Note that has fewer than $2^3 = 8$ states.

N.B. Reading: 47–58.
Then, 58–63.
We want to show that the construction is correct. Assume that $N$ has no $\gamma$-moves. We prove by induction on the length of the input string $x$ that
\[ \sigma'((q'_0, x)) = \sigma(q_0, x). \]
For then, $\sigma'(q'_0, x) \in F'$.

\[ \iff \sigma'(q'_0, x) \text{ contains an accept state of } N \]
\[ \iff \sigma(q_0, x) \cap F \neq \emptyset. \]
Thus, $M$ accepts $w$ $\iff N$ accepts $w$ follows.

**basis** \hspace{1cm} \text{Let } \lambda = \varepsilon. \text{ Then, } x = \lambda.
\[ \sigma'(q'_0, \lambda) = q'_0 = \{ q_0 \} = \sigma(q_0, \lambda). \]

**induction step** Assume the hypothesis is true for all inputs of length $\leq m$. Let $xa$ be a string of length $m+1$ with $a$ in $\Sigma$. Then,
\[ \sigma'(q'_0, xa) = \sigma'(\sigma'(q'_0, x), a) \]
\[ = \sigma'\left( \sigma(q'_0, x), a \right) \]
\[ = \bigcup_{\gamma \in \sigma(q_0, x)} \sigma(\gamma, a) \]
\[ = \sigma(q_0, xa). \]
This completes the proof for the case that \( N \) has no \( n \)-moves.
Suppose $\lambda$ has $\lambda$-moves.
For each state $r \in Q$, let
$E(r) = \{ q \mid q \text{ is reachable from } r \text{ by traveling along zero or more } \lambda\text{-moves} \}$

Let $C = G$. Define
$E(p) = \bigcup_{r \in p} E(r)$.

accepts $0^*1^*2^*$
\[ G = \{ q_0, q_1, q_2 \} \]
\[ \Sigma = \{ 1, 2, 3 \} \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_0 )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( \emptyset )</td>
<td>( { q_1 } )</td>
<td>( \emptyset )</td>
<td>( { q_2 } )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( { q_2 } )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

\( q_0 \) is the start state

\[ F = \{ q_2 \} \]

\[ E(q_0) = \{ q_0, q_1, q_2 \} \]
\[ E(q_1) = \{ q_1, q_2 \} \]
\[ E(q_2) = \{ q_2 \} \]
Given NFA $N = (Q, \Sigma, \delta, q_0, F)$,
the equivalent DFA is $M = (Q', \Sigma, \delta', q_0', F')$, where

$Q' = P(Q)$

$q_0' = E(q_0)$

$F' = \{ q' \in Q' \mid \exists q \in F, q \subseteq q' \}$

$\delta'(R, a) = \{ q \in Q' \mid \exists q \in E(\delta(r, a)) \text{, for some } r \in R \}$

$= \bigcup \{ E(\delta(r, a)) \mid R \in Q', a \in \Sigma \}$

eg. continued.

$q_0' = E(q_0) = \{ q_0, q_1, q_2 \}$

$\delta'(\{ q_0, q_1, q_2 \}, 0) = E(\delta(q_0, 0)) \cup E(\delta(q_1, 0)) \cup E(\delta(q_2, 0))$

$= E(\{ q_0 \}) \cup E(\emptyset) \cup E(\emptyset)$

$= \{ q_0, q_0, q_0 \}$
\[ \sigma'([q_0, q_n q_2]) \]

\[ = E([q_0, q_1]) \cup E([q_1, q_2]) \cup \ldots \cup E([q_n, q_1]) \]

\[ = E(\varnothing) \cup E([q_1]) \cup E(\varnothing) \]

\[ = \varnothing \cup [q_1, q_2] \cup \varnothing \]

\[ = [q_1, q_2] \]
We define the equivalent DFA as before, except

2'. $\delta'(R, a) = \{ q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R \} \setminus \cup_{r \in R} E(\delta(r, a)), \text{ for } R \subseteq Q, a \in \Sigma.$

3'. $q' = E(\lambda, q_0).$
Important: Every NFA is equivalent to one with exactly one final state.
Another way to see that regular languages are closed under union:

Let \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \)

and \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \)

be NFAs.

Construct \( N \) by taking a new state \( q_0 \) to be the initial state.

Draw \( \lambda \)-moves from \( q_0 \) to \( q_1 \) and to \( q_2 \).
Theorem: Regular languages are closed under concatenation.

Let $N_1$ and $N_2$ be NFAs.

For each final state of $N_1$, allow for the possibility of staying in $N_1$ or for entering the start state of $N_2$. We construct an NFA $M$ so that $L(M) = L(N_1) L(N_2)$.

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

\[ Q = Q_1 \cup Q_2 \]

\[ \delta(q, a) = \delta_1(q, a), \quad q \in Q_1, \quad q \notin F \]

\[ \delta(q, a) = \delta_1(q, a), \quad q \in F_1, \quad a \notin \lambda \]

\[ \delta(q, \lambda) = \delta_1(q, \lambda) \cup \{ q_2 \}, \quad q \in F_1 \]

\[ \delta(q, a) = \delta_2(q, a), \quad q \in Q_2 \]

\[ N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \]

\[ N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \]
Theorem. The class of regular languages is closed under star.

Let $M_i$ be an NFA. We construct $N$ to accept $L(N)^*$. 
\[ N_i = (Q_i, \Sigma, \delta_i, q_{i0}, \{ t_i \}) \]

\[ N = (Q, \Sigma, \delta, q_0, \{ t_0 \}) \]

\[ Q = \{ q_0, t_0 \} \cup Q_i \), where \( q_0 \notin Q \), and \( t_0 \notin Q \)

\[ F = \{ t_0 \} \]

\[ \delta(q, a) = \delta_i(q, a), \quad \text{if } q \in Q_i, \quad q \neq t_i \]

\[ \delta(t_i, a) = \delta_i(t_i, a), \quad \text{if } a \notin \Sigma \]

\[ \sigma(t_i, \lambda) = \sigma_i(t_i, \lambda) \cup \{ q_i \} \cup \{ t_i \} \]

\[ \sigma(q_{i0}, \lambda) = \{ q_i \} \cup \{ t_i \} \]
Theorem: If a language $L$ is described by a regular expression, then $L$ is regular.

For each regular expression $r$, we construct an NFA $N$ s.t.
$L(r) = L(N)$.

**Basis Step.** If $r = a$, $a \in \Sigma$, then $L(r) = \{a\}$, NFA is:

```
\rightarrow q_0 \xrightarrow{a} q_1
```

If $r = \lambda$, $L(r) = \{\lambda\}$, NFA is:

```
\rightarrow q_0
```

If $r = \emptyset$, $L(r) = \emptyset$. NFA is:

```
\rightarrow q_0
```
**induction step.** Let \( r_1 \) and \( r_2 \) be regular expressions and let \( N_1 \) and \( N_2 \) be NFAs s.t.
\[
L(N_1) = r_1 \quad \text{and} \quad L(N_2) = r_2.
\]

Then, NFAs for
\[
L(r_1 \cup r_2) = L(r_1) \cup L(r_2)
\]
\[
L(r_1 r_2) = L(r_1) L(r_2)
\]
\[
\text{and} \quad L(r_1^*) = L(r_1)^*
\]
are given by the theorems.

**Example:** \( 01^* \cup 1 \)

\( r_1 = 01^* \) \quad \text{and} \quad r_2 = 1

\( N_2 = \quad 1 \rightarrow \quad 0 \)

\( r_3 = r_3 \quad \text{where} \quad r_3 = 0 \) \quad \text{and} \quad r_4 = 1^*.

\( N_3 = \quad 0 \rightarrow \quad 1 \)

\( r_4 = r_5^* , \quad r_5 = 1. \)

\( N_5 = \quad 1 \rightarrow \quad 0 \)

\[ \text{We need to name states in } N_5 \]
\[ \text{differ from } N_1. \]
\[ r = (01 \cup 1)^* \]

\[ r \cdot 0_1^* \cdot r = r_2 \cup r_3 \]

\[ r_2 = 0_1 \quad r_3 = 1 \]

\[ r = r_4 \cdot r_5 \quad r_4 = 0 \quad r_5 = 1 \]

Diagram:

- \( N_4 \) -> \( q_0 \) -> (\( q_1 \))
- \( N_5 \) -> (\( q_2 \))
- \( N_3 \) -> (\( q_3 \))
- \( N_2 \) -> (\( q_0 \)) -> (\( q_1 \)) -> (\( q_2 \)) -> (\( q_3 \))
- \( N \) -> (\( q_0 \)) -> (\( q_1 \)) -> (\( q_2 \)) -> (\( q_3 \))
At this point we know the following:

Now we will show how to construct a regular expression from any DFA.

Given DFA $M$,

1. Add a new start state $S$. Draw an arrow from $S$ to the original start state. Label with $\lambda$.

   N.B. No arrow goes into state $S$.

2. Add a final state $F$. Draw an arrow from every final state to $F$, label the arrow with $\lambda$; original final states are no longer final.

   N.B. There is exactly one final state $F$ and no arrow from $F$. 
Order the remaining states $\mathcal{Q} = \{1, \ldots, n\}$. We will successively perform bypass operations until all states of $\mathcal{Q}$ are removed. (A bypass operation removes a state in $\mathcal{Q}$).

Given

union operation

$\mathcal{Q}$

replace with

More generally:

from

(3) Order states $\mathcal{Q} = \{1, \ldots, n\}$

(4) Perform all union operations s.t. for all $i, j$ diagram has at most one arrow from $i$ to $j$. 

$\mathcal{Q}$
Bypass operation (on node $j$)

For each arc into 4 out of $i$

\[ j = 1 \]

For $j = 1$ to $n$ do

for every pair of states $i, k \in Q$,

if there is an arrow from $i$ to $j$

and there is an arrow from $j$ to $k$,

perform a bypass operation to remove state $j$.

Then, perform a union operation

When we are done, we have

Thus, every regular language is described
by a regular expression.