Equivalence of DFA and NFA

- NFA’s are usually easier to “program” in.

- Surprisingly, for any NFA $N$ there is a DFA $D$, such that $L(D) = L(N)$, and vice versa.

- This involves the subset construction, an important example how an automaton $B$ can be generically constructed from another automaton $A$.

- Given an NFA

  $$N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

  we will construct a DFA

  $$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

  such that

  $$L(D) = L(N)$$
The details of the subset construction:

• \( Q_D = \{ S : S \subseteq Q_N \} \).

Note: \( |Q_D| = 2^{|Q_N|} \), although most states in \( Q_D \) are likely to be garbage.

• \( F_D = \{ S \subseteq Q_N : S \cap F_N \neq \emptyset \} \)

• For every \( S \subseteq Q_N \) and \( a \in \Sigma \),

\[
\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)
\]
Let’s construct $\delta_D$ from the NFA on slide 27

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\rightarrow {q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_1}$</td>
<td>$\emptyset$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>$\star{q_2}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>$\star{q_0, q_2}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>$\star{q_1, q_2}$</td>
<td>$\emptyset$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>$\star{q_0, q_1, q_2}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
</tbody>
</table>
Note: The states of $D$ correspond to subsets of states of $N$, but we could have denoted the states of $D$ by, say, $A – F$ just as well.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$A$</td>
<td>$A$</td>
</tr>
<tr>
<td>$B$</td>
<td>$E$</td>
<td>$B$</td>
</tr>
<tr>
<td>$C$</td>
<td>$A$</td>
<td>$D$</td>
</tr>
<tr>
<td>$D$</td>
<td>$A$</td>
<td>$A$</td>
</tr>
<tr>
<td>$E$</td>
<td>$E$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$E$</td>
<td>$B$</td>
</tr>
<tr>
<td>$G$</td>
<td>$A$</td>
<td>$D$</td>
</tr>
<tr>
<td>$H$</td>
<td>$E$</td>
<td>$F$</td>
</tr>
</tbody>
</table>
We can often avoid the exponential blow-up by constructing the transition table for $D$ only for accessible states $S$ as follows:

**Basis:** $S = \{q_0\}$ is accessible in $D$

**Induction:** If state $S$ is accessible, so are the states in $\bigcup_{a \in \Sigma} \delta_D(S, a)$

Example: The “subset” DFA with accessible states only.
**Theorem 2.11:** Let $D$ be the “subset” DFA of an NFA $N$. Then $L(D) = L(N)$.

**Proof:** First we show by an induction on $|w|$ that

$$
\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)
$$

**Basis:** $w = \epsilon$. The claim follows from def.
Induction:

\[ \tilde{\delta}_D(\{q_0\}, xa) \overset{\text{def}}{=} \delta_D(\tilde{\delta}_D(\{q_0\}, x), a) \]

\[ \overset{\text{i.h.}}{=} \delta_D(\tilde{\delta}_N(q_0, x), a) \]

\[ \overset{\text{cst}}{=} \bigcup_{p \in \tilde{\delta}_N(q_0, x)} \delta_N(p, a) \]

\[ \overset{\text{def}}{=} \tilde{\delta}_N(q_0, xa) \]

Now (why?) it follows that \( L(D) = L(N) \).
**Theorem 2.12:** A language $L$ is accepted by some DFA if and only if $L$ is accepted by some NFA.

**Proof:** The “if” part is Theorem 2.11.

For the “only if” part we note that any DFA can be converted to an equivalent NFA by modifying the $\delta_D$ to $\delta_N$ by the rule

- If $\delta_D(q, a) = p$, then $\delta_N(q, a) = \{p\}$.

By induction on $|w|$ it will be shown in the tutorial that if $\tilde{\delta}_D(q_0, w) = p$, then $\tilde{\delta}_N(q_0, w) = \{p\}$.

The claim of the theorem follows.

**How do you convert an NFA to C/C++ code?**
There is an NFA $N$ with $n + 1$ states that has no equivalent DFA with fewer than $2^n$ states.

$L(N) = \{x_1c_2c_3 \cdots c_n : x \in \{0, 1\}^*, c_i \in \{0, 1\}\}$

Suppose an equivalent DFA $D$ with fewer than $2^n$ states exists.

$D$ must remember the last $n$ symbols it has read.

There are $2^n$ bitsequences $a_1a_2 \cdots a_n$

$$\exists q, a_1a_2 \cdots a_n, b_1b_2 \cdots b_n : q = \hat{\delta}_D(q_0, a_1a_2 \cdots a_n),$$

$$q = \hat{\delta}_D(q_0, b_1b_2 \cdots b_n),$$

$$a_1a_2 \cdots a_n \neq b_1b_2 \cdots b_n$$
Case 1:

\[ 1a_2 \cdots a_n \]
\[ 0b_2 \cdots b_n \]

Then \( q \) has to be both an accepting and a nonaccepting state.

Case 2:

\[ a_1 \cdots a_{i-1} 1a_{i+1} \cdots a_n \]
\[ b_1 \cdots b_{i-1} 0b_{i+1} \cdots b_n \]

Now \( \hat{\delta}_D(q_0, a_1 \cdots a_{i-1} 1a_{i+1} \cdots a_n 0^{i-1}) = \hat{\delta}_D(q_0, b_1 \cdots b_{i-1} 0b_{i+1} \cdots b_n 0^{i-1}) \)

and \( \hat{\delta}_D(q_0, a_1 \cdots a_{i-1} 1a_{i+1} \cdots a_n 0^{i-1}) \in F_D \)

\[ \hat{\delta}_D(q_0, b_1 \cdots b_{i-1} 0b_{i+1} \cdots b_n 0^{i-1}) \notin F_D \]
An $\epsilon$-NFA accepting decimal numbers consisting of:

1. An optional + or - sign
2. A string of digits
3. a decimal point
4. another string of digits

One of the strings (2) are (4) are optional
Example:

$\epsilon$-NFA accepting the set of keywords \{ebay, web\}

We can have an $\epsilon$-moves for each keyword.
An $\epsilon$-NFA is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where $\delta$ is a function from $Q \times (\Sigma \cup \{\epsilon\})$ to the powerset of $Q$.

Example: The $\epsilon$-NFA from the previous slide

$$E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\} \delta, q_0, \{q_5\})$$

where the transition table for $\delta$ is

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$\pm$, $-$</th>
<th>.</th>
<th>$0, \ldots, 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow q_0$</td>
<td>${q_1}$</td>
<td>${q_1}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_2}$</td>
<td>${q_1, q_4}$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_3}$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>${q_5}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_3}$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_3}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$*q_5$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
ECLOSE or $\epsilon$-closure

We close a state by adding all states reachable by a sequence $\epsilon \epsilon \cdots \epsilon$

Inductive definition of $\text{ECLOSE}(q)$

**Basis:**

$q \in \text{ECLOSE}(q)$

**Induction:**

$p \in \text{ECLOSE}(q)$ and $r \in \delta(p, \epsilon) \Rightarrow r \in \text{ECLOSE}(q)$
Example of $\epsilon$-closure

For instance,

$$\text{ECLOSE}(1) = \{1, 2, 3, 4, 6\}$$
• Inductive definition of $\hat{\delta}$ for $\epsilon$-NFA’s

**Basis:**

$$\hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$$

**Induction:**

$$\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q, x), a} \text{ECLOSE}(p)$$

where $\hat{\delta}(q, x, a) = \bigcup_{r \in \hat{\delta}(q, x)} \delta(r, a)$

Let’s compute on the blackboard in class $\hat{\delta}(q_0, 5.6)$ for the NFA on slide 43

$\hat{\delta}(q_0, \epsilon) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 5) = \text{ECLOSE}([q_1, q_4]) = \{q_1, q_4\}$, because $\hat{\delta}(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$

$\hat{\delta}(q_0, 5.) = \text{ECLOSE}([q_2, q_3]) = \{q_2, q_3, q_5\}$

$\hat{\delta}(q_0, 5.6) = \text{ECLOSE}([q_3]) = \{q_3, q_5\}$
Given an $\epsilon$-NFA

$$E = (Q_E, \Sigma, \delta_E, q_0, F_E)$$

we will construct a DFA

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

such that

$$L(D) = L(E)$$

Details of the construction:

- $Q_D = \{ S : S \subseteq Q_E \text{ and } S = \text{ECLOSE}(S) \}$
- $q_D = \text{ECLOSE}(q_0)$
- $F_D = \{ S : S \in Q_D \text{ and } S \cap F_E \neq \emptyset \}$
- $\delta_D(S, a) = \bigcup \{ \text{ECLOSE}(p) : p \in \delta_E(t, a) \text{ for some } t \in S \}$
Example: $\epsilon$-NFA $E$

DFA $D$ corresponding to $E$
**Theorem 2.22:** A language $L$ is accepted by some $\epsilon$-NFA $E$ if and only if $L$ is accepted by some DFA.

**Proof:** We use $D$ constructed as above and show by induction that $\tilde{\delta}_D(q_D, w) = \tilde{\delta}_E(q_0, w)$

**Basis:** $\tilde{\delta}_E(q_0, \epsilon) = \text{ECLOSE}(q_0) = q_D = \tilde{\delta}(q_D, \epsilon)$
**Induction:**

\[ \tilde{\delta}_E(q_0, xa) \overset{\text{DEF}}{=} \bigcup_{p \in \delta_E(\tilde{\delta}_E(q_0, x), a)} \text{ECLOSE}(p) \]

I.H.

\[ = \bigcup_{p \in \delta_E(\hat{\delta}_D(q_D, x), a)} \text{ECLOSE}(p) \]

CST

\[ = \hat{\delta}_D(\hat{\delta}_D(q_D, x), a) \]

DEF

\[ = \tilde{\delta}_D(q_D, xa) \]
An FA (NFA or DFA) is a “blueprint” for constructing a machine recognizing a regular language.

A regular expression is a “user-friendly,” declarative way of describing a regular language.

Example: $01^* + 10^*$

Regular expressions are used in e.g.

1. UNIX grep command

   \begin{verbatim}
   grep PATTERN FILE
   \end{verbatim}

2. UNIX Lex (Lexical analyzer generator) and Flex (Fast Lex) tools.

3. Text/email mining (e.g., for HomeUnion)
Operations on languages

Union:

\[ L \cup M = \{ w : w \in L \text{ or } w \in M \} \]

Concatenation:

\[ L.M = \{ w : w = xy, x \in L, y \in M \} \]

Powers:

\[ L^0 = \{ \epsilon \}, \ L^1 = L, \ L^{k+1} = L.L^k \]

Kleene Closure:

\[ L^* = \bigcup_{i=0}^{\infty} L^i \]

**Question:** What are \( \emptyset^0 \), \( \emptyset^i \), and \( \emptyset^* \)

**Question:** What is \( \{0^2,0^3\}^* \)?
Inductive definition of regex's:

**Basis:** $\epsilon$ is a regex and $\emptyset$ is a regex. 
$L(\epsilon) = \{\epsilon\}$, and $L(\emptyset) = \emptyset$.

If $a \in \Sigma$, then $a$ is a regex. 
$L(a) = \{a\}$.

**Induction:**

If $E$ is a regex's, then $(E)$ is a regex. 
$L((E)) = L(E)$.

If $E$ and $F$ are regex's, then $E + F$ is a regex. 
$L(E + F) = L(E) \cup L(F)$.

If $E$ and $F$ are regex's, then $E.F$ is a regex. 
$L(E.F) = L(E).L(F)$.

If $E$ is a regex's, then $E^*$ is a regex. 
$L(E^*) = (L(E))^*$.
Example: Regex for

\[ L = \{ w \in \{0, 1\}^* : 0 \text{ and } 1 \text{ alternate in } w \} \]

\[ (01)^* + (10)^* + 0(10)^* + 1(01)^* \]

or, equivalently,

\[ (\epsilon + 1)(01)^*(\epsilon + 0) \]

Order of precedence for operators:

1. Star
2. Dot
3. Plus

Example: \( 01^* + 1 \) is grouped \( (0(1^*)) + 1 \)

Ex. Regex's for \( L_1 = \{ w \mid w \in \{0, 1\}^*, w \text{ contains no consecutive 0's} \} \)
\[ L_2 = \{ w \mid w \in \{0, 1\}^*, \text{ the number of 0's in } w \text{ is even} \} \].
Equivalence of FA’s and regex’s

We have already shown that DFA’s, NFA’s, and $\epsilon$-NFA’s all are equivalent.

To show FA’s equivalent to regex’s we need to establish that

1. For every DFA $A$ we can find (construct, in this case) a regex $R$, s.t. $L(R) = L(A)$.

2. For every regex $R$ there is an $\epsilon$-NFA $A$, s.t. $L(A) = L(R)$.
**Theorem 3.4:** For every DFA $A = (Q, \Sigma, \delta, q_0, F)$ there is a regex $R$, s.t. $L(R) = L(A)$.

**Proof:** Let the states of $A$ be \{1, 2, ..., $n$\}, with 1 being the start state.

- Let $R_{ij}^{(k)}$ be a regex describing the set of labels of all paths in $A$ from state $i$ to state $j$ going through intermediate states \{1, ..., $k$\} only. 

\[ \text{Note that, i and j don't have to be in \{1, ...,k\}.} \]
$R_{ij}^{(k)}$ will be defined inductively. Note that

$$L \left( \bigoplus_{j \in F} R_{1j}^{(n)} \right) = L(A)$$

**Basis:** $k = 0$, i.e. no intermediate states.

- **Case 1:** $i \neq j$
  
  $R_{ij}^{(0)} = \bigoplus_{\{a \in \Sigma : \delta(i,a) = j\}} a$

- **Case 2:** $i = j$
  
  $R_{ii}^{(0)} = \left( \bigoplus_{\{a \in \Sigma : \delta(i,a) = i\}} a \right) + \epsilon$
Induction:

\[ R^{(k)}_{ij} = R^{(k-1)}_{ij} + R^{(k-1)}_{ik} \left( R^{(k-1)}_{kk} \right)^* R^{(k-1)}_{kj} \]

- At least one line of \( k \) is in \( R^{(k-1)}_{ik} \)
- At least one line of \( k \) is in \( R^{(k-1)}_{kk} \)
- More than or equal to one line of \( k \) is in \( R^{(k-1)}_{kj} \)
Example: Let’s find $R$ for $A$, where

$L(A) = \{x0y : x \in \{1\}^* \text{ and } y \in \{0, 1\}^*\}$

```
R(0)
11  \epsilon + 1
R_{12}  0
R_{21}  \emptyset
R_{22}  \epsilon + 0 + 1
```
We will need the following simplification rules:

- \((\epsilon + R)^* = R^*\)  \((\epsilon + R)R^* = R^*\)
- \(R + RS^* = RS^*\)  \(\epsilon + R + R^* = R^*\)
- \(\emptyset R = R\emptyset = \emptyset\) (Annihilation)
- \(\emptyset + R = R + \emptyset = R\) (Identity)
\[
R^{(0)}_{11} \quad \epsilon + 1 \\
R^{(0)}_{12} \quad 0 \\
R^{(0)}_{21} \quad \emptyset \\
R^{(0)}_{22} \quad \epsilon + 0 + 1 \\
\]

\[
R^{(1)}_{ij} = R^{(0)}_{ij} + R^{(0)}_{i1}(R^{(0)}_{11})^* R^{(0)}_{1j}
\]

<table>
<thead>
<tr>
<th></th>
<th>By direct substitution</th>
<th>Simplified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^{(1)}_{11}$</td>
<td>$\epsilon + 1 + (\epsilon + 1)(\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1)$</td>
<td>1*</td>
</tr>
<tr>
<td>$R^{(1)}_{12}$</td>
<td>$0 + (\epsilon + 1)(\epsilon + 1)^*0$</td>
<td>1*0</td>
</tr>
<tr>
<td>$R^{(1)}_{21}$</td>
<td>$\emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$R^{(1)}_{22}$</td>
<td>$\epsilon + 0 + 1 + \emptyset(\epsilon + 1)^*0$</td>
<td>$\epsilon + 0 + 1$</td>
</tr>
<tr>
<td></td>
<td>Simplified</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>$R_{11}^{(1)}$</td>
<td>$1^*$</td>
<td></td>
</tr>
<tr>
<td>$R_{12}^{(1)}$</td>
<td>$1^*0$</td>
<td></td>
</tr>
<tr>
<td>$R_{21}^{(1)}$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$R_{22}^{(1)}$</td>
<td>$\epsilon + 0 + 1$</td>
<td></td>
</tr>
</tbody>
</table>

$$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} (R_{22}^{(1)})^* R_{2j}^{(1)}$$

<table>
<thead>
<tr>
<th></th>
<th>By direct substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{11}^{(2)}$</td>
<td>$1^* + 1^<em>0(\epsilon + 0 + 1)^</em>\emptyset$</td>
</tr>
<tr>
<td>$R_{12}^{(2)}$</td>
<td>$1^*0 + 1^<em>0(\epsilon + 0 + 1)^</em>(\epsilon + 0 + 1)$</td>
</tr>
<tr>
<td>$R_{21}^{(2)}$</td>
<td>$\emptyset + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*\emptyset$</td>
</tr>
<tr>
<td>$R_{22}^{(2)}$</td>
<td>$\epsilon + 0 + 1 + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$</td>
</tr>
</tbody>
</table>
By direct substitution

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{11}^{(2)}$</td>
<td>$1^* + 1^<em>0(\epsilon + 0 + 1)^</em>\emptyset$</td>
</tr>
<tr>
<td>$R_{12}^{(2)}$</td>
<td>$1^*0 + 1^<em>0(\epsilon + 0 + 1)^</em>(\epsilon + 0 + 1)$</td>
</tr>
<tr>
<td>$R_{21}^{(2)}$</td>
<td>$\emptyset + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*\emptyset$</td>
</tr>
<tr>
<td>$R_{22}^{(2)}$</td>
<td>$\epsilon + 0 + 1 + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Simplified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{11}^{(2)}$</td>
<td>$1^*$</td>
</tr>
<tr>
<td>$R_{12}^{(2)}$</td>
<td>$1^<em>0(0 + 1)^</em>$</td>
</tr>
<tr>
<td>$R_{21}^{(2)}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$R_{22}^{(2)}$</td>
<td>$(0 + 1)^*$</td>
</tr>
</tbody>
</table>

The final regex for $A$ is

$$R_{12}^{(2)} = 1^*0(0 + 1)^*$$
Observations

There are $n^3$ expressions $R_{ij}^{(k)}$.

Each inductive step grows the expression 4-fold.

$R_{ij}^{(n)}$ could have size $4^n$.

For all $\{i, j\} \subseteq \{1, \ldots, n\}$, $R_{ij}^{(k)}$ uses $R_{kk}^{(k-1)}$, so we have to write $n^2$ times the regex $R_{kk}^{(k-1)}$.

but most of them can be removed by annihilation!

We need a more efficient approach: the state elimination technique.
The state elimination technique

Let’s label the edges with regex’s instead of symbols
Now, let’s eliminate state $s$.

For each accepting state $q$ eliminate from the original automaton all states except $q_0$ and $q$. 
For each $q \in F$ we’ll be left with an $A_q$ that looks like

![Diagram](image)

that corresponds to the regex $E_q = (R + SU^*T)^*SU^*$

or with $A_q$ looking like

![Diagram](image)

corresponding to the regex $E_q = R^*$

- The final expression is

$$\bigoplus_{q \in F} E_q$$
Note that the algorithm also works for NFAs and ε-NFAs.

Example: \( A \), where \( L(A) = \{ W : w = x1b, \text{ or } w = x1bc, \ x \in \{0, 1\}^*, \{b, c\} \subseteq \{0, 1\}\} \)

We turn this into an automaton with regex labels
Let’s eliminate state $B$

Then we eliminate state $C$ and obtain $A_D$

with regex $(0 + 1)^*1(0 + 1)(0 + 1)$
From

we can eliminate $D$ to obtain $A_C$

with regex $(0 + 1)^*1(0 + 1)$

• The final expression is the sum of the previous two regex’s:

$$(0 + 1)^*1(0 + 1)(0 + 1) + (0 + 1)^*1(0 + 1)$$
From regex’s to $\epsilon$-NFA’s

**Theorem 3.7:** For every regex $R$ we can construct an $\epsilon$-NFA $A$, s.t. $L(A) = L(R)$.

**Proof:** By structural induction:

**Basis:** Automata for $\epsilon$, $\emptyset$, and $a$.

1. Automaton for $\epsilon$:
   - Start state = Final state
   - Accepts any input

2. Automaton for $\emptyset$:
   - No states, no transitions
   - Accepts nothing

3. Automaton for $a$:
   - Single state
   - Single transition labeled $a$
   - Accepts any input

**$\epsilon$-NFAs with properties:**
- Unique start and final states
- No arcs into the start state
- No arcs out of the final state
Induction: Automata for $R + S$, $RS$, and $R^*$
Example: We convert \((0 + 1)^*1(0 + 1)\)