Equivalence Between DFAs and NFAs

Every DFA is (a special case of) an NFA, hence $DFA-s \subseteq NFA-s$, but $NFA-s \not\subseteq DFA-s$.

Nevertheless, these classes are *Equivalent*.

This means that for any NFA $N$ there exists a DFA $D$ satisfying: $L(D) = L(N)$.

Equivalence Between DFAs and NFAs

Thus, to prove equivalence of the classes we prove:

**Theorem:** For every NFA $N$ there exists a DFA $D$ satisfying $L(D) = L(N)$.

**Proof Idea:** The proof is *Constructive:* We assume that we know $N$, and construct a simulating DFA, $D$. 

Roadmap for Lecture

In this lecture we:

- Prove that NFA-s and DFA-s are *equivalent*.
- Present the three regular operations.
- Prove that each of the regular operations preserves regularity.
Proof

Let $N = (Q, \Sigma, \delta, q_0, F)$ recognizing some language $A$. The state set of the simulating DFA $D$, should reflect the fact that at each step of the computation, $N$ may occupy several states.

Thus we define the state set of $D$ as $P(Q)$, the power-set of the state set of $N$.

Proof (cont.)

Our next task is to define $D$'s transition function, $\delta'$:

For any $R \in Q'$ and $a \in \Sigma$ define

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

If $R$ is a state of $M$, then it is a set of states of $N$.

When $M$ in state $R$ processes an input symbol $a$, $M$ goes to the set of states to which $N$ will go in any of the branches of its computation.

Proof (cont.)

An alternative way to write the definition of $M$'s transition function, $\delta'$ is:

For any $R \in Q'$ and $a \in \Sigma$ define

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

And the explanation is just the same.

Note: if $\bigcup_{r \in R} \delta(r, a) = \phi$, then $\delta'(R, a) = \phi$

Which is OK since $\phi \in P(Q)$. 

Proof (cont.)

Let $N = (Q, \Sigma, \delta, q_0, F)$ recognizing some language $A$. First we assume that $N$ has no $\varepsilon$ - transitions.

Define $D = (Q', \Sigma, \delta', q_0', F)$ where $Q' = P(Q)$.
The initial state of \( M \) is: 

\[ q_0' = \{ q_0 \} . \]

Finally, the final state of \( M \) is: 

\[ F' = \{ R \in Q' | R \text{ contains a finite state of } N \} \]

Since \( D \) accepts if \( N \) reaches at least one accepting state.

The reader can verify for her/him self that \( D \) indeed simulates \( N \).

It remains to consider \( \varepsilon \) - transitions. For any state \( R \) of \( D \) define \( E(R) \) to be the collection of states of \( R \) unified with the states reachable from \( R \) by \( \varepsilon \)- transitions.

The old definition of \( \delta'(R, a) \) is:

\[ \delta'(R, a) = \{ q \in \delta(R, a) | q \text{ for some } r \in R \} \]

And the new definition is:

\[ \delta'(R, a) = \{ q \in \delta(R, a) | q \in E(\delta(R, a)) \text{ for some } r \in R \} \]

In addition, we have to change the definition of \( q_0' \), the initial state of \( M \). The previous definition, \( q_0' = \{ q_0 \} \), is replaced with \( q_0' = E(\{ q_0 \}) \).

Once again the reader can verify that the new definition of \( D \) satisfies all requirements.

A language \( L \) is regular if and only if there exists an NFA recognizing \( L \).
The Regular Operations

Let $A$ and $B$ be 2 regular languages above the same alphabet, $\Sigma$. We define the 3 Regular Operations:

- **Union**: $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$.
- **Concatenation**: $A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$.
- **Star**: $A^* = \{ x_1, x_2, \ldots, x_k \mid k \geq 0 \text{ and } x_k \in A \}$.

Elaboration

- **Union** is straightforward.
- **Concatenation** is the operation in which each word in $A$ is concatenated with every word in $B$.
- **Star** is a unary operation in which each word in $A$ is concatenated with every other word in $A$ and this happens any finite number of times.

The Regular Operations - Examples

- $A = \{ \text{good, bad} \}$  
  $B = \{ \text{girl, boy} \}$
- $A \cup B = \{ \text{good, bad, girl, boy} \}$
- $A \circ B = \{ \text{goodgirl, goodboy, badgirl, badboy} \}$
- $A^* = \{ \epsilon, \text{good, bad, goodgood, goodbad,} \}$
  \[ \text{goodgoodgoodbad, badbadgoodbad,} \ldots \]  

Motivation for Nondeterminism

We want to use the regular operations for a systematic construction of all regular expressions.

Given a two DFA-s, their product DFA can recognizes their union, but we do not know how to construct a DFA recognizing either concatenation or star.

This can be proved by using NFA-s.
**Theorem**

The class of Regular languages is **closed** under the all three **regular operations**.

**Proof for union Using NFA-s**

If \( A_1 \) and \( A_2 \) are regular, each has its own recognizing automaton \( N_1 \) and \( N_2 \), respectively.

In order to prove that the language \( A_1 \cup A_2 \) is regular we have to construct an FA that accepts exactly the words in \( A_1 \cup A_2 \).

**A Pictorial proof**

![Diagram showing NFA for union of languages]

Let \( N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) recognizing \( A_1 \), and \( N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) recognizing \( A_2 \).

Construct \( N = (Q, \Sigma, \delta, q, F) \) to recognize \( A_1 \cup A_2 \),

Where \( Q = \{q_0\} \cup Q_1 \cup Q_2 \), \( F = F_1 \cup F_2 \),

\[
\delta(q,a) = \begin{cases} 
\delta_1(q,a) & q \in Q_1 \\
\delta_2(q,a) & q \in Q_2 \\
\{q_1,q_2\} & q = Q_i \text{ and } a = \varepsilon \\
\emptyset & q = Q_i \text{ and } a \neq \varepsilon 
\end{cases}
\]
The class of Regular languages is **closed** under the *concatenation* operation.

**Proof idea**

Given an input word to be checked whether it belongs to \( A_1 \circ A_2 \), we may want to run \( N_1 \) until it reaches an accepting state and then to move to \( N_2 \).

The problem: Whenever an accepting state is reached, we cannot be sure whether the word of \( A_1 \) is finished yet.

The idea: Use non-determinism to choose the right point in which the word of \( A_1 \) is finished and the word of \( A_2 \) starts.

![A Pictorial proof](diagram.png)
Proof using NFAs

Let $N_1 = (Q_1, \Sigma, \delta, q_1, F_1)$ recognizing $A_1$, and $N_2 = (Q_2, \Sigma, \delta, q_2, F_2)$ recognizing $A_2$.

Construct $N = (Q, \Sigma, \delta, q_1, F)$ to recognize $A_1 \circ A_2$, where

$Q = Q_1 \cup Q_2$, $F = F_2$, 

$$\delta(q,a) = \begin{cases} 
\delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta_2(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\
\delta_1(q,a) \cup q_2 & q = F_1 \text{ and } a = \epsilon \\
\delta_1(q,a) & q = Q_2 
\end{cases}$$

A Pictorial proof

![Diagram of NFA](image)

Proof using NFAs

Let $N_1 = (Q_1, \Sigma, \delta, q_1, F_1)$ recognizing $A_1$.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1^*$

Where $Q = \{q_0\} \cup Q_1$, $F = \{q_0\} \cup F_1$, and

$$\delta(q,a) = \begin{cases} 
\delta(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\
\delta(q,a) \cup \{q_0\} & q \in F_1 \text{ and } a = \epsilon \\
q_0 & q = q_0 \text{ and } a = \epsilon \\
\phi & q = q_0 \text{ and } a \neq \epsilon 
\end{cases}$$

Theorem

The class of Regular languages is \textbf{closed} under the \textit{star} operation.
Wrap Up

In this lecture we:
• Proved equivalence between DFA-s and NFA-s.
• Motivated and defined the three Regular Operations.
• Proved that the regular operations preserve regularity.