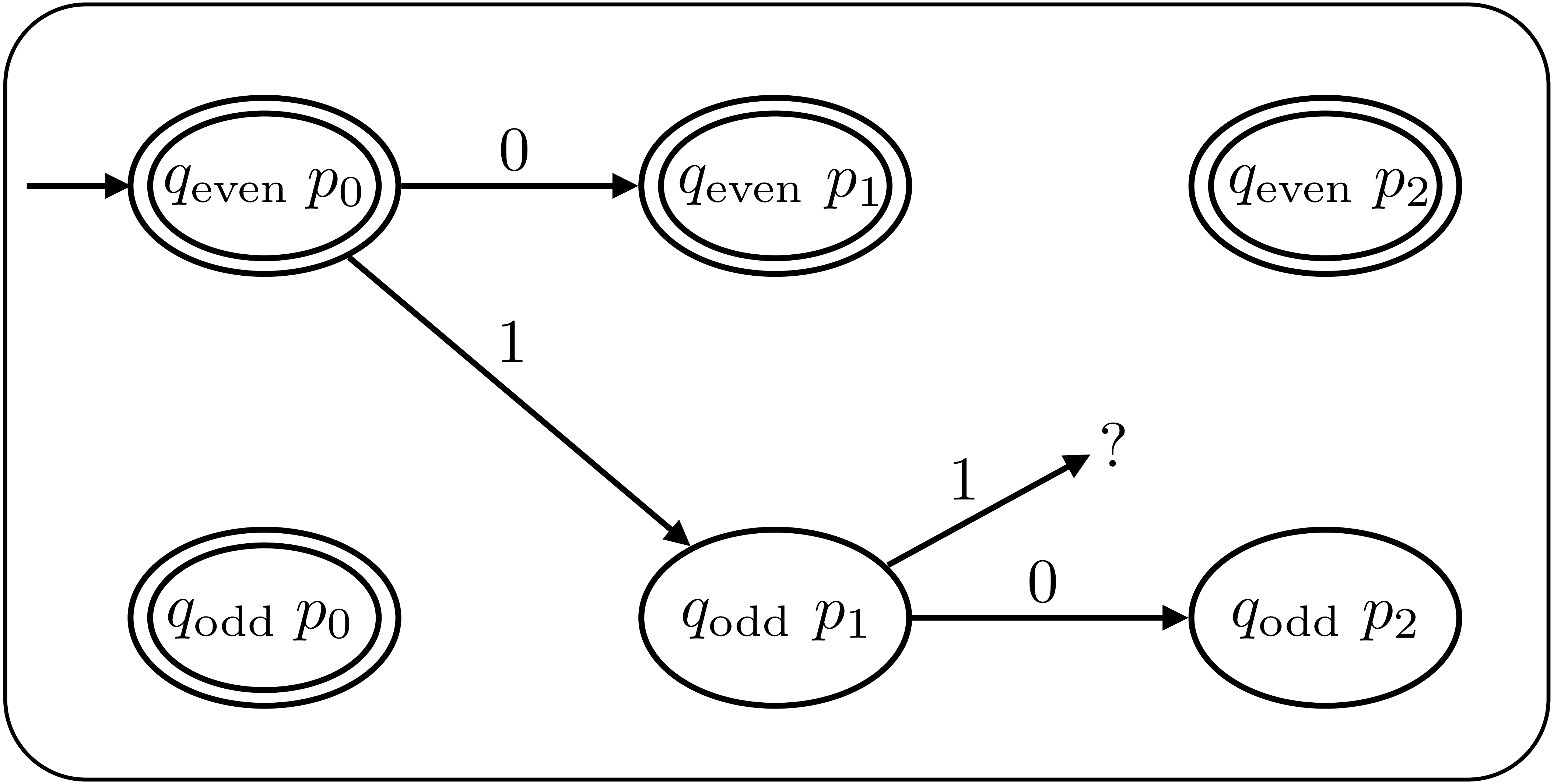
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

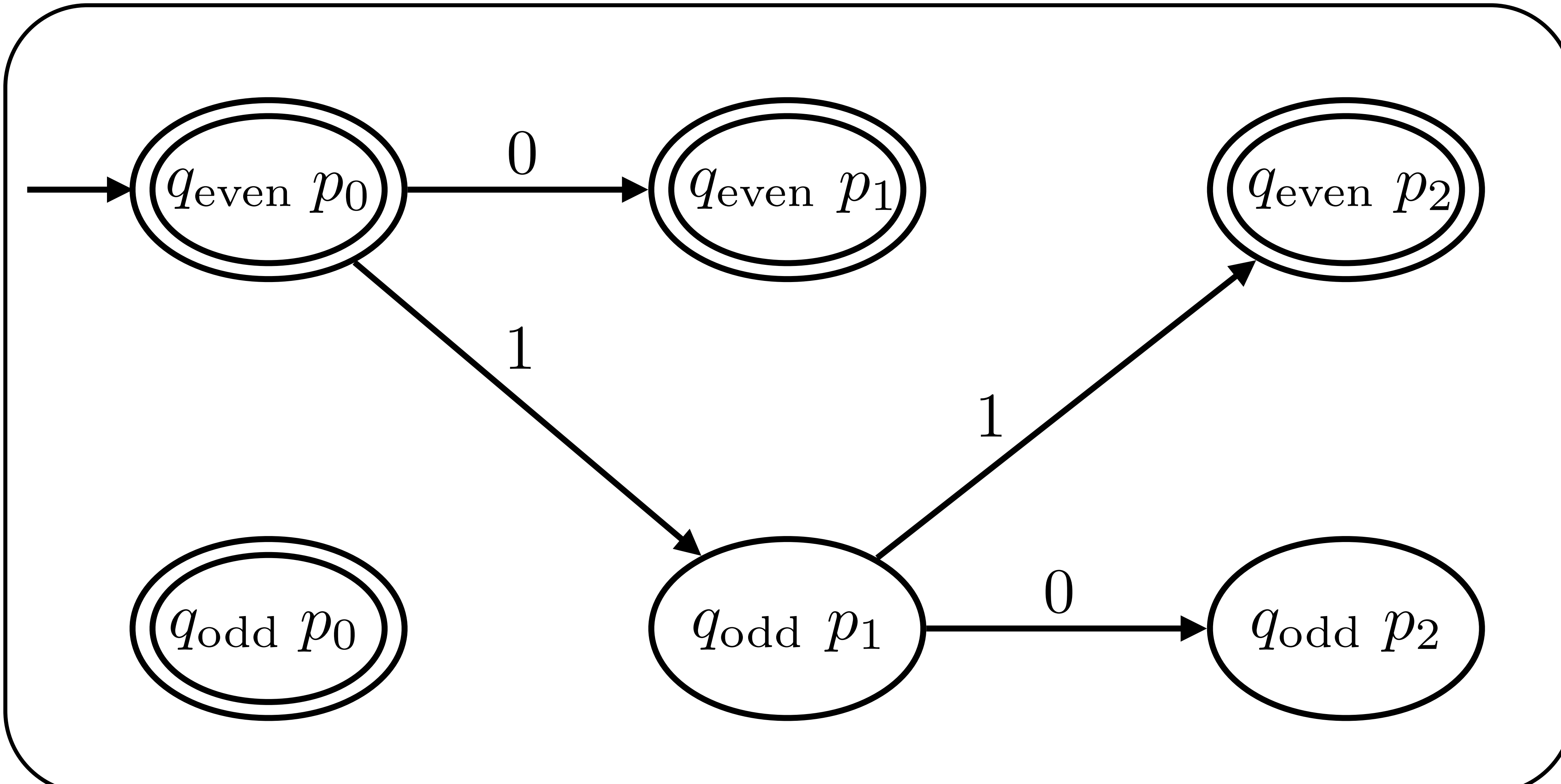
Construct a DFA that keeps track of both at once.



Closed under union

Main idea:

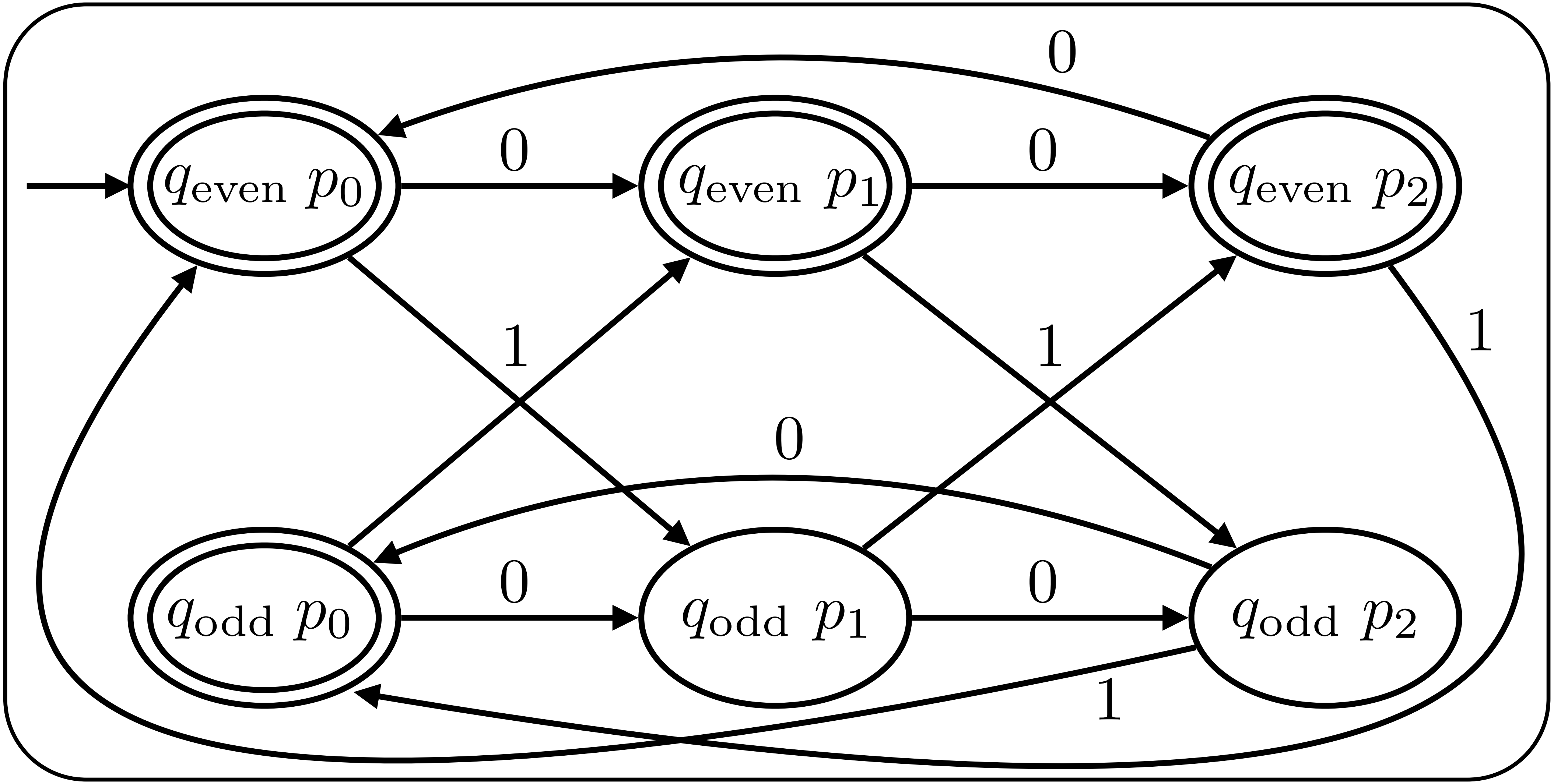
Construct a DFA that keeps track of both at once.



Closed under union

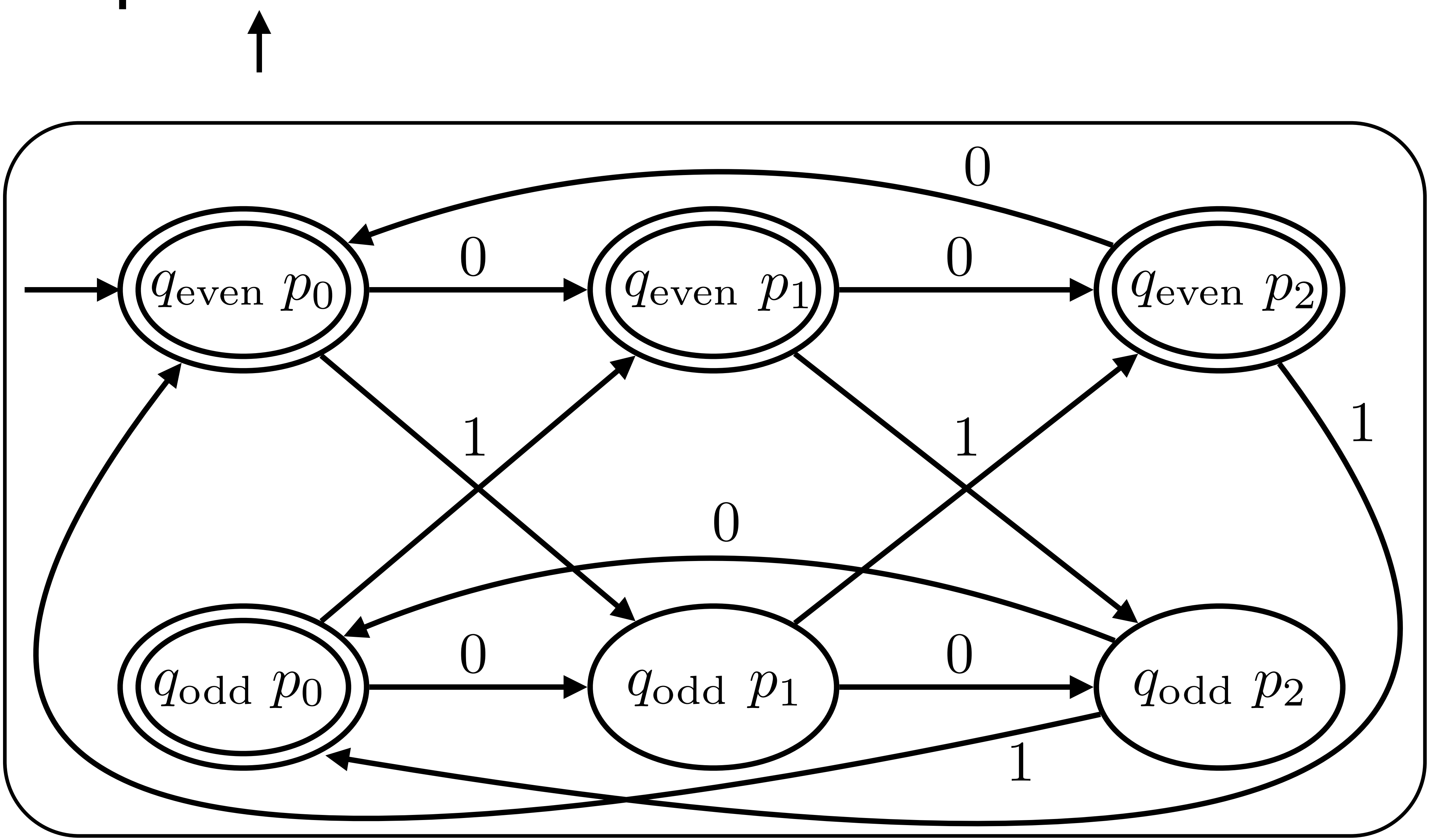
Main idea:

Construct a DFA that keeps track of both at once.



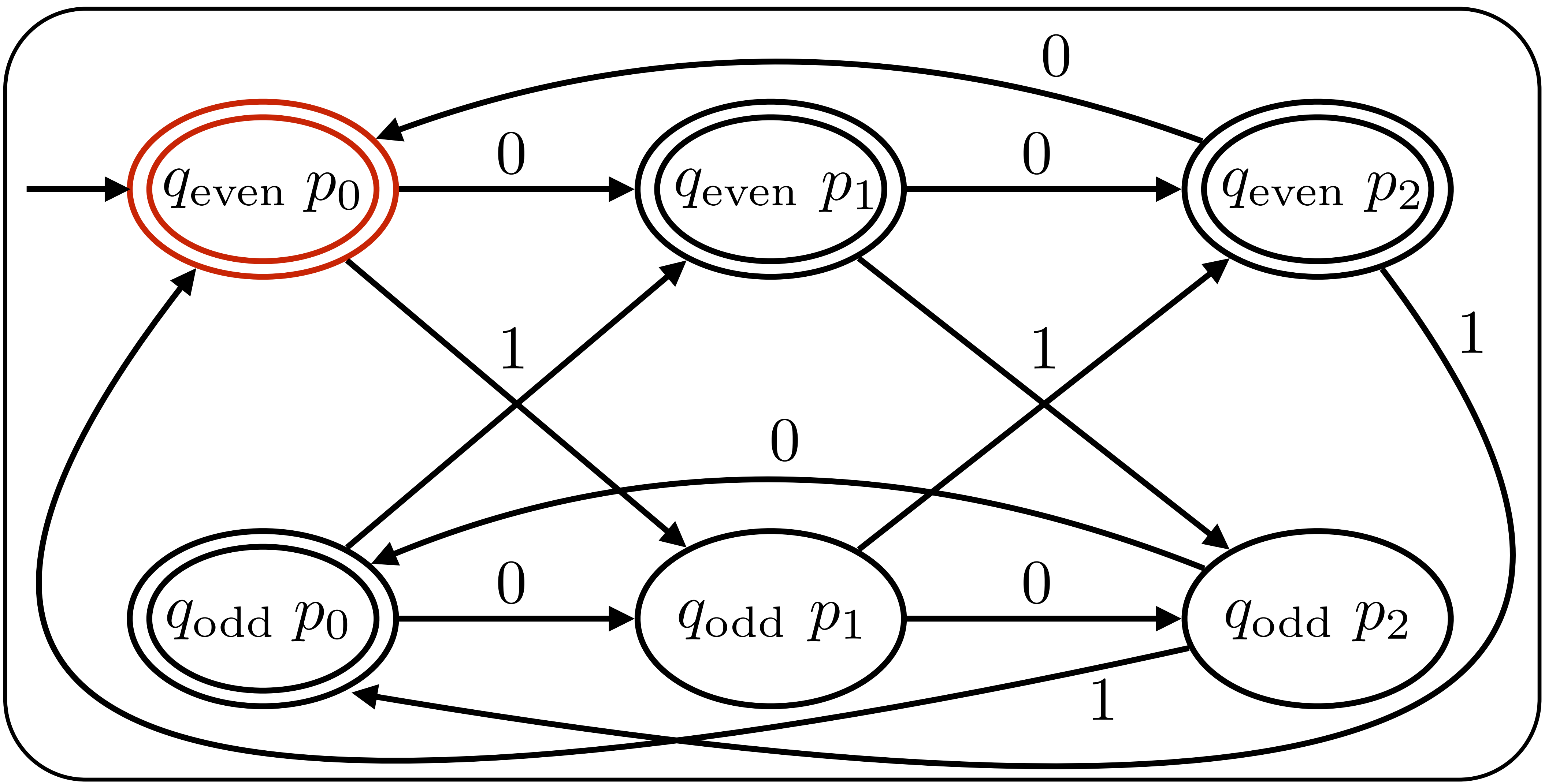
Closed under union

Input: 101001



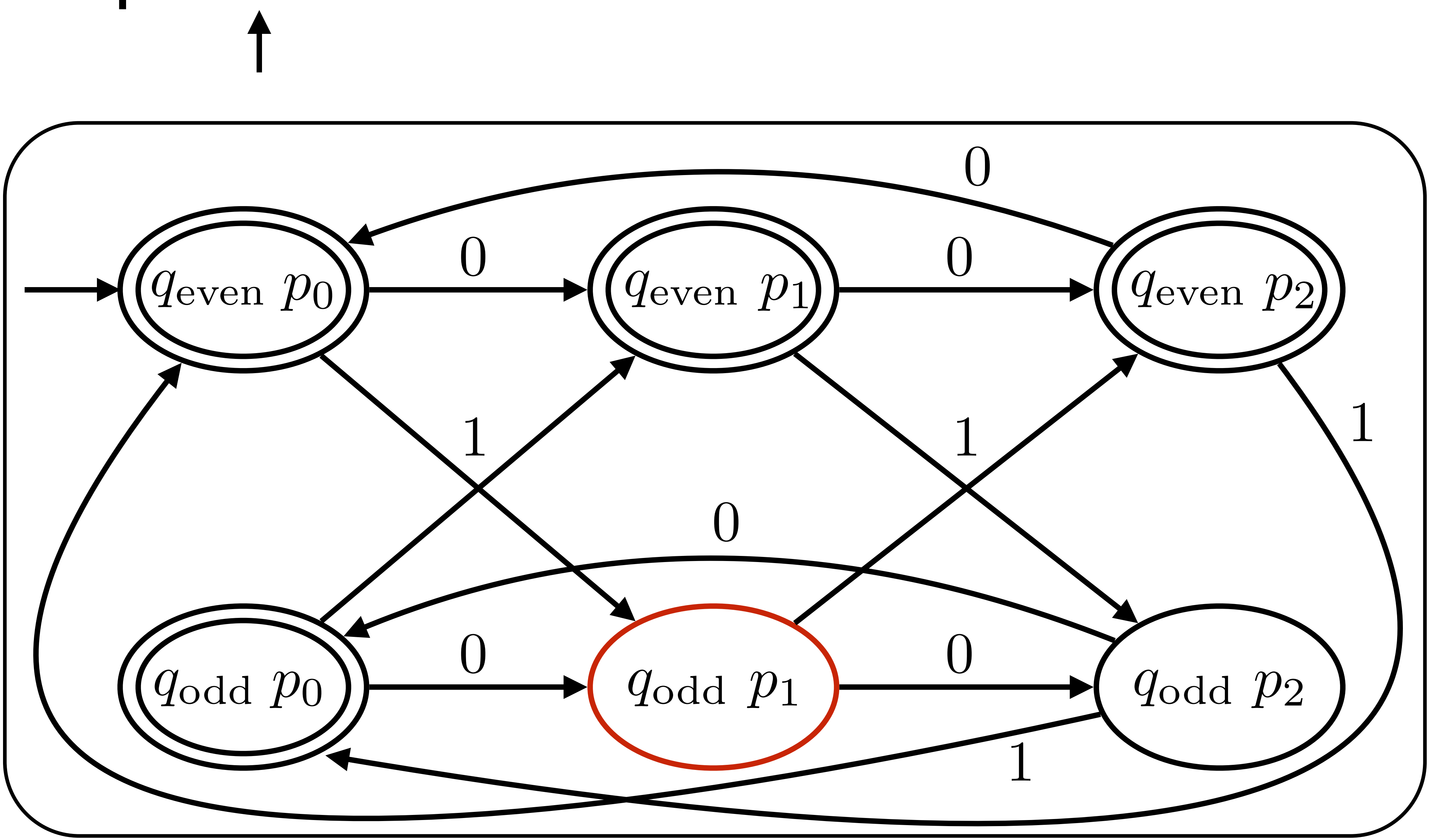
Closed under union

Input: 101001



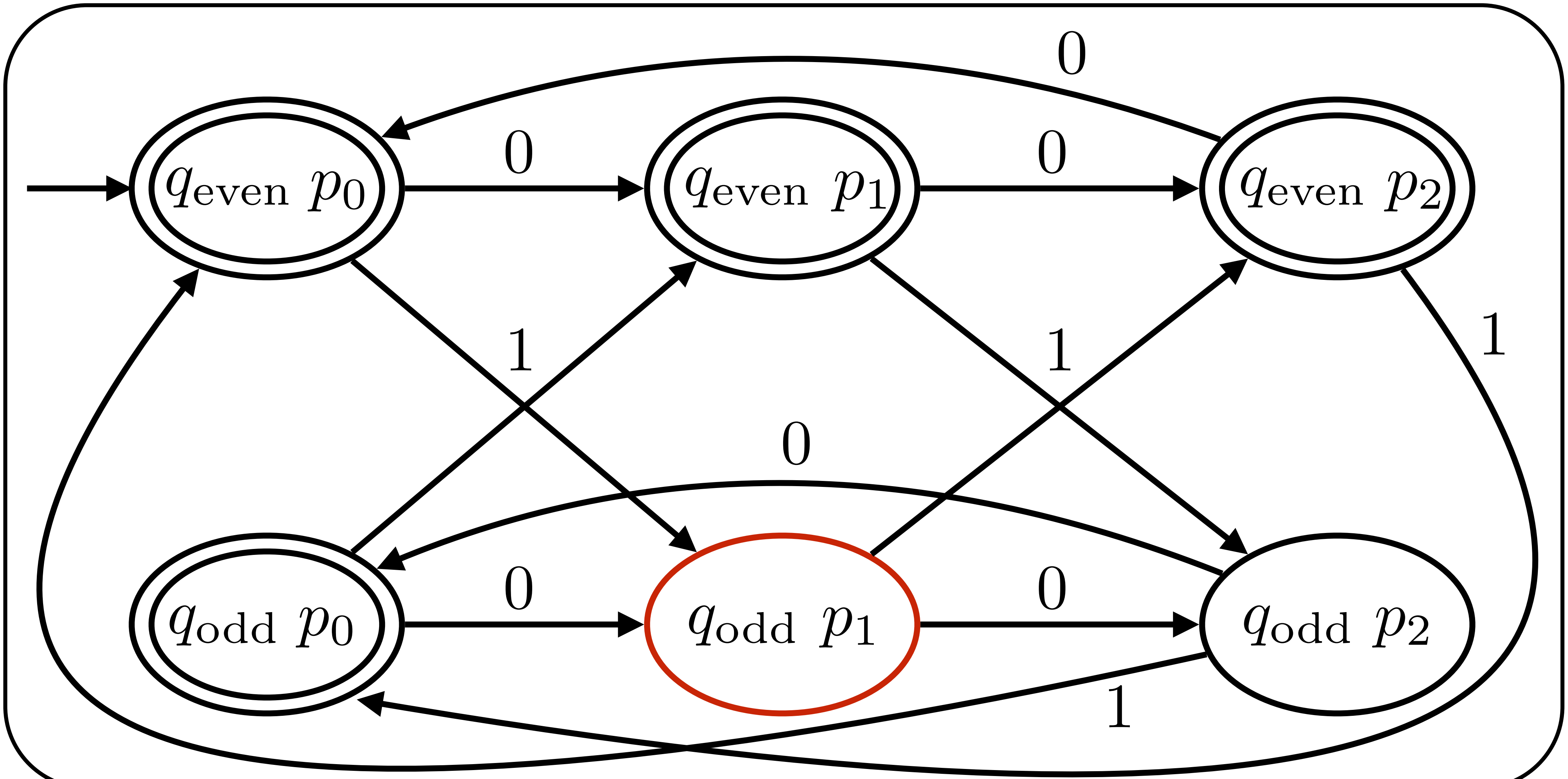
Closed under union

Input: 101001



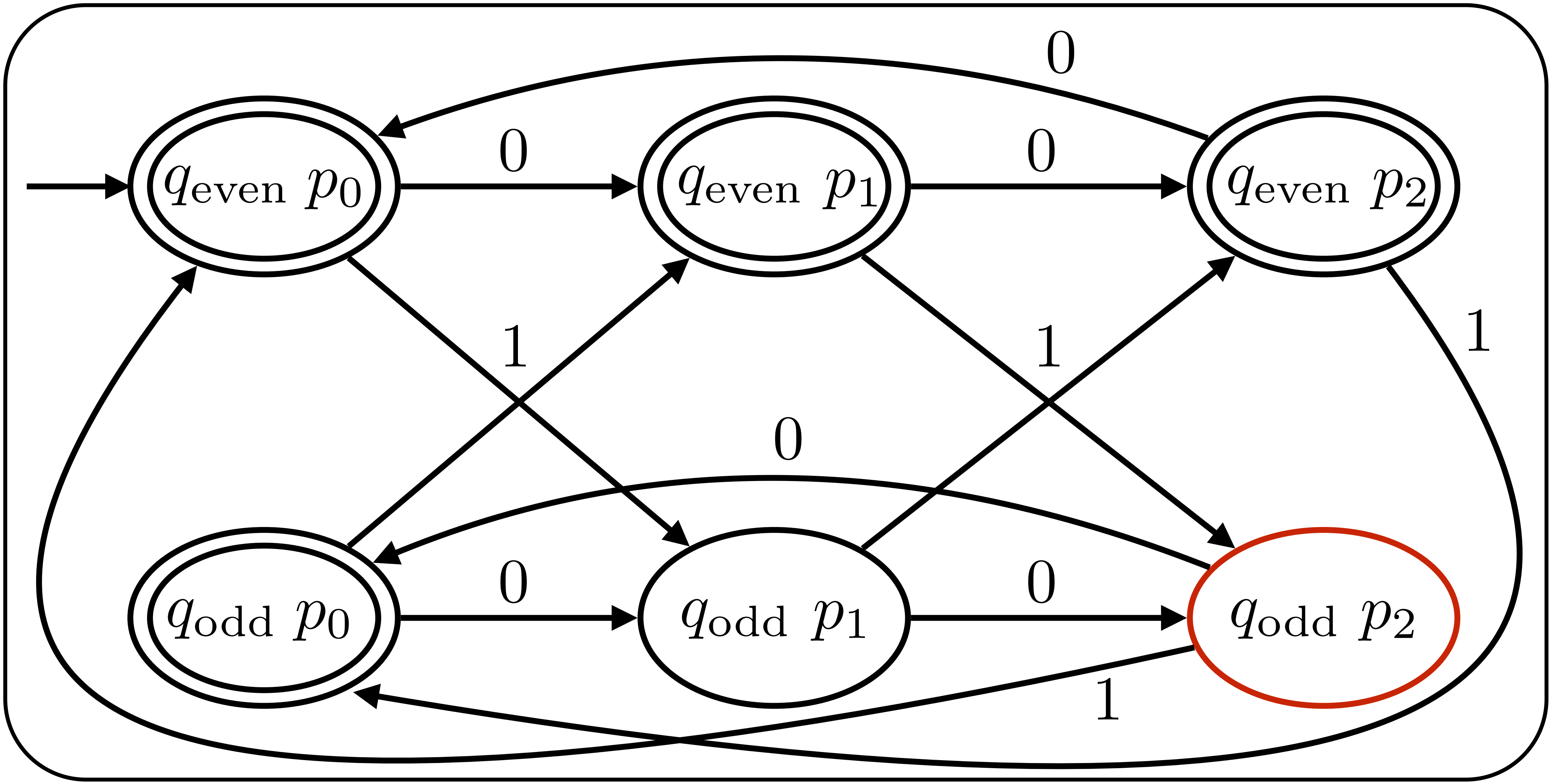
Closed under union

Input: 101001



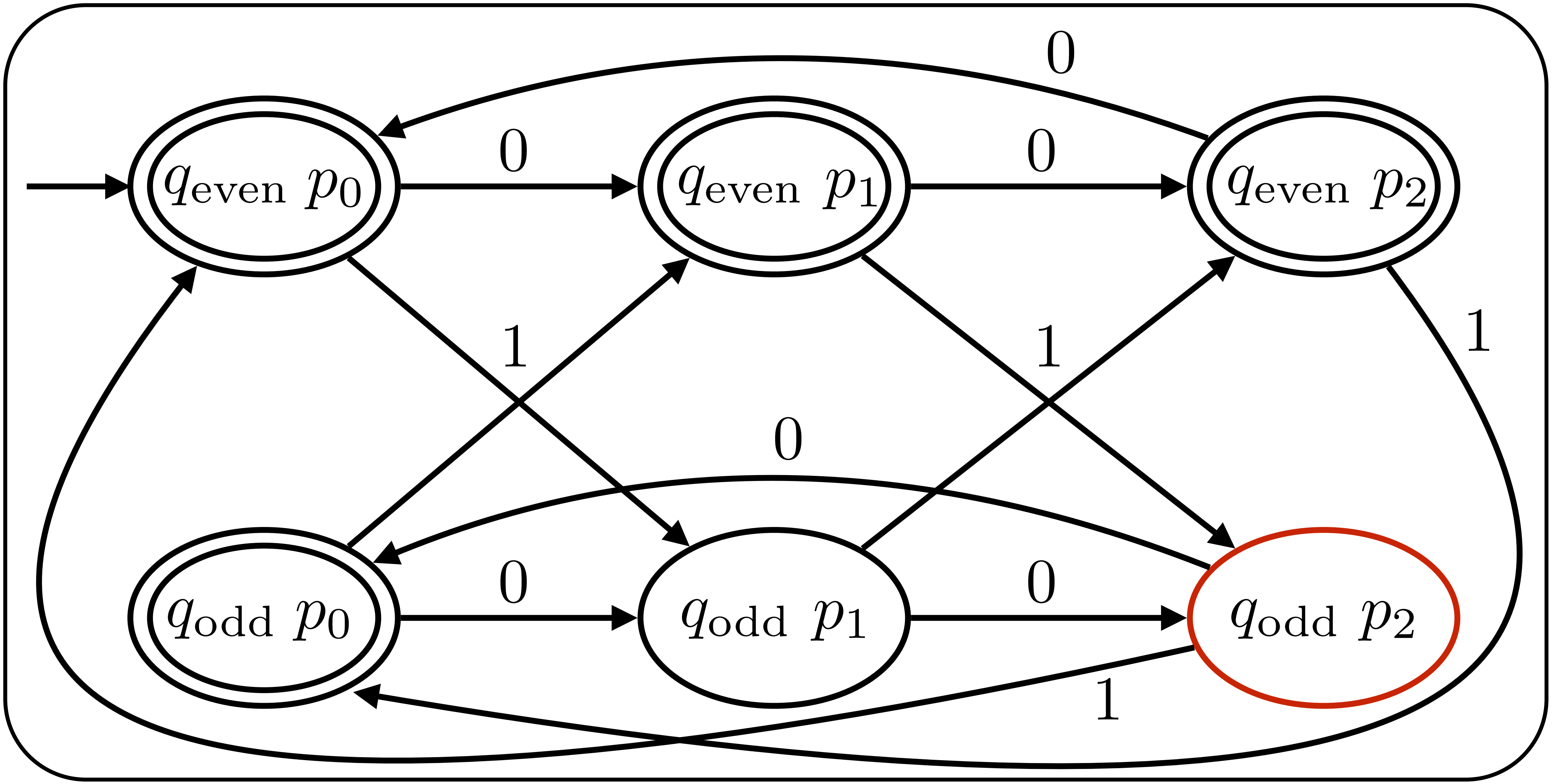
Closed under union

Input: 101001



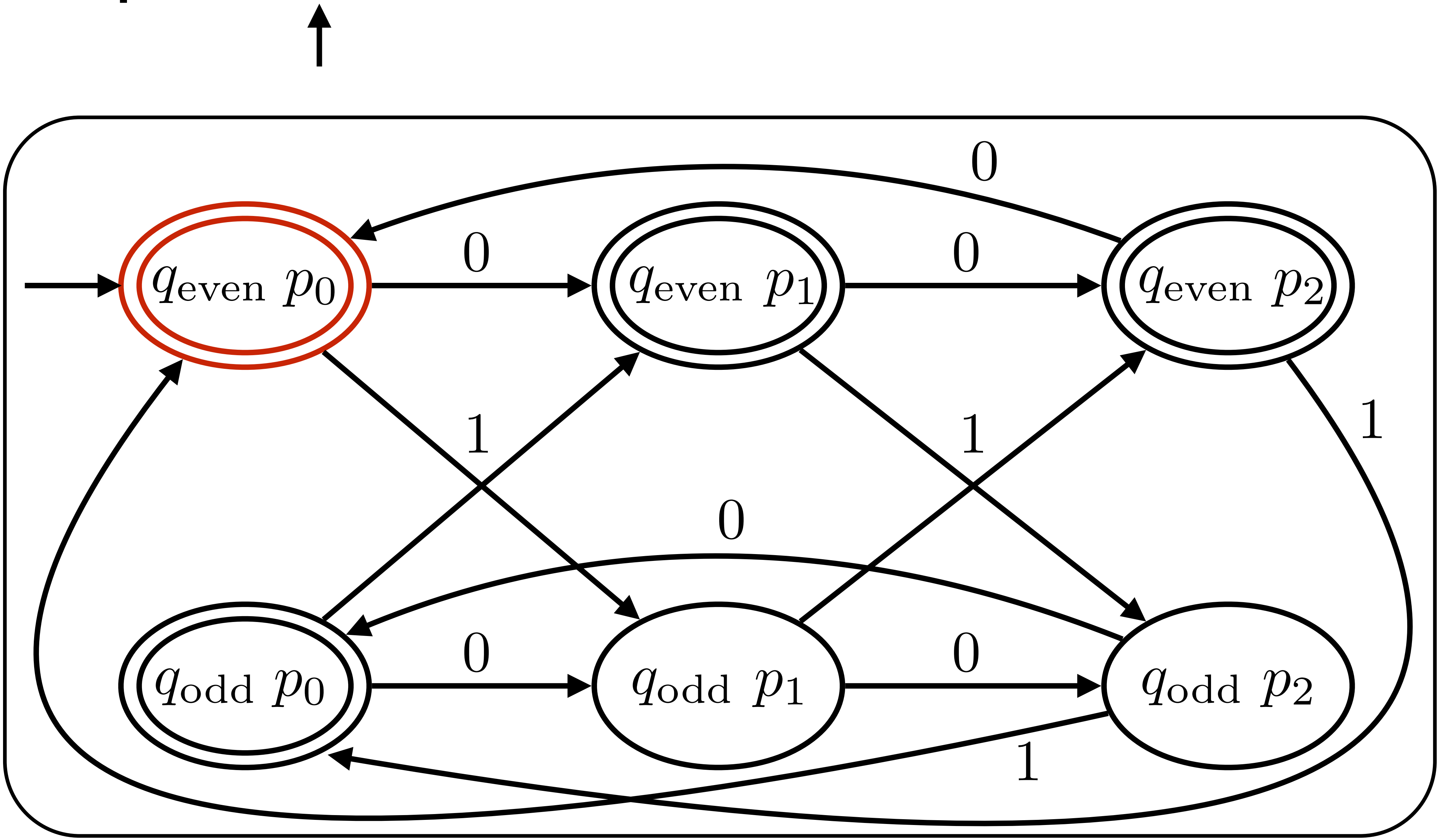
Closed under union

Input: 101001



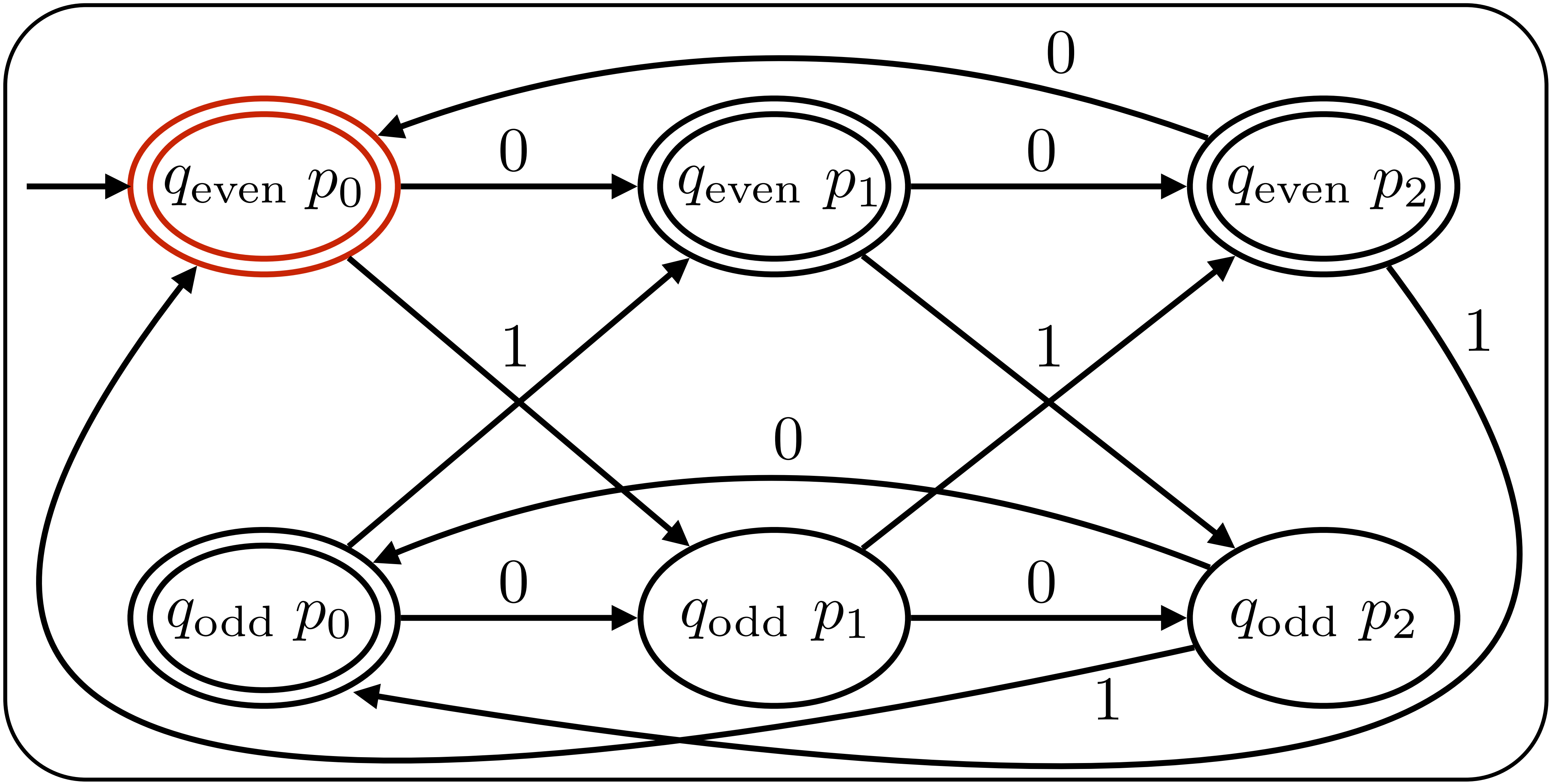
Closed under union

Input: 101001



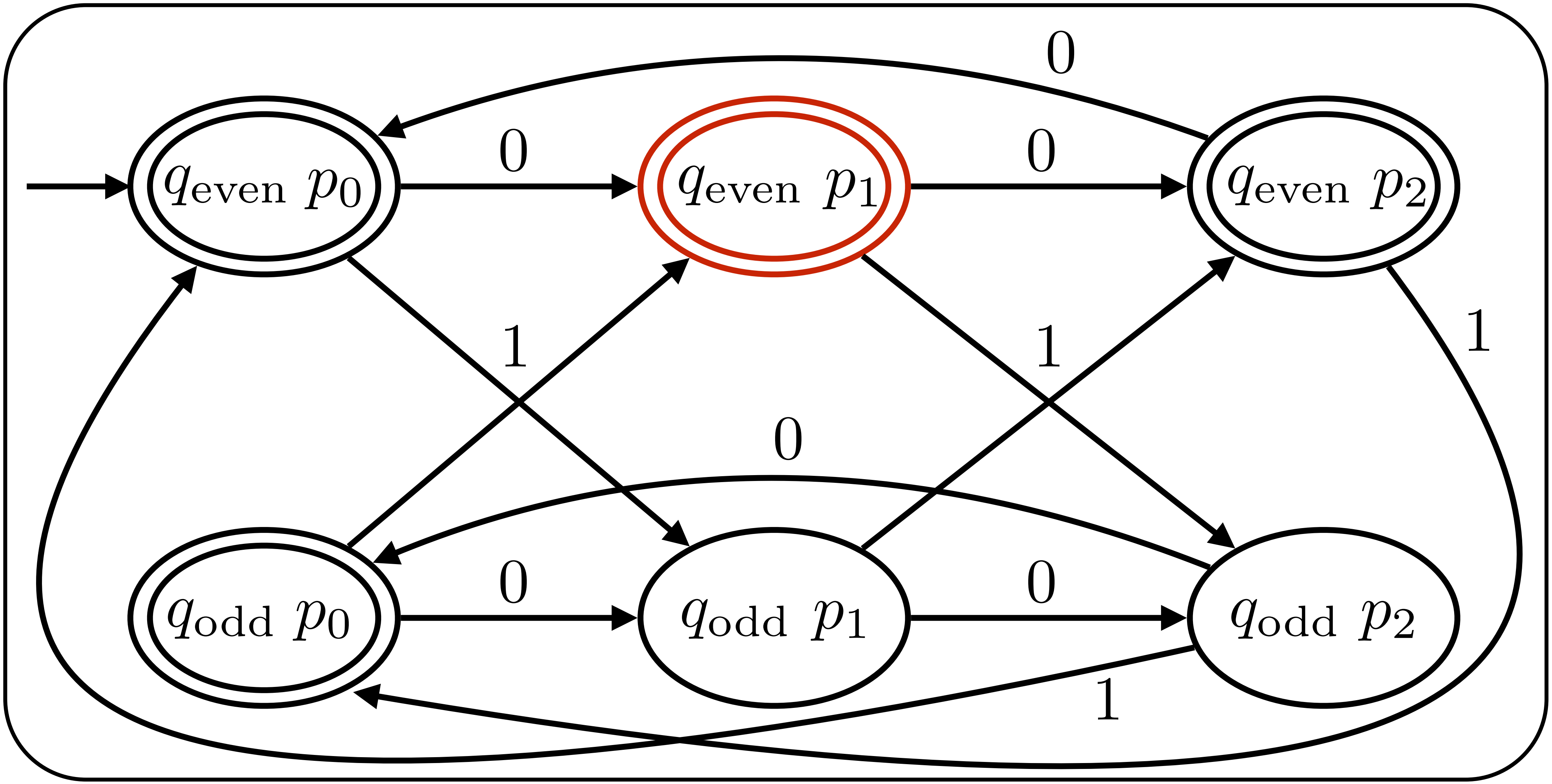
Closed under union

Input: 101001



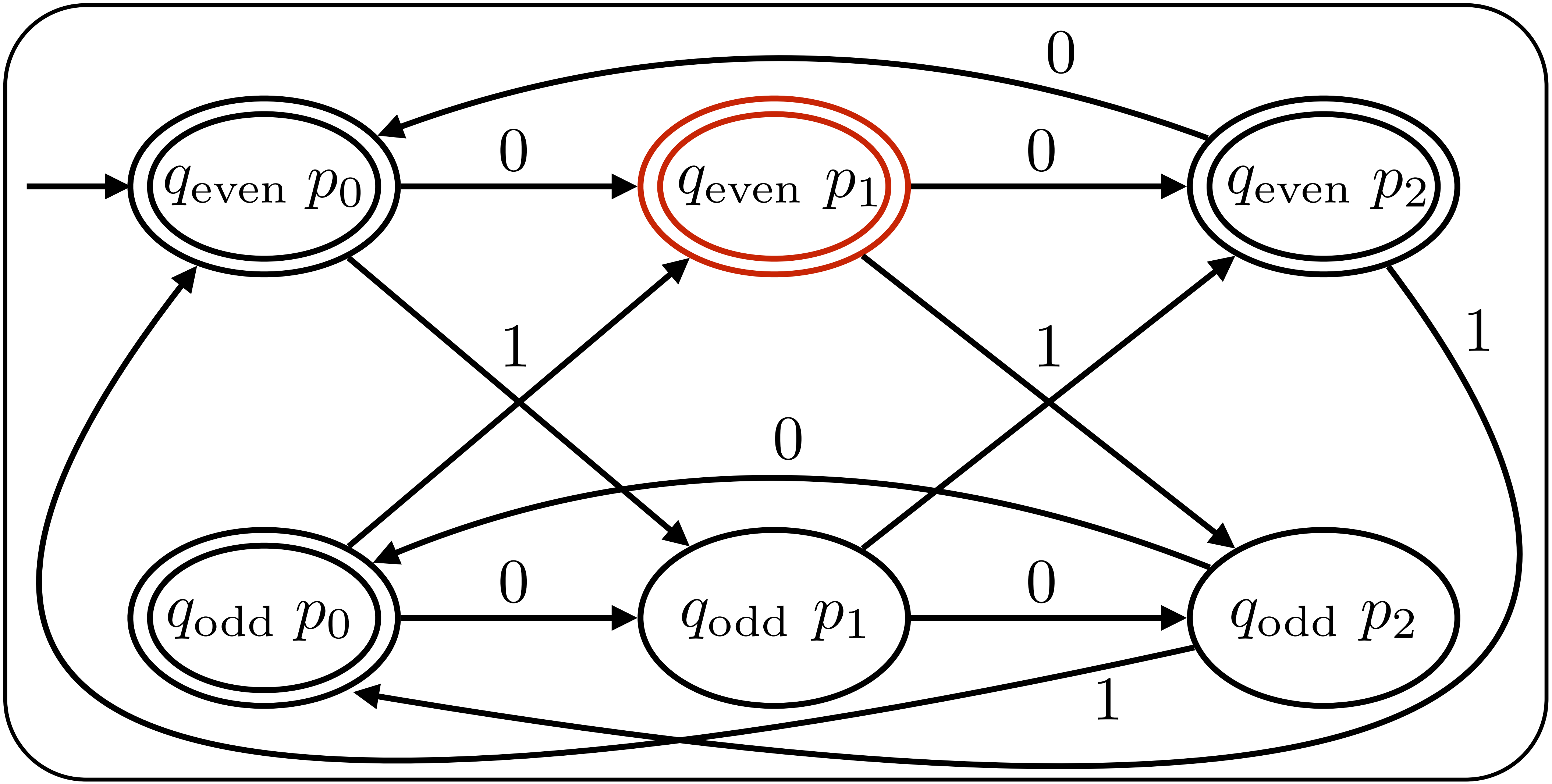
Closed under union

Input: 101001



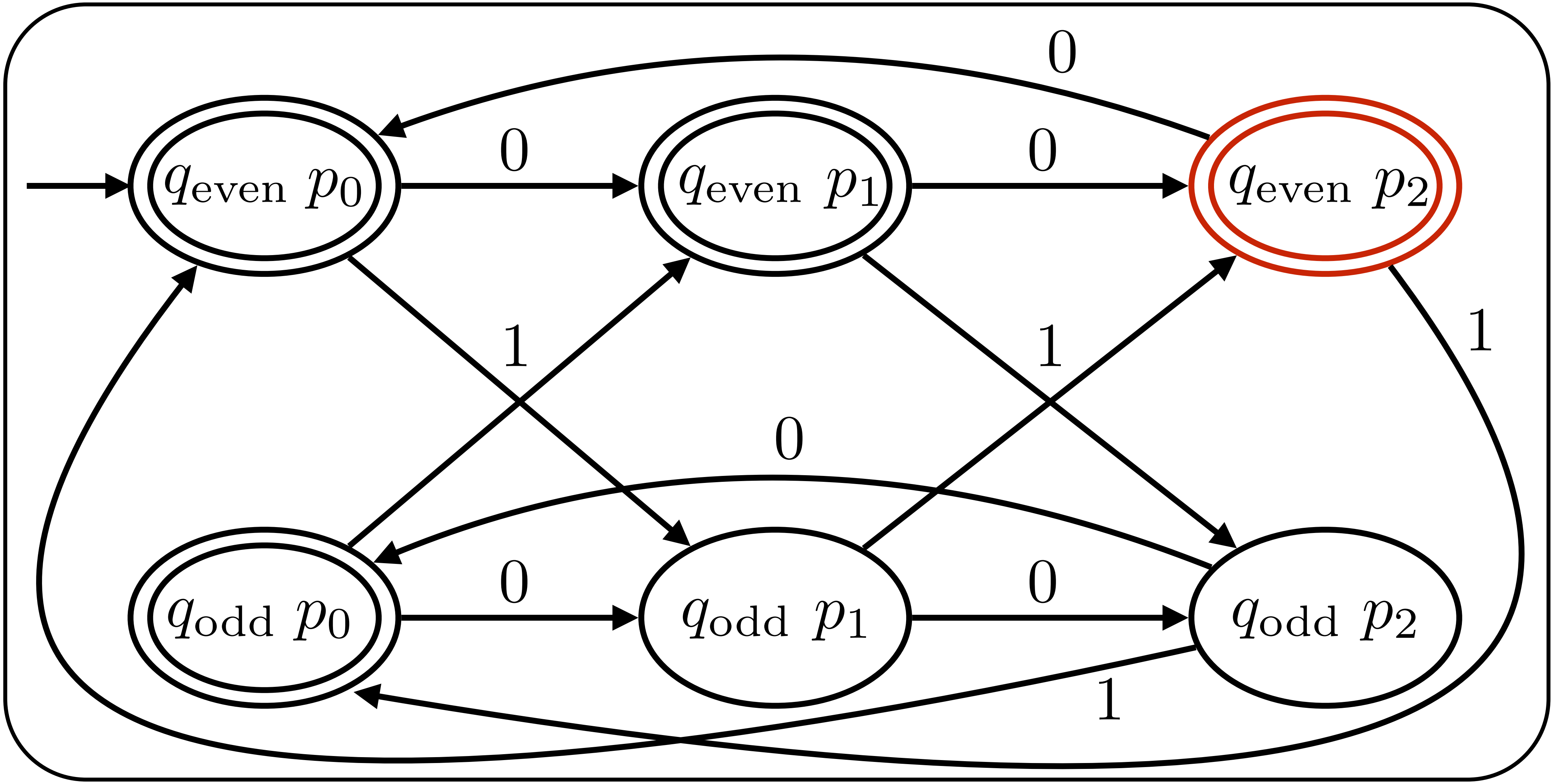
Closed under union

Input: 101001



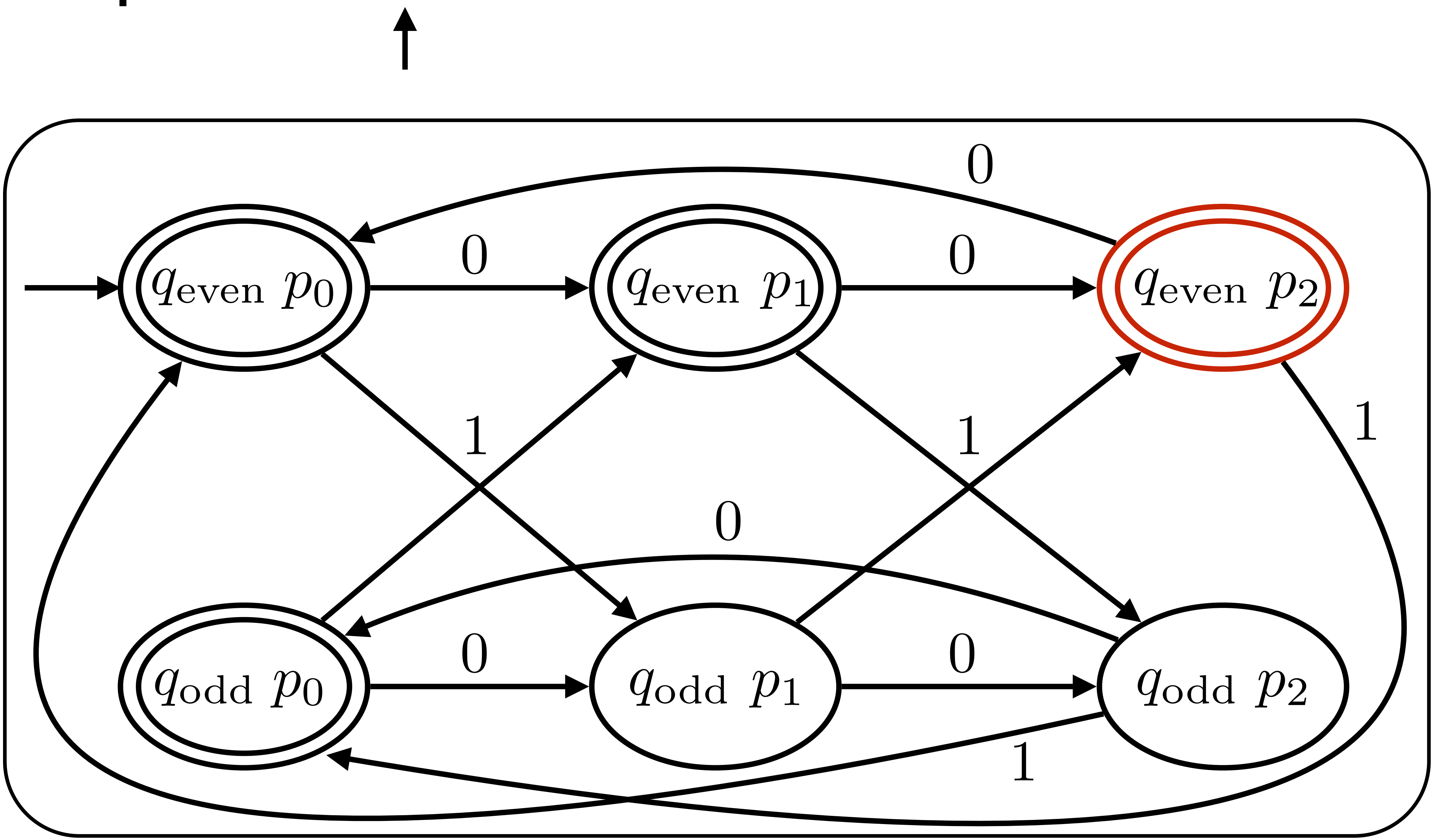
Closed under union

Input: 101001



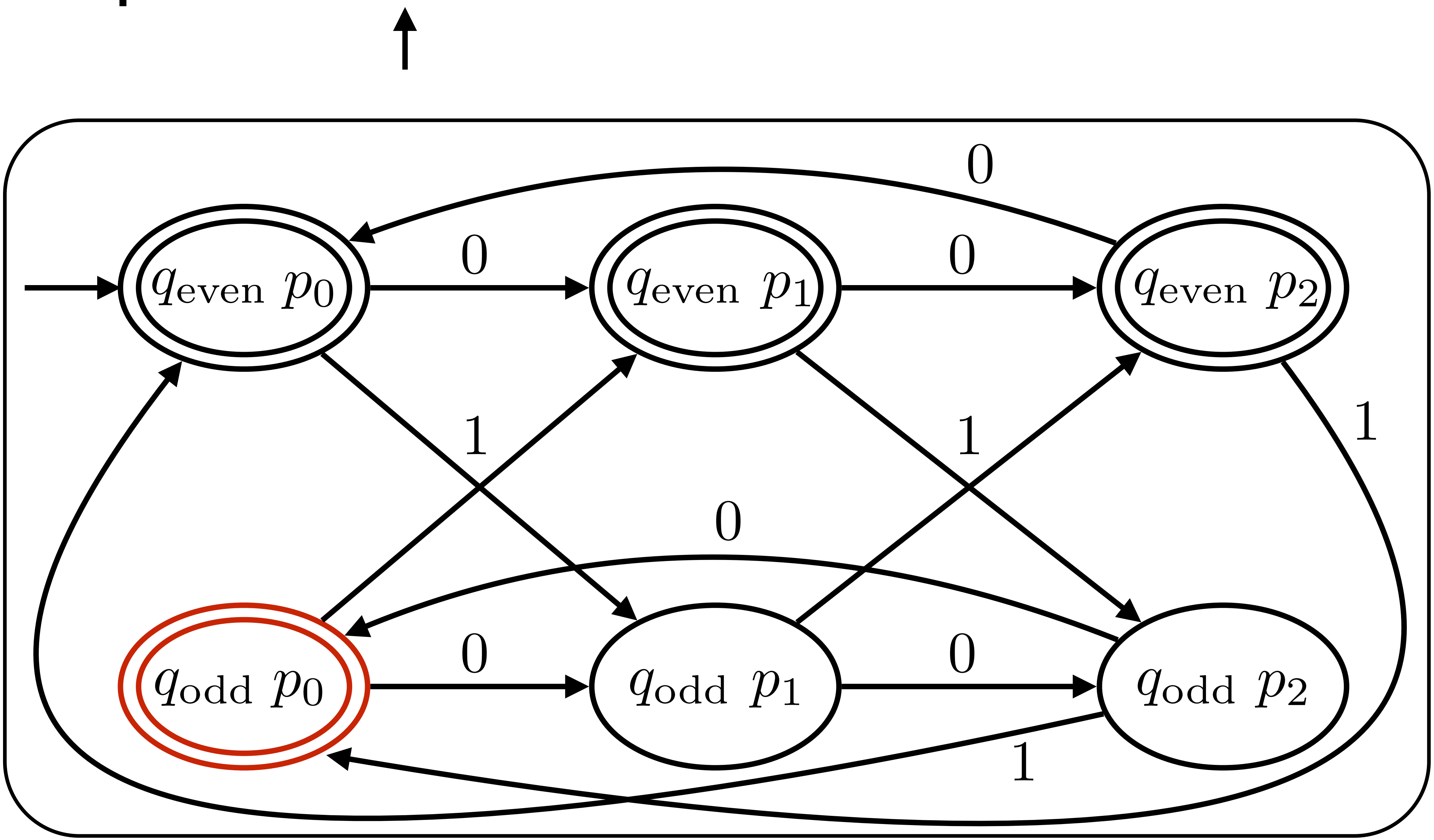
Closed under union

Input: 101001



Closed under union

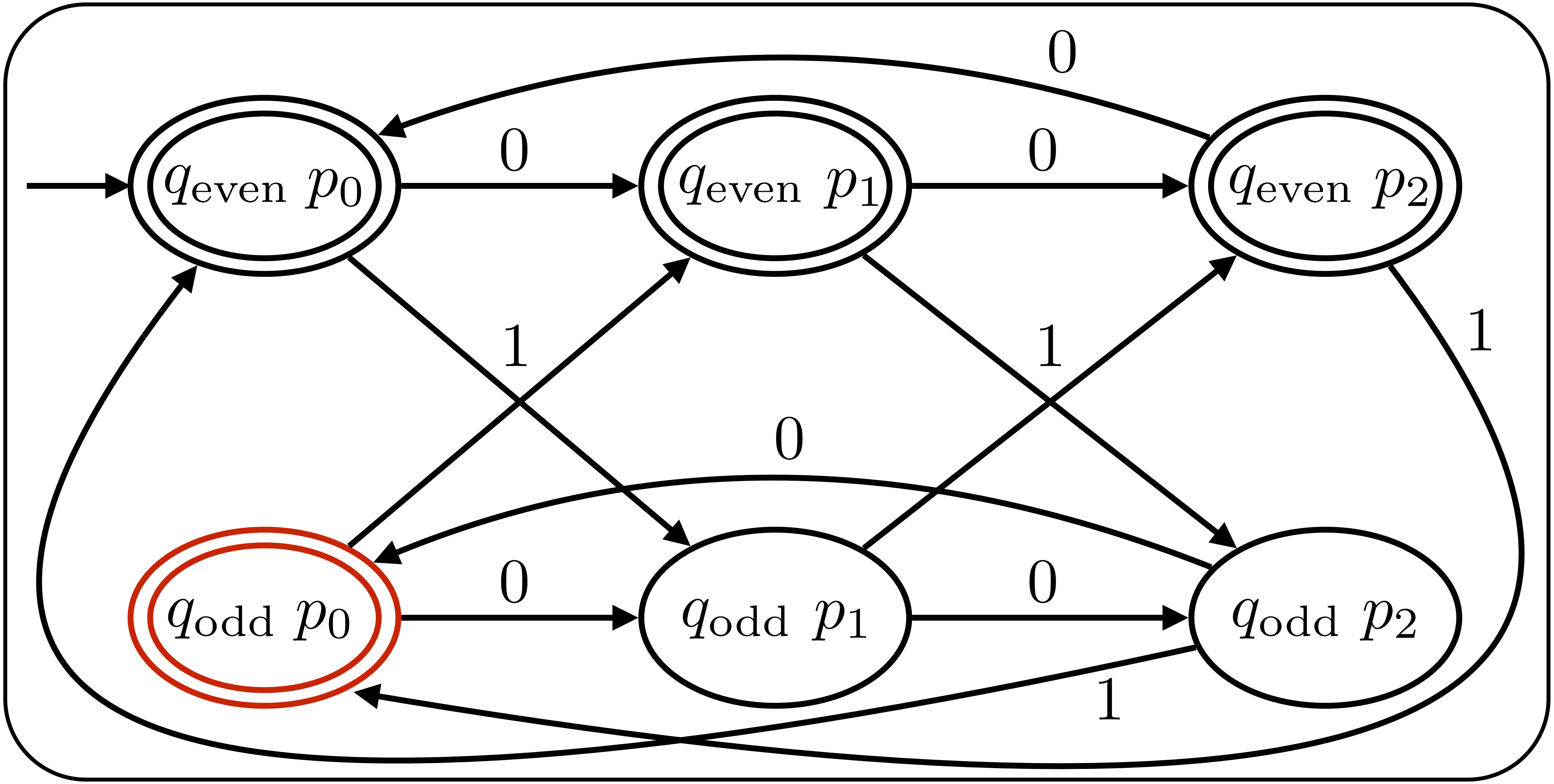
Input: 101001



Closed under union

Input: 101001

Decision: Accept



<u>**Step 2**</u>: Formally defining the DFA

Closed under union

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA deciding L_1 and $M' = (Q', \Sigma, \delta', q_0', F')$ be a DFA deciding L_2 .
We construct a DFA $M'' = (Q'', \Sigma, \delta'', q_0'', F'')$ that decides $L_1 \cup L_2$, as follows:

- $Q'' = Q \times Q' = \{(q, q') : q \in Q, q' \in Q'\}$
- $\delta''((q, q'), a) = (\delta(q, a), \delta'(q', a))$
- $q_0'' = (q_0, q_0')$
- $F'' = \{(q, q') : q \in F \text{ or } q' \in F'\}$

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

$L(M'') \subseteq L_1 \cup L_2 : \quad \ldots$

$L_1 \cup L_2 \subseteq L(M'') : \quad \ldots$

$\square$

Closed under intersection

Corollary:

Let Σ be some finite alphabet.
If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are regular, then so is $L_1 \cap L_2$.

Proof: Follows from:

- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
- regular languages are closed under complementation
- regular languages are closed under union

$\square$

Closed under intersection

Closure properties can be used to show languages are not regular.

Example:

Let $L \subseteq \{0,1\}^*$ be the language consisting of all words with an equal number of 0's and 1's.

We claim L is not regular. Suppose it was regular.

$$\{0^n 1^m : n, m \in \mathbb{N}\} \quad \cap \quad L \quad = \quad \{0^n 1^n : n \in \mathbb{N}\}$$

regular **regular** **regular** $\lightning$

contradiction

More closure properties

Closed under union:

$$L_1, L_2 \text{ regular} \implies L_1 \cup L_2 \text{ regular.}$$

$$L_1 \cup L_2 = \{x \in \Sigma^* : x \in L_1 \text{ or } x \in L_2\}$$

Closed under concatenation:

$$L_1, L_2 \text{ regular} \implies L_1 \cdot L_2 \text{ regular.}$$

$$L_1 \cdot L_2 = \{xy : x \in L_1, y \in L_2\}$$

Closed under star:

$$L \text{ regular} \implies L^* \text{ regular.}$$

$$L^* = \{x_1 x_2 \cdots x_k : k \geq 0, \forall i \; x_i \in L\}$$

awesome vs regular

What is the relationship between awesome and regular ?

$$\text{awesome} \subseteq \text{regular}$$

In fact:

$$\text{awesome} = \text{regular}$$

awesome = regular

Theorem:

Can define regular languages recursively as follows:

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

Regular expressions

Definition:

A regular expression is defined recursively as follows:

- $\emptyset$ is a regular expression.
- ϵ is a regular expression.
- For every $a \in \Sigma$, a is a regular expression.
- R_1, R_2 regular expr. $\implies$ $(R_1 \cup R_2)$ regular expr.
- R_1, R_2 regular expr. $\implies$ $(R_1 R_2)$ regular expr.
- R regular expr. $\implies$ (R^*) regular expr.

Regular expressions

Examples:

$$(((0 \cup 1)^* 1)(0 \cup 1)^*) \;=\; \Sigma^* 1 \Sigma^*$$

$$\{w \in \{0,1\}^* : w \text{ has at least one } 1\}$$

$$0^* 1 0^*$$

$$\{w \in \{0,1\}^* : w \text{ has exactly one } 1\}$$

$$0\Sigma^* 0 \cup 1\Sigma^* 1 \cup 0 \cup 1$$

$$\{w \in \{0,1\}^* : w \text{ starts and ends with same symbol}\}$$

Closed under concatenation

Theorem:

Let Σ be some finite alphabet.

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

The mindset

Imagine yourself as a DFA.

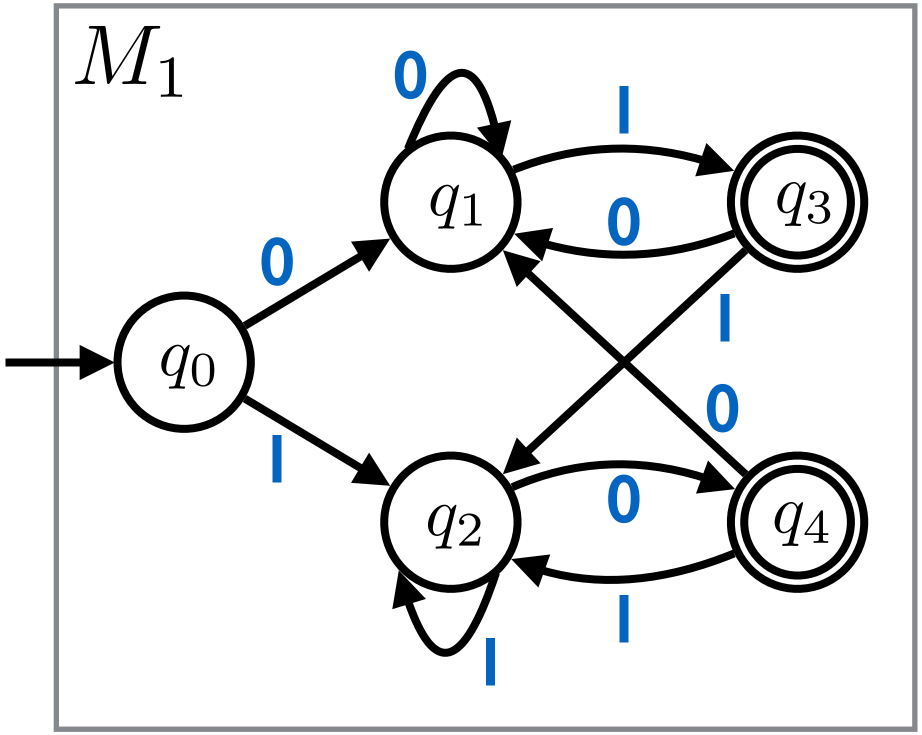
Rules:

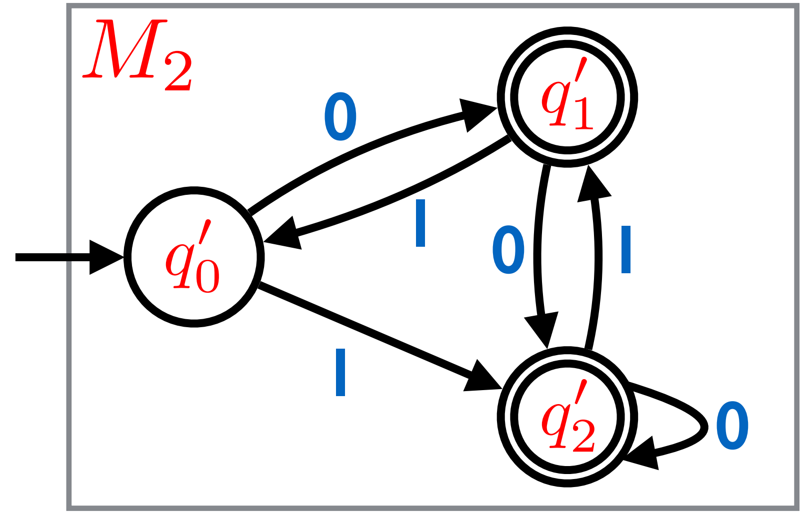
1) Can only scan the input once, from left to right.

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

should not change
based on input length

Step 1: Imagining ourselves as a DFA



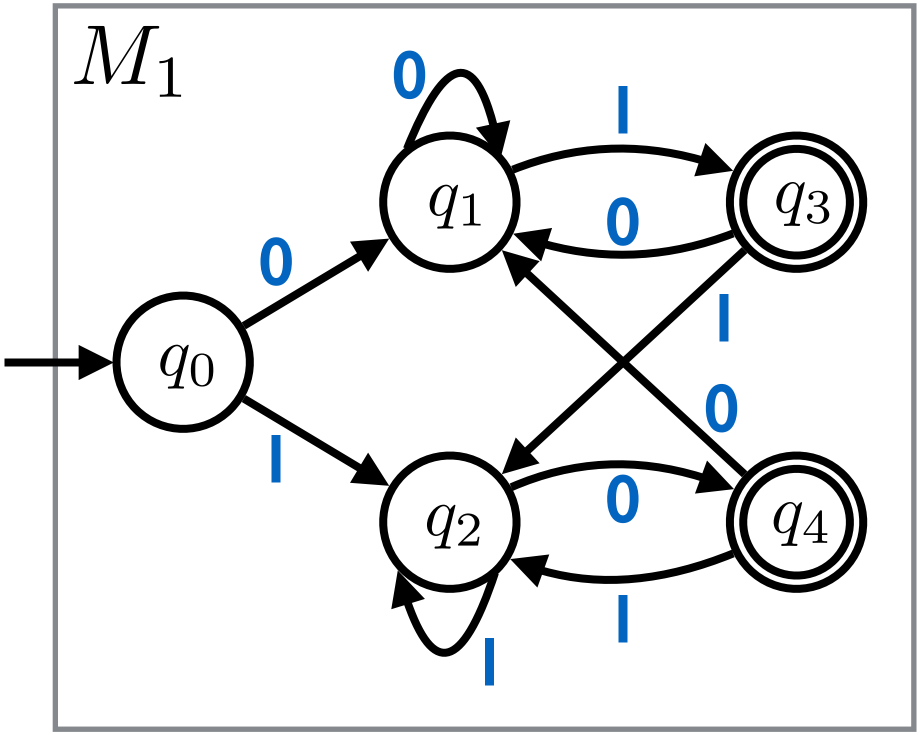


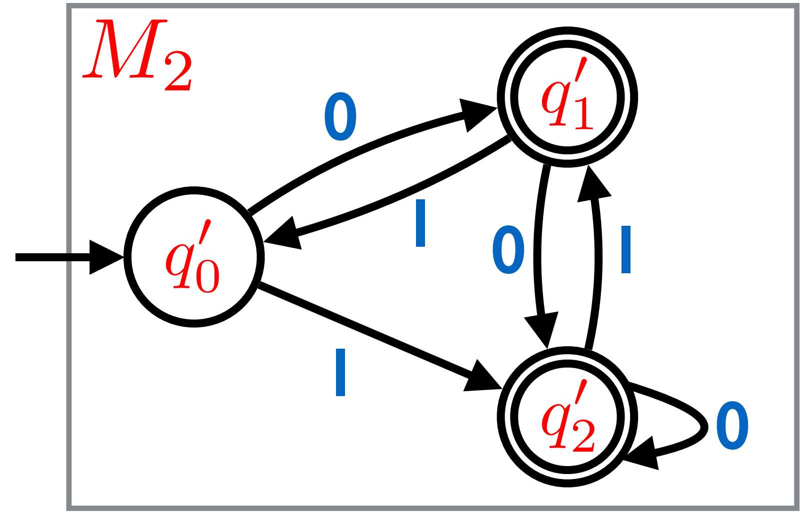
Given $w \in \Sigma^*$, we need to decide if

$$w = uv \quad \text{for} \quad u \in L_1,\ v \in L_2.$$

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

When do you stop simulating M_1 and start simulating M_2?

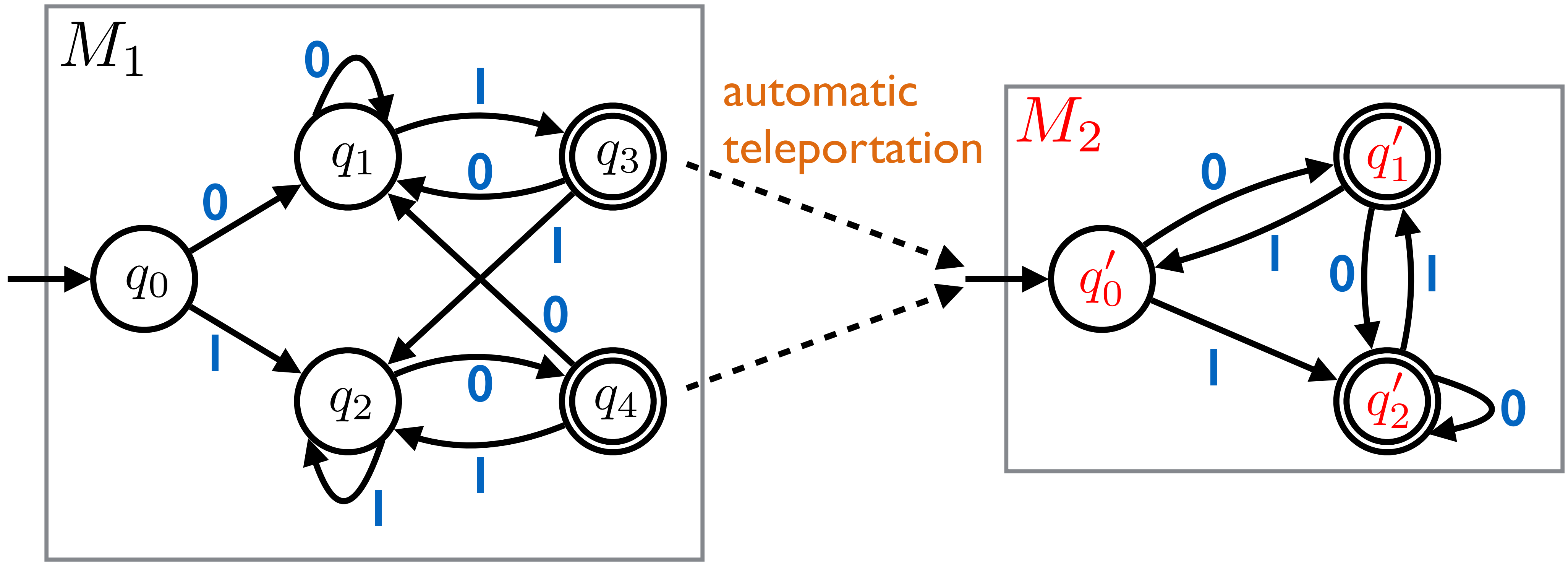




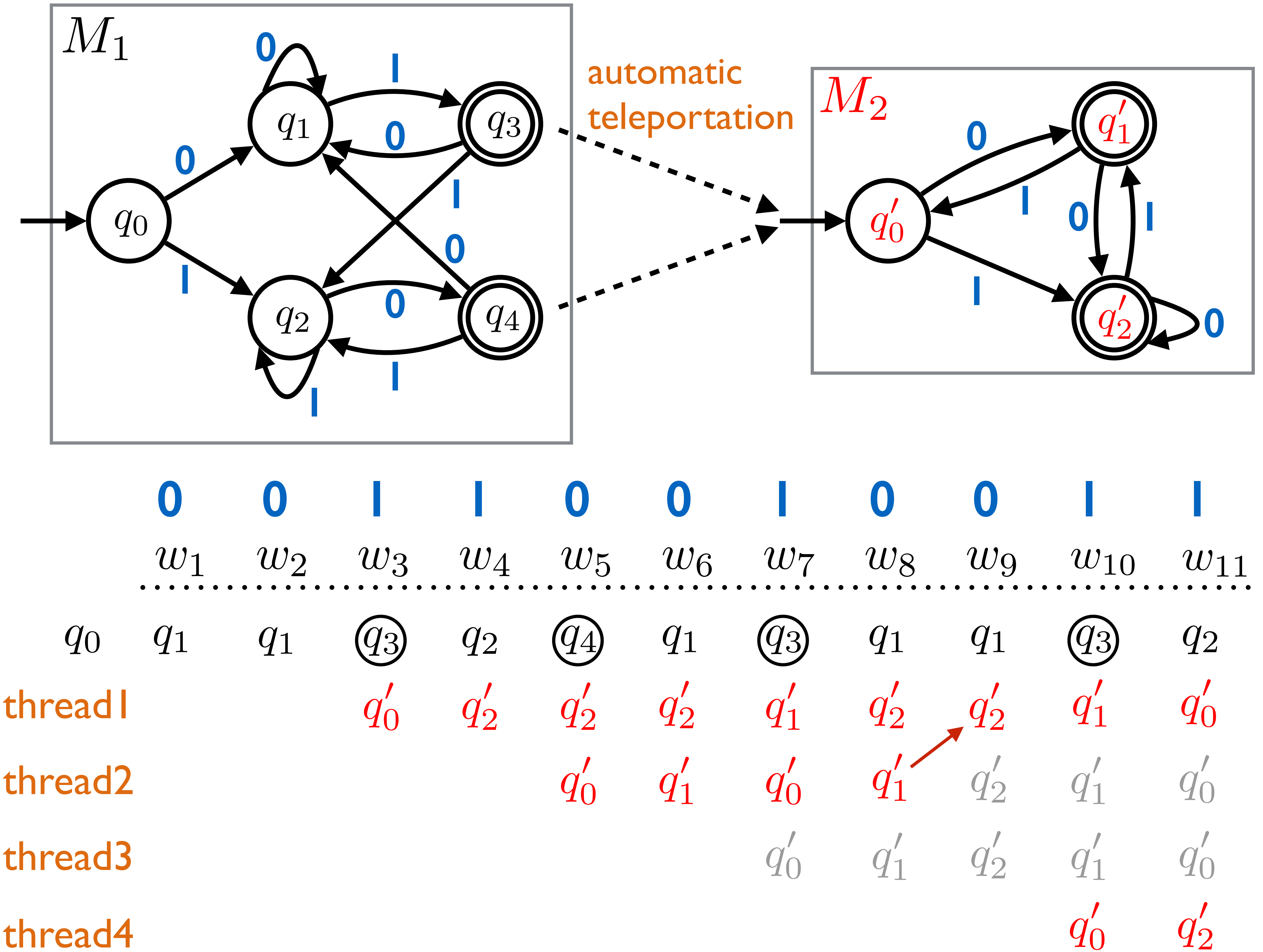
Suppose God tells you u ends at w_3.

	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

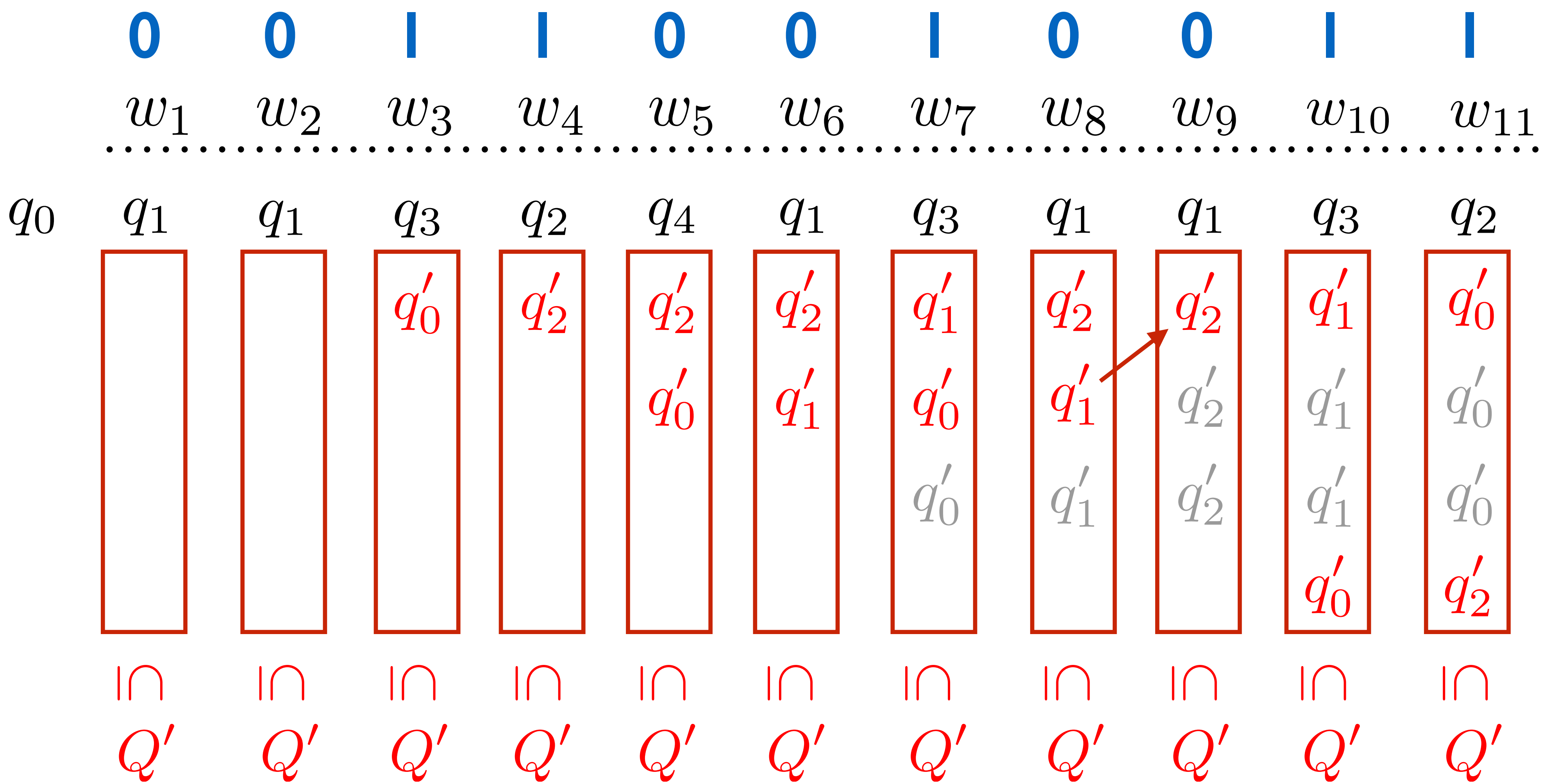
thread: a simulation of M_1 and then M_2 that corresponds to breaking up w into uv where $u \in L_1$.



		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	q_3	q_2	q_4	q_1	q_3	q_1	q_1	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'



M_1
q_0 q_1 q_2 q_3 q_4
0 1
automatic
teleportation
M_2
q'_0 q'_1 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 q_3 q_2 q_4 q_1 q_3 q_1 q_1 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



This keeps track of every possible thread.

At any point, need to remember:

- an element of Q
- a subset of Q'

constant amount of information

<u>Step 2</u>: Formally defining the DFA

$M_1 = (Q, \Sigma, \delta, q_0, F)$ $\quad$ $M_2 = (Q', \Sigma, \delta', q_0', F')$

$$Q'' = Q \times \mathcal{P}(Q')$$

$$\delta'' : Q \times \mathcal{P}(Q') \times \Sigma \to Q \times \mathcal{P}(Q')$$

for $q \in Q$, $S \in \mathcal{P}(Q')$, $a \in \Sigma$

$$(q, S, a) \xrightarrow{\delta''} (\delta(q,a), \{\delta'(s,a) : s \in S\}) \quad \text{if } \delta(q,a) \notin F$$

$$(q, S, a) \xrightarrow{\delta''} (\delta(q,a), \{\delta'(s,a) : s \in S\} \cup \{q_0'\}) \quad \text{otherwise}$$

$$q_0'' = (q_0, \emptyset) \quad \text{if } q_0 \notin F$$

$$q_0'' = (q_0, \{q_0'\}) \quad \text{otherwise}$$

$$F'' = \{(q, S) : q \in Q,\ S \in \mathcal{P}(Q'),\ S \cap F' \neq \emptyset\}$$

Next Time