To achieve our goal, we need the concept of Non-deterministic Finite Automaton with ε-moves (NFAε).

An NFAε is a tuple $M = (Q, \Sigma, q_0, \delta, F)$, where $\delta$ is modified to be a function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$.

There are two aspects to non-determinism:
- one-to-many transitions
- spontaneous transitions

For an NFAε, $\delta(q, a)$ is an element of $P(Q)$, i.e. a subset of $Q$, for example $\delta(q, a) = \{q_1, q_2, q_3, q_4\}$.

If we are in state $q$ and the input is $a$, we can choose to move to any one of $q_1$, $q_2$, $q_3$ or $q_4$.

Of course, if there is no input, we can also choose to stay at state $q$.

Thus, if we reach state $q$ while processing an input string, then the next transition could be from any of the states $q$, $q_1$, $q_2$ or $q_3$. 
First we define the ε-closure of a state:

(i) $q$ is in $\varepsilon$-clos($q$)

(ii) if $q_1 \in \varepsilon$-clos($q$) and $q_2 \in \delta(q_1, \varepsilon)$ then $q_2 \in \varepsilon$-clos($q$)

Thus $\varepsilon$-clos($q$) has $q$ and all the states that can be reached from $q$ by sequences of ε-transitions.

Example

The states colored gray are all in the set $\varepsilon$-clos($q$).

ε-closure of a set of states

We can extend the definition of $\varepsilon$-clos($q$) to $\varepsilon$-clos($S$) for a set of states $S$.

$\varepsilon$-clos($S$) consists of $S$ and all states reachable from a state in $S$ by ε-transitions:

$$\varepsilon$-clos($S$) = \bigcup_{q \in S} \varepsilon$-clos($q$)$

Extending $\delta$

We can modify $\delta$ to $\delta^*$ in three steps as follows:

- Define $\delta^*:Q \times \Sigma \rightarrow P(Q)$ by
  $$\delta^*(q, a) = \varepsilon$-clos(\delta(q, a))$$

- Next we extend as follows:
  $$\delta^*:P(Q) \times \Sigma \rightarrow P(Q)$$ by defining
  $$\delta^*(S, a) = \bigcup_{q \in S} \delta^*(q, a)$$

- Finally we extend to strings in $\Sigma^*$
The language of an NFA

- Define $\delta^* : P(Q) \times \Sigma^* \rightarrow P(Q)$ by
  - $\delta^*(S, \varepsilon) = \varepsilon$-clos($S$)
  - $\delta^*(S, wa) = \delta^*(\delta^*(S, w), a)$
  - Remember that every application of $\delta^*$ always adds the $\varepsilon$-closure to the set of points reached by a transition
- For an NFA $\varepsilon$ called $M$, the language of $M$ is denoted $L(M)$.
  
  $L(M) = \{ w \in \Sigma^* : \delta^*(\varepsilon$-clos($q_0, w)) \cap F \neq \emptyset \}$

Example-I

$\delta^*(\varepsilon$-clos($q_0), a) = \delta^*(\{q_0, q_2\}, a)$
$= \varepsilon$-clos($\delta(q_0, a)) \cup \varepsilon$-clos($\delta(q_2, a))$
$= \varepsilon$-clos($\emptyset) \cup \varepsilon$-clos($\{q_0, q_1\}) = \{q_0, q_1, q_2\}$

Example-II

$\delta^*(\varepsilon$-clos($q_0, ab) = \delta^*(\delta^*(\varepsilon$-clos($q_0, a), b)$
$= \delta^*(\{q_0, q_1, q_2\}, b)$
$= \varepsilon$-clos($\{q_0, q_3\})$
$= \{q_0, q_2, q_3\}$, so $ab$ is accepted
Example-III

\[ \delta^*(\varepsilon\text{-clos}(q_0), abb) = \delta^*(\delta^*(\varepsilon\text{-clos}(q_0), ab), b) \]
\[ = \delta^*(\{q_0, q_2, q_3\}, b) = \varepsilon\text{-clos}(\{q_0, q_1, q_3\}) \]
\[ = \{q_0, q_1, q_2, q_3\}, \text{ so } abb \text{ is accepted} \]

What is going on?

- The definition of acceptance by an NFA with \( \varepsilon \)-transitions is subtle and hidden by all the formal definitions.
- Given a string \( w = a_1 a_2 ... a_n \), it is accepted if you can start at \( q_0 \) and finish at a final state, possibly using \( \varepsilon \)-transitions along the way.
- So we may really be using transitions labeled something like \( \varepsilon, \varepsilon, a_1, \varepsilon, a_2, \varepsilon, ..., \varepsilon, \varepsilon, a_m, \varepsilon \)

Why non-determinism?

- We will see in our examples that an NFA with \( \varepsilon \)-transitions can be more compact and intuitive than a DFA for the same language.
- However, we have to show that they recognize the same class of languages, i.e. all an NFA with \( \varepsilon \)-transitions can do is recognize a regular language.
The following is one of many NFA$\varepsilon$’s that recognize $10^*10^*1 + 0^*1110^*1$

- The trick is simply to consider subsets of $Q$, i.e. the elements of $P(Q)$ to be states of a new automaton.
- Given an NFA$\varepsilon$ $M_1 = (Q, \Sigma, q_0, \delta, F)$, consider the automaton:
  - $M_2 = (P(Q), \Sigma, \varepsilon$-clos$(q_0), \delta_2, F_2)$, where $\delta_2 = \delta^*: P(Q) \times \Sigma \rightarrow P(Q)$ and $F_2$ is the collection of all subsets of $Q$ that contain an element of $F$
- $M_2$ IS a DFA

Recall that $\delta^*: P(Q) \times \Sigma^* \rightarrow P(Q)$ and so we can restrict the function to words of length 1 to obtain a function $\delta^*: P(Q) \times \Sigma \rightarrow P(Q)$

This last $\delta^*$ is deterministic because it takes an element of $P(Q)$ and a symbol in $\Sigma$ to another element of $P(Q)$

Now, $P(Q)$ is big: if $Q$ has 8 elements, $P(Q)$ has $2^8 = 256$ elements, which is a lot of states.
We will give a constructive method that only puts the *reachable* states into $M_2$, which is often more manageable.

However, we want $L(M_1) = L(M_2)$.

First, it is a tedious symbol-dense exercise to check that $\delta_2^* = \delta^*: P(Q) \times \Sigma^* \rightarrow P(Q)$ so acceptance of $w$ by $M_2$ requires $\delta^*(\epsilon\text{-clos}(q_0), w) \in F_2$

Symbol-dense verification

By definition $\delta_2 = \delta^*: P(Q) \times \Sigma \rightarrow P(Q)$, so consider $S$ in $P(Q)$ (i.e. $S \subseteq Q$), so $\delta_2^* = \delta^*$ agree on string of length 1

We proceed by induction, suppose $\delta_2^* = \delta^*$ agree on all strings of length up to $n$

Pick a string $w = w_1a$, where $|w| = n + 1$ and so $|w_1| = n$.

Symbol-dense verification - 2

From slide 2-34

$\delta_2^*(S, w) = \delta_2^*(S, w_1a)$

$= \delta_2(\delta_2^*(S, w_1), a)$

$= \delta^*(\delta_2^*(S, w_1), a)$,

since $\delta_2(\ldots, a) = \delta^*(\ldots, a)$ by definition

By the inductive hypothesis

$\delta_2^*(S, w_1) = \delta^*(S, w_1)$

Therefore

$\delta_2^*(S, w) = \delta^*(\delta^*(S, w_1), a)$

$= \delta^*(S, w_1a) = \delta^*(S, w)$

Symbol-dense verification - 3

Also, look at the definition of $\delta_2^*$ (for a deterministic transition function)

$\delta_2^*(S, \epsilon) = S$ (see slide 2-34) while

$\delta^*(S, \epsilon) = \epsilon\text{-clos}(S)$ (from 3-14)

These two expressions coincide if $S = \epsilon\text{-clos}(q_0)$ since $\epsilon\text{-clos}(\epsilon\text{-clos}(q_0))$

Altogether,

$\delta_2^*(\epsilon\text{-clos}(q_0), w) = \delta^*(\epsilon\text{-clos}(q_0), w)$
Equivalence is easy

- Look at the definition of $F_2$; it tells us that $\delta^*(\varepsilon\text{-clos}(q_0), w)$ is an element of $F_2$ precisely when $\delta^*(\varepsilon\text{-clos}(q_0), w)$ contains an element of $F$, i.e. $\delta^*(\varepsilon\text{-clos}(q_0), w) \cap F \neq \emptyset$
- This second criterion defines the acceptance of $w$ by $M_1$
- The language of the NFA $\varepsilon$ is the language of the DFA

Practical conversion NFA $\varepsilon$ to DFA

- Suppose we have an NFA $\varepsilon$ transition table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$\emptyset$</td>
<td>${q_4}$</td>
<td>${q_1}$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>${q_5}$</td>
<td>${q_2}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>${q_3}$</td>
<td>$\emptyset$</td>
<td>${q_5}$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\emptyset$</td>
<td>${q_2, q_4}$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$\emptyset$</td>
<td>${q_6}$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>$q_5$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>$q_6$</td>
<td>${q_6}$</td>
<td>$\emptyset$</td>
<td>${q_0}$</td>
</tr>
</tbody>
</table>

Transition graph

- Transition graph

DFA: initial state

- The initial state is $\varepsilon\text{-clos}\{q_0\} = \{q_0, q_1\}$
- We fill in the $\delta^*$-transitions from this initial state (so we always add the $\varepsilon$-closure)

\[
\begin{array}{ccc}
\{q_0, q_1\} & \{q_0, q_1, q_5\} & \{q_0, q_1, q_2, q_4, q_5\} \\
 a & b & \\
\{q_0, q_1\} & \{q_0, q_1, q_5\} & \{q_0, q_1, q_2, q_4, q_5\} \\
 b & b & \\
\{q_0, q_1\} & \{q_0, q_1, q_5\} & \{q_0, q_1, q_2, q_4, q_5\} \\
\end{array}
\]
- This is unwieldy, so we abbreviate as follows:
  $q_{0.1}$, $q_{0.1.5}$, $q_{0.1.2.4.5}$, etc.
Keep adding reachable states

- Every state we reach gets added to the left-hand column:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_{0.1}</td>
<td>q_{0.1.5}</td>
<td>q_{0.1.2.4.5}</td>
</tr>
<tr>
<td>q_{0.1.5}</td>
<td>q_{0.1.5}</td>
<td>q_{0.1.2.4.5}</td>
</tr>
<tr>
<td>q_{0.1.2.4.5}</td>
<td>q_{0.1.2.3.5}</td>
<td>q_{0.1.2.4.5.6}</td>
</tr>
<tr>
<td>q_{0.1.2.3.5}</td>
<td>q_{0.1.2.3.5}</td>
<td>q_{0.1.2.4.5}</td>
</tr>
<tr>
<td>q_{0.1.2.3.5.6}</td>
<td>q_{0.1.2.3.5.6}</td>
<td>q_{0.1.2.4.5}</td>
</tr>
</tbody>
</table>

* final states

- The DFA is smaller (unusual)!

The converted DFA

- The final DFA:

getting an NFA\(\varepsilon\) from a regular expression-

- If we have a regular expression \(r\), we need to construct an NFA\(\varepsilon\) \(M\) so that \(L(M) = L(r)\)

- A regular expression which represents a language with only one element denotes the language accepted by an NFA with a linear sequence of states:

- Regular expression \(abca\)

- NFA\(\varepsilon\) (in fact a DFA):

Notes

- DFA final states are those containing one of the final states of the NFA\(\varepsilon\)

- A second example is provided on the course web pages

- In \textit{that} example, there is a state \(\emptyset\), which is a dead state:

- once you enter a dead state you can never reach a final state

- any string that takes \(q_0\) to a dead state in a DFA can never be accepted
getting an NFA from a regular expression-II

Finite languages, which are not singletons, are also easy: abc + cbb

Kleene closure - I

- The Kleene closure is often simple: consider $r_1a^*r_2$, where $r_1$ and $r_2$ are regular expressions

Kleene closure - II

- Next consider $r_1(abc)^*r_2$, where $r_1$ and $r_2$ are regular expressions

Kleene closure - III

- The following is an NFA for $(1+ab+100)^*$
  - It is not unique
  - The next slide has an alternative
The following slides have a full-DFA and a DFA with the dead state removed.

Kleene closure - IV

- \((1+ab+100)^*\)

Equivalent full-DFA

- This second DFA should demonstrate that the graphic is clearer without the dead states showing.

A simple sum

- The NFAε for \(abc + bac\) could be either of the following, among others.
A simple concatenation

- The NFAε for $abc^* cab$ could be the following, among others

\[a \rightarrow b \rightarrow c\]
\[b \leftarrow a\]

Equivalent DFA

- The following is the DFA derived from the previous NFAε

\[a \rightarrow b \rightarrow c\]
\[b \leftarrow a\]

The standard construction

- We want to show that any regular expression has an NFAε
- We do this by always creating NFAε’s that have a SINGLE FINAL STATE that is different from $q_0$
- We show how to combine them to build larger regular expressions
- The basic regular expressions are $\varnothing$, $\varepsilon$, and $a$ and the simplest NFAε’s with a single initial and final state are:

NFAε’s for the basic regular expressions

\[
\begin{align*}
\varnothing & \quad \varepsilon & \quad a
\end{align*}
\]

NOTE: the following is simpler for $\varepsilon$ but does not fit the separate initial/final states property:
The NFAε for a sum of regular expressions

- If we have constructed the NFAε’s (each with a single final state) for two regular expressions $r_1$ and $r_2$ then the NFAε for $r_1 + r_2$ is shown next.
- The two final states from $r_1$ and $r_2$ are no longer final. A new single final state is added with ε-transitions from the old final state.

The NFAε for a sum of regular expressions - II

- What were final states in the graphs of $r_1$ and $r_2$ are no longer final states (they are shown with dotted lines around them).
- There is only one final state.

The NFAε for a concatenation of reg. expr.

- If we have constructed the NFAε’s (each with a single final state) for two regular expressions $r_1$ and $r_2$ then the NFAε for $r_1 r_2$ is shown next.
The NFAε for the Kleene closure of a regular expression.

- If we have constructed the NFAε (each with a single final state) for the regular expression $r$ then the NFAε for $r^*$ is shown next.