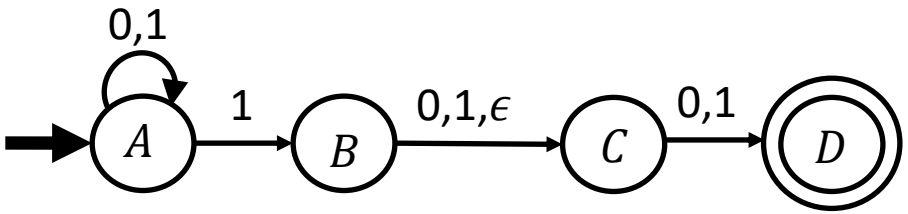
Administrivia

Read Section 1.3 of the textbook.

Converting an NFA to a DFA



What is the language $L(N)$? $\{w: w \text{ has at least 2 symbols and either the second last or third last symbol is } 1\}$

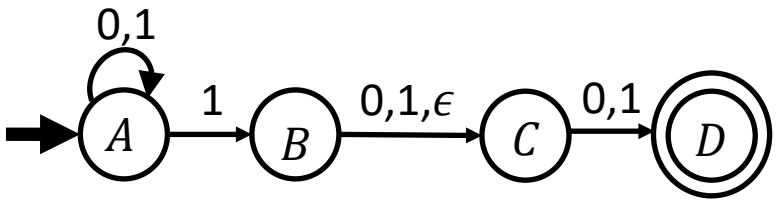
At any step, the machine can be in multiple states.

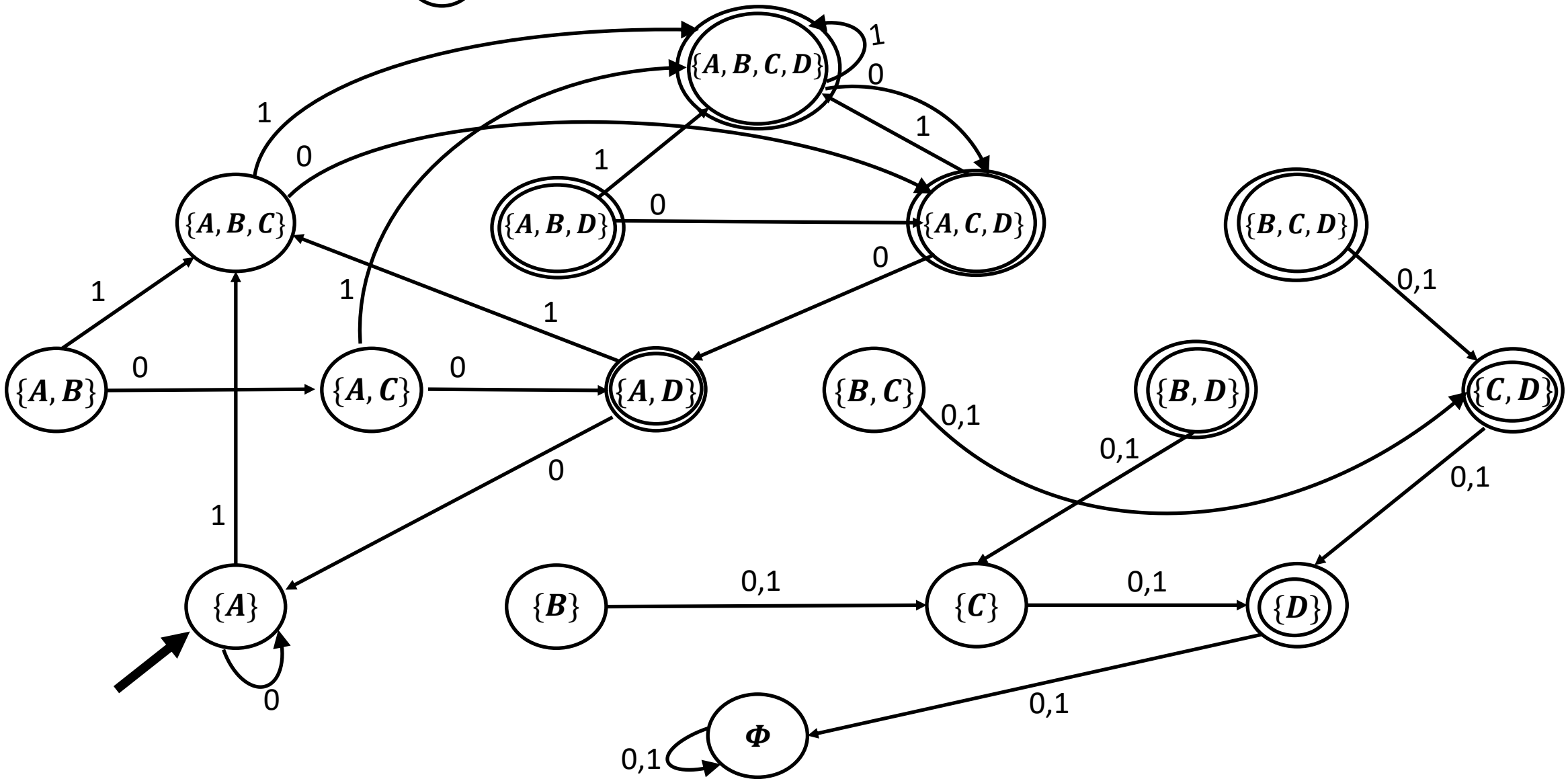
One state of the DFA for each subset of states of the NFA

The start state of the DFA is $\{A\}$

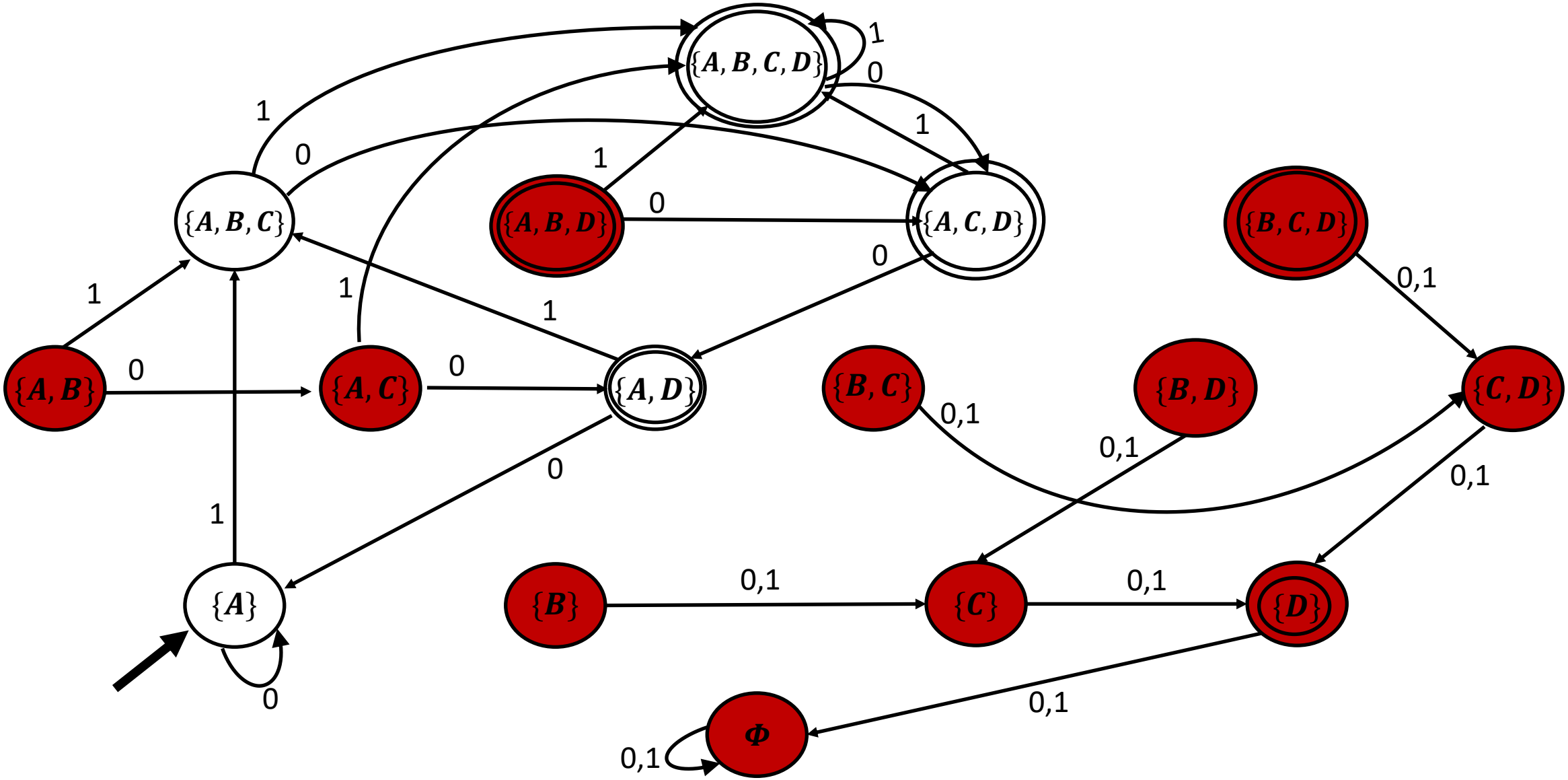
Every DFA state that represents a subset containing D is an accept state.

Constructing the Equivalent DFA M





Removing Redundant States of M



Equivalence of NFA and DFAs

Theorem 1.39. Every NFA has an equivalent DFA.

Proof Sketch: Let NFA $N = (Q, \Sigma, \delta, q_0, F)$.

Define DFA $M = (\wp(Q), \Sigma, \delta', q_0', F')$

$F' = \{R \subseteq Q: R \cap F \neq \Phi\}$ all subsets that contain a final state of N.

From state $r \in Q, a \in \Sigma$:

$\delta(r, a) =$ set of states in N directly following an a-transition from r

$\delta'(R, a) =$ set of states in N reached by following an a-transition from any state in $R \subseteq Q$

$= \bigcup_{r \in R} \delta(r, a)$

But what about freebies: States reached by ϵ-transitions following the a-transition?

Equivalence of NFA and DFAs … 2

Define:

$E(R) = \{q \in N: q$ is reachable from $r \in R$ using a sequence of 0 or more ϵ-transitions$\}$

Now, $\delta'(R, a) = E(\bigcup_{r \in R} \delta(r, a))$

$= \bigcup_{r \in R} E(\delta(r, a)$

Finally, $q_0' = E(\{q_0\})$

Putting it All Together

Definition: Regular Languages are those recognized by DFAs.

Obvious Fact: Every DFA is also an NFA

Last Lecture: If A, B are each recognized by an NFA then so are $A \cup B, A \circ B, A^*$

From today: every NFA has an equivalent DFA

Therefore, if A, B are each recognized by a DFA then so are $A \cup B, A \circ B, A^*$

In other words, regular languages are closed under regular operations

Moral: Think NFA!

Question: So is there any sense in which NFAs more powerful than DFAs?

DFA vs. NFA

Every NFA can be converted into an equivalent DFA.

(Their languages are the same.)

NFAs are often simpler to design.

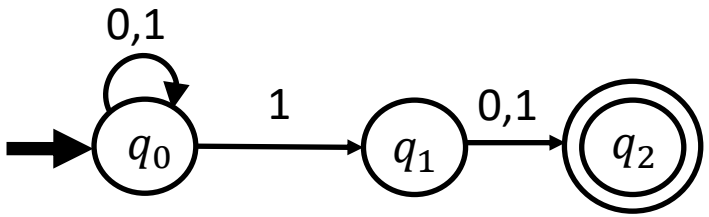
Question: Is there any sense in which NFAs more powerful than DFAs?

We simulated a k-state NFA by a 2^k-state DFA. Can we do better?

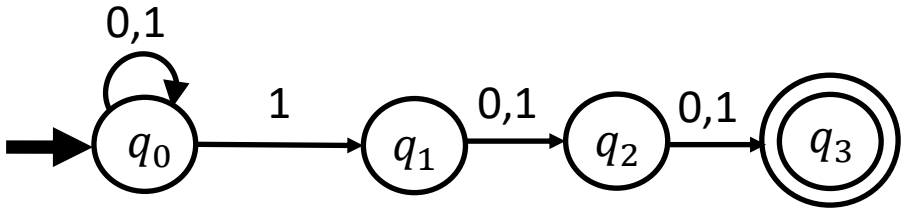
Is an exponential blowup necessary?

Consider the language $L_k = \{w: length\ of\ w \geq k\ and\ the\ k^{th}\ last\ symbol\ is\ 1\}$

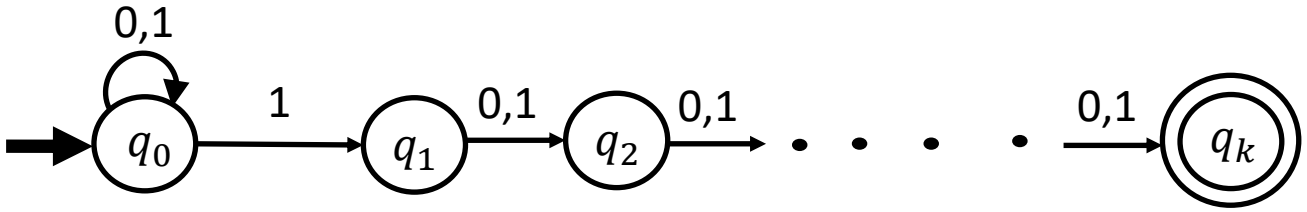
3-state NFA for L_2



4-state NFA for L_3



$(k+1)$-state NFA for L_k

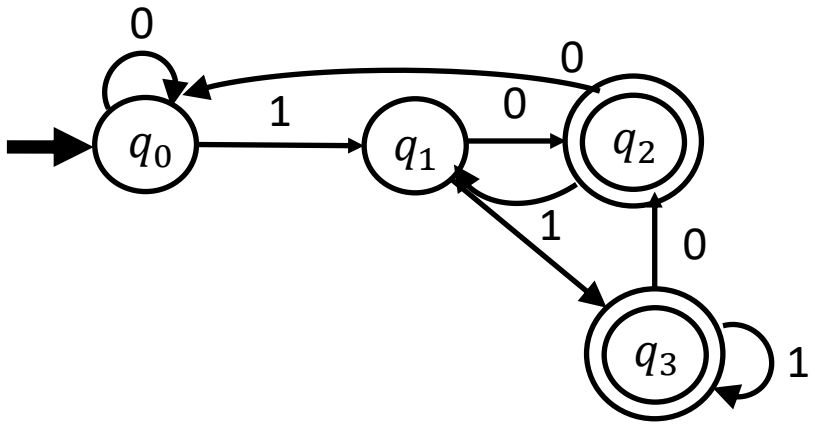


How many states must an DFA for L_k have?

DFAs for L_k

Consider the language $L_k = \{w: length\ of\ w \geq k\ and\ the\ k^{th}\ last\ symbol\ is\ 1\}$

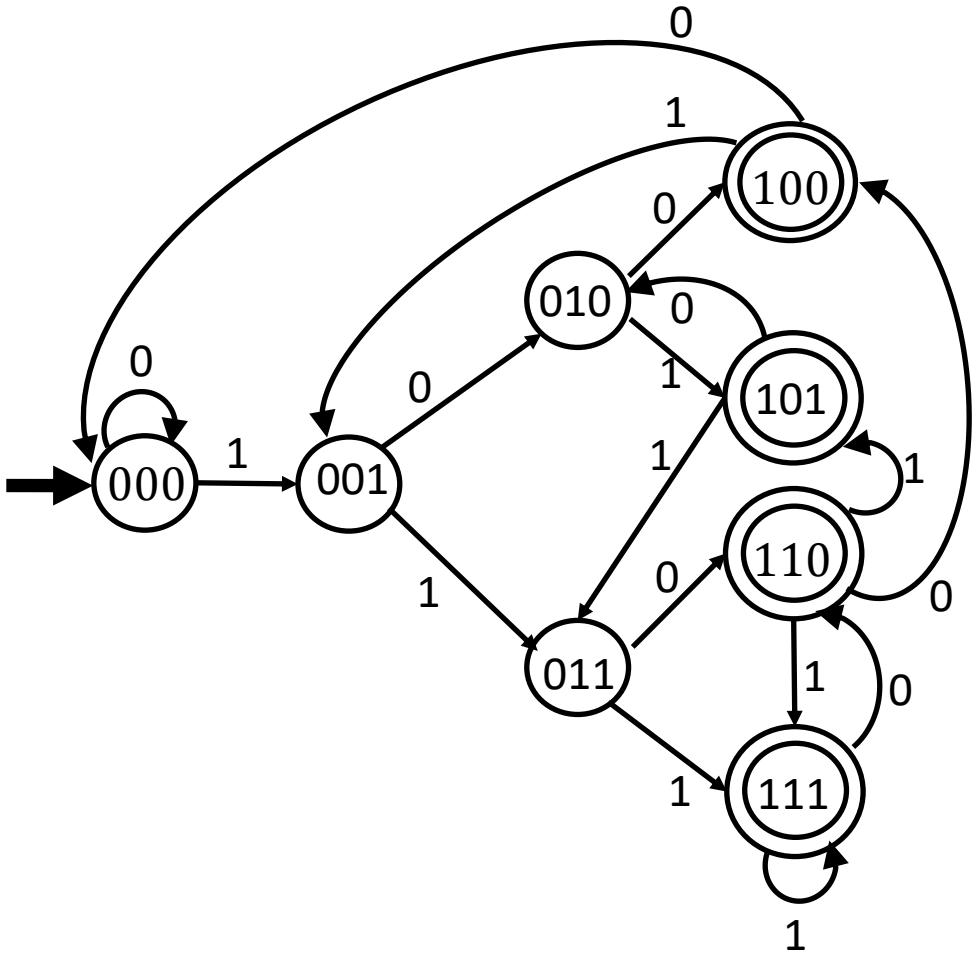
4-state DFA for L_2



Each state tracks the last two symbols: $q_0\ (00)$; $q_1\ (01)$; $q_2\ (10)$; $q_3\ (11)$

DFA for L_3

Each state tracks the last 3 bits of the input



The DFAs Grow Exponentially!

This strategy uses 8 states for L_3
16 states for L_4
…
2^k states for L_k

The NFA for L_k needed only $k + 1$ states while this FSA needs 2^k states!

Is there a better method that requires fewer states for a DFA?
Question: do we ***have to*** remember each of the last k bits of the input?

Intuition: If the i^{th} last bit seen thus far eventually turns out to be the k^{th} last bit of the entire input string, we must remember it – otherwise we won't know whether or not to accept the input!

Every DFA for L_k has $\geq 2^k$ States!

Theorem. Every DFA for L_k has $\geq 2^k$ states.

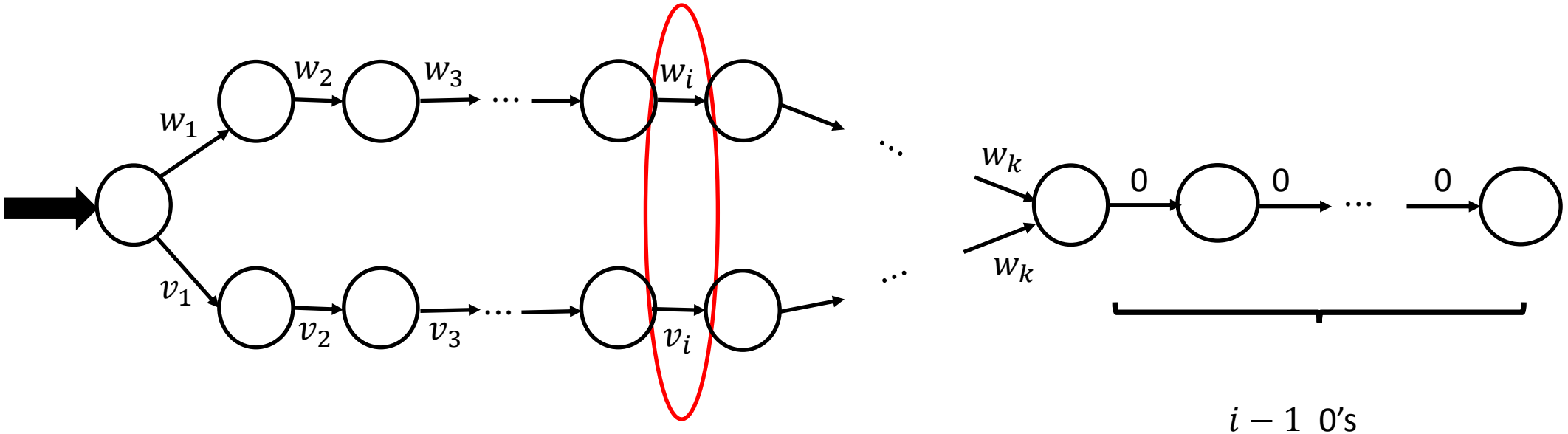
Proof: (by contradiction).

- Suppose that DFA M with fewer states recognizes L_k.
- There are 2^k strings of length k, but M has strictly less than 2^k states.
- By the pigeonhole principle, $\exists w, v$ both of length k that end up in the same state of M.
- Let i be the last (rightmost) bit that strings w, v differ in.

$$w = w_1 w_2 \cdots w_i w_{i+1} \cdots w_k$$

$$v = \quad v_1 v_2 \cdots v_i w_{i+1} \cdots w_k$$

No DFA with less than 2^k states for L_k



kth last symbol

$i-1$ 0's

$w' = w0^{i-1}$ and $v' = v0^{i-1}$ both end in the same final state. Both are either accepted or both are rejected.

But w', v' differ in the k^{th} last bit. So one is in L_k and the other is not.

This means the DFA with less than 2^k states does not recognize L_k.