Module 2

By:
Dr. Archana Gupta

Closure Properties of RE

- REs are closed under the following operations.
  - Complementation.
  - Union.
  - Intersection.
  - Difference.
  - Kleene star.
  - Concatenation.
  - Homomorphism
  - Inverse Homomorphism.

Closure Properties

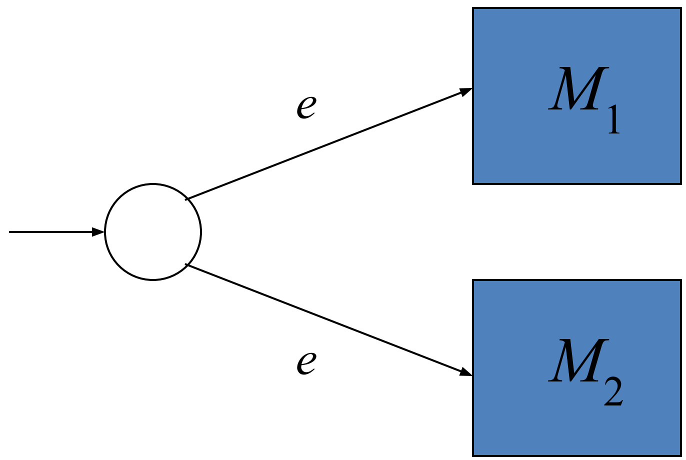
- Let L_1 and L_2 be languages that are accepted by DFAs M_1 and M_2, respectively.
- Is $L_1 \cup L_2$ necessarily accepted by some DFA?
- Is $L_1 \cap L_2$ necessarily accepted by some DFA?
- What about complements, concatenation, Kleene star, etc.?

Closure under Complementation

- Is the complement of L_1 accepted by some DFA?
- Reverse the acceptance status of the states of M_1.
  - Accepting states become rejecting.
  - Rejecting states become accepting.
- The language of the resulting DFA is the complement of L_1.

Closure under Union

- Is $L_1 \cup L_2$ accepted by some DFA?
- Design a new machine (NFA):



- Create a new initial state.

Closure under Union

- Create *e*-moves from the new initial state to the (old) initial states of M_1 and M_2.
- The input is accepted if execution halts in a final state of either M_1 and M_2.
- The language of this machine is $L_1 \cup L_2$.

Closure under Intersection

- Is $L_1 \cap L_2$ accepted by some DFA?
- In the preceding construction, change

$$F = \{(p, q) \mid p \in F_1 \text{ or } q \in F_2\}.$$

to

$$F = \{(p, q) \mid p \in F_1 \text{ and } q \in F_2\}.$$

- The language of this DFA is $L_1 \cap L_2$.

- Use DeMorgan's Law:

$$L_1 \cap L_2 = (L_1' \cup L_2')'.$$

- Warning: To find the complement of an NFA, it is necessary first to convert it to a DFA.

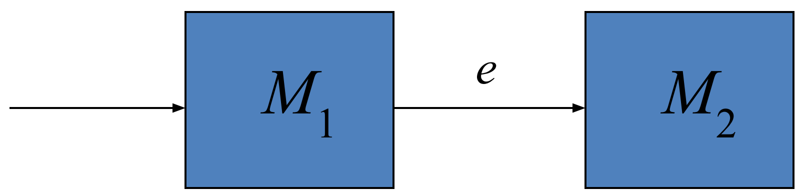
Closure under Difference

- Is $L_1 - L_2$ accepted by some DFA?
- Use the set identity

$$L_1 - L_2 = L_1 \cap L_2'.$$

Concatenation

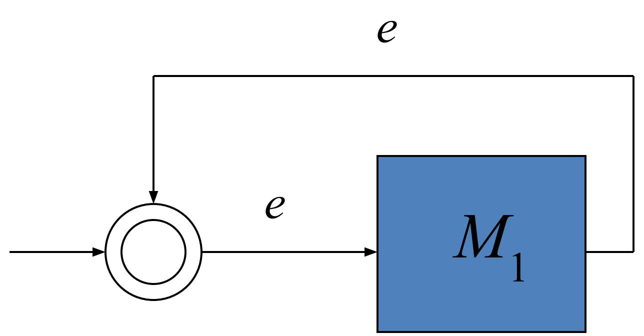
- Is L_1L_2 accepted by some DFA?
- Design a new machine (NFA):



- Create e-moves from every final state of M_1 to the initial state of M_2.
- The input is accepted if execution halts in a favorable state of M_2.

Closure under Kleene Star

- Is L_1^* accepted by some DFA?
- Design a new machine (NFA):



- Create a new initial state.
- Make it an accepting state.

Closure under Kleene Star

- Create an *e*-move from each favorable state of M_1 back to the new initial state.
- The language of this NFA is L_1^*.

Homomorphisms

- A *homomorphism* on an alphabet is a function that gives a string for each symbol in that alphabet.
- Example: $h(0) = ab$; $h(1) = \varepsilon$.
- Extend to strings by $h(a_1 \ldots a_n) = h(a_1) \ldots h(a_n)$.
- Example: $h(01010) = ababab$.

Closure Under Homomorphism

- If L is a regular language, and h is a homomorphism on its alphabet, then h(L) = {h(w) | w is in L} is also a regular language.
- Proof: Let E be a regular expression for L.
- Apply h to each symbol in E.
- Language of resulting RE is h(L).

Example: Closure under Homomorphism

- Let h(0) = ab; h(1) = ε.
- Let L be the language of regular expression **01*** + **10***.
- Then h(L) is the language of regular expression **ab**ε* + ε(**ab**)*.

Note: use parentheses to enforce the proper grouping.

Example – Continued

- **ab**ε* + ε(**ab**)* can be simplified.
- ε* = ε, so **ab**ε* = **ab**ε.
- ε is the identity under concatenation.
  - That is, εE = Eε = E for any RE *E*.
- Thus, **ab**ε* + ε(**ab**)* = **ab**ε + ε(**ab**)* = **ab** + (**ab**)*.
- Finally, L(**ab**) is contained in L((**ab**)*), so a RE for h(L) is (**ab**)*.

Inverse Homomorphisms

- Let h be a homomorphism and L a language whose alphabet is the output language of h.
- $h^{-1}(L)$ = {w | h(w) is in L}.

Example: Inverse Homomorphism

- Let $h(0) = ab$; $h(1) = \varepsilon$.
- Let $L = \{abab, baba\}$.
- $h^{-1}(L)$ = the language with two 0's and any number of 1's = $L(\mathbf{1^*01^*01^*})$.

Notice: no string maps to baba; any string with exactly two 0's maps to abab.

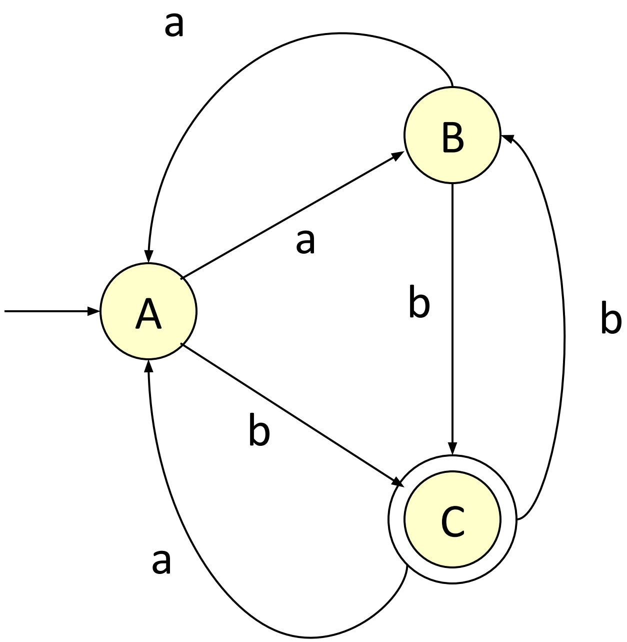
Closure Proof for Inverse Homomorphism

- Start with a DFA A for L.
- Construct a DFA B for $h^{-1}(L)$ with:
  - The same set of states.
  - The same start state.
  - The same final states.
  - Input alphabet = the symbols to which homomorphism h applies.

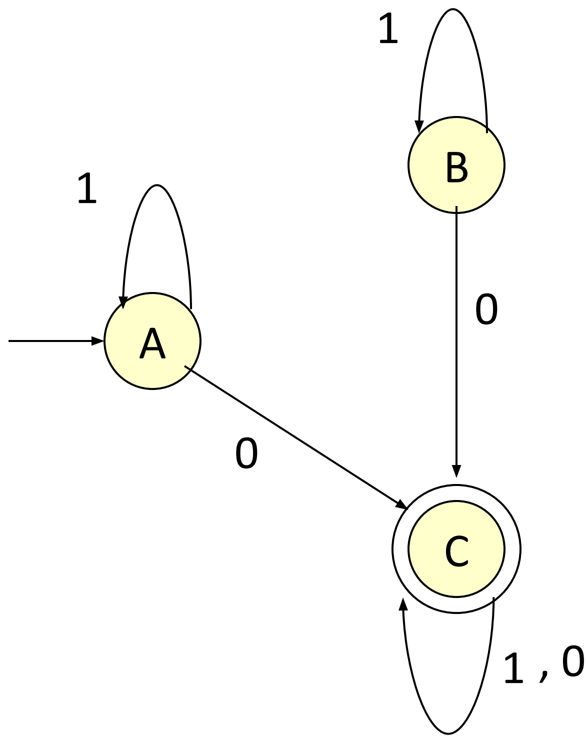
Proof – (2)

- The transitions for B are computed by applying h to an input symbol *a* and seeing where A would go on sequence of input symbols h(a).
- Formally, $\delta_B(q, a) = \delta_A(q, h(a))$.

Example: Inverse Homomorphism Construction



$h(0) = ab$
$h(1) = \varepsilon$



Since
$h(1) = \varepsilon$

Since
$h(0) = ab$

Proof – (3)

- Induction on |w| shows that $\delta_B(q_0, w) = \delta_A(q_0, h(w))$.
- Basis: $w = \varepsilon$.
- $\delta_B(q_0, \varepsilon) = q_0$, and $\delta_A(q_0, h(\varepsilon)) = \delta_A(q_0, \varepsilon) = q_0$.

Proof – (4)

- Induction: Let w = xa; assume IH for x.
- $\delta_B(q_0, w) = \delta_B(\delta_B(q_0, x), a)$.
- $= \delta_B(\delta_A(q_0, h(x)), a)$ by the IH.
- $= \delta_A(\delta_A(q_0, h(x)), h(a))$ by definition of the DFA B.
- $= \delta_A(q_0, h(x)h(a))$ by definition of the extended delta.
- $= \delta_A(q_0, h(w))$ by def. of homomorphism.

Conclusion

- Theorem: The class of languages that are accepted by DFAs is closed under the following operations.
  - Complementation.
  - Union.
  - Intersection.
  - Difference.
  - Kleene star.
  - Concatenation.
  - Homomorphi
  - Inverse Homomorphism.

Pumping Lemma

Properties of Regular Languages

Draw DFA for L={e}

- *for a regular language, L={e, 01}*
- *for the r.l., L={e, 01, 0011}*
- *for the r.l., L={e, 01, 0011, 000111}*
- *…*

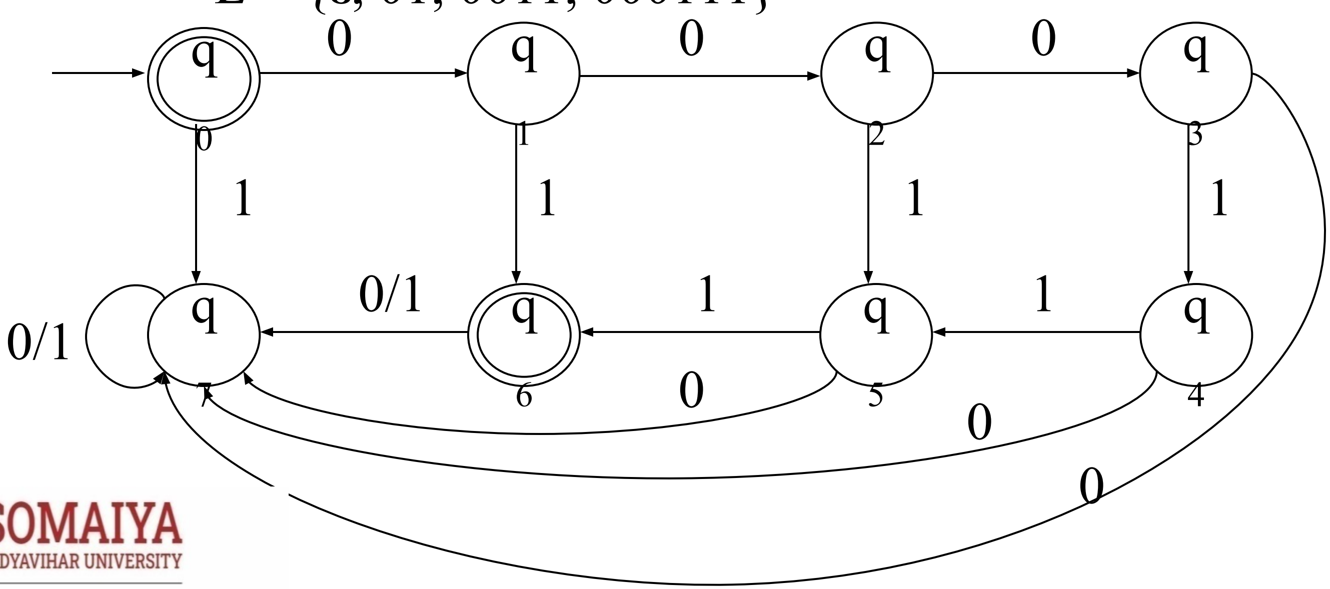
- **Example:**

  $\{0^n1^n \mid 0 =< n\}$ is *not* regular, but

  $\{0^n1^n \mid 0 \leq n \leq k$, for some fixed $k\}$ is regular, for any fixed *k*.

- **For k=3:**

  L = {ε, 01, 0011, 000111}



Properties of Regular Languages

Drawn DFA for L={e}

for a regular language, L={e, 01}

for the r.l., L={e, 01, 0011}

for the r.l., L={e, 01, 0011, 000111}

...

now, try for the r.l., $L=\{0^k 1^k \mid 0 \le k \le \text{infinity}\}$

$\{0^n 1^n \mid 0 \le n \le k$, for some fixed $k\}$ is regular, for any fixed k

but, $\{0^n 1^n \mid 0 \le n\}$ is *not* regular,

to be proved using *Pumpimg Lemma*

Pumping Lemma for Regular Languages

- Pumping Lemma relates the *size of string* accepted
  with the *number of states* in a DFA

- For accepting a string of length m how many states do you need on a path?

string $a_1a_2a_3a_4$ below

$\text{-->}q1 - a_1 \text{->} q2 \text{ -- } a_2$ □ $q3 - a_3$ □ $q4 - a_4 \text{-->} q_F{}^*$

- What is the largest possible string accepted by a DFA with n states,
  presuming there is NO loop?

Pumping Lemma for Regular Languages

- If there is a loop in the path for accepting a string,
  what type of strings are accepted *via* the loop(s)?

- Think of a string in the language: 001 10 111, with middle *10* on a loop

  q0 001 q5 10 q5 111

  Now, what type of strings should also be accepted?

- What is the largest string accepted by a DFA with *n* states,
  presuming there is a ***LOOP*** ?

Pumping Lemma for Regular Languages

- Pumping lemma quantifies two observations toward a path of accepting a string by a DFA:

- 1. If the path is longer than the number of states available in the DFA, then there is a repetition of some state in the path, or a 'loop'

- 2. If there is such a loop in the path, then the substring on the loop may appear 0 or more number of times for the corresponding string to be accepted
  - string: 001 10 111, with middle *10* on a loop

□q0 □ 001 □ q5 □ 10 □ q5 □ 111

  - Other corresponding *accepted* strings are:

001 (10)*

Pumping Lemma for Regular Languages

- **Lemma:** (the pumping lemma)
  Let M be a DFA with $|Q| = n$ states.
  If there exists a string x in L(M), **such that** $|x| \geq n$,
  then there exists a way to write it as $x = uvw$,
  where u, v, and w are such that:
  - $1 \leq |uv| \leq n$
  - $|v| \geq 1$
  - AND, all the strings $\mathbf{uv^iw}$ are also in L(M), for all $i \geq 0$

Pumping Lemma for Regular Languages

- **Proof:**

  Let $x = a_1a_2 \dots a_m$ where $m \geq n$, x is in L(M), and $\delta(q_0, a_1a_2 \dots a_p) = q_{jp}$

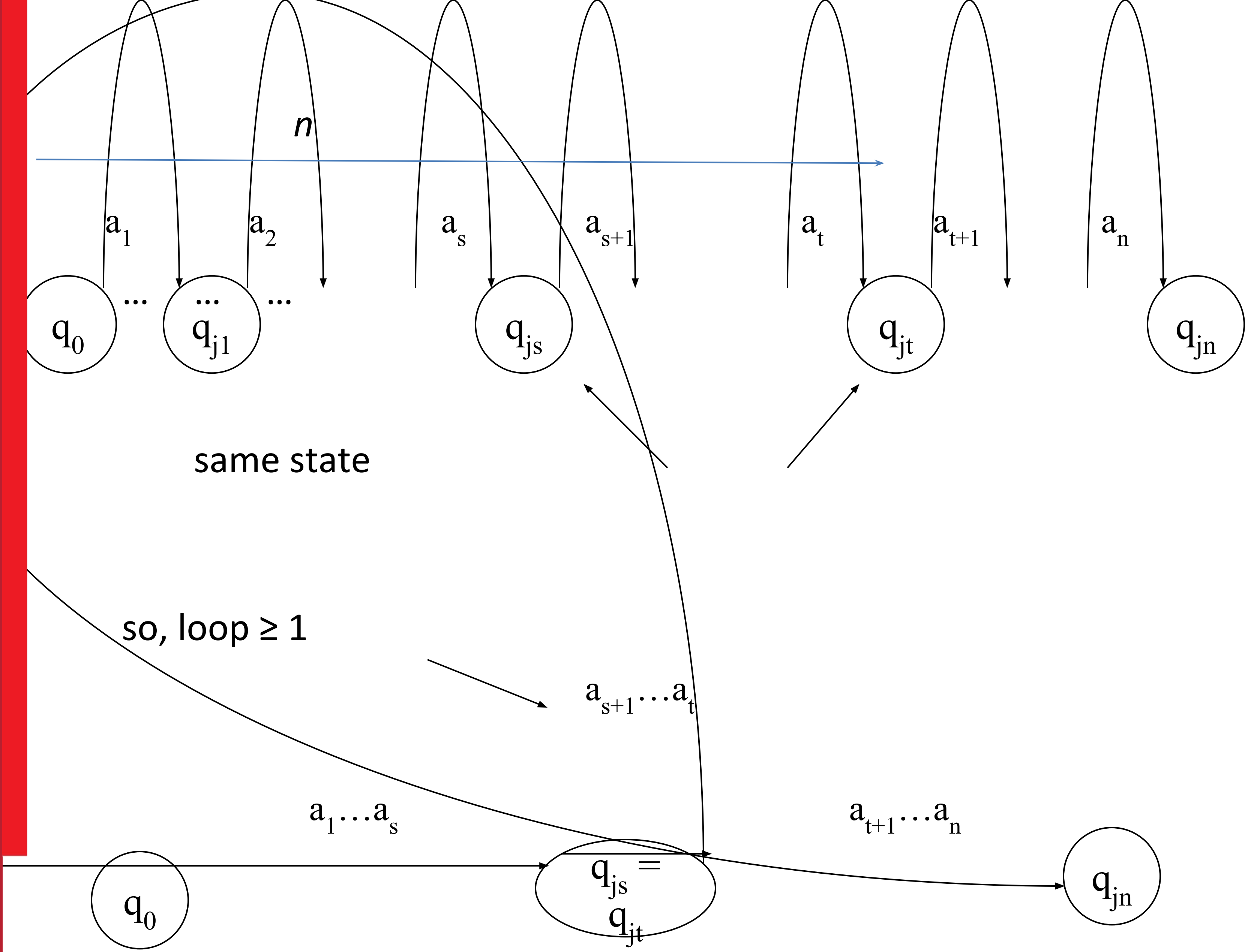
$$
\begin{array}{lcccccccc}
 & & a_1 & & a_2 & & a_3 & \dots & a_m \\
\dashrightarrow & q_0 & q_{j1} & q_{j2} & q_{j3} & \dots & & q_{jm} & \qquad m \geq n
\end{array}
$$

  Consider the first n symbols, and first n+1 states on the above path:

$$
\begin{array}{cccccccc}
 & a_1 & & a_2 & & a_3 & \dots & a_n \\
q_0 & q_{j1} & q_{j2} & q_{j3} & \dots & q_{jn} & &
\end{array}
$$

  Since $|Q| = n$, it follows from the pigeon-hole principle that $j_s = j_t$ for some

  $0 \leq s<t \leq n$, i.e., some state appears on this path twice (perhaps many states appear more than once, but at least one does).


n
a_1
a_2
a_s
a_{s+1}
a_t
a_{t+1}
a_n
q_0 … q_{j1} … …
q_{js}
q_{jt}
q_{jn}
same state
so, loop ≥ 1
$a_{s+1} \ldots a_t$
$a_1 \ldots a_s$
$a_{t+1} \ldots a_n$
q_0
$q_{js} = q_{jt}$
q_{jn}

- Let:
  - $u = a_1 \ldots a_s$
  - $v = a_{s+1} \ldots a_t$

- Since $0 \leq s<t \leq n$ and $uv = a_1 \ldots a_t$ it follows that:

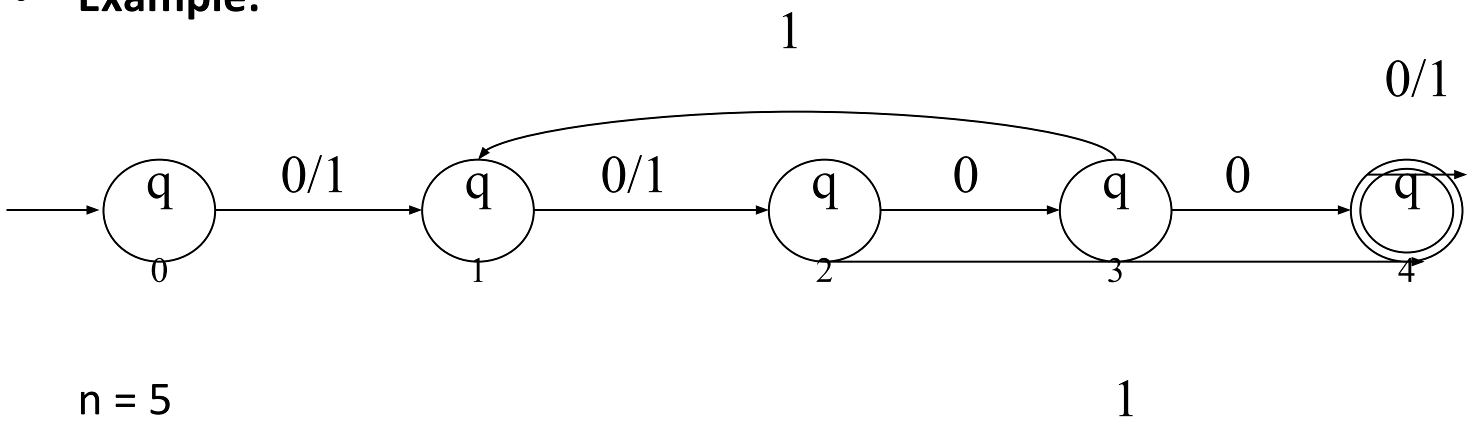
  - $1 \leq |v|$ and therefore $1 \leq |uv|$
  - $|uv| \leq n$ and therefore $1 \leq |uv| \leq n$
- In addition, let:
  - $w = a_{t+1} \ldots a_m$

- It follows that $uv^iw = a_1 \ldots a_s(a_{s+1} \ldots a_t)^i a_{t+1} \ldots a_m$ is in L(M), for all $i \geq 0$.

  In other words, when processing the accepted string x, the loop was traversed once, but it could be allowed to traverse as many times as desired, and the corresponding strings would be accepted.

"as many times" >= 0

- **Example:**



n = 5

x = 0001000 is in L(M)

0 0 0 1 0 0 0

q_0 $\mathbf{q_1}$ q_2 q_3 $\mathbf{q_1}$ q_2 q_3 q_4

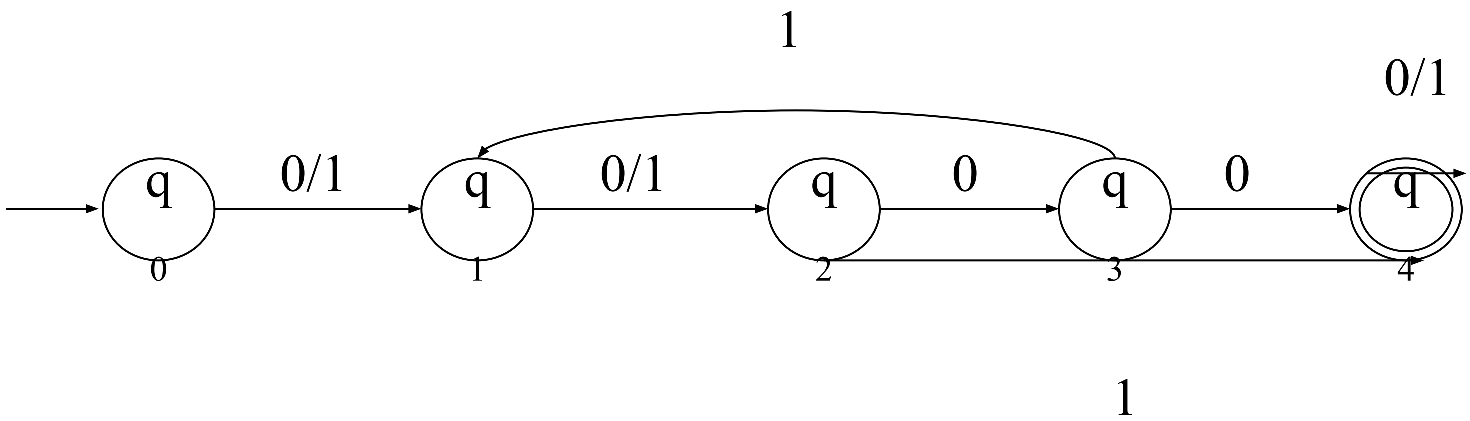
first n+1 states

u = 0

v = 001

w = 000

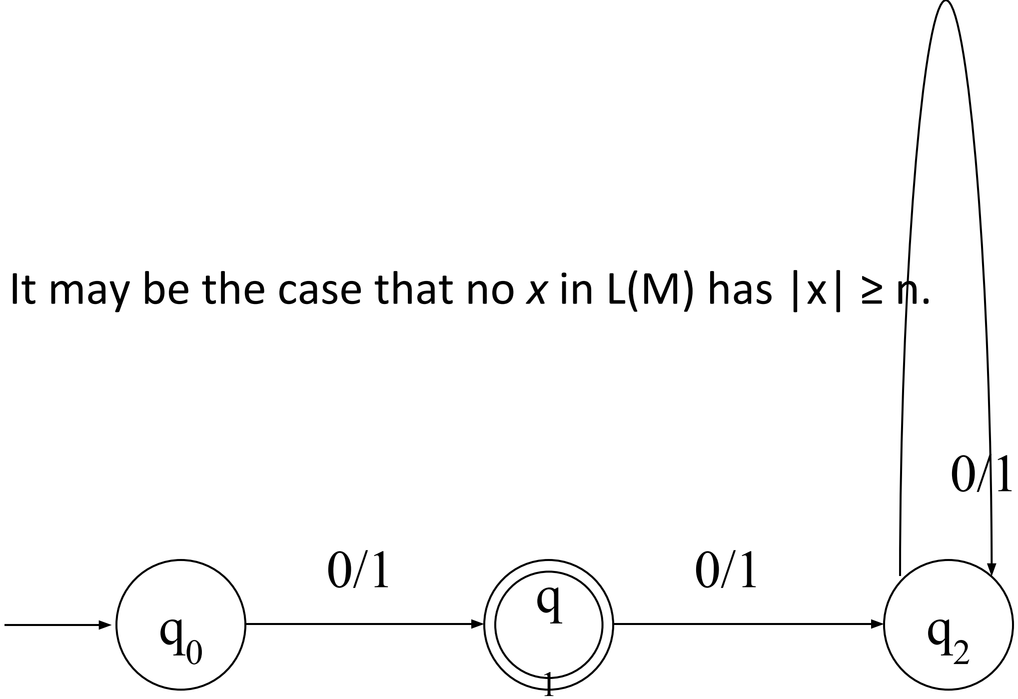
uv^iw is in L(M), i.e., $0(001)^i000$ is in L(M), for all $i \geq 0$



- Note that this does not mean that every string accepted by the DFA has this form:
  - 001 is in L(M) but is not of the form $0(001)^i000$

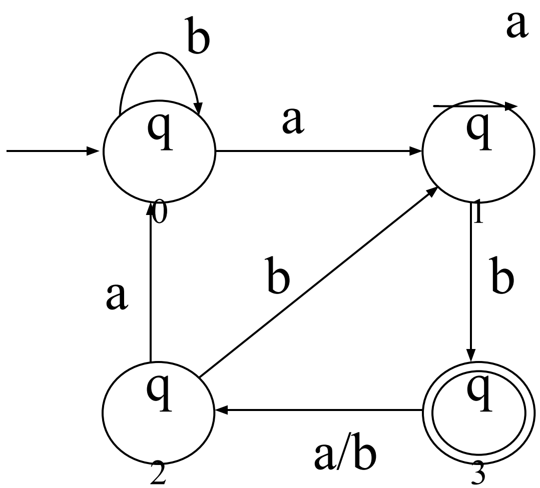
- Similarly, this does not mean that every long string accepted by the DFA has this form:
  - 0011111 is in L(M), is very long, but is not of the form $0(001)^i000$
- Note, however, in this latter case 0011111 could be similarly decomposed.

- **Note:** It may be the case that no *x* in L(M) has |x| ≥ n.



What is the language?

- **Example:**



n = 4

x = bbbab is in L(M)

b b b a b

|x| = 5

q0 q0 q0 q0 q1 q3

u = ε

u = b

u = bb

v = b

or

v = b

v = b

w = bbab

w = bab

w = ab

$(b)^i$bbab is in L(M), for all i ≥ 0

b$(b)^i$bab is in L(M), for all i ≥ 0

Non-Regular Language: Example

- **Theorem:** The language:

$$L = \{0^k 1^k \mid k \geq 0\} \qquad (1)$$

is not regular.

- **Proof:** (*by contradiction*) Suppose that L is regular. Then there exists a DFA *M* such that:

$$L = L(M) \qquad (2)$$

We will show that *M* accepts some strings not in L, contradicting (2).

Suppose that M has *n* states, and <u>choose</u> a string $x=0^m1^m$,

where the

constant $m>>n$.

By (1), x is in L.

By (2), x is also in L(M), *note that the machine accepts a language not just a string*

Since $|x| = m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 \leq |uv| \leq n$
- $1 \leq |v|$, and
- uv^iw is in L(M), for all $i \geq 0$

Since $1 \leq |uv| \leq n$ and $n << m$, it follows that $1 \leq |uv| < m$.

Also, since $x = 0^m1^m$ it follows that uv is a substring of 0^m.

In other words $v=0^j$, for some $j \geq 1$.

Since uv^iw is in L(M), for all $i \geq 0$, it follows that $0^{m+cj}1^m$ is in L(M), for all $c \geq 1$ (no. of loops), and $j \geq 1$ (length of the loop)

But by (1) and (2), $0^{m+cj}1^m$ is not in L(M), for any $c \geq 1$, i.e., $m+cj > m$, a contradiction.

- Note that L basically corresponds to balanced parenthesis.

Non-Regularity Example

- **Theorem:** The language:

  $$L = \{0^k 1^k 2^k \mid k \geq 0\} \qquad (1)$$

  is not regular.

- **Proof:** (by contradiction) Suppose that L is regular. Then there exists a DFA M such that:

  $$L = L(M) \qquad (2)$$

  We will show that M accepts some strings not in L, contradicting (2).

  Suppose that M has n states, and consider a string $x=0^m1^m2^m$, where the constant $m>>n$.

  By (1), x is in L.

  By (2), x is also in L(M), *note that the machine accepts a language not just a string*

Since $|x| = m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 \leq |uv| \leq n$
- $1 \leq |v|$, and
- uv^iw is in L(M), for all $i \geq 0$

Since $1 \leq |uv| \leq n$ and $n<<m$, it follows that $1 \leq |uv| \leq m$.

Also, since $x = 0^m1^m2^m$ it follows that uv is a substring of 0^m.

In other words $v=0^j$, for some $j \geq 1$.

Since uv^iw is in L(M), for all $i \geq 0$, it follows that $0^{m+cj}1^m2^m$ is in L(M), for all $c \geq 1$ and $j \geq 1$.

But by (1) and (2), $0^{m+cj}1^m2^m$ is not in L(M), for any integer $c \geq 1$, a contradiction.

- Note that the above proof is almost identical to the previous proof.

NonRegularity Example

- **Theorem:** The language:

$$L = \{0^m 1^n 2^{m+n} \mid m,n \geq 0\} \qquad (1)$$

is not regular.

- **Proof:** (by contradiction) Suppose that L is regular. Then there exists a DFA M such that:

$$L = L(M) \qquad (2)$$

We will show that M accepts some strings not in L, contradicting (2).

Suppose that M has n states, and consider a string $x=0^m1^n2^{m+n}$, where m>>n.

By (1), x is in L.

By (2), x is also in L(M).

Since $|x| = m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 \leq |uv| \leq n$
- $1 \leq |v|$, and
- uv^iw is in L(M), for all $i \geq 0$

Since $1 \leq |uv| \leq n$ and $n<<m$, it follows that $1 \leq |uv| \leq m$.

Also, since $x = 0^m1^n2^{m+n}$ it follows that uv is a substring of 0^m.

In other words $v=0^j$, for some $j \geq 1$.

Since uv^iw is in L(M), for all $i \geq 0$, it follows that $0^{m+cj}1^m2^{m+n}$ is in L(M), for all $c \geq 1$. In other words v can be "pumped" as many times as we like, and we still get a string in L(M).

But by (1) and (2), $0^{m+cj}1^n2^{m+n}$ is not in L(M), for any $c \geq 1$, *because the acceptable expression should be* $0^{m+cj}1^n2^{m+\mathbf{cj}+n}$, a contradiction.

- *What about* $\{0^m 1^n \mid m, n \geq 0\}$?
- $\{0^m 1^n \mid m, n \geq 0, \text{ and } m<n\}$?
- $\{0^m 1^n \mid m, n \geq 0, \text{ and } m = n\text{^}2\}$?
- $\{0^m 1^n \mid m, n \geq 0, \text{ and } m>n\}$?
- *Are these regular languages, or not?*