As another example consider a binary serial adder. At any time it gets two binary inputs $x_1$ and $x_2$. The adder can be in any one of the states ‘carry’ or ‘no carry’. The four possibilities for the inputs $x_1x_2$ are 00, 01, 10, 11. Initially the adder is in the ‘no carry’ state. The working of the serial adder can be represented by the following diagram.
denotes that when the adder is in state $p$ and gets input $x_1x_2$, it goes to state $q$ and outputs $x_3$. The input and output on a transition from $p$ to $q$ is denoted by i/o. It can be seen that suppose the two binary numbers to be added are 100101 and 100111.

\[
\begin{array}{ccccccc}
\text{Time} & 6 & 5 & 4 & 3 & 2 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

The input at time $t = 1$ is 11 and the output is 0 and the
machine goes to ‘carry’ state. The output is 0. Here at time $t = 2$, the input is 01; the output is 0 and the machine remains in ‘carry’ state. At time $t = 3$, it gets 11 and output 1 and remains in ‘carry’ state. At time $t = 4$, the input is 00; the machine outputs 1 and goes to ‘no carry’ state. At time $t = 5$, the input is 00; the output is 0 and the machine remains in ‘no carry’ state. At time $t = 6$, the input is 11; the machine outputs 0 and goes to ‘carry’ state. The input stops here. At time $t = 7$, no input is there (and this is taken as 00) and the output is 1.

It should be noted that at time $t = 1, 3, 6$ input is 11, but the output is 0 at $t = 1, 6$ and is 1 at time $t = 3$. At time $t = 4, 5$, the input is 00, but the output is 1 at time $t = 4$ and 0 at $t = 5$. 
So it is seen that the output depends both on the input and the state. The diagrams we have seen are called state diagrams.

The above diagram indicates that in state $q$ when the machine gets input $i$, it goes to state $p$ and outputs 0. Let us consider one more example of a state diagram given in

```
q   i/o   p
```

Introduction to Formal Languages, Automata and Computability – p.5/51
The input and output alphabet are \( \{0, 1\} \). For the input 011010011, the output is 00110100 and machine is in state \( q_1 \). It can be seen that the first is 0 and afterwards, the output is the symbol read at the previous instant. It can also be noted that the machine goes to \( q_1 \) after reading a 1 and goes to \( q_0 \) after reading a 0.
It should also be noted that when it goes from state $q_0$ it outputs a 0 and when it goes from state $q_1$, it outputs a 1. This machine is called a one moment delay machine.
Deterministic Finite State Automaton

**Definition** A Deterministic Finite State Automaton (DFSA) is a 5-tuple

\[ M = (K, \Sigma, \delta, q_0, F) \]

where

- \( K \) is a finite set of states
- \( \Sigma \) is a finite set of input symbols
- \( q_0 \) in \( K \) is the start state or initial state
- \( F \subseteq K \) is set of final states
- \( \delta \), the transition function is a mapping from
  \[ K \times \Sigma \rightarrow K. \]

\[ \delta(q, a) = p \]
means, if the automaton is in state \( q \) and reading a symbol \( a \), it goes to state \( p \) in the next...
instant, moving the pointer one cell to the right. \(\hat{\delta}\) is an extension of \(\delta\) and \(\hat{\delta} : K \times \Sigma^* \rightarrow K\) as follows:

- \(\hat{\delta}(q, \epsilon) = q\) for all \(q\) in \(K\)
- \(\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)\) \(x \in \Sigma^*, q \in K, a \in \Sigma\).

Since \(\hat{\delta}(q, a) = \delta(q, a)\), without any confusion we can use \(\delta\) for \(\hat{\delta}\) also. The language accepted by the automaton is defined as

\[ T(M) = \{ w \mid w \in T^*, \delta(q_0, w) \in F \} \].
Example Let a DFSA have state set \( \{q_0, q_1, q_2, D\} \); \( q_0 \) is the initial state; \( q_2 \) is the only final state. The state diagram of the DFSA is given in the following figure.
contd

\[
a a a b \\
\uparrow \\
q_0 \\
a a a b \\
\uparrow \\
q_1 \\
a a a b \\
\uparrow \\
q_1 \\
a a a b \\
\uparrow \\
q_1 \\
a a a b \\
\uparrow \\
q_2
\]
After reading $aaabb$, the automaton reaches a final state. It is easy to see that

$$T(M) = \{a^n b^m / n, m \geq 1\}$$

There is a reason for naming the fourth state as $D$. Once the control goes to $D$, it cannot accept the string, as from $D$ the automaton cannot go to a final state. On further reading any symbol the state remains as $D$. Such a state is called a dead state or a sink state.
Consider the following state diagram of a nondeterministic FSA.

On a string *aaabb* the transition can be looked at as follows.
**Definition** A Nondeterministic Finite State Automaton (NFSA) is a 5-tuple $M = (K, \Sigma, \delta, q_0, F)$ where $K, \Sigma, \delta, q_0, F$ are as given for DFSA and $\delta$, the transition function is a mapping from $K \times \Sigma$ into finite subsets of $K$. The mappings are of the form $\delta(q, a) = \{p_1, \ldots, p_r\}$ which means if the automaton is in state $q$ and reads ‘$a$’ then it can go to any one of the states $p_1, \ldots, p_r$. $\delta$ is extended as $\hat{\delta}$ to $K \times \Sigma^*$ as follows:

$$\hat{\delta}(q, \epsilon) = \{q\} \text{ for all } q \text{ in } K.$$
If $P$ is a subset of $K$

$$\delta(P, a) = \bigcup_{p \in P} \delta(p, a)$$

$$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

$$\hat{\delta}(P, x) = \bigcup_{p \in P} \hat{\delta}(p, x)$$

Since $\delta(q, a)$ and $\hat{\delta}(q, a)$ are equal for $a \in \Sigma$, we can use the same symbol $\delta$ for $\hat{\delta}$ also.

The set of strings accepted by the automaton is denoted by $T(M)$. 
\[ T(M) = \{ w/w \in T^*, \delta(q_0, w) \text{ contains a state from } F \} \]

The automaton can be represented by a state table also. For example the state diagram given in the figure can be represented as the state table given below

\[
\begin{array}{c|cc}
\text{a} & \text{b} \\
\hline
\{q_0, q_1\} & \phi \\
\phi & \{q_0, q_1\} \\
\phi & \phi \\
\end{array}
\]
Example The state diagram of an NFSA which accepts binary strings which have at least one pair ‘00’ or one pair ‘11’ is
Theorem If $L$ is accepted by a NFSA then $L$ is accepted by a DFSA. Let $L$ be accepted by a NFSA $M = (K, \Sigma, \delta, q_0, F)$. Then we construct a DFSA $M' = (K', \Sigma', \delta', q'_0, F')$ as follows: $K' = \mathcal{P}(K)$, power set of $K$. Corresponding to each subset of $K$, we have a state in $K'$. $q'_0$ corresponds to the subset containing $q_0$ alone. $F'$ consists of states corresponding to subsets having at least one state from $F$. We define $\delta'$ as follows:

$$\delta'([q_1, \ldots, q_k], a) = [r_1, r_2, \ldots, r_s]$$

if and only if

$$\delta\left(\{q_1, \ldots, q_k\}, a\right) = \{r_1, r_2, \ldots, r_s\}.$$

We show that $T(M) = T(M')$. 

Introduction to Formal Languages, Automata and Computability – p.18/51
We prove this by induction on the length of the string. We show that
\[ \delta'(q_0', x) = [p_1, \ldots, p_r] \]
if and only if \[ \delta(q_0, x) = \{p_1, \ldots, p_r\} \]

**Basis**

\[ |x| = 0 \text{ i.e., } x = \epsilon \]
\[ \delta'(q_0', \epsilon) = q_0' = [q_0] \]
\[ \delta(q_0, \epsilon) = \{q_0\} \]

**Induction**

Assume that the result is true for strings \( x \) of length upto \( m \). We have to prove for string of length \( m + 1 \).

By induction hypothesis
\[ \delta'(q'_0, x) = [p_1, \ldots, p_r] \]

if and only if \( \delta(q_0, x) = \{p_1, \ldots, p_r\} \).

\[ \delta'(q'_0, xa) = \delta'([p_1, \ldots, p_r], a), \]

\[ \delta(q_0, xa) = \bigcup_{p \in P} \delta(p, a), \]

where \( P = \{p_1, \ldots, p_r\} \).

Suppose \( \bigcup_{p \in P} \delta(p, a) = \{s_1, \ldots, s_m\} \)

\[ \delta(\{p_1, \ldots, p_r\}, a) = \{s_1, \ldots, s_m\} \]  

By our construction

\[ \delta'([p_1, \ldots, p_r], a) = [s_1, \ldots, s_m] \]  

and hence

\[ \delta'(q'_0, xa) = \delta'([p_1, \ldots, p_r], a) = [s_1, \ldots, s_m]. \]
In $M'$, any state representing a subset having a state from $F$ is in $F'$.

So if a string $w$ is accepted in $M$, there is a sequence of states which takes $M$ to a final state $f$ and $M'$ simulating $M$ will be in a state representing a subset containing $f$. Thus $L(M) = L(M')$. 
Example Let us construct the DFSA for the NFSA given by the table in previous figure. We construct the table for DFSA.

\[
\begin{array}{c|cc}
  & a & b \\
\hline
[q_0] & [q_0, q_1] & [\phi] \\
[q_0, q_1] & [q_0, q_1] & [q_1, q_2] \\
[q_1, q_2] & [\phi] & [q_1, q_2] \\
[\phi] & [\phi] & [\phi] \\
\end{array}
\]

\[
\delta'( [q_0, q_1], a ) = [\delta(q_0, a) \cup \delta(q_1, a)] \\
= [\{q_0, q_1\} \cup \phi] \\
= [q_0, q_1]
\]
\[
\delta'(\[q_0, q_1\], b) = \left[ \delta(q_0, b) \cup \delta(q_1, b) \right] \\
= [\phi \cup \{q_1, q_2\}] \\
= [q_1, q_2]
\] (4)
(5)
(6)

The state diagram is given
Definition An NFSA with \( \varepsilon \)-transition is a 5-tuple \( M = (K, \Sigma, \delta, q_0, F) \). where \( K, \Sigma, \delta, q_0, F \) are as defined for NFSA and \( \delta \) is a mapping from \( K \times (\Sigma \cup \{\varepsilon\}) \) into finite subsets of \( K \). \( \delta \) can be extended as \( \hat{\delta} \) to \( K \times \Sigma^* \) as follows. First we define the \( \varepsilon \)-closure of a state \( q \). It is the set of states which can be reached from \( q \) by reading \( \varepsilon \) only. Of course, \( \varepsilon \)-closure of a state includes itself. \( \hat{\delta}(q, \varepsilon) = \varepsilon \)-closure\( (q) \).
For $w$ in $\Sigma^*$ and $a$ in $\Sigma$, $\hat{\delta}(q, wa) = \epsilon$-closure$(P)$, where

$P = \{p | \text{ for some } r \text{ in } \hat{\delta}(q, w), p \text{ is in } \delta(r, a)\}$

Extending $\delta$ and $\hat{\delta}$ to a set of states, we get

$\delta(Q, a) = \bigcup_{q \text{ in } Q} \delta(q, a)$

$\delta(Q, w) = \bigcup_{q \text{ in } Q} \delta(q, w)$

The language accepted is defined as

$T(M) = \{w | \hat{\delta}(q, w) \text{ contains a state in } F\}$.

**Theorem** Let $L$ be accepted by a NFSA with $\epsilon$-moves. Then $L$ can be accepted by a NFSA without $\epsilon$-moves.

Let $L$ be accepted by a NFSA with $\epsilon$-moves

$M = (K, \Sigma, \delta, q_0, F)$. Then we construct a NFSA
$M' = (K, \Sigma, \delta', q_0, F')$ without $\epsilon$-moves for accepting $L$ as follows.

$$F' = F \cup \{q_0\}$$ if $\epsilon$-closure of $q_0$ contains a state from $F$.

$$= F$$ otherwise.

$\delta'(q, a) = \hat{\delta}(q, a)$.

We should show $T(M) = T(M')$.

We wish to show by induction on the length of the string $x$ accepted that $\delta'(q_0, x) = \hat{\delta}(q_0, x)$. We start the basis with $|x| = 1$ because for $|x| = 0$, i.e., $x = \epsilon$ this may not hold. We may have $\delta'(q_0, \epsilon) = \{q_0\}$ and $\hat{\delta}(q_0, \epsilon) = \epsilon$-closure of $q_0$ which may include other states.
Basis
\[ |x| = 1. \text{ Then } x \text{ is a symbol of } \Sigma \text{ say } a, \text{ and } \delta'(q_0, a) = \hat{\delta}(q_0, a) \text{ by our definition of } \delta'. \]

Induction
\[ |x| > 1. \text{ Then } x = ya \text{ for some } y \in \Sigma^* \text{ and } a \in \Sigma. \]

Then \( \delta'(q_0, ya) = \delta'(\delta'(q_0, y), a) \).

By the inductive hypothesis \( \delta'(q_0, y) = \hat{\delta}(q_0, y) \).

Let \( \hat{\delta}(q_0, y) = P \).
\[
\delta'(P, a) = \bigcup_{p \in P} \delta'(p, a) = \bigcup_{p \in P} \delta(p, a)
\]

\[
\bigcup_{p \in P} \hat{\delta}(p, a) = \hat{\delta}(q_0, ya)
\]

Therefore \(\delta'(q_0, ya) = \hat{\delta}(q_0, ya)\)

It should be noted that \(\delta'(q_0, x)\) contains a state in \(F'\) if and only if \(\hat{\delta}(q_0, x)\) contains a state in \(F\).
Consider the $\epsilon$-NFSA of previous example. By our construction we get the NFSA without $\epsilon$-moves given in the following figure.
\[ \varepsilon\text{-closure of } (q_0) = \{q_0, q_1\} \]
\[ \varepsilon\text{-closure of } (q_1) = \{q_1\} \]
\[ \varepsilon\text{-closure of } (q_2) = \{q_2, q_3\} \]
\[ \varepsilon\text{-closure of } (q_3) = \{q_3\} \]

It is not difficult to see that the language accepted by the above \( NFSA = \{a^nb^mc^pd^q/m \geq 1, n, p, q \geq 0\} \).
**Definition** Let $\Sigma$ be an alphabet. For each $a$ in $\Sigma$, $a$ is a regular expression representing the regular set $\{a\}$. $\phi$ is a regular expression representing the empty set. $\epsilon$ is a regular expression representing the set $\{\epsilon\}$. If $r_1$ and $r_2$ are regular expressions representing the regular sets $R_1$ and $R_2$ respectively, then $r_1 + r_2$ is a regular expression representing $R_1 \cup R_2$. $r_1 r_2$ is a regular expression representing $R_1 R_2$. $r_1^*$ is a regular expression representing $R_1^*$. Any expression obtained from $\phi$, $\epsilon$, $a (a \in \Sigma)$ using the above operations and parentheses where required is a regular expression.

**Example** $(ab)^*abcd$ represent the regular set $\{(ab)^n cd \mid n \geq 1\}$.
Theorem: If $r$ is a regular expression representing a regular set, we can construct an NFSA with $\epsilon$-moves to accept $r$.

$r$ is obtained from $a$, $(a \in \Sigma)$, $\epsilon$, $\phi$ by finite number of applications of $+$, $.$ and $\ast$ ($.$ is usually left out).

For $\epsilon$, $\phi$, $a$ we can construct NFSA with $\epsilon$-moves are.

- $\{\epsilon\}$ is accepted
- $\phi$ is accepted
- $a$ is accepted
Let \( r_1 \) represent the regular set \( R_1 \) and \( R_1 \) is accepted by the NFSA \( M_1 \) with \( \epsilon \)-transitions.

\[ M_1 \]

\( q_{01} \rightarrow \]

Without loss of generality we can assume that each such NFSA with \( \epsilon \)-moves has only one final state. \( R_2 \) is similarly accepted by an NFSA \( M_2 \) with \( \epsilon \)-transition.

\[ M_2 \]

\( q_{02} \rightarrow \]

Introduction to Formal Languages, Automata and Computability – p.33/51
Now we can easily see that $R_1 \cup R_2$ (represented by $r_1 + r_2$) is accepted by the NFSA given in next figure
For this NFSA \(q_0\) is the start state and \(q_f\) is the final state.

\(R_1 R_2\) represented by \(r_1 r_2\) is accepted by the NFSA with \(\epsilon\)-moves given as

For this NFSA with \(\epsilon\)-moves \(q_{01}\) is the initial state and \(q_{f2}\) is the final state.

\[
R^*_1 = R^0_1 \cup R^1_1 \cup R^2_1 \cup \cdots \cup R^k_1 \cup \cdots
\]

\[
R^0_1 = \{\epsilon\} \text{ and } R'_1 = R_1.
\]
$R_1^*$ represented by $r_1^*$ is accepted by the NFSA with $\varepsilon$-moves given as

For this NFSA with $\varepsilon$-moves $q_0$ is the initial state and $q_f$ is the final state. It can be seen that $R_1^*$ contains strings of the form $x_1, x_2, \ldots, x_k$ each $x_i \in R_1$. To accept
this string, the control goes from $q_0$ to $q_{01}$ and then after reading $x_1$ and reaching $q_{f1}$, it goes to $q_{01}$, by an $\epsilon$-transition. From $q_{01}$, it again reads $x_2$ and goes to $q_{f1}$. This can be repeated a number ($k$) of times and finally the control goes to $q_f$ from $q_{f1}$ by an $\epsilon$-transition. $R_{1}^{0} = \{\epsilon\}$ is accepted by going to $q_f$ from $q_0$ by an $\epsilon$-transition.

Thus we have seen that given a regular expression one can construct an equivalent NFSA with $\epsilon$-transitions.
Consider a regular expression $aa^*bb^*$. $a$ and $b$ are accepted by NFSA with $\epsilon$-moves given in the following figure.
$aa^*bb^*$ will be accepted by NFSA with $\epsilon$-moves given in next figure.

But we have already seen that a simple NFSA can be drawn easily for this as in next figure.
**Definition** Let $L \subseteq \Sigma^*$ be a language and $x$ be a string in $\Sigma^*$. Then the derivative of $L$ with respect to $x$ is defined as

$$L_x = \{ y \in \Sigma^* / xy \in L \}.$$  

It is sometimes denoted as $\partial_x L$.

**Theorem** If $L$ is a regular set $L_x$ is regular for any $x$. Consider a DFSA accepting $L$. Let this FSA be $M = (K, \Sigma, \delta, q_0, F)$. Start from $q_0$ and read $x$ to go to state $q_x \in K$.

Then $M' = (K, \Sigma, \delta, q_x, F)$ accepts $L_x$. This can be seen easily as below.
\[ \delta(q_0, x) = q_x, \]
\[ \delta(q_0, xy) \in F \iff xy \in L, \]
\[ \delta(q_0, xy) = \delta(q_x, y), \]
\[ \delta(q_x, y) \in F \iff y \in L_x, \]
\[ \therefore M' \text{ accepts } L_x. \]

**Lemma** Let \( \Sigma \) be an alphabet. The equation \( X = AX \cup B \) where \( A, B \subseteq \Sigma^* \) has a unique solution \( A^*B \) if \( \epsilon \notin A. \)
Let

\[
X = AX \cup B
= A(AX \cup B) \cup B
= A^2X \cup AB \cup B
= A^2(AX \cup B) \cup AB \cup B
= A^3X \cup A^2B \cup AB \cup B
\vdots
= A^{n+1}X \cup A^nB \cup A^{n-1}B \cup \cdots \cup AB \cup B(7)
\]

Since \( \epsilon \notin A \), any string in \( A^k \) will have minimum length \( k \).

To show \( X = A^*B \).
Let \( w \in X \) and \( |w| = n \). We have

\[
X = A^{n+1}X \cup A^n B \cup \cdots \cup AB \cup B \tag{8}
\]

Since any string in \( A^{n+1}X \) will have minimum length \( n + 1 \), \( w \) will belong to one of \( A^k B \), \( k \leq n \). Hence \( w \in A^* B \). On the other hand let \( w \in A^* B \). To prove \( w \in X \). Since \( |w| = n \), \( w \in A^k B \) for some \( k \leq n \). Therefore from (8) \( w \in X \).

Hence we find that the unique solution for \( X = AX + B \) is \( X = A^* B \).

**Note:** If \( \epsilon \in A \), the solution will not be unique. Any \( A^* C \), where \( C \supseteq B \), will be a solution.
Next we give an algorithm to find the regular expression corresponding to a DFSA.

**Algorithm**

Let $M = (K, \Sigma, \delta, q_0, F)$ be the DFSA.

$\Sigma = \{a_1, a_2, \ldots, a_k\}$, $K = \{q_0, q_1, \ldots, q_{n-1}\}$.

**Step 1** Write an equation for each state in $K$.

$q = a_1q_{i1} + a_2q_{i2} + \cdots + a_kq_{ik}$

if $q$ is not a final state and $\delta(q, a_j) = q_{ij}$, $1 \leq j \leq k$.

$q = a_1q_{i1} + a_2q_{i2} + \cdots + a_kq_{ik} + \lambda$

if $q$ is a final state and $\delta(q, a_j) = q_{ij}$, $1 \leq j \leq k$.

**Step 2** Take the $n$ equations with $n$ variables $q_i$, $1 \leq i \leq n$, and solve for $q_0$ using the above lemma and substitution.
Step 3  Solution for $q_0$ gives the desired regular expression. Let us execute this algorithm for the following DFSA given in the figure.
Step 1

\[ q_0 = aq_1 + bD \quad (9) \]
\[ q_1 = aq_1 + bq_2 \quad (10) \]
\[ q_2 = aD + bq_2 + \lambda \quad (11) \]
\[ D = aD + bD \quad (12) \]

Step 2
Solve for \( q_0 \). From (12)

\[ D = (a + b)D + \phi \]
Using previous lemma

\[ D = (a + b)^* \phi = \phi. \]  \hspace{1cm} (13)

Using them we get

\[ q_0 = aq_1 \]  \hspace{1cm} (14)

\[ q_1 = aq_1 + bq_2 \]  \hspace{1cm} (15)

\[ q_2 = bq_2 + \lambda \]  \hspace{1cm} (16)

Note that we have got rid of one equation and one variable.
In 16 using the lemma we get

\[ q_2 = b^* \]  \hspace{1cm} (17)

Now using 17 and 15

\[ q_1 = aq_1 + bb^* \]  \hspace{1cm} (18)

We now have 14 and 18. Again we eliminated one equation and one variable.

Using the above lemma in (18)

\[ q_1 = a^*bb^* \]  \hspace{1cm} (19)
Using 19 in 14

\[ q_0 = aa^*bb^* \]  \hspace{1cm} (20)

This is the regular expression corresponding to the given FSA.

Next, we see, how we are justified in writing the equations.

Let \( q \) be the state of the DFSA for which we are writing the equation,

\[ q = a_1q_{i1} + a_2q_{i2} + \cdots + a_kq_{ik} + Y. \]  \hspace{1cm} (21)

\[ Y = \lambda \text{ or } \phi. \]
Let $L$ be the regular set accepted by the given DFSA. Let $x$ be a string such that starting from $q_0$, after reading $x$, state $q$ is reached. Therefore $q$ represents $L_x$, the derivative of $L$ with respect to $x$. From $q$ after reading $a_j$, the state $q_{ij}$ is reached.

$$L_x = q = a_1L_{xa_1} + a_2L_{xa_2} + \cdots + a_kL_{xa_k} + Y. \quad (22)$$

$a_jL_{xa_j}$ represents the set of strings in $L_x$ beginning with $a_j$. Hence equation (22) represents the partition of $L_x$ into strings beginning with $a_1$, beginning with $a_2$ and so on. If $L_x$ contains $\epsilon$, then $Y = \epsilon$ otherwise $Y = \emptyset$. 
It should be noted that when $L_x$ contains $\epsilon$, $q$ is a final state and so $x \in L$. It should also be noted that considering each state as a variable $q_j$, we have $n$ equation in $n$ variables. Using the above lemma, and substitution, each time one equation is removed while one variable is eliminated. The solution for $q_0$ is $L_\epsilon = L$. This gives the required regular expression.