Regular Expressions

Reading: Chapter 3

Operations on Languages

- Let L, L_1, L_2 be subsets of Σ^*
- Concatenation: $L_1L_2 = \{xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2\}$
- Concatenating a language with itself: $L^0 = \{\varepsilon\}$
  $L^i = LL^{i-1}$, for all i >= 1
- Kleene Closure: $L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \ldots$
- Positive Closure: $L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup \ldots$
- Question: Does L^+ contain ε?

Regular Expressions

A regular expression is:

- a finite length sequence of symbols
- used to specify a language
- very precise, intuitive, and useful in a lot of contexts
- easy to convert to an NFA-ε, algorithmically; and consequently to an NFA, a DFA, and a corresponding program

Definition of a Regular Expression

- If r is a regular expression, then L(r) is used to denote the corresponding language.

- Σ be an alphabet. The regular expressions over Σ are:

Ø	Represents the empty set { }
ε	Represents the set {ε}
a	Represents the set {a}, for any symbol a in Σ

Let r and s be regular expressions.

r+s	Represents the set L(r) U L(s)
rs	Represents the set L(r)L(s)
r^*	Represents the set L(r)*
(r)	Represents the set L(r)

- Note that the operators are listed in increasing precedence.

- **Examples:** Let $\Sigma = \{0, 1\}$

(0 + 1)*	All strings of 0's and 1's
0(0 + 1)*	All strings of 0's and 1's, beginning with a 0
(0 + 1)*1	All strings of 0's and 1's, ending with a 1
(0 + 1)*0(0 + 1)*	All strings of 0's and 1's containing at least one 0
(0 + 1)*0(0 + 1)*0(0 + 1)*	All strings of 0's and 1's containing at least two 0's
(0 + 1)*01*01*	All strings of 0's and 1's containing at least two 0's
1*(01*01*)*	All strings of 0's and 1's containing an even number of 0's
(1*01*0)*1*	All strings of 0's and 1's containing an even number of 0's
(1 + 01*0)*	All strings of 0's and 1's containing an even number of 0's

- Question: Is there a unique minimum regular expression for a given language?

- How do the above regular expressions "parse" based on the formal definition?

- **Other examples:**

  011

  $010 + 1100 + \varepsilon$

  $010 + 1100 + \varnothing$

  $\varepsilon(0 + 1)^*$

  $\varepsilon 0(0 + 1)^*$

  $(0 + 1 + \varepsilon)^*1$

  $\varnothing(0 + 1)^*$

  $(\varnothing + 1)^*$

  $\varnothing(0 + \varepsilon)^* + \varepsilon^*\varnothing^*$

- An almost completely useless program…

- Generating a Random String for a Regular Expression (Example):

1*(01*01*)*

```
// generate something from 1*
int n = random(0,inf);
for (int i=0; i<=n-1; i++) {
    print('1');
}
// generate something from (01*01*)*
int m = random(0,inf);
for (int i=0; i<=m-1; i++) {
    // generate a single 0
    print('0');
    // generate something from 1*
    int k = random(0,inf);
    for (int i=0; i<=k-1; i++) {
        print('1');
    }
    // generate a single 0
    print('0');
    // generate something from 1*
    int k = random(0,inf);
    for (int i=0; i<=k-1; i++) {
       print('1');
    }
}
```

- Algebraic Laws for Regular Expressions:

1.	$u + v = v + u$	commutativity
2.	$(u + v) + w = u + (v + w)$	associativity
3.	$(uv)w = u(vw)$	associativity
4.	$\emptyset + u = u + \emptyset = u$	identity
5.	$\varepsilon u = u\varepsilon = u$	identity
6.	$\emptyset u = u\emptyset = \emptyset$	annihilator
7.	$u(v+w) = uv+uw$	distributive
8.	$(u+v)w = uw+vw$	distributive
9.	$u + u = u$	idempotent
10.	$(u^*)^* = u^*$	
11.	$\emptyset^* = \varepsilon$	
12.	$\varepsilon^* = \varepsilon$	

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \ldots$$

- Such laws can be used to prove equivalences between regular expressions:

$\varepsilon + 1^* = 1^*$

$0 + 01^* = (\varepsilon + 1^*)0$

$(0 + 1)^* = (0^* + 10^*)^*$

- **Other Laws:**
  1. $(uv)^*u = u(vu)^*$
  2. $(u+v)^* = (u^*+v)^*$

     $= u^*(u+v)^*$

     $= (u+vu^*)^*$

     $= (u^*v^*)^*$

     $= u^*(vu^*)^*$

     $= (u^*v)^*u^*$
  3. $L^+ = LL^* = L^*L$
  4. $L^* = L^+ + \varepsilon$

Equivalence of Regular Expressions and NFA-εs

- **Note:**

  Throughout the following, keep in mind the definition of *string acceptance* for an NFA-ε…what is it?

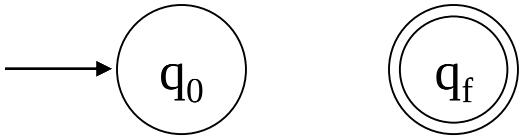
  Lemma 1: Let r be a regular expression. Then there exists an NFA-ε M such that L(M) = L(r). Furthermore, M has exactly one final state with no transitions out of it.

  Proof: (by induction on the number of operators, denoted by OP(r), in r).

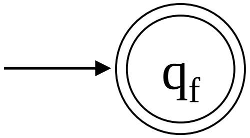
Basis: OP(r) = 0

Then r is either Ø, ε, or **a**, for some symbol **a** in Σ

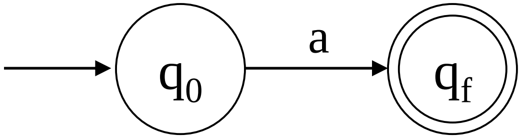
For Ø:



For ε:



For **a**:



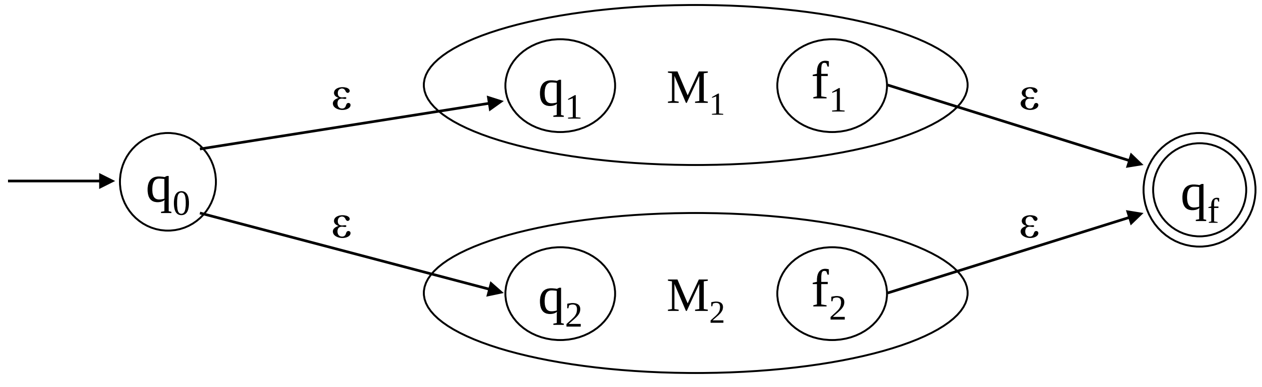
Inductive Hypothesis: Suppose there exists a $k >= 0$ such that for any regular expression r where $0 <= OP(r) <= k$, there exists an NFA-ε such that $L(M) = L(r)$. Furthermore, suppose M has exactly one final state with no transitions out of it.

Inductive Step: Let r be a regular expression with k + 1 operators ($OP(r) = k + 1$). Since $k>=0$, it follows that $k + 1 >= 1$, and therefore r has at least one operator.

Case 1) $r = r_1 + r_2$

Since $OP(r) = k +1$, it follows that $0 <= OP(r_1) <= k$ and $0 <= OP(r_2) <= k$. By the inductive hypothesis there exist NFA-ε machines M_1 and M_2 such that $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$. Furthermore, both M_1 and M_2 have one final state.

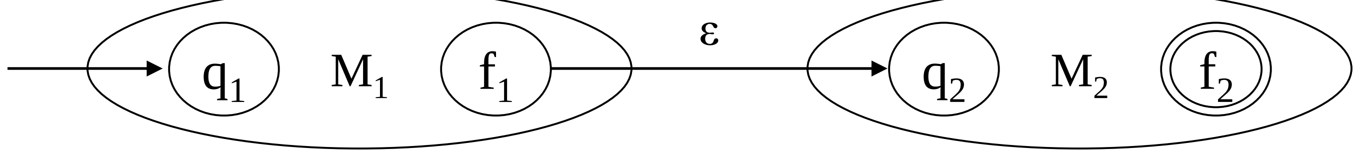
Construct M as:



Case 2) $r = r_1 r_2$

Since $OP(r) = k+1$, it follows that $0 <= OP(r_1) <= k$ and $0 <= OP(r_2) <= k$. By the inductive hypothesis there exist NFA-ε machines M_1 and M_2 such that $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$. Furthermore, both M_1 and M_2 have exactly one final state.

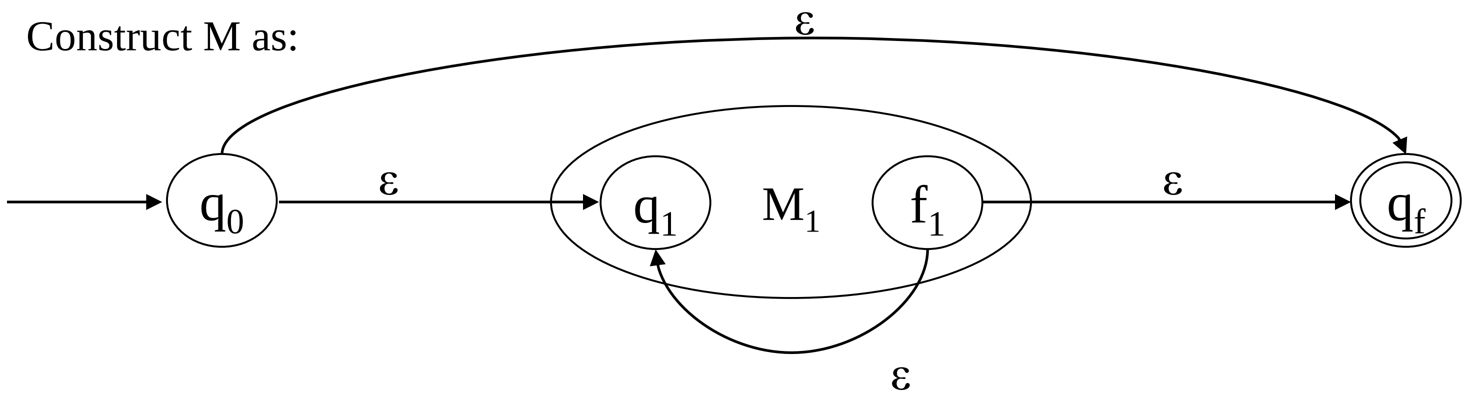
Construct M as:



Case 3) $r = r_1^*$

Since $OP(r) = k+1$, it follows that $0 <= OP(r_1) <= k$. By the inductive hypothesis there exists an NFA-ε machine M_1 such that $L(M_1) = L(r_1)$. Furthermore, M_1 has exactly one final state.

Construct M as:



- Note that the previous proof is "constructive" in that it shows us how to construct the NFA-ε from the regular expression.

- Given a regular expression, first decompose it based on the recursive definition:

$r = 0(0+1)^*$

$r = r_1r_2$

$r_1 = 0$

$r_2 = (0+1)^*$

$r_2 = r_3^*$

$r_3 = 0+1$

$r_3 = r_4 + r_5$

$r_4 = 0$

$r_5 = 1$

$r = 0(0+1)^*$

$r = r_1 r_2$

$r_1 = 0$

$r_2 = (0+1)^*$

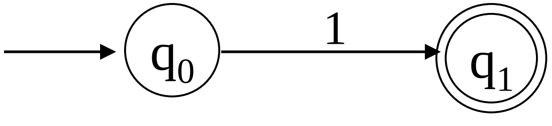
$r_2 = r_3^*$

$r_3 = 0+1$

$r_3 = r_4 + r_5$

$r_4 = 0$

$\mathbf{r_5 = 1}$



$r = 0(0+1)^*$

$r = r_1 r_2$

$r_1 = 0$

$r_2 = (0+1)^*$

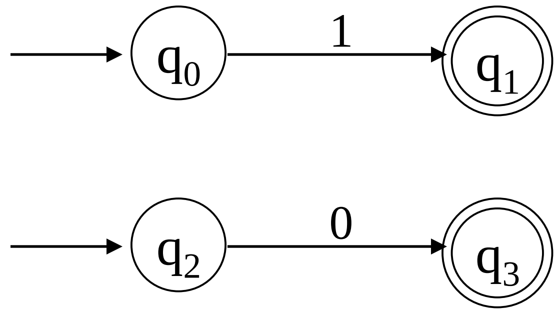
$r_2 = r_3^*$

$r_3 = 0+1$

$r_3 = r_4 + r_5$

$r_4 = 0$

$r_5 = 1$



$r = 0(0+1)^*$

$r = r_1 r_2$

$r_1 = 0$

$r_2 = (0+1)^*$

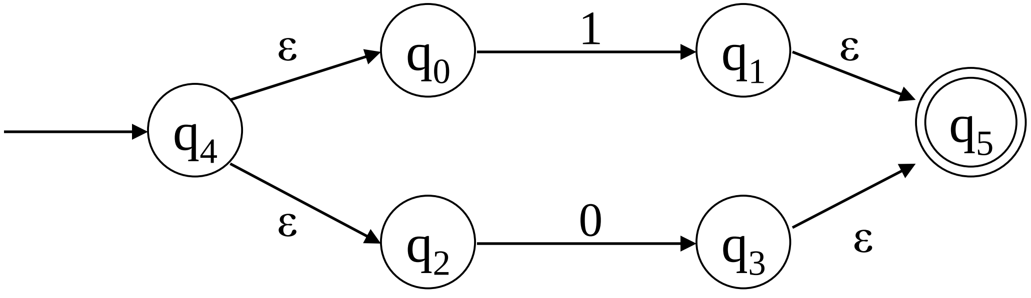
$r_2 = r_3^*$

$r_3 = 0+1$

$\mathbf{r_3 = r_4 + r_5}$

$r_4 = 0$

$r_5 = 1$



$r = 0(0+1)^*$

$r = r_1 r_2$

$r_1 = 0$

$r_2 = (0+1)^*$

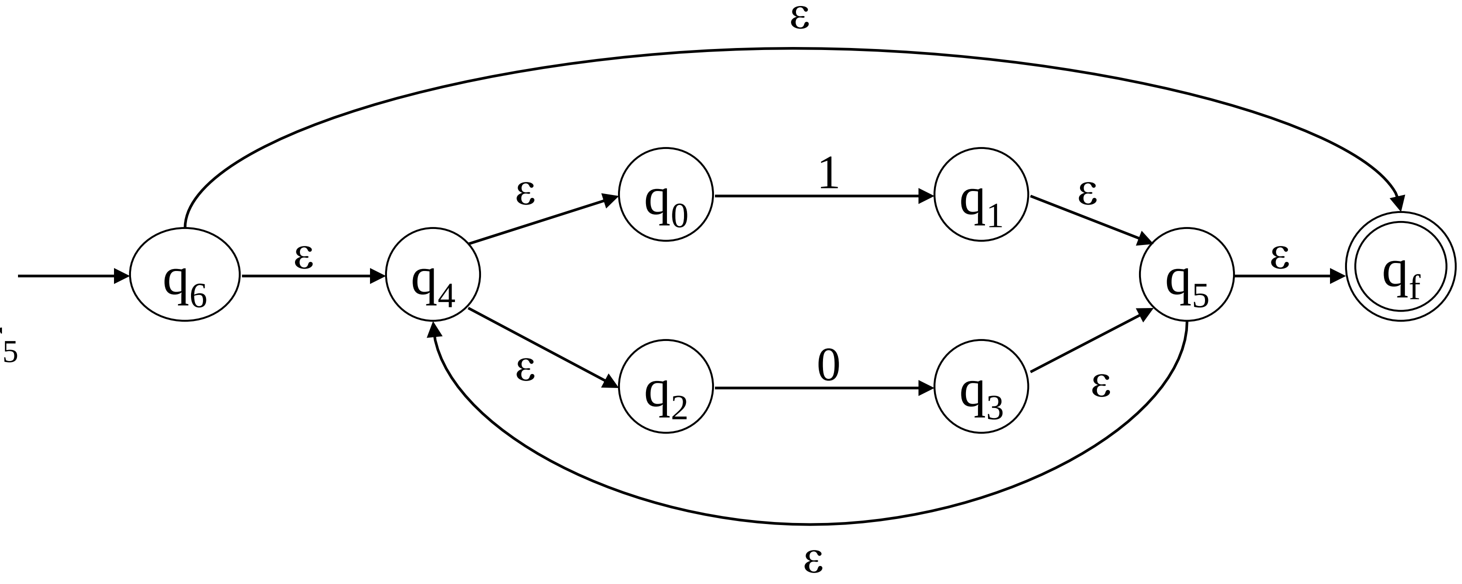
$\mathbf{r_2 = r_3^*}$

$r_3 = 0+1$

$r_3 = r_4 + r_5$

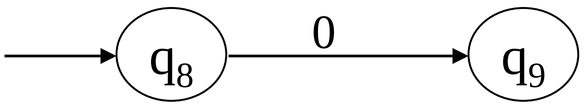
$r_4 = 0$

$r_5 = 1$



r = 0(0+1)*

r = $r_1 r_2$



r_1 = 0

r_2 = (0+1)*

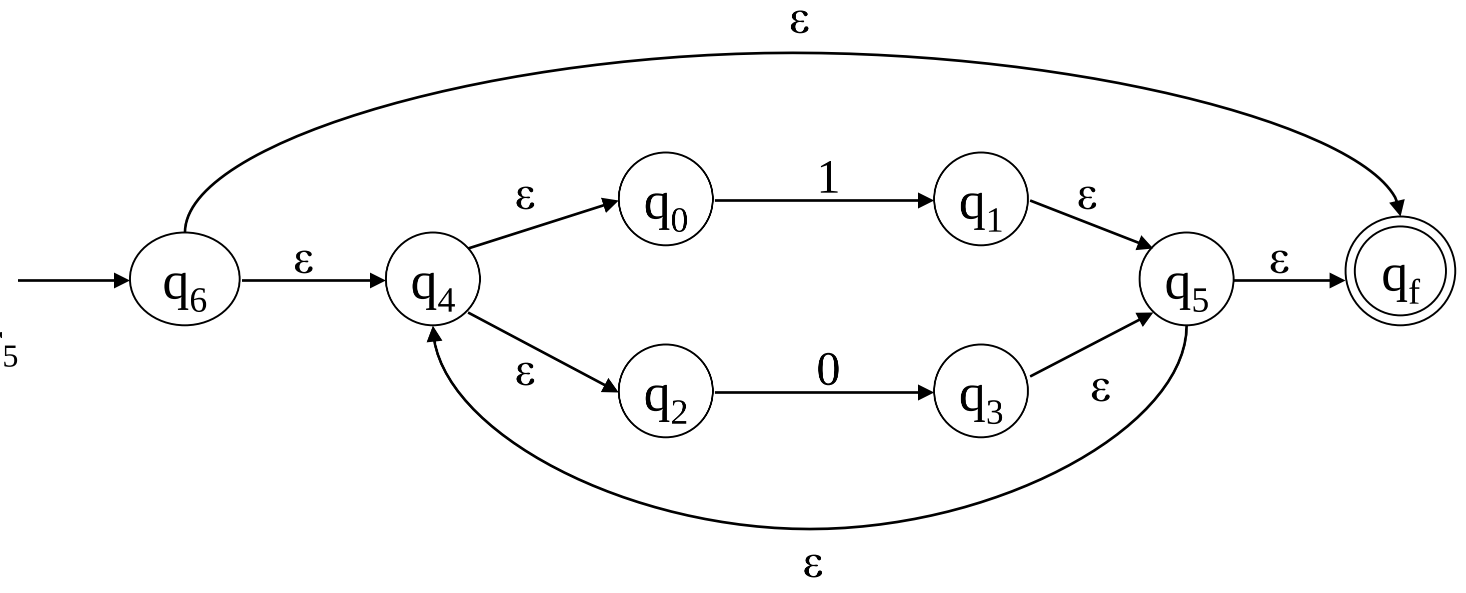
$r_2 = r_3$*

r_3 = 0+1

$r_3 = r_4 + r_5$

r_4 = 0

r_5 = 1



$r = 0(0+1)^*$

$\mathbf{r = r_1 r_2}$

$r_1 = 0$

$r_2 = (0+1)^*$

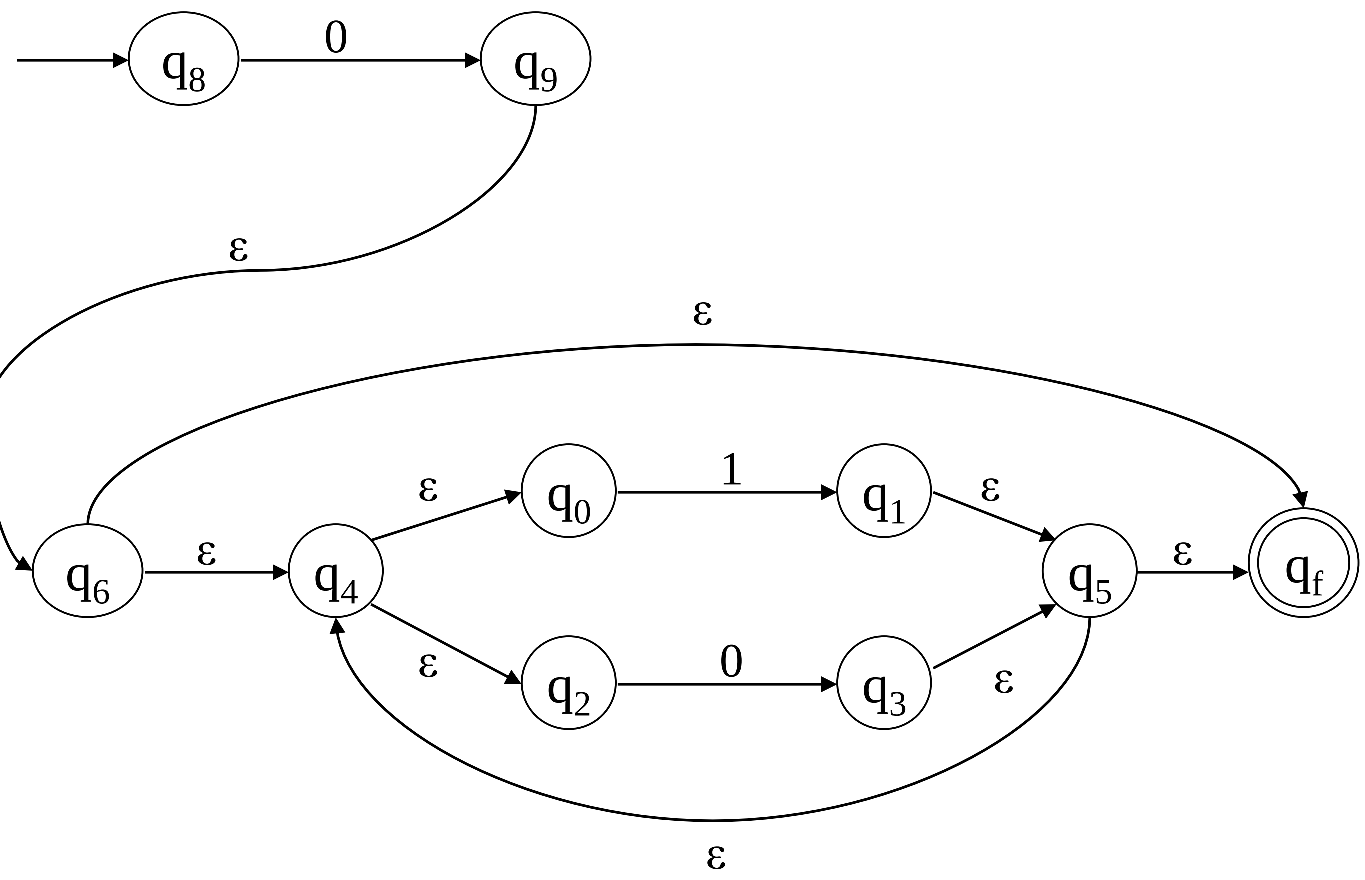
$r_2 = r_3^*$

$r_3 = 0+1$

$r_3 = r_4 + r_5$

$r_4 = 0$

$r_5 = 1$



Definitions Required to Convert a DFA to a Regular Expression

- Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA with state set $Q = \{q_1, q_2, \ldots, q_n\}$, and define:

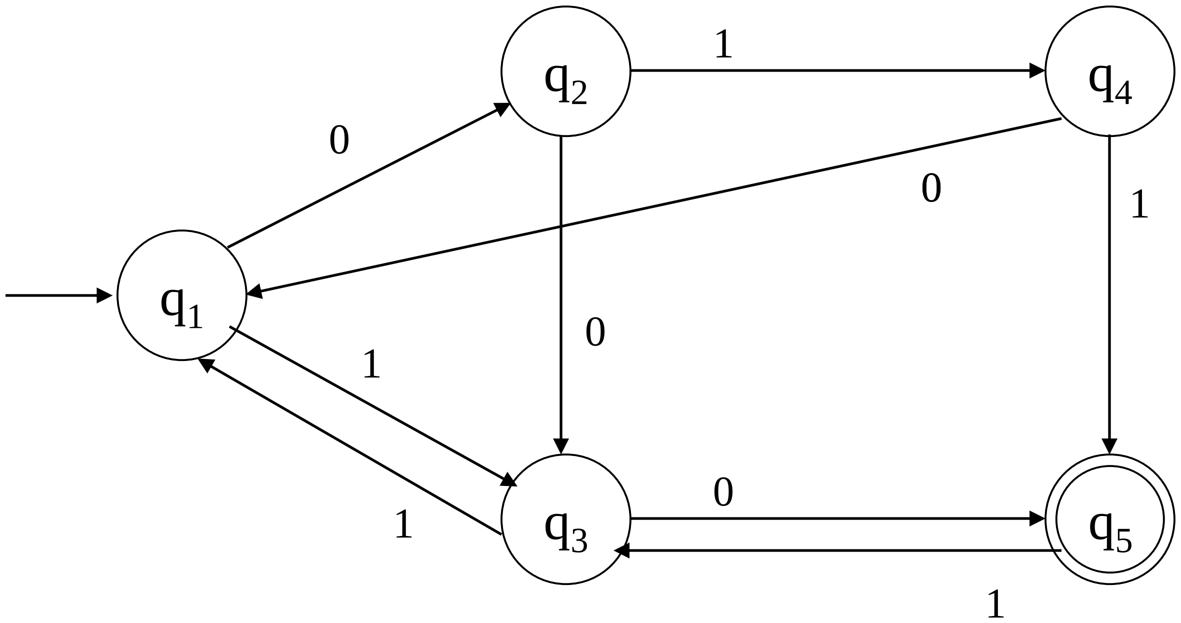
  $R_{i,j} = \{ x \mid x \text{ is in } \Sigma^* \text{ and } \delta(q_i,x) = q_j\}$ for any i,j, where $1 <= i,j <= n$

  $R_{i,j}$ is the set of all strings that define a path in M from q_i to q_j.

- Note that states have been numbered starting at 1!

- This has been done simply for convenience, and it is "without loss of generality."

- Example:



$R_{2,3} = \{0, 001, 00101, 011, \ldots\}$

$R_{1,4} = \{01, 00101, \ldots\}$

$R_{3,3} = \{11, 100, \ldots\}$

- Another definition:

  $R^k_{i,j}$ = { x | x is in Σ^* and $\delta(q_i,x) = q_j$, and for no u where 1 <= |u| < |x| and x = uv is it the case that $\delta(q_i,u) = q_p$ where p>k}

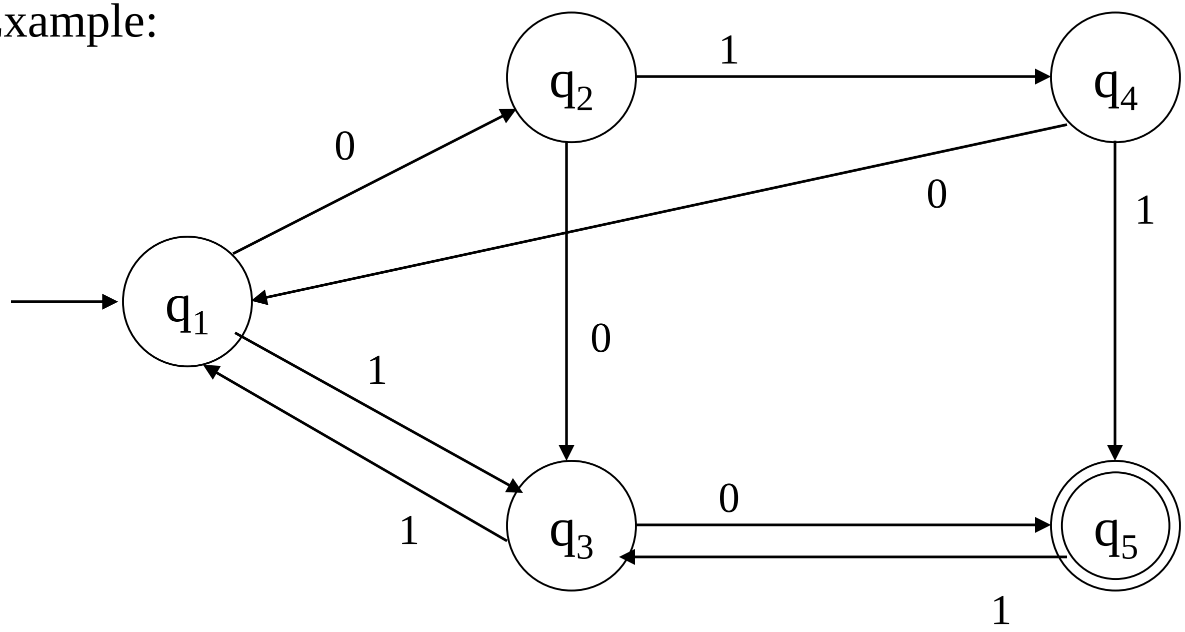
  for any i,j,k, where 1 <= i,j <= n and 0 <= k <= n

- In other words, $R^k_{i,j}$ is the set of all strings that define a path in M from q_i to q_j but that pass through no state numbered greater than k.

- Here, the phrase *pass through a state q* means that the machine enters the state *q* at some point, and then (subsequently) leaves that state *q*.

- Consequently it may be the case that i>k or j>k for $R^k_{i,j}$.

- Example:



$R^4_{2,3} = \{0, 1000, 011, \ldots\}$

111 is not in $R^4_{2,3}$

$R^2_{1,5} = \{\}$

$R^1_{2,3} = \{0\}$

111 is not in $R^1_{2,3}$

101 is not in $R^1_{2,3}$

$R^5_{2,3} = R_{2,3}$

- Observations:

1) $R^n_{i,j} = R_{i,j}$ -- More generally, $R^k_{i,j} = R_{i,j}$ for any k>= n.

2) $R^{k-1}_{i,j}$ is a subset of $R^k_{i,j}$

3) $L(M) = \bigcup_{q \in F} R^n_{1,q}$

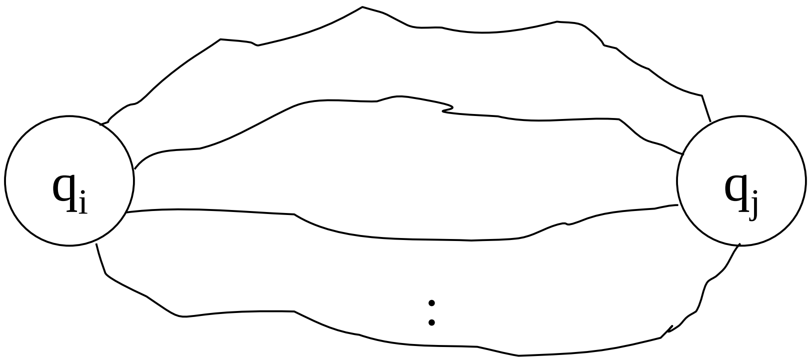
4) $R^0_{i,j} = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \bigcup \{\varepsilon\} & i = j \end{cases}$ -- Easily computed from the DFA!

5) $R^k_{i,j} = R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j} \cup R^{k-1}_{i,j}$ For k>=1

- Explanation of 5:

  5) $R^{k}_{i,j} = R^{k-1}_{i,k} (R^{k-1}_{k,k})^{*} R^{k-1}_{k,j} \cup R^{k-1}_{i,j}$

- Consider paths represented by the strings in $R^{k}_{i,j}$:



- If x is a string in $R^{k}_{i,j}$ then no state numbered > k is passed through when processing x.

- Any state numbered <= k, on the other hand, may or may not appear on the path while processing x; this includes, in particular, state q_k

- So there are two cases:
  - q_k is not passed through, i.e., x is in $R^{k-1}_{i,j}$
  - q_k is passed through one or more times, i.e., x is in $R^{k-1}_{i,k} (R^{k-1}_{k,k})^{*} R^{k-1}_{k,j}$

- **Lemma 2:** Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA. Then there exists a regular expression r such that L(M) = L(r).

- **Proof:**

  First we will show (by induction on k) that for all i,j, and k, where $1 <= i,j <= n$ and $0 <= k <= n$, there exists a regular expression r such that $L(r) = R^k_{i,j}$.

  Throughout the following, the regular expression representing $R^k_{i,j}$ will be denoted by $r^k_{i,j}$.

Basis: k=0

$R^0_{i,j}$ contains single symbols, one for each transition from q_i to q_j, and possibly ε if i=j.

case 1) No transitions from q_i to q_j and $i \neq j$

$$r^0_{i,j} = \emptyset$$

case 2) At least one (m>=1) transition from q_i to q_j and $i \neq j$

$$r^0_{i,j} = a_1 + a_2 + a_3 + \ldots + a_m$$

where $\delta(q_i, a_p) = q_j$, for all $1 <= p <= m$

case 3) No transitions from q_i to q_j and $i = j$

$$r^0_{i,j} = \varepsilon$$

case 4) At least one (m>=1) transition from q_i to q_j and $i = j$

$$r^0_{i,j} = a_1 + a_2 + a_3 + \ldots + a_m + \varepsilon$$ where $\delta(q_i, a_p) = q_j$, for all $1 <= p <= m$

Inductive Hypothesis:

Suppose there exists a k>=1 such that $R^{k-1}_{i,j}$ can be represented by a regular expression, for all 1<= i,j <=n. Let that regular expression be denoted by $r^{k-1}_{i,j}$.

Inductive Step:

Consider $R^{k}_{i,j} = R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j} \cup R^{k-1}_{i,j}$.

By the inductive hypothesis $R^{k-1}_{i,k}$ can be represented by a regular expression, denoted $r^{k-1}_{i,k}$.

Similarly, $R^{k-1}_{k,k}$, $R^{k-1}_{k,j}$, and $R^{k-1}_{i,j}$ can all be represented by regular expressions, denoted $r^{k-1}_{k,k}$, $r^{k-1}_{k,j}$, and $r^{k-1}_{i,j}$, respectively.

Thus, if we let

$$r^{k}_{i,j} = r^{k-1}_{i,k} (r^{k-1}_{k,k})^* r^{k-1}_{k,j} + r^{k-1}_{i,j}$$

then $r^{k}_{i,j}$ is a regular expression generating $R^{k}_{i,j}$,i.e., $L(r^{k}_{i,j}) = R^{k}_{i,j}$.

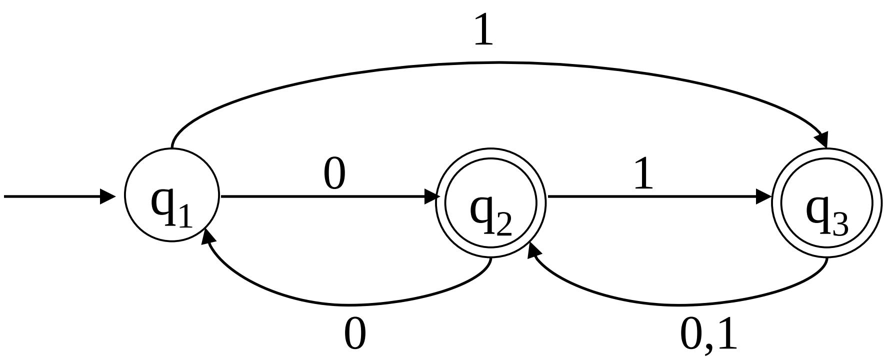
- Finally, if $F = \{q_{j1}, q_{j2}, \ldots, q_{jr}\}$, then

$$r^n_{1,j1} + r^n_{1,j2} + \ldots + r^n_{1,jr}$$

is a regular expression generating L(M).

- Not only does this prove that the regular expressions generate the regular languages, but it also provides an algorithm for computing it!

- **Example:**



First table column is computed from the DFA.

	k = 0	k = 1	k = 2
$r^k_{1,1}$	ε		
$r^k_{1,2}$	0		
$r^k_{1,3}$	1		
$r^k_{2,1}$	0		
$r^k_{2,2}$	ε		
$r^k_{2,3}$	1		
$r^k_{3,1}$	Ø		
$r^k_{3,2}$	0 + 1		
$r^k_{3,3}$	ε		

- All remaining columns are computed from the previous column using the formula.

$$
\begin{aligned}
r^1_{2,3} &= r^0_{2,1}\,(r^0_{1,1})^*\, r^0_{1,3} + r^0_{2,3} \\
&= 0\,(\varepsilon)^*\,1 + 1 \\
&= 01 + 1
\end{aligned}
$$

	k = 0	k = 1	k = 2
$r^k_{1,1}$	ε	ε	
$r^k_{1,2}$	0	0	
$r^k_{1,3}$	1	1	
$r^k_{2,1}$	0	0	
$r^k_{2,2}$	ε	ε + 00	
$r^k_{2,3}$	1	01 + 1	
$r^k_{3,1}$	Ø	Ø	
$r^k_{3,2}$	0 + 1	0 + 1	
$r^k_{3,3}$	ε	ε	

$$r^2_{1,3} = r^1_{1,2} (r^1_{2,2})^* r^1_{2,3} + r^1_{1,3}$$
$$= 0 (\varepsilon + 00)^* (1 + 01) + 1$$
$$= 0^*1$$

	k = 0	k = 1	k = 2
$r^k_{1,1}$	ε	ε	(00)*
$r^k_{1,2}$	0	0	0(00)*
$r^k_{1,3}$	1	1	0*1
$r^k_{2,1}$	0	0	0(00)*
$r^k_{2,2}$	ε	ε + 00	(00)*
$r^k_{2,3}$	1	1 + 01	0*1
$r^k_{3,1}$	Ø	Ø	(0 + 1)(00)*0
$r^k_{3,2}$	0 + 1	0 + 1	(0 + 1)(00)*
$r^k_{3,3}$	ε	ε	ε + (0 + 1)0*1

- To complete the regular expression, we compute:

$r^3_{1,2} + r^3_{1,3}$

	k = 0	k = 1	k = 2
$r^k_{1,1}$	ε	ε	(00)*
$r^k_{1,2}$	0	0	0(00)*
$r^k_{1,3}$	1	1	0*1
$r^k_{2,1}$	0	0	0(00)*
$r^k_{2,2}$	ε	ε + 00	(00)*
$r^k_{2,3}$	1	1 + 01	0*1
$r^k_{3,1}$	Ø	Ø	(0 + 1)(00)*0
$r^k_{3,2}$	0 + 1	0 + 1	(0 + 1)(00)*
$r^k_{3,3}$	ε	ε	ε + (0 + 1)0*1

- **Theorem:** Let L be a language. Then there exists an a regular expression r such that L = L(r) if and only if there exits a DFA M such that L = L(M).

- **Proof:**

  (if) Suppose there exists a DFA M such that L = L(M). Then by Lemma 2 there exists a regular expression r such that L = L(r).

  (only if) Suppose there exists a regular expression r such that L = L(r). Then by Lemma 1 there exists a DFA M such that L = L(M).•

- **Corollary:** The regular expressions define the regular languages.

- **Note:** With the completion of Lemma 1, the conversion from a regular expression to a DFA and a program accepting L(r) is now complete, and fully automated!