$NFA - \Lambda$

S. Pramanik
$NFA - \Lambda$

- Concatanation of languages $0^*$ and $(01)^*$: $0^*(01)^*$

Figure 1:

- $NFA - \Lambda$ is an NFA but it also allows $\Lambda$-transitions.
- Advantage of NFA plus flexibility of $\Lambda$-transitions.
• **Definition:** A nondeterministic finite automata with \( \Lambda \)-transitions (abbreviated \( NFA-\Lambda \)) is a 5-tuple \((Q, \Sigma, q_0, A, \delta)\), where \( Q \) and \( \Sigma \) are finite sets, \( q_0 \in Q \), \( A \subseteq Q \), and
\[
\delta : Q \times (\Sigma \cup \{\Lambda\}) \to 2^Q
\]

• Note: \( \Lambda \) is not part of the alphabet. It is defined for any \( NFA - \Lambda \).

![Diagram of NFA-\Lambda](image)

Figure 2:

For the above NFA-\( \Lambda \)

\[
\begin{align*}
q_0 & \quad \{q_0\} \\
q_1 & \quad \{q_2\} \\
q_2 & \quad \{q_1\}
\end{align*}
\]

• Example:
\[
\begin{align*}
\delta(q_0, \Lambda) &= \{q_1\} \\
\delta(q_0, 0) &= \{q_0\} \\
\delta(q_1, 0) &= \{q_2\} \\
\delta(q_2, 1) &= \{q_1\}
\end{align*}
\]

• \( \delta^* \) for an NFA-\( \Lambda \)
For $x \in \Sigma$ and $p \in Q$, $p$ is the set of all states $q \in Q$ such that there is a sequence of transitions corresponding to $x$ by which the $NFA - \Lambda$ moves from $p$ to $q$.

- For the above NFA-$\Lambda$:
  
  $x=0$
  
  $\delta^*(q_0, 0) = \{q_0, q_1\}$
  
  $\delta^*(q_0, 00) = \{q_0, q_1, q_2\}$
  
  $\delta^*(q_0, 10) =$ not defined

- Nondeterminism due to $\Lambda$-transitions ($\delta^*(q_0, 0) = \{q_0, q_1\}$).

- accept $x=0?$, $x=00?$, $x=10?$

![Diagram of NFA](attachment:image.png)

**Figure 3:**

Language $= 0^* (1 + 0) 0^*$ same as $(0 + 1) 0^*$ or $(0 + 1)^*(0 + 1)(0 + 1)^*$
GOAL: Find Equivalent NFA

1. NFA-Λ can make transitions on Λ

2. How can the equivalent NFA simulate these Λ-Transitions.

3. Basic Idea:

   Λ-transition is a type of nondeterminism. If the transitions of NFA-Λ is as follows: \( p^0 \rightarrow q^\Lambda \rightarrow r \)
   then from state \( p \), the input symbol 0 allows us to go to either \( q \) or \( r \). Thus, we can eliminate the NFA-Λ without changing the states, by simply adding the transitions from \( p \) to \( r \) on input 0. Thus, we have the following:

   - For each state \( q \) of NFA-Λ and each character \( \alpha \) of \( \Sigma \), figure out which states are reachable from \( q \) taking any number of Λ-transitions and exactly one transition on that character \( \alpha \).

   - In the equivalent NFA, directly connect \( q \) to each of these states using an arc labeled with \( \alpha \).

4. 
Example:

NFA-

Process state 1:

Process state 2

Process state 3

Similarly process states 4 and 5, NFA: Combine the 5 NFA’s
Lambda Closure of a Set of States

- Lambda Closure, $\Lambda(S)$, for a set of states $S$ is defined as:
  1. Every element of $S$ is an element of $\Lambda(S)$.
  2. For any $q \in \Lambda(S)$, every element of $\delta(q, \Lambda)$ is in $\Lambda(S)$.
  3. No other elements of $Q$ are in $\Lambda(S)$.
- All states that can be reached through $\Lambda$ transitions only.

$$\Lambda(\{s\}) = \{s, w, q_0, p, t\}$$
$NFA - \Lambda$ to NFA

Input $NFA - \Lambda = (Q, \Sigma, q_0, \delta, A)$
Output $NFA = (Q_1, \Sigma_1, q_1, \delta_1, A_1)$

$Q_1 = Q$
$\Sigma_1 = \Sigma$
$q_1 = q_0$

What is $A_1$?

**Computing $\delta_1(q, a)$:**

$\delta_1(q, a)$ = the set of states reachable from state $q$ in the NFA-$\Lambda$ taking 0 or more $\Lambda$-transitions and exactly one transition on the character $a = \Lambda(\delta(\Lambda(q), a))$

Break this down into three steps:

- First compute all states reachable from $q$ using 0 or more $\Lambda$-transitions
  - We call this set of states $\Lambda(q)$

- Next, compute all states reachable from any element of $\Lambda(q)$ using the character $a$
  
  We can denote this states as $\delta(\Lambda(q), a)$

- Finally, compute all states reachable from states in $\delta(\Lambda(q), a)$ using 0 or more $\Lambda$-transitions.
  - We denote these states as $\Lambda(\delta(\Lambda(q), a)$
  - This is the desired answer
Example of Computing $\delta_1$

$\delta_1(1, b) = \{3, 4, 5\}$

1. Compute $\Lambda(1)$, all states reachable from state 1 using 0 or more $\Lambda$-transitions
   $\Lambda(1) = \{1, 2\}$

2. Compute $\delta(\Lambda(1), b)$, all states reachable from any element of $\Lambda(1)$ using the character $b$
   $\delta(\Lambda(1), b) = \delta(\{1, 2\}, b) = \delta(1, b) \cup \delta(2, b) = \emptyset \cup \{3\} = \{3\}$

3. Compute $\Lambda(\delta(\Lambda(1), b))$, all states reachable from states in $\delta(\Lambda(1), b)$ using 0 or more $\Lambda$-transitions.
   $\Lambda(\delta(\Lambda(1), b)) = \Lambda(3) = \{3, 4, 5\}$
Recursive Definition of $\delta^*$ for an $NFA - \Lambda$

- **Definition of $\delta^*$**
  1. For any $q \in Q$, $\delta^*(q, \Lambda) = \Lambda(\{q\})$
  2. For any $q \in Q$, $y \in \Sigma^*$, and $a \in \Sigma$,
     $$\delta^*(q, ya) = \Lambda(\cup_{r \in \delta^*(q, y)} \delta(r, a))$$

A string $x$ is accepted by $M$ if $\delta^*(q_0, x) \cap A \neq \emptyset$.

- **Similar to $\delta^*$ of NFA except**
  1. Start with $\Lambda$ in front of the input.
  2. After each input symbol do $\Lambda$ closure.

We will use recursive definition of $\delta^*$ to calculate the set $\delta^*(q_0, abc)$. This set is defined in terms of $\delta^*(q_0, ab)$, which is defined in terms of $\delta^*(q_0, a)$ and which is then defined in terms of $\delta^*(q_0, \Lambda)$ which is $\Lambda(q_0)$ (the base case). We, therefore approach the calculation from bottom up, calculating $\delta^*(q_0, \Lambda)$ first.

- **$\delta^*(q_0, abc)$**

  $$\delta^*(p, \Lambda/abc) \text{ where } p \in P = \Lambda(\delta(q_0, \Lambda))$$

  $$\delta^*(r, \Lambda a/bc) \text{ where } r \in R = \Lambda((\delta(p, a), p \in P)$$

  $$\delta^*(s, \Lambda ab/c) \text{ where } s \in S = \Lambda((\delta(r, b), r \in R)$$

  $$\delta^*(m, \Lambda abc/) \text{ where } m \in M = \Lambda((\delta(s, c), s \in S)$$
\begin{itemize}
\item \(\delta^*(q_0, 01) = \delta^*(r, \Lambda/01), r \in \{q_0, p, t\}\)
\end{itemize}

\[
= \delta^*(s, \Lambda 0/1), s \in \Lambda(\delta(q_0, 0) \cup \delta(p, 0) \cup \delta(t, 0)) = \Lambda(\{p, u\}) = \{p, u,\}
\]

\[
= \delta^*(m, \Lambda 01/), m \in \Lambda(\delta(p, 1) \cup \delta(u, 1)) = \Lambda(\{r\}) = \{r\}
\]
Constructing NFA from $NFA - \Lambda$

- Theorem: If $L$ is accepted by an $NFA - \Lambda$, there is an equivalent NFA that also accepts $L$.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$\delta(q, \Lambda)$</th>
<th>$\delta(q, 0)$</th>
<th>$\delta(q, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>${B}$</td>
<td>${A}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>B</td>
<td>${D}$</td>
<td>${C}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>C</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${B}$</td>
</tr>
<tr>
<td>D</td>
<td>$\emptyset$</td>
<td>${D}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

- $\delta^*(A, 0) = \{A, B, C, D\}$, $\delta^*(A, 1) = \{??\}$, $\delta^*(B, 0) = \{C, D\}$
- $\delta^*(B, 1) = \{??\}$, $\delta^*(C, 0) = \{??\}$, $\delta^*(C, 1) = \{?\}$
- $\delta^*(D, 0) = \{??\}$, $\delta^*(D, 1) = \{??\}$

- Draw NFA using the above transitions. Show $\delta^*_1$ for the same inputs as above for NFA are the same.

- Prove that $\delta^*_1(q, x) = \delta^*(q, x)$ for any $x$ except $x = \Lambda$.

- See the difference: $\delta^*(A, \Lambda = \{A, B, D\}$ but $\delta^*_1(A, \Lambda) = \{A\}$

- Justify conditions for acceptance states:

1. Acceptance state of the $NFA - \Lambda$ becomes the acceptance states of the NFA.

2. If $\Lambda(\{q_0\})$ contains an acceptance state then $q_0$ also becomes an acceptance state of the NFA. Justification:
3. For the exercise problem:

\[ \Lambda(\{A\}) = \{A, B, D\} \] containing the final state D

Therefore, Both D and A are going to be final states.
Theorem: For every language $L \subseteq \Sigma^*$ accepted by an NFA-$\Lambda$: $M = (Q, \Sigma, q_0, A, \delta)$, there is an NFA (without $\Lambda$-transitions): $M_1 = (Q_1, \Sigma_1, q_1, A_1, \delta_1)$ that also accepts $L$.

Proof:
We define $M_1$ as

$Q_1 = Q$
$\Sigma = \Sigma_1$
$q_0 = q_1$
$\delta_1(q, \Lambda) = \emptyset$ (there is no $\Lambda$-transitions in $M_1$
$\forall q \in Q$ and $\forall \sigma \in \Sigma$
$\delta_1(q, \sigma) = \delta^*(q, \sigma)$
and
$A_1 = A \cup \{q_0\}$ if $\Lambda \in L$ else $A_1 = A$

We want to show $L(M_1) = L(M) = L$

First, assume $|x| = 0$

This is the base case for the proof by induction later. If $\Lambda$ is accepted by $M$ then $\Lambda(\{q_0\}) \subseteq A$ which includes $q_0$. By definition of $M_1$, $q_0 = q_1$. Therefore, $q_1 \in A_1$ by definition of $M_1$. $\Lambda$ is accepted by $M_1$

If the string $\Lambda \notin \epsilon L(M)$, than $A = A_1$; therefore, $q_0 \notin \epsilon A_1$, and $\Lambda \notin \epsilon L(M_1)$

Second, assume $|x| \geq 1$

We prove $\delta_1^*(q, x) = \delta^*(q, x)$ for $|x| \geq 1$
We prove this by induction:
Assume \(| y | = n \geq 1, \delta_1^*(q, y) = \delta^*(q, y)\) (hypothesis)
Then \(| y\sigma | = n + 1\) and
\[
\delta_1^*(q, ya) = \bigcup \{\delta_1(p, \sigma) \mid p \in \delta_1^*(q, y)\} \text{ by definition of } \delta_1^*
\]
\[
= \bigcup \{\delta_1(p, \sigma) \mid p \in \delta^*(q, y)\} \text{ by the induction hypothesis}
\]
\[
= \bigcup \{\delta^*(p, \sigma) \mid p \in \delta^*(q, y)\} \text{ by definition of } \delta_1.
\]

Now if \(x \in L(M)\) we want to show \(x \in L(M_1)\)
Because \(x \in L(M)\), \(\delta^*(q_0, x)\) contains an element of \(A\);
therefore since \(\delta^*(q_0, x) = \delta_1^*(q_0, x)\) and \(A \subseteq A_1\), \(x \in L(M_1)\).

Now if \(x \in L(M_1)\) we show \(x \in L(M)\)
If \(x \in L(M_1)\) then \(\delta_1^*(q_0, x)\) contains an element of \(A_1\). If
\(A_1 = A\) then \(x \in L(M)\). However, if \(A_1 \neq A\) then \(A_1\) has
\(q_0\) and \(A\) does not. If \(A_1\) contains \(q_0\) then \(\Lambda \in L(M_1)\) i.e.,
\(\delta_1^*(q_0, x)\) contains \(q_0\). However, we have shown
\(\delta_1^*(q_0, x) = \delta^*(q_0, x)\) Thus,
\(\delta^*(q_0, x)\) contains \(q_0\) and therefore, contains \(\Lambda(\{q_0\})\)
Thus, even when \(A_1 \neq A\) \(x \in L(M)\)

NOTE: even if \(| x | \geq 1\), \(M_1\) can go to \(q_1\) (for example
string \(x = 0^*\), and \(x\) is accepted by reaching \(q_1\)). In \(M\),
the string \(\Lambda\) does not have to be accepted by reaching \(q_0\);
it is accepted by reaching any of the states in \(\Lambda(\{q_0\})\).
Theorem 4.3

1. L can be recognized by an FA
2. L can be recognized by an NFA
3. L can be recognized by an \( NFA - \Lambda \)
Kleene’s Theorem

• Examples:

![Diagram of three states and transitions for examples](image)

Figure 7:

Does Not work:

![Diagram showing a non-working example](image)

Figure 8:

• Kleene’s Theorem 4.4 (Part 1):

Any regular language can be accepted by a finite automata.

Proof:

– By theorem 4.3 (same L can be recognized by FA, NFA and $\Lambda$-NFA) it is sufficient to show that any regular language can be accepted by an $NFA - \Lambda$.

– Definition 3.1 gives the definition of regular languages and we construct $NFA - \Lambda$ based on this definition.

– $NFA - \Lambda$ for the three basic languages (figure 11):

$\emptyset, \{\Lambda\}, \{a\}(a\epsilon\Sigma)$

Now if $L_1$ and $L_2$ are two regular languages recognized by two $NFA - \Lambda$’s $M1: (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M2: (Q_2, \Sigma, q_2, A_2, \delta_2)$
Figure 9:

, then we give construction for $NFA - \Lambda$ s union, concatenation and kleen *, i.e., $M_u, M_c, \text{ and } M_k$, recognizing languages $L_1 \cup L_2, L_1.L_2$ and $L_1^*$, respectively.

We will give construction for $M_c : (Q_c, \Sigma, q_c, A_c, \delta_c)$ only:

By renaming states we can assume $Q_1 \cap Q_2 = \emptyset$.

1. $Q_c = Q_1 \cup Q_2$
2. $q_c = q_1$
3. $A_c = A_2$
4. Include all transitions of $M_1$ and $M_2$ in $M_c$
5. Add $\Lambda$ transitions from each of the states in $A_1$ to $q_2$. Refer to the following figure:

Figure 10:

Show that if $x_1$ is accepted by $M_1$ and $x_2$ is accepted by $M_2$ then $x_1x_2$ is accepted by $M_c$

Since $x_1$ is accepted by $M_1$ and $M_c$ has all the states and the transitions of $M_1$, $M_c$ will reach the state cor-
responding to an acceptance state of $M_1$. It will then follow the transition from this state to the state of $M_c$ that corresponds to the start state of $M_2$. Since $x_2$ is accepted by $M_2$ and $M_c$ has all the states and the transitions of $M_2$, $M_c$ will reach the state that corresponds to the acceptance state of $M_2$ which is also an acceptance state of $M_c$. Thus $M_c$ accepts $x_1x_2$.

**Show if $x$ is accepted by $M_c$ then $x = x_1x_2$, and $x_1$ is accepted by $M_1$ and $x_2$ is accepted by $M_2$:**

Since $x$ is accepted by $M_c$, it starts in $q_c = q_1$ and ends in $q = A_2$. This can only be done through an intermediate transition as defined above. Since $Q_1 \cap Q_2 = \emptyset$, this crossing can happen only once. All transitions before crossing are due to those of machine $M_1$ and after crossing are due to those of $M_2$. Thus, $x = x_1x_2$, $x$ is accepted by $M_c$ and $x_1$ is used before crossing and is accepted by $M_1$, $x_2$ is used after crossing and is accepted by $M_2$.

**Proof for $M_k$:**

Base case is similar to above that there exists $NFA - \Lambda$ for empty set, etc. We assume there is a $NFA - \Lambda$ for $L$ then there is an $NFA - \Lambda$ for $L^*$.

If $x \in L^*$ then $x \in L(M_k)$

and if $x \in L(M_k)$ then $x \in L^*$

$x \in L^*$ means that $x = x_1x_2\ldots x_m$, where each $x_i \in L$

x can be written as $(\Lambda x_1 \Lambda)(\Lambda x_2 \Lambda)\ldots(\Lambda x_m \Lambda)$

Where each one of them $(\Lambda x_m \Lambda)$ gets accepted by M and comes back to start state of $L(M_k)$. Therefore $x$ is accepted by $L(M_k)$
• Construct an \( NFA - \lambda \) for the following regular expression using Kleene's theorem:
\[(1 + 0)^*1\]

![Figure 11: NFA Diagram](image-url)