# Closure Property

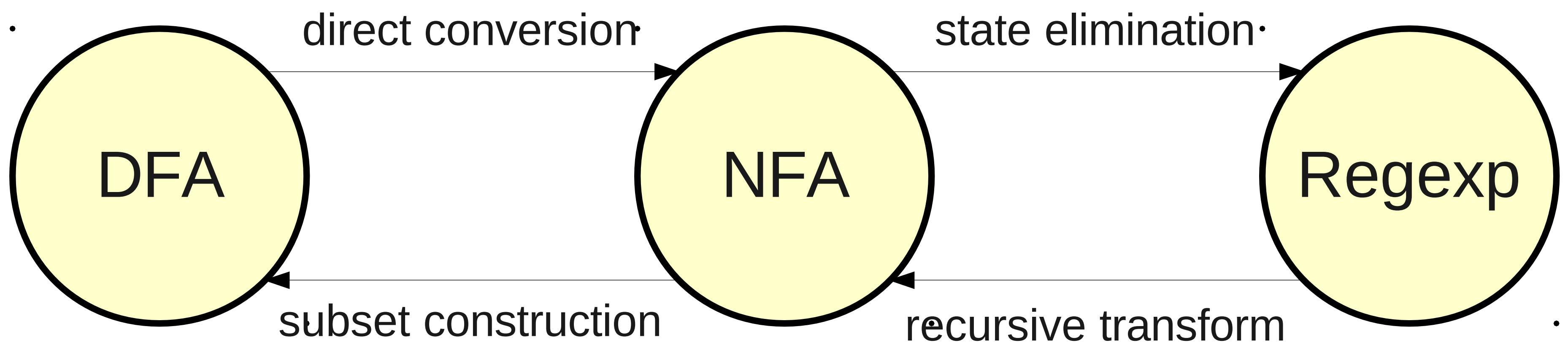
Lecturer: Manar Elkady, Ph.D.

## NFA -> DFA: The Catch

If $N$ is an NFA with $s$ states, how many states does the DFA obtained using the subset construction have? (In the worst case.)

a) $s$
b) $s^2$
c) $2^s$
d) None of the above

# Our Transformations



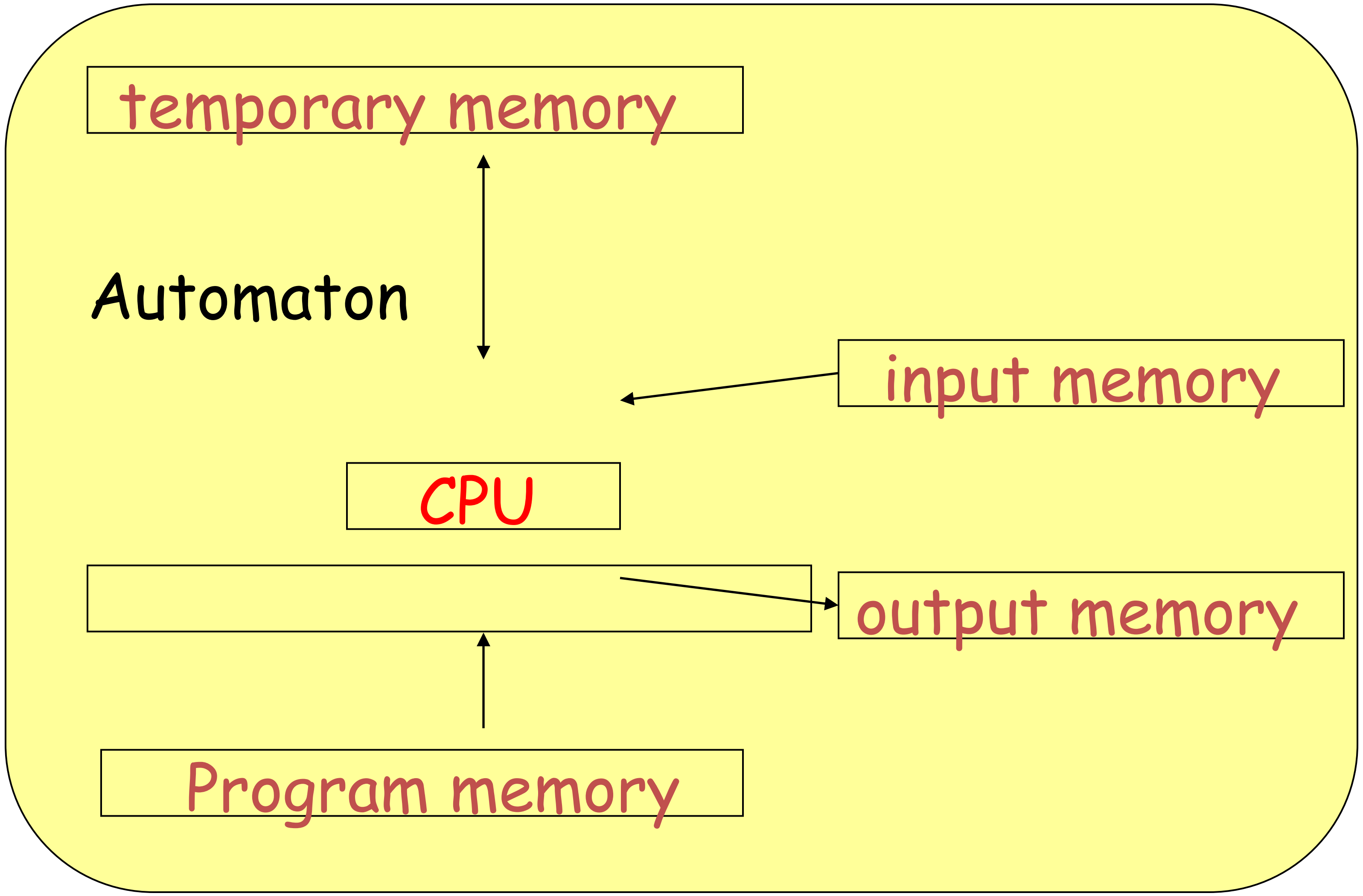
***Theorem:*** The following are all equivalent:

- $L$ is a regular language.
- There is a DFA $D$ such that $\mathcal{L}(D) = L$.
- There is an NFA $N$ such that $\mathcal{L}(N) = L$.
- There is a regular expression $R$ such that $\mathcal{L}(R) = L$.

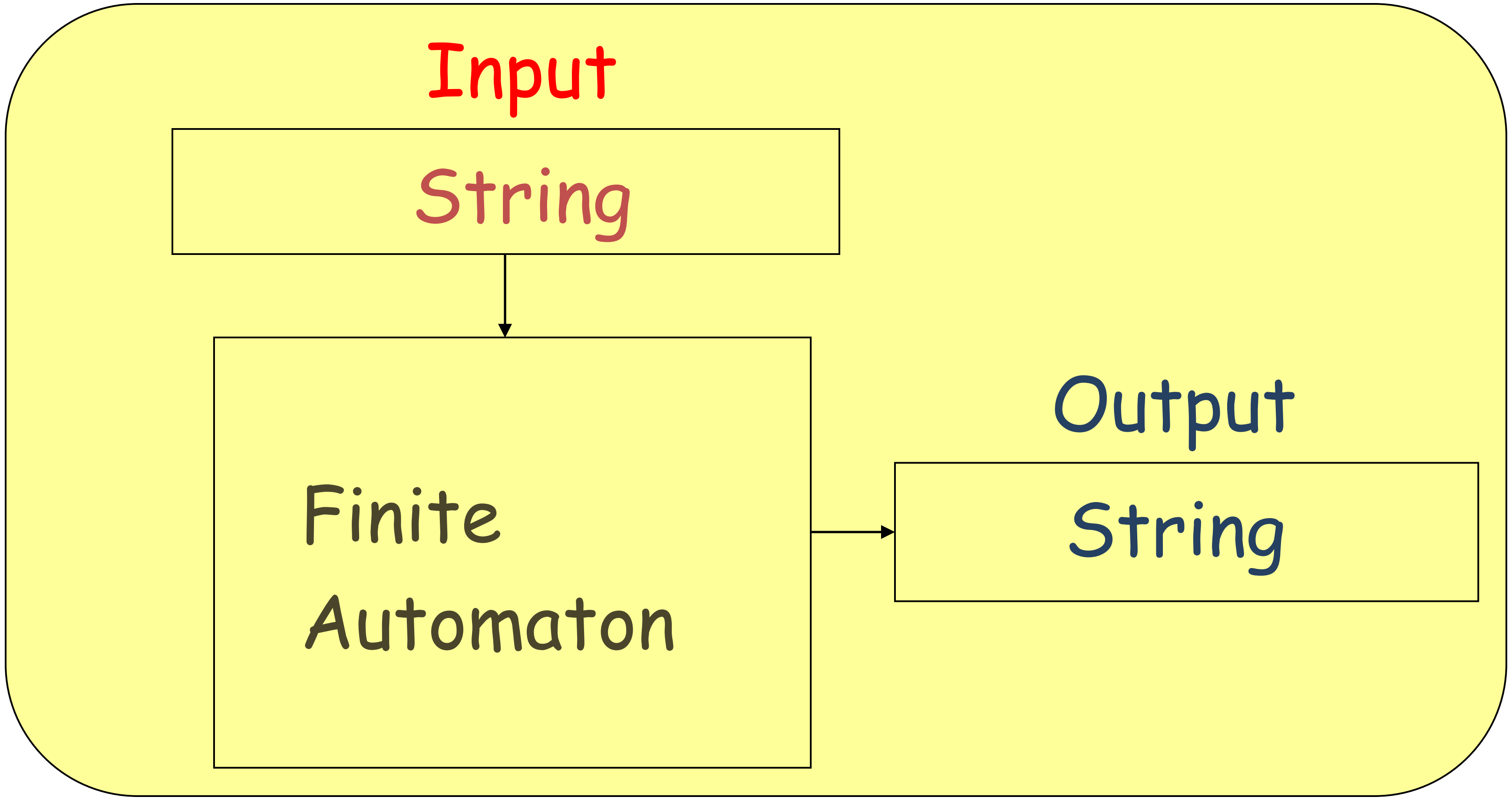
## Why This All Matters

- DFAs correspond to computers with **finite memory**.
- The equivalence of DFAs and NFAs tells us that given finite memory, nondeterminism does not increase computational power.
  - Though it might save on memory.
- The equivalence of DFAs and regular expressions tells us that all problems solvable by finite computers can be assembled out of smaller building blocks.

## Automaton


temporary memory
Automaton
input memory
CPU
output memory
Program memory

## Finite Automaton



## Different Kinds of Automata

Automata are distinguished by the temporary memory:

- **Finite Automata**: no temporary memory
- **Pushdown Automata**: stack
- **Turing Machines**: random access memory

# Power of Automata

Finite Automata $<$ Pushdown Automata $<$ Turing Machine

## Closure Properties

## 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: $x / y$

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

Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

Star: $A^* =$

## Other operations

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

Complement: $A^c = \{w \mid w \notin A\}$

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

Reverse: $A^R = \{w \mid w^R \in A\}$

## 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

## Regular Languages

- A language that can be defined by a RE is called a **regular language.**

1. If $L_1$ and $L_2$ are regular languages then

   $L_1 + L_2$ , $L_1 L_2$ and $L_1^*$ are also regular languages.

2. If L is a regular language then L′ (L complement) is also a regular language.

- L′ : is the language that is not accepted by r where r is the RE that accepts language L.
- Note: (L′)′ =L and (r′)′ =r

## Regular Languages (cont.)

3. If $L_1$ and $L_2$ are regular languages then $L_1 \cap L_2$ is also a regular language.

- $L_1$ =words starting with a
  - $r_1$ =a(a+b)*
- $L_2$ =words ending with a
  - $r_2$ = (a+b)*a
- $L_3 = L_1 \cap L_2$ = a(a+b)*a
- $L_3$ = words that start and end with a.

## How powerful is RE?

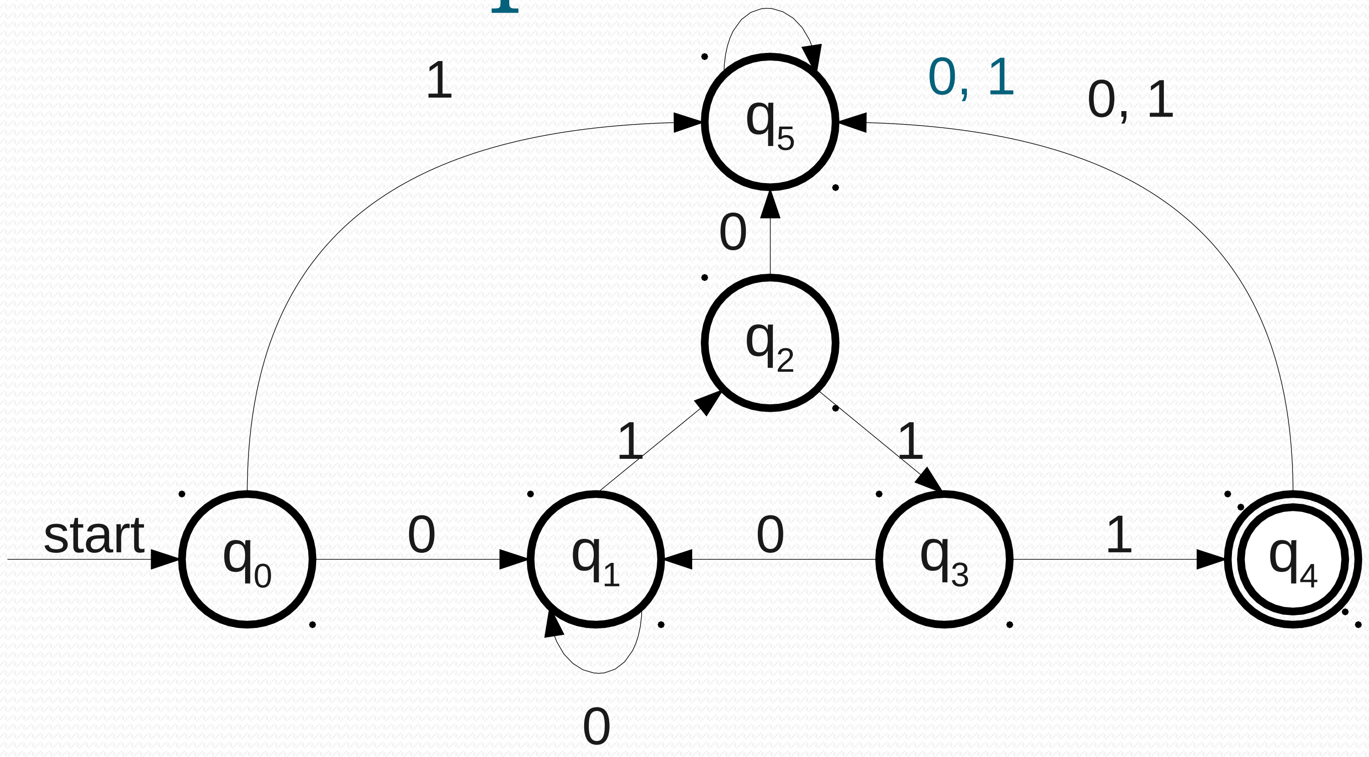
- RE can not express some languages such as plaindrome strings $a^n\ b^n$ .
- How can we prove that a language L is not regular?
- Prove that there is no FA or RE that accepts L
- **Solution:** use a Lemma called pumping Lemma.
- It depends on pumping a string into a template. If the string could not fit into the template then the string is not an RE.

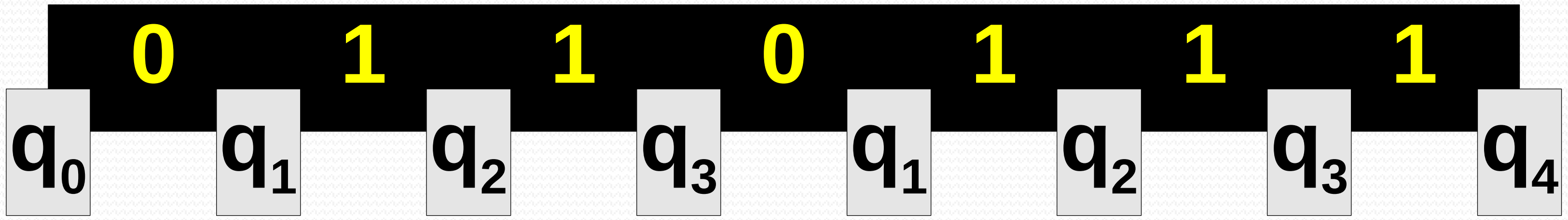
## Why This All Matters

- DFAs correspond to computers with finite memory.
- The equivalence of DFAs and NFAs tells us that given finite memory, nondeterminism does not increase computational power.
  - Though it might save on memory.
- The equivalence of DFAs and regular expressions tells us that all problems solvable by finite computers can be assembled out of smaller building blocks.

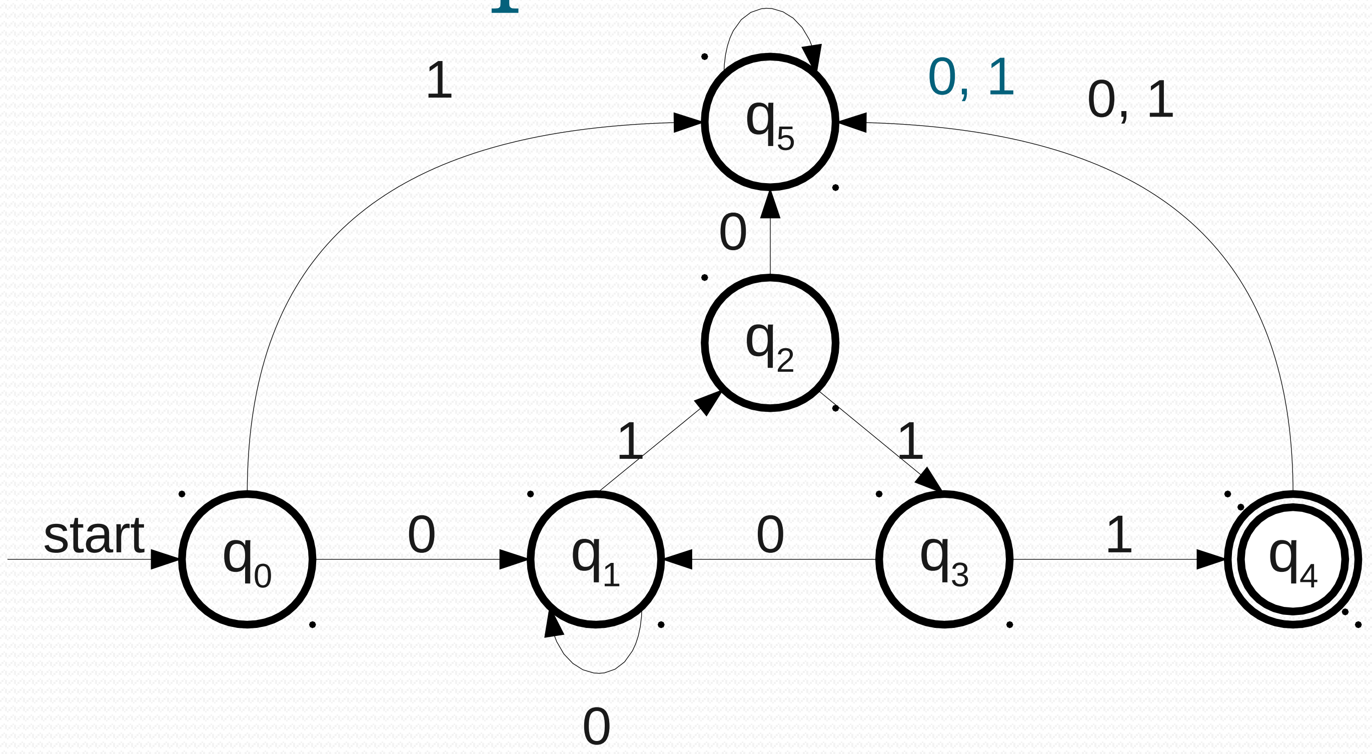
## Is every language regular?

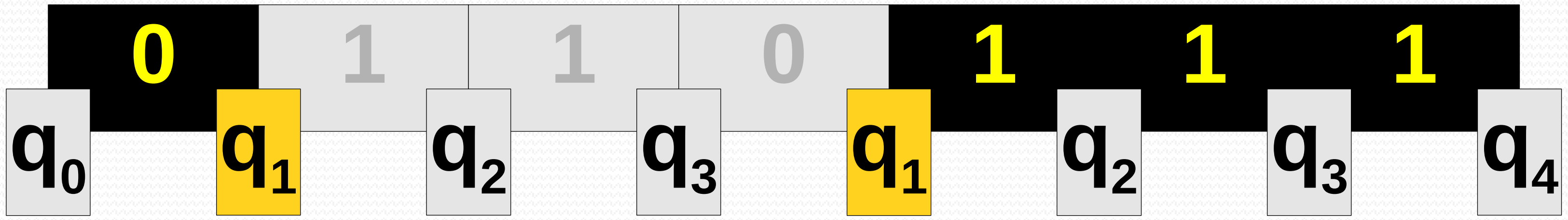
## An Important Observation



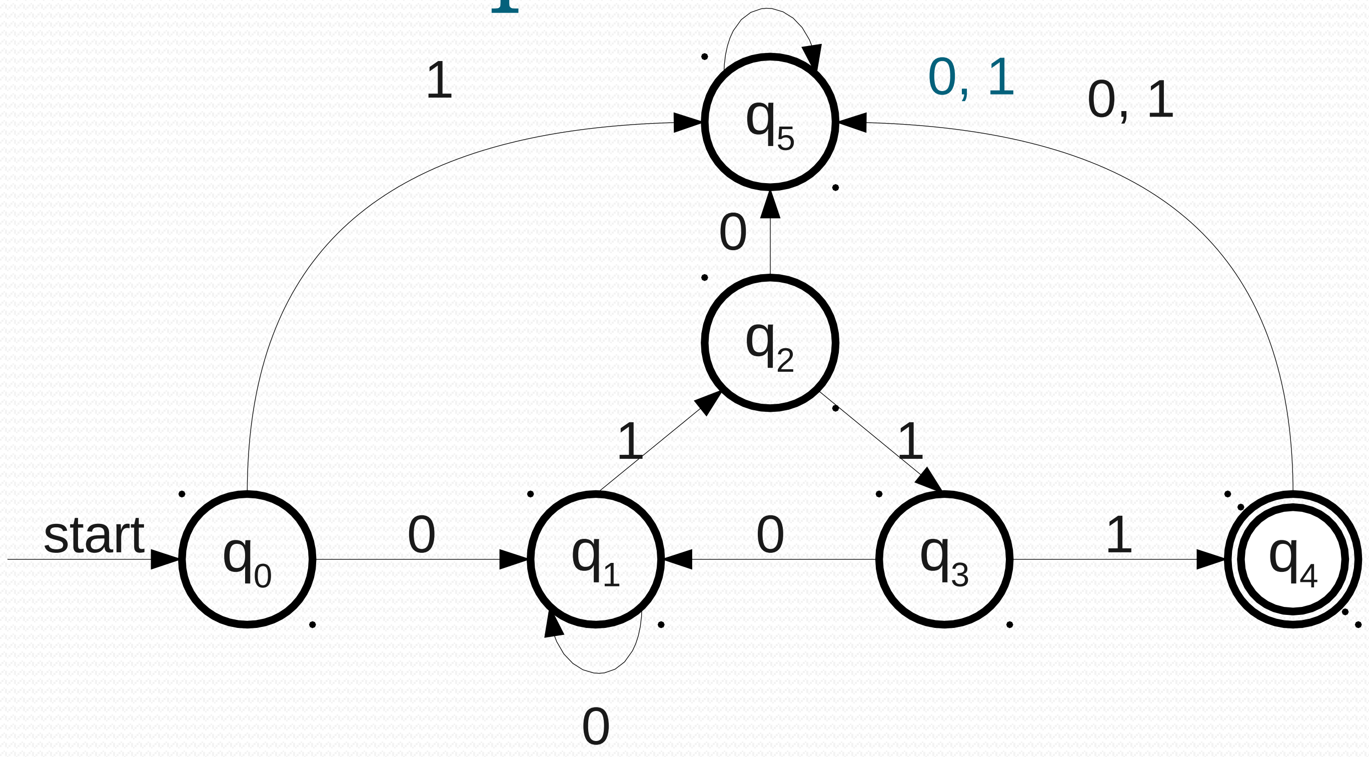


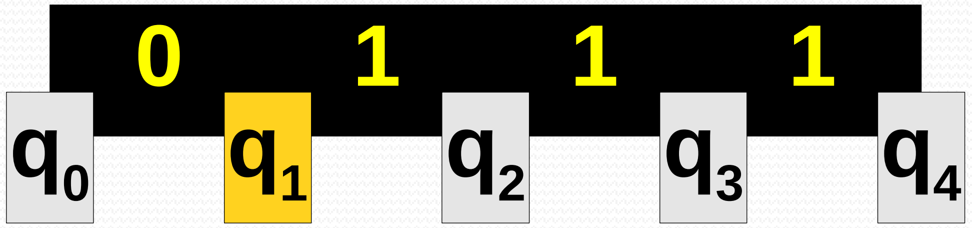
# An Important Observation



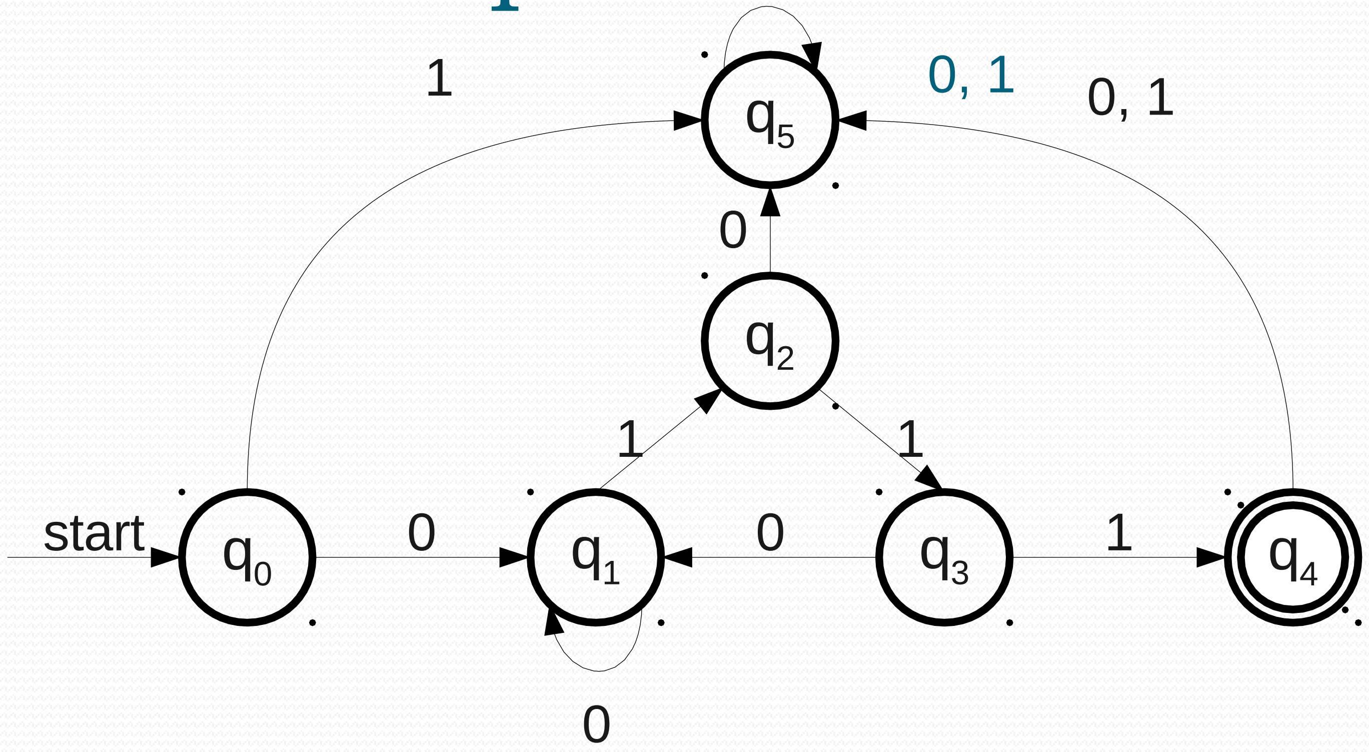


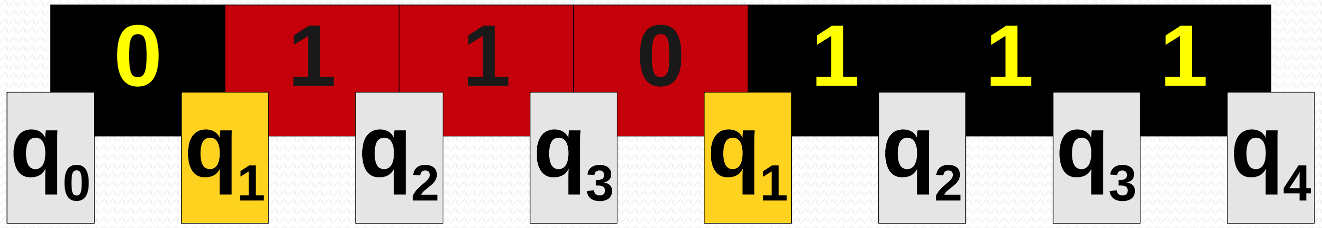
# An Important Observation



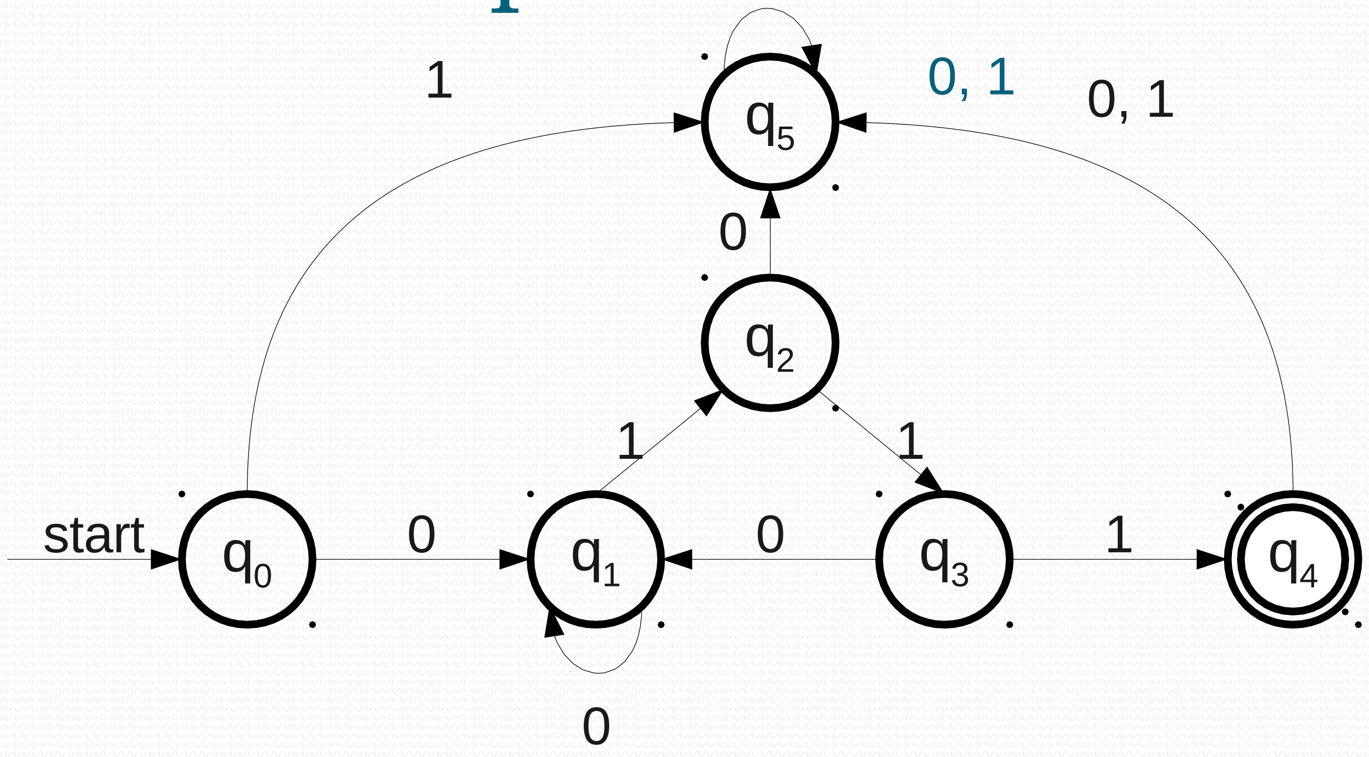


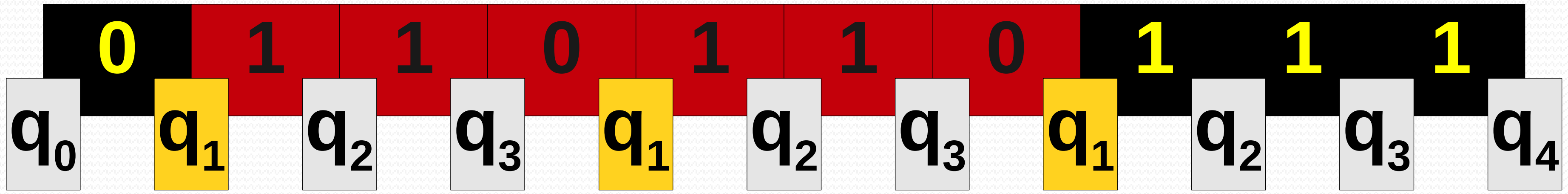
# An Important Observation





## An Important Observation

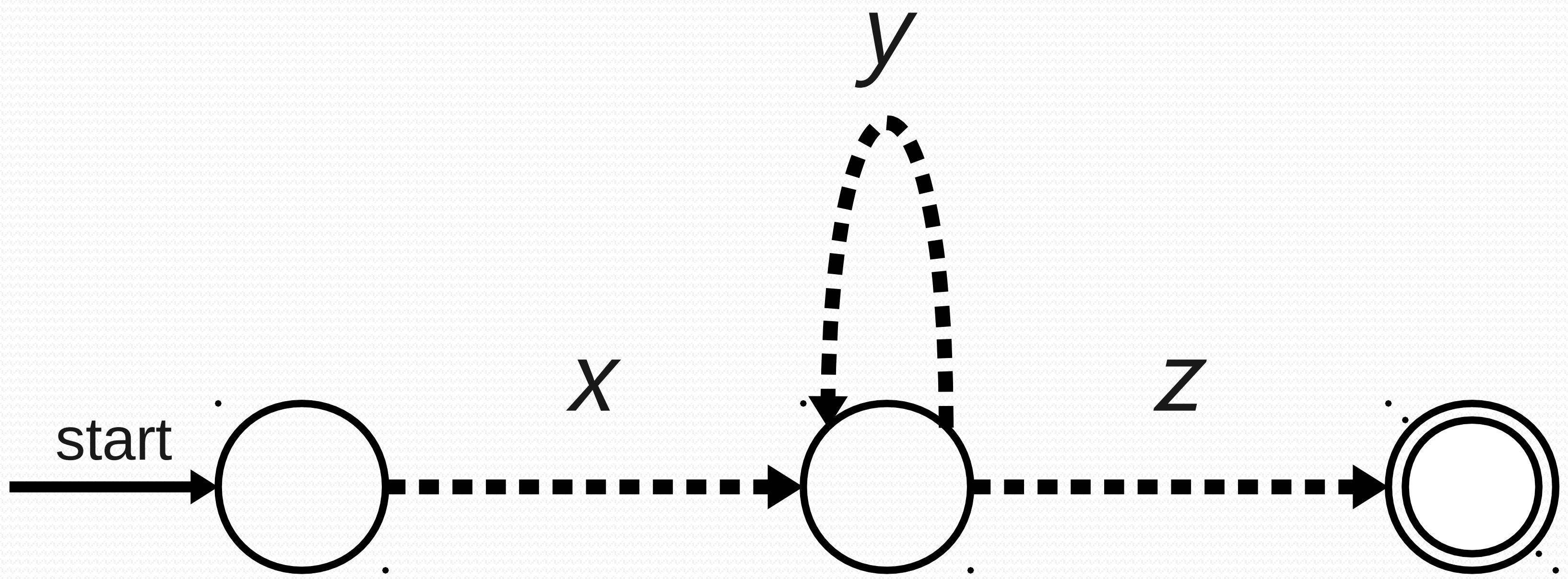




# Visiting Multiple States

- Let $D$ be a DFA with $n$ states.
- Any string $w$ accepted by $D$ that has length at least $n$ must visit some state twice.
    - Number of states visited is equal to the length of the string plus one.
    - By the pigeonhole principle, some state is duplicated.
- The substring of $w$ between those revisited states can be removed, duplicated, tripled, etc. without changing the fact that $D$ accepts $w$.

## Intuitively



## Informally

- Let $L$ be a regular language.
- If we have a string $w \in L$ that is "sufficiently long," then we can split the string into three pieces and "pump" the middle.
- We can write $w = xyz$ such that $xy^0z$, $xy^1z$, $xy^2z$, …, $xy^nz$, … are all in $L$.
  - **Notation**: $y^n$ means "$n$ copies of $y$."

# The Weak Pumping Lemma

- The Weak Pumping Lemma for Regular Languages states that
- For any regular language $L$,
  - There exists a positive natural number $n$ such that
    - For any $w \in L$ with $|w| \geq n$,
      - There exists strings $x$, $y$, $z$ such that
        - For any natural number $i$,

$w = xyz$,

$y \neq \varepsilon$

$xy^iz \in L$

# The Weak Pumping Lemma

- The Weak Pumping Lemma for Regular Languages states that
- For any regular language $L$,
  - There exists a positive natural number $n$ such that
    - For any $w \in L$ with $|w| \geq n$,
      - There exists strings $x$, $y$, $z$ such that
        - For any natural numbe

          $w = xyz$,

          $y \neq \varepsilon$

          $xy^iz \in L$

This number n is sometimes called the **pumping length**.

## The Weak Pumping Lemma

- The Weak Pumping Lemma for Regular Languages states that
- For any regular language $L$,
  - There exists a positive natural number $n$ such that
    - For any $w \in L$ with $|w| \geq n$,
      - There exists strings $x$, $y$, $z$ such that
        - For any natural number $i$,

$w = xyz$,

$y \neq \varepsilon$

$xy^iz \in L$

Strings longer than the pumping length must have a special property.

## The Weak Pumping Lemma

- The Weak Pumping Lemma for Regular Languages states that
- For any regular language $L$,
  - There exists a positive natural number $n$ such that
    - For any $w \in L$ with $|w| \geq n$,
      - There exists strings $x$, $y$, $z$ such that
        - For any natural number $i$,

$w = xyz$, w can be broken into three pieces,

$y \neq \varepsilon$ where the middle piece isn't empty,

$xy^iz \in L$ where the middle piece can be replicated zero or more times.

# What is Pumping Lemma?

**Formal Definition** "The pumping lemma for regular languages is a lemma that makes use of a property shared by all regular languages *RL*. The property specifies that any string $w$ that belongs to a Regular Language $L$ can be broken into $xyz$ "

**Formal Statement**" Let $L$ be a RL, then there exists some constant integer $n \geq 1$ (that depends on L) such that any string $w$ in L with $|w| \geq n$ (where $n$ is a "pumping length") can be written as

$w = xyz$ with substrings $x$, $y$ and $z$, such that

1. $y \neq \varepsilon$; *OR* $|y| \geq 1$; OR $y$ is not empty
2. $|xy| \leq$ n,
3. For every $i \geq 0$, any $xy^{i}z \in L$

## Example 1

- Use pumping Lemma to prove that $L_{|a|(odd)}$ is a RL. Any word *w* of $L_{|a|(odd)}$ have count of a is odd.
- L = {a, aaa, aaaaa, …}

Solution

w = *xyz, where*

1. *y* = aa
2. |*xy*|<= n where *x is a , z is ε, where n*=3
3. For every $i \geq 0$, any $xy^iz \in L$

*Therefore* $a(aa)^i \in L$

*Note that Point#3 covers the word with length 1 ➔"a"*

## How to prove that a given language $L$ in not regular ?

- Assume the opposite: $L$ is regular
- The pumping lemma should hold for $L$
- Use the pumping lemma to obtain a contradiction
- Therefore, $L$ is not regular

# Example 2

- prove that the language $a^n b^n$ is not regular
- From the pumping lemma we can write
- $w = a^m b^m$ in form of xyz that satisfies the three conditions of the pumping Lemma

$$w = xyz = a^m b^m = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots aa\ldots a}_{y^k}}^{m}\overbrace{\underbrace{b\ldots b}_{z}}^{m}$$

**Thus:** $y = a^k,\ 1 \pounds k \pounds n$

# Example 2

Solution

Is there any valid $n$?

w = $xyz$, *where*

1. $y$ = a
2. $|xy|$ <= n where $x$ *is* $a^j$, y is $a^k$ for any j, k
3. For every $i \geq 0$, any $x\,y^i\,z \in L$

*Therefore* $a..a(a)^k\, b^m \in L$

$$w = xyz = a^m b^m = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots aa\ldots a}_{y^k}}^{m}\overbrace{\underbrace{b\ldots b}_{z}}^{m}$$

# Example 2

- From the Pumping Lemma:

1. *Since i could be zero then xz* ∈ L [Rule3]
2. $y \neq \varepsilon$ *[Rule1]*
3. *In case i =0, x must have same the length as z which is indicated by m*
4. *In cases i=k where k >=1, |xy| must be equal m but since |x| is m then the formula becomes* ***m + k = m***

# Example 2

**Thus:**

$$xy^kz = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots a}_{y}\ldots\ldots\underbrace{a\ldots a}_{y}}^{m+k}\underbrace{a\overbrace{b\ldots b}^{m}}_{z} \in L$$

Is there such k integer where m+k=m ?

**CONTRADICTION!!!**

- $m + k \neq m$
- Therefore: Our assumption that $L$ is a regular language is not true

**Conclusion:** $L$ is not a regular language

## Non-Regular Languages

Not all languages are regular”

- The language *w* where *w*=(a|b)*

Non-regular languages cannot be described using REs, NFAs and DFAs.