Nondeterminism and Kleene’s Theorem

We’ll want to show that every regular language is accepted by some FA, and that every FA accepts a regular language.

We already have the tools we need to show that every FA accepts a regular language, although the proof is not so easy. Better to look at it after we’ve had more experience with FA’s. (The proof implies an algorithm for constructing regular expressions from FA’s. We’ll also look at a nicer algorithm.)

On the other hand, we already know that every regular language is characterized by some regular expression (and that every regular expression characterizes a regular language). So, in the other direction, we will, essentially, construct an FA from a given regular expression. But it is not convenient to do this directly.

Instead, we’ll introduce nondeterministic finite automata (NFA’s), and show that NFA’s are equivalent to FA’s.

Then we’ll introduce an extension of NFA’s, called (awkwardly) nondeterministic FA’s with δ-transitions (NFA-δ‘s), and show that they are equivalent to NFA’s.

Then we’ll show, essentially, how to construct an NFA-δ from a given regular expression.

NFA’s

A nondeterministic finite automaton (NFA) is a 5-tuple

\[(Q, \Sigma, q_0, A, \delta)\]

where

1. \(Q\) is a finite set
2. \(\Sigma\) is a finite set of symbols
3. \(q_0 \in Q\)
4. \(A \subseteq Q\)
5. \(\delta: Q \times \Sigma \rightarrow 2^Q\)

So the definition of an NFA is the same as for an FA, except for the transition function \(\delta\). Now \(\delta\) takes a state and a symbol to a set of states (instead of a single state).

Does our diagram for \((0 + 1)^*1(0 + 1)^2\) fit this definition?

Example

Recall the family of \(L_n\) languages: \((0 + 1)^*1(0 + 1)^n\).

We used Theorem 3.2 to show that any FA for \(L_n\) must have at least \(2^n+1\) states.

What about the following “nondeterministic” machine for \((0 + 1)^*1(0 + 1)^{n^2}\)?

Is it an FA?

But it can be understood to accept \((0 + 1)^*1(0 + 1)^{n^2}\), and does it using only 4 states. (And in general, it appears that for each language \(L_n\) there should be such a machine with only \(n + 2\) states!)

As with FA’s, we want to define a multi-step transition function.

But in this case it will be a little harder, since reading a string in an NFA may take us to a number of different states (i.e. a set of states).

Recall that the definition of \(\delta\) for FA’s is recursive. The base case says

\[\delta^*(q, \varepsilon) = q\]

and the recursive equation says what to do for (nonempty) strings \(xs\), in terms of what to do on \(x\) and what to do on \(a\).

We’ll do the same sort of thing for NFA’s.

Example

Consider this problem wrt our diagram for \((0 + 1)^*1(0 + 1)^2\) and the string 011.

That is, what should we have for \(\delta^*(q_0, 011)\)?
Definition of $\delta'$ for NFA's

Given an NFA

$$M = (Q, \Sigma, q_0, A, \delta),$$

we define the multi-step transition function

$$\delta' : Q \times \Sigma^* \rightarrow 2^Q$$

as follows:

1. For all $q \in Q$, $\delta'(q, \Lambda) = \{q\}$.
2. For all $q \in Q$, $x \in \Sigma^*$ and $a \in \Sigma$,

$$\delta'(q, xa) = \bigcup_{r \in \delta'(q, x)} \delta(r, a).$$

Some properties you might want to verify:

$$\delta'(q, a) = \delta(q, a)$$
$$\delta'(q, xy) = \bigcup_{r \in \delta'(q, x)} \delta'(r, y)$$

If $\delta'(q, x) = \emptyset$, then $\delta'(q, xa) = \emptyset$.

Acceptance for NFA's

Roughly:

An NFA $M$ accepts $x$ if there is a sequence of moves $M$ can make on $x$ that ends in an accepting state.

Precisely:

**Definition** Given an NFA $M = (Q, \Sigma, q_0, A, \delta)$ and a string $x \in \Sigma^*$, $M$ accepts $x$ if

$$\delta'(q_0, x) \cap A \neq \emptyset.$$

As before, $L(M)$ denotes the set of strings accepted by $M$. For any $L \subseteq \Sigma^*$, we say $M$ recognizes $L$ if $L = L(M)$.

Reducing NFA's to FA's

**Example** Any NFA can be reduced to an FA that accepts the same language. Before we look at the general ("subset") construction, let's try an example. Consider again the NFA for

$$(0+1)^*1(0+1)^2.$$
NFA $M = (Q, \Sigma, q_0, A, \delta)$. FA $M_1 = (2^Q, \Sigma, \{q_0\}, A_1, \delta_1)$, where $A_1 = \{q \in 2^Q \mid q \cap A \neq \emptyset\}$ and for every $q \in 2^Q$ and $a \in \Sigma$,

$$\delta_1(q, a) = \bigcup_{p \in q} \delta(p, a).$$

According to def of multi-step transition function for FA's:

1. For any $q \in 2^Q$, $\delta_1(q, \lambda) = q$.
2. For any $q \in 2^Q$, $y \in \Sigma^*$ and $a \in \Sigma$, $\delta_1(q, ye) = \delta_1(\delta_1(q, y), a)$.

NFA's def of $\delta'$:

1. For all $q \in Q$, $\delta'(q, \lambda) = \emptyset$.
2. For all $q \in Q, x \in \Sigma^*$ and $a \in \Sigma$,

$$\delta'(q, xa) = \bigcup_{r \in \delta(q, x)} \delta(r, a).$$

**Case 1:**

To show: $\delta_1(\{q_0\}, \lambda) = \delta'(q_0, \lambda)$.

$$\delta_1(\{q_0\}, \lambda) = \{q_0\} \quad \text{(def \(\delta_1\) for FA)}$$

$$\delta'(q_0, \lambda) = \delta'(q_0, \lambda) \quad \text{(def \(\delta'\) for NFA)}$$

**Case 2: $x \in \Sigma^*$ and $a \in \Sigma$.**

II: $\delta_1(\{q_0\}, x) = \delta'(q_0, x)$.

To show: $\delta_1(\{q_0\}, xa) = \delta'(q_0, xa)$.

$$\delta_1(\{q_0\}, xa) = \delta_1(\delta_1(\{q_0\}, x), a) \quad \text{(def \(\delta_1\) for FA)}$$

$$= \delta_1(\delta'(q_0, x), a) \quad \text{[II]}$$

$$= \bigcup_{r \in \delta'(q_0, x)} \delta(r, a) \quad \text{(def \(\delta_1\))}$$

$$= \delta'(q_0, xa) \quad \text{(def \(\delta'\) for NFA)}$$

So we can conclude that for all $x \in \Sigma^*$, $\delta_1(\{q_0\}, x) = \delta'(q_0, x)$.

---

**Reducing an NFA for $1^* + 1(0 + 1)^*0$ to an FA**

---

**Concatenation and Kleene star**

Given a regular expression, it is much easier to produce a corresponding NFA than it was to produce a corresponding FA.

And we can convert an NFA to a corresponding FA.

So given an arbitrary regular expression, we may at this point feel fairly confident that we could produce a corresponding FA.

But how to prove that?

We need an algorithm, first of all.

Second, since the definition of regular expressions and regular languages is recursive, it would be simplest to have a construction based on the cases of the recursive definition. Then we could prove by structural induction that every regular language is accepted by some FA.

How might the proof go?
Recall: Every NFA can be reduced to an FA. So we might try to show, by structural induction on the recursive definition of the set of regular languages, that every regular language is accepted by some NFA.

Case 1: \( L = \emptyset \). NFA for \( \emptyset \)? Yes.

Case 2: \( L = \{ a \} \). NFA for \( \{ a \} \)? Yes.

Case 3: \( L = \{ a \} \) for some \( a \in \Sigma \). NFA for \( \{ a \} \)? Yes.

Case 4a: \( L = L_1 \cup L_2 \) for some regular languages \( L_1, L_2 \). NFA for \( L_1 \cup L_2 \)? Well, here the IH "gives us the existence of NFA's for \( L_1 \) and \( L_2 \), and we have constructions that we know can take these two NFA's and produce two FA's and from them one FA for \( L_1 \cup L_2 \).

Case 4b: \( L = L_1 \cdot L_2 \) for some regular languages \( L_1, L_2 \). NFA for \( L_1 \cdot L_2 \)? I believe I could do this, but I need to prove that I can. (In fact, we'll use NFA-L's instead.)

Case 4c: \( L = L_1^* \) for some regular language \( L_1 \). NFA for \( L_1^* \)? Again, I think I could probably do this, but I need to prove that I can. (In fact, we'll use NFA-L's instead.)

---

**Kleene star made easy (NFA-L)**

Consider \( (0[01]^* + [11]^*)^* \):

---

**How to capture \( \Lambda \)-transitions (mathematically)**

**Definition** A nondeterministic finite automaton with \( \Lambda \)-transitions (NFA-\( \Lambda \)) is a 5-tuple \( (Q, \Sigma, q_0, A, \delta) \) where

- \( Q \) is a finite set
- \( \Sigma \) is a finite set of symbols
- \( q_0 \in Q \)
- \( A \subseteq Q \)
- \( \delta : Q \times (\Sigma \cup \{ A \}) \to 2^Q \).

Once again, we will want to define a multi-step transition function. But consider the kind of difficulty we face:

For NFA's we had the base case \( \delta^*(q, \Lambda) = \{ q \} \).

For NFA-L's it is clear that we still have \( q \in \delta^*(q, \Lambda) \), but it is also clear that there may be other states "reachable" from \( q \) on the empty string — states reachable from \( q \) by one or more \( \Lambda \)-transitions.
\( \Lambda \)-closure

So we want to define the set of states reachable from a given state by zero or more \( \Lambda \)-transitions.

This set of states is most naturally defined recursively...

Since in general we will want to know what states are reachable by \( \Lambda \)-transitions from any member of a set of states, we define the following:

**Definition** Given an NFA-LA \((Q, \Sigma, q_0, A, \delta)\), and a subset \(S\) of \(Q\), the \( \Lambda \)-closure of \(S\), written \( \Lambda(S) \), is the smallest set satisfying the following two conditions:

1. \( S \subseteq \Lambda(S) \)
2. For any \( q \in \Lambda(S) \), every element of \( \delta(q, \Lambda) \) belongs to \( \Lambda(S) \).

Using this definition, the base case of the definition of \( \delta^* \) is now easily expressed:

For any \( q \in Q \),

\[
\delta^*(q, \Lambda) = \Lambda(\{q\}).
\]

\( \delta^* \) for NFA-LA's

In the recursive case, if we assume that \( \delta^*(q, x) \) is the set of all states "reachable" by reading \( x \) starting in \( q \) (using the symbols of \( x \) along with \( \Lambda \)-transitions), then

\[
\delta^*(q, x) = \delta(q, x) \quad \text{for all } x \in \Sigma
\]

is the set of all states reachable in one step more by reading the additional symbol \( a \), and

\[
\Lambda = \bigcup_{q \in \Lambda(q)} \delta(q, a)
\]

includes in addition the states that can then be reached by a finite sequence of \( \Lambda \)-transitions.

**Definition** Given an NFA-LA \((Q, \Sigma, q_0, A, \delta)\), we define

\[
\delta^* : Q \times \Sigma^* \rightarrow 2^Q
\]

as follows:

1. For any \( q \in Q \), \( \delta^*(q, \Lambda) = \Lambda(\{q\}) \).
2. For any \( q \in Q \), \( x \in \Sigma^* \) and \( a \in \Sigma \),

\[
\delta^*(q, ax) = \Lambda \left( \bigcup_{r \in \delta^*(q, a)} \delta(r, a) \right)
\]

Some properties of \( \delta^* \) for NFA-LA’s?

You might wish to verify the following:

\( \Lambda(\emptyset) = \emptyset, \Lambda(Q) = Q \).

For all \( S \subseteq Q \), \( \Lambda(S) = \Lambda(\Lambda(S)) \).

For all \( S \subseteq T \subseteq Q \), if \( S \subseteq T \), then \( \Lambda(S) \subseteq \Lambda(T) \).

For any subset \( S \) of \( 2^Q \),

\[
\bigcup_{s \in S} \Lambda(s) = \Lambda \left( \bigcup_{s \in S} s \right)
\]

No, not always. (Can you find a counterexample?) But we do have

\[
\delta(q, a) \subseteq \delta^*(q, a).
\]

To see this, notice first that

\[
\delta(q, a) = \delta(q, A_a) = \Lambda \left( \bigcup_{r \in \Lambda(q)} \delta(r, a) \right) = \Lambda \left( \bigcup_{r \in \Lambda(q)} \delta(r, a) \right).
\]

Then notice that

\[
\delta(q, a) \subseteq \Lambda \left( \bigcup_{r \in \Lambda(q)} \delta(r, a) \right)
\]

since \( q \in \Lambda(\{q\}) \) and \( \delta(q, a) \subseteq \Lambda(\delta(q, a)) \).

\[
\delta^*(q, xy) \subseteq \bigcup_{a \in \Sigma} \delta^*(q, y)
\]

Yes, but a little tricky to show. (We’ll use this fact soon.)

If \( \delta^*(q, x) = \emptyset \), then \( \delta^*(q, xy) = \emptyset \). Yes.
NFA-Λ language acceptance and reducibility to NFA’s

Definition For any NFA-Λ M = (Q, Σ, q₀, A, δ), a string x is accepted by M if δ∗(q₀, x) ∩ A ≠ ∅. The language recognized by M is the set L(M) of strings over Σ accepted by M.

Theorem 4.2 For any NFA-Λ M = (Q, Σ, q₀, A, δ), there is an NFA M₁ = (Q, Σ, q₀, A₁, δ₁) s.t. L(M₁) = L(M).

Proof. Define δ₁ in terms of δ∗: for all q ∈ Q and a ∈ Σ,

δ₁(q, a) = δ∗(q, a).

Notice that, for any q ∈ Q, by definition of the multi-step transition function for NFA’s,

δ[q, A] = {q},

whereas for NFA-Λ’s

δ(q, A) = Λ{q},

by the definition of the multi-step transition function for NFA-Λ’s.

Accordingly, take A₁ =

\[ A , \text{ if } Λ\{q₀\} ∩ A ≠ ∅ \]
\[ A ∪ \{q₀\} , \text{ otherwise.} \]

Main lemma: for any x ∈ Σ∗ s.t. x ≠ Λ, we will show that

δ₁(q, x) = δ(q, x).

So let’s prove this...

\[ \text{Case 1: } Λ\{q₀\} ∩ A ≠ ∅. \]

In this case, A₁ = A. Also, neither M nor M₁ accepts the empty string. For any nonempty string x,

M₁ accepts x ≡ δ₁[q₀, x] ∩ A₁ ≠ ∅ \hspace{1cm} (\text{def acceptance for NFA M₁})
\[ \text{iff } δ(q₀, x) ∩ A ≠ ∅ \hspace{1cm} (A₁ = A, δ₁[q₀, x] = δ(q₀, x)) \]
\[ \text{iff } M \text{ accepts } x \hspace{1cm} (\text{def acceptance for NFA-Λ M}) \]

\[ \text{Case 2: } Λ\{q₀\} ∩ A ≠ ∅. \]

In this case, A₁ = A ∪ {q₀}, and both M and M₁ accept the empty string. Now consider any x ∈ Σ∗ s.t. x ≠ Λ. Two subcases:

Subcase 1: q₀ ∈ δ∗(q₀, x).

Since {q₀} ⊆ δ∗(q₀, x), we know that Λ\{q₀\} ⊆ Λδ∗(q₀, x). (Since for all S, T ⊆ Q, if S ⊆ T, then Λ(S) ⊆ Λ(T).) Hence, Λδ∗(q₀, x) ∩ A ≠ ∅. We also know by the definition of δ∗ that Λδ(q₀, x) = Λδ∗(q₀, x). (Since Λ(S) = Λ(Λ(S)), for all S ⊆ Q.) Hence, δ(q₀, x) ∩ A ≠ ∅. That is, M accepts x. And since δ∗(q₀, x) = δ(q₀, x), δ(q₀, x) ∩ A ≠ ∅ also. Hence δ₁(q₀, x) ∩ A₁ ≠ ∅, and M₁ accepts x. So M and M₁ both accept x.

Subcase 2: q₀ ∉ δ∗(q₀, x).

In this case,

M₁ accepts x ≡ δ₁[q₀, x] ∩ A₁ ≠ ∅ \hspace{1cm} (\text{def acceptance for NFA M₁})
\[ \text{iff } δ(q₀, x) ∩ A₁ ≠ ∅ \hspace{1cm} (δ₁[q₀, x] = δ(q₀, x)) \]
\[ \text{iff } M \text{ accepts } x \hspace{1cm} (\text{def acceptance for NFA-Λ M}) \]

Equivalence of FA’s, NFA’s and NFA-Λ’s

Theorem 4.3 For any alphabet Σ and any language L ⊆ Σ∗, the following are equivalent:

• There is an FA that recognizes L.
• There is an NFA that recognizes L.
• There is an NFA-Λ that recognizes L.

Proof. Theorems 4.2 and 4.1 show that the third statement implies the second and that the second implies the first. So it is enough to show that the first implies the third.

Assume that FA M = (Q, Σ, q₀, A, δ), where δ : Q × Σ → 2Q is defined as follows:

For any q ∈ Q, δ(q, a) = ∅.

For any q ∈ Q, a ∈ Σ, δ(q, a) = {δ(q, a)}.

As you would expect, we want to show that for any q ∈ Q, x ∈ Σ∗,

δ(q, x) = \{δ*(q, x)\}.
To begin, recall that, for any $S \subseteq Q$, $\Lambda(S)$ (for NFA-A $M_1$) is the smallest set satisfying the following two conditions:

1. $S \subseteq \Lambda(S)$
2. For any $q \in \Lambda(S)$, every element of $\delta(q, A)$ belongs to $\Lambda(S)$.

Clearly, since there are no $\Lambda$-transitions in $M_1$, we can conclude that in this case, for any $S \subseteq Q$, $\Lambda(S) = S$.

This will allow us to simplify the expressions for $\delta_f$. Recall that, by definition, we have the following:

1. For any $q \in Q$, $\delta_f[q, A] = \Lambda(\{q\})$.
2. For any $q \in Q$, $x \in \Sigma^*$ and $a \in \Sigma$, 

\[ \delta_f[q, xa] = \Lambda \left( \bigcup_{r \in \delta_f[q, a]} \delta_f[r, a] \right). \]

Since $\Lambda(S) = S$, for any $S \subseteq Q$, we can instead write:

1. For any $q \in Q$, $\delta_f[q, A] = \Lambda(\{q\})$.
2. For any $q \in Q$, $x \in \Sigma^*$ and $a \in \Sigma$, 

\[ \delta_f[q, xa] = \bigcup_{r \in \delta_f[q, a]} \delta_f[r, a]. \]

Now we are ready to complete the proof of the claim that, for all $q \in Q$, $x \in \Sigma^*$, $\delta_f(q, x) = \delta^*(q, x)$. Proof is by structural induction on $x$ (using the recursive definition of $\Sigma^*$).

Case 1: $x = \Lambda$

\[ \delta_f[q, \Lambda] = \{q\} \] (By 1’ above)
\[ \delta^*(q, \Lambda) = \{\delta^*(q, A)\} \] (def $\delta^*$ for FA)

Case 2: $x \in \Sigma$ and $a \in \Sigma$

III: $\delta_f[q, x] = \delta^*(q, x)$

To show: $\delta_f[q, xa] = \delta^*(q, xa)$

[i] $\delta_f[q, xa] = \bigcup_{r \in \delta_f[q, a]} \delta_f[r, a]$ (By 2’ above)

[ii] $\delta_f[q, xa] = \delta_f[\delta^*(q, a), a]$ (def of $\delta$ in terms of $\delta$)

[iii] $\delta^*(q, xa) = \delta^*(q, xa)$ (def of $\delta^*$ for FA)

Now to complete the proof of the theorem, notice that $M$ accepts $x$ iff $\delta^*(q, x) \in A$ (def acceptance for FA's)

iff $\delta^*(q, x) \in \Lambda \neq \emptyset$

iff $\delta_f[q, x] \in \Lambda \neq \emptyset$ (def of $\delta_f$ in terms of $\delta$)

iff $M_1$ accepts $x$ (def acceptance for NFA-A's)

Case 4b: $L = L_L L_L$ for some regular languages $L_L, L_R$.

III: There are NFA-A's $L_L$ and $L_R$.

To show: There is an NFA-A that accepts $L_L L_R$.

We will soon have a construction that can take the two NFA-A's for $L_L$ and $L_R$ and produce one for $L_L L_R$.

Case 4c: $L = L_L^*$ for some regular language $L_L$.

III: There is an NFA-A accepting $L_L$.

To show: There is an NFA-A that accepts $L_L^*$.

We will soon have a construction that can take the NFA-A for $L_L$ and produce one for $L_L^*$.

This sketch of the proof lacks only the three lemmas that show how to construct NFA-A's for union, concatenation, and Kleene star.

So we complete our discussion of the proof of the first part of Kleene's Theorem (every regular language is accepted by an FA) by considering the three crucial lemmas...
**Lemma: NFA-Ł for union**

Consider any two languages $L_1, L_2 \subseteq \Sigma^*$ recognized by NFA-Ł's

\[ M_1 = (Q_1, \Sigma, q_0, A_1, \delta_1) \]

and

\[ M_2 = (Q_2, \Sigma, q_0, A_2, \delta_2) \]

respectively. WLOG, assume that $Q_1 \cap Q_2 = \emptyset$.

**Union Lemma.** The language $L_1 \cup L_2$ is recognized by the NFA-Ł

\[ M_u = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, q_0, A_1 \cup A_2, \delta_u) \]

where $q_0 \notin Q_1 \cup Q_2$ and $\delta_u$ is defined as follows:

1. $\delta_u(q_0, A) = \{q_0, q_0\}$
2. For all $q \in \Sigma$, $\delta_u(q, a) = \emptyset$
3. For all $q \in Q_1 \cup Q_2$ and $a \in \Sigma \cup \{A\}$,

\[ \delta_u(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \\ \delta_2(q, a), & \text{if } q \in Q_2 \end{cases} \]

**Proof sketch.**

One approach would be to show that, for all $x \in \Sigma^*$,

\[ \delta_u^*(q_0, x) \cap Q_1 = \delta_1^*(q_0, x) \]

and

\[ \delta_u^*(q_0, x) \cap Q_2 = \delta_2^*(q_0, x) . \]

Since $A_1 \subseteq Q_1$, it follows that

\[ \delta_u^*(q_0, x) \cap A_1 \neq \emptyset \iff \delta_1^*(q_0, x) \cap A_1 \neq \emptyset . \]

Similarly, since $A_2 \subseteq Q_2$,

\[ \delta_u^*(q_0, x) \cap A_2 \neq \emptyset \iff \delta_2^*(q_0, x) \cap A_2 \neq \emptyset . \]

We can conclude that $M_u$ accepts $x$ if and only if $M_1$ or $M_2$ does.

---

**Lemma: NFA-Ł for concatenation**

Consider any two languages $L_1, L_2 \subseteq \Sigma^*$ recognized by NFA-Ł's

\[ M_1 = (Q_1, \Sigma, q_0, A_1, \delta_1) \]

and

\[ M_2 = (Q_2, \Sigma, q_0, A_2, \delta_2) \]

respectively. WLOG, assume that $Q_1 \cap Q_2 = \emptyset$.

**Concatenation Lemma.** The language $L_1L_2$ is recognized by the NFA-Ł

\[ M_c = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, q_0, A_1A_2, \delta_c) \]

where $\delta_c$ is defined as follows:

1. For all $q \in A_1$, $\delta_c(q, A) = \{q_0\}$
2. For all $q \in A_1$ and $a \in \Sigma$, $\delta_c(q, a) = \delta_1(q, a)$.
3. For all $q \in (Q_1 \setminus A_1) \cup Q_2$ and $a \in \Sigma \cup \{A\}$,

\[ \delta_c(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \\ \delta_2(q, a), & \text{if } q \in Q_2 \end{cases} \]

**Proof sketch.**

Here you might verify first that for all $x \in \Sigma^*$,

\[ \delta_c^*(q_0, x) \cap Q_1 = \delta_1^*(q_0, x) . \]

It follows that if $\delta_1^*(q_0, x) \cap A_1 \neq \emptyset$, then $q_2 \in \delta_2^*(q_0, x)$.

Similarly, one can show that, for all $y \in \Sigma^*$,

\[ \delta_c^*(q_0, y) - \delta_2^*(q_0, y) . \]

From these observations, it follows in straightforward fashion that if $x \in L_1$ and $y \in L_2$, then $xy \in L(M_c)$. So $L_1L_2 \subseteq L(M_c)$.

For the other direction, it appears one must verify that, for all $w \in \Sigma^*$, if $\delta_c^*(q_0, w) \cap A_2 \neq \emptyset$, then there are $x, y \in \Sigma^*$ s.t.

- $w = xy$
- $\delta_c^*(q_0, x) \cap A_1 \neq \emptyset$
- $\delta_c^*(q_0, y) \cap A_2 \neq \emptyset$

Then, in light of the previous observations, it follows in straightforward fashion that $L(M_c) \subseteq L_1L_2$. 

---

(31)
Lemma: NEA-Λ for Kleene star

Consider any language $L \subseteq \Sigma^*$ recognized by NEA-Λ

$$M = \langle Q, \Sigma, q_0, A, \delta \rangle$$

**Kleene Star Lemma.** The language $L^*$ is recognized by the NEA-Λ

$$M_k = \langle Q \cup \{ q_k \}, \Sigma, q_0, \{ q_k \}, \delta_k \rangle$$

where $q_k \notin Q$ and $\delta_k$ is defined as follows:

1. $\delta((q_0, A) = \{ q_0 \}$
2. For all $a \in \Sigma$, $\delta((q_k, a) = \emptyset$
3. For all $q \in A$, $\delta(q, \Lambda) = \delta(q, A)$
4. For all $q \in A$ and $a \in \Sigma$, $\delta(q, a) = \delta(q, a)$.
5. For all $q \in Q - A$ and $a \in \Sigma$, $\delta(q, a) = \delta(q, a)$.

**Proof sketch.**

First show that for any $x \in \Sigma^*$, if $\delta^*(q_0, x) \cap A \neq \emptyset$, then $q_k \in \delta^*(q_0, x)$.

It is then straightforward to show by induction that, for every $n \in N$, $M_k$ accepts every string in $L^n$. Hence, $L^* \subseteq L(M_k)$.

The other direction seems harder...

---

Every FA recognizes a regular language?

Now let’s prove the second part of Kleene’s Theorem...

Consider an arbitrary FA $M = \langle Q, \Sigma, q_0, A, \delta \rangle$. For convenience later on, we will assume that $Q = \{ 1, 2, \ldots, k \}$ for some $k \in N$.

Notice:

$$L[M] = \{ x \in \Sigma^* | \delta(q_0, x) \in A \}$$

We can also write this as

$$L[M] = \bigcup_{r \in A} \{ x \in \Sigma^* | \delta(q_0, x) = r \}$$

This suggests the first idea of the proof:

Since every finite union of regular languages is a regular language, we can show that $L[M]$ is regular by proving that, for all $p, q \in Q$, the language

$$L[p, q] = \{ x \in \Sigma^* | \delta(q, x) = q \}$$

is regular.

To see how this helps, we need another idea.

One approach might be to begin by showing [1] that every nonempty string that belongs to $L(M_p)$ has a nonempty prefix that belongs to $L$. Then show [2] that for all $x, y \in \Sigma$, if $xy \in L(M_p)$ and $x \in L$ then $y \in L(M_k)$.

With these lemmas, you can show by strong induction on the length of $w \in \Sigma^*$ that if $w \in L(M_k)$, then $w \in L^*$.

**Case 1:** $w = \Lambda$.

Then $w \in L^*$. Trivial.

**Case 2:** $w \neq \Lambda$.

Assume $w \in L(M_k)$. Since $w$ is nonempty, we know by [1] that $w$ has a nonempty prefix $x$ that belongs to $L$. There is a $y \in \Sigma^*$ s.t. $w = xy$. Since $xy \in L(M_k)$ and $x \in L$, we know by [2] that $y \in L(M_k)$. Since $|y| < |w|$, we conclude by the IH that $y \in L^*$. Since $x \in L$ and $y \in L^*$, $xy \in L^*$.

---

For any $p, q, r \in Q$, and any string $w \in \Sigma^*$, we say $w$ goes from $p$ to $q$ through $r$ if there are nonempty strings $x, y \in \Sigma^*$ s.t.

- $w = xy$
- $\delta^*(p, x) = r$
- $\delta^*(r, y) = q$

Recall that $Q = \{ 1, 2, \ldots, k \}$ for some $k \in N$.

For any $n \in N$, let $L[p, q, n]$ be the language

$\{ w \in L[p, q] | \text{there is no } r \in Q \text{ s.t. } r > n \text{ and } w \text{ goes from } p \text{ to } q \text{ through } r \}$.

So

$$L[p, q] = \bigcup_{n=0}^{k} L[p, q, n]$$

(In fact, $L[p, q] = L[p, q, k]$.)

It follows that we can show that $L[p, q]$ is a regular language, by showing that, for all $n \in N$, $L[p, q, n]$ is a regular language.

This sounds like a candidate for inductive proof...
We'll show that for all \( n \in \mathbb{N} \), for all \( p, q \in Q \), \( L(p, q, n) \) is a regular language.

**Proof.** By strong induction on \( n \).

**IH:** For all \( m < n \), for all \( p, q \in Q \), \( L(p, q, m) \) is a regular language.

To show: For all \( p, q \in Q \), \( L(p, q, n) \) is a regular language.

**Case 1:** \( n = 0 \).

Every string in \( L(p, q, 0) \) has length less than 2, since otherwise the string would go from \( p \) to \( q \) through some state (and every state exceeds 0). So \( L(p, q, 0) \) is finite, and thus regular.

**Case 2:** \( n > 0 \).

Claim:

\[
L(p, q, n) = L(p, q, n - 1) \cup L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1)
\]

Notice that if this claim holds, it follows by the IH along with the recursive definition of the regular languages over \( \Sigma \) that \( L(p, q, n) \) is a regular language.

What we will show is:

\[
L(p, q, n) - L(p, q, n - 1) \subseteq L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1)
\]

and

\[
L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1) \subseteq L(p, q, n) .
\]

To see that \( y \in L(n, n, n - 1)^* \), consider two cases.

**Case 1:** \( y \in L(n, n, n - 1) \). Then \( y \in L(n, n, n - 1)^* \).

**Case 2:** \( y \notin L(n, n, n - 1) \). Since \( y \in L(n, n, n) - L(n, n, n - 1) \), we can conclude that \( y \) goes from \( n \) to \( n \) through \( n \). Consider the set \( S \) of all nonempty prefixes \( y' \) of \( y \) s.t. \( \delta(n, y') = n \). Let \( j = |S| \). Then \( y \) can be written in the form \( y_1y_2 \cdots y_j \) s.t. each \( y_i \) is nonempty and \( S = \{y_1 \cdots y_i \mid 1 \leq i \leq j\} \). One can show that each \( y_i \) belongs to \( L(n, n, n - 1) \). It follows from this observation that \( y \in L(n, n, n - 1)^* \).

That completes the proof that

\[
L(p, q, n) - L(p, q, n - 1) \subseteq L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1) .
\]

It remains to show that

\[
L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1) \subseteq L(p, q, n) .
\]

Assume that \( w \in L(p, n, n - 1)L(n, n, n - 1)^*L(n, q, n - 1) \). So \( w \) can be written in the form \( xyz \), where \( x \in L(p, n, n - 1) \), \( y \in L(n, n, n - 1)^* \), and \( z \in L(n, q, n - 1) \). From the fact that \( y \in L(n, n, n - 1)^* \), we can conclude also that \( y \in L(n, n, n) \). Hence, \( xyz \in L(p, n, n - 1)L(n, n, n) \). From this fact that \( y \not\in L(n, n, n) \), we can conclude also that \( y \in L(n, n, n) \). Hence, \( xyz \in L(p, n, n - 1)L(n, n, n) \). It follows that \( w = xyz \in L(p, q, n) \).