CHAPTER 2
FINITE AUTOMATA

1. Deterministic Finite Automata DFA
2. Nondeterministic Finite Automata NFA
3. Finite Automata and Regular Expressions
4. Languages that are Not Regular
5. State Minimization
CHAPTER 2
PART 2: Nondeterministic Finite Automata NDFA
NDFA: Nondeterministic Finite Automata

Now we add a new powerful feature to the finite automata. This feature is called nondeterminism. 

Nondeterminism is essentially the ability to change states in a way that is only partially determined by the current state and input symbol, or a string of symbols, empty string included.

The automaton, as it reads the input string, may choose at each step to go to any of its states. The choice is not determined by anything in our model, and therefore it is said to be nondeterministic.

At each step there is always a finite number of choices, hence it is still a finite automaton.
Class Definition
A Nondeterministic Finite Automata is a quintuple

\[ M = (K, \Sigma, \Delta, s, F) \]

where

- \( K \) is a finite set of states
- \( \Sigma \) as an alphabet
- \( s \in K \) is the initial state
- \( F \subseteq K \) is the set of final states
- \( \Delta \) is a finite set and

\[ \Delta \subseteq K \times \Sigma^* \times K \]

\( \Delta \) is called the transition relation
We usually use different symbols for \( K, \Sigma, \) i.e. we have that

\( K \cap \Sigma = \emptyset \)
NDFA Definition

Class Definition revisited
A Nondeterministic Finite Automata is a quintuple

\[ M = (K, \Sigma, \Delta, s, F) \]

where
- \( K \) is a finite set of states
- \( K \neq \emptyset \) because \( s \in K \)
- \( \Sigma \) as an alphabet
- \( \Sigma \) can be \( \emptyset \) - case to consider
- \( s \in K \) is the initial state
- \( F \subseteq K \) is the set of final states
- \( F \) can be \( \emptyset \) - case to consider
- \( \Delta \) is a finite set and \( \Delta \subseteq K \times \Sigma^* \times K \)
- \( \Delta \) is called the transition relation
- \( \Delta \) can be \( \emptyset \) - case to consider
Some Remarks

R1 We must say that $\Delta$ is a finite set because the set $K \times \Sigma^* \times K$ is countably infinite, i.e. $|K \times \Sigma^* \times K| = \aleph_0$ and we want to have a finite automata and we defined it as

$$\Delta \subseteq K \times \Sigma^* \times K$$

R2 The DFA transition function $\delta : K \times \Sigma \rightarrow K$ is (as any function!) a relation

$$\delta \subseteq K \times \Sigma \times K$$

R3 The set $\delta$ is always finite as the set $K \times \Sigma \times K$ is finite

R4 The DFA transition function $\delta$ is a particular case of the NDFA transition relation $\Delta$, hence similarity of notation
We extend the notion of the **state diagram** to the case of the NDFA in natural was as follows:

\((q_1, w, q_2) \in \Delta\) means that \(M\) in a state \(q_1\) reads the word \(w \in \Sigma^*\) and goes to the state \(q_2\).

**Picture**

![NDFA Diagram](image)

**Remember** that in particular \(w = e\)
Examples

Example 1
Let $M$ be given by a diagram

By definition $M$ is not a deterministic DFA as it reads $e \in \Sigma^*$

$L(M) = \{e\}$
Examples

Example 2
Let $M_1$ be given by a diagram

Observe that $M_1$ is not a deterministic DFA as $(q, a, q_1) \in \Delta$ and $(q, a, q_2) \in \Delta$ what proves that $\Delta$ is not a function

$$L(M_1) = \{a\}$$
Examples

Example 3
Let $M$ be given by a diagram

$M$ is not a deterministic DFA as $(q_2, e, q_0) \in \Delta$ and this is not admitted in DFA

$\Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0), (q_2, e, q_0)\}$
Examples

Example 4
Let $M$ be given by a diagram

$M$ is not a deterministic DFA as $(q, ab, q_1) \in \Delta$ and this is not admitted in DFA

$\Delta = \{(q, ba, q), (q, ab, q_1), (q, e, q_3)\}$ and $F = \emptyset$

$L(M1) = \emptyset$
A Nondeterministic Finite Automata is a quintuple 

\[ M = (K, \Sigma, \Delta, s, F) \]

where

- \( K \) is a finite set of states
- \( \Sigma \) as an alphabet
- \( s \in K \) is the initial state
- \( F \subseteq K \) is the set of final states

\( \Delta \), the transition relation is defined as

\[ \Delta \subseteq K \times (\Sigma \cup \{e\}) \times K \]

Observe that \( \Delta \) is finite set as both \( K \) and \( \Sigma \cup \{e\} \) are finite sets
Example

Let $M$ be automaton from Example 3 given by a diagram

$M$ follows the Book Definition as

$$\Delta \subseteq K \times (\Sigma \cup \{e\}) \times K$$
Equivalence of Definitions

The Class and the Book definitions are equivalent

1. We get the **Book Definition** as a particular case of the **Class Definition** as

   \[ \Sigma \cup \{e\} \subseteq \Sigma^* \]

2. We will show later a general method how to transform any automaton defined by the **Class Definition** into an equivalent automaton defined by the **Book Definition**

When solving problems you can use any of these definitions
Configuration and Transition Relation

Given a **NDFA** automaton

\[ M = (K, \Sigma, \Delta, s, F) \]

We define as we did in the case of **DFA** the notions of a **configuration**, and a **transition relation**

**Definition**

A **configuration** in a **NDFA** is any tuple

\[ (q, w) \in K \times \Sigma^* \]
Configuration and Transition Relation

Definition
A transition relation in $M = (K, \Sigma, \Delta, s, F)$ defined by the Class Definition is a binary relation

$$\vdash_M \subseteq (K \times \Sigma^*) \times (K \times \Sigma^*)$$

such that $q, q' \in K, \ u, w \in \Sigma^*$

$$(q, uw) \vdash_M (q', w)$$

if and only if

$$(q, u, q') \in \Delta$$

For $M$ defined by the Book Definition definition of the Transition Relation is the same but for the fact that

$$u \in \Sigma \cup \{e\}$$
Language Accepted by M

We define, as in the case of the deterministic DFA, the language accepted by the non-deterministic M as follows.

Definition

\[ L(M) = \{ w \in \Sigma^* : (s, w) \vdash_M^* (q, e) \text{ for } q \in F \} \]

where \( \vdash_M^* \) is the reflexive, transitive closure of \( \vdash_M \)
Equivalency of Automata

We define now formally an equivalency of automata as follows.

**Definition**
For any two automata $M_1, M_2$ (deterministic or nondeterministic)

$$M_1 \simeq M_2 \quad \text{if and only if} \quad L(M_1) = L(M_2)$$

Now we are going to formulate and prove the main theorem of this part of the Chapter 2, informally stated as

**Equivalency Statement**
The notions of a deterministic and a non-deterministic automata are equivalent.
The Equivalency Statement consists of two Equivalency Theorems.

Equivalency Theorem 1
For any DFA $M$, there is an NDFA $M'$, such that $M \approx M'$, i.e. such that

$$L(M) = L(M')$$

Equivalency Theorem 2
For any NDFA $M$, there is a DFA $M'$, such that $M \approx M'$, i.e. such that

$$L(M) = L(M')$$
Equivalency of Automata Theorems

Equivalency Theorem 1
For any DFA $M$, there is a NDFA $M'$, such that $M \approx M'$, i.e. such that $L(M) = L(M')$

Proof
Any DFA $M$ is a particular case of a DFA $M'$ because any function $\delta$ is a relation.

Moreover $\delta$ and its a particular case of the relation $\Delta$ as $\Sigma \subseteq \Sigma \cup \{e\}$ (for the Book Definition) and $\Sigma \subseteq \Sigma^*$ (for the Class Definition).

This ends the proof.
Equivalency of Automata Theorems

Equivalency Theorem 2
For any NDFA \( M \), there is a DFA \( M' \), such that \( M \approx M' \), i.e. such that

\[
L(M) = L(M')
\]

Proof
The proof is far from trivial. It is a constructive proof; We will describe, given a NDFA \( M \), a general method of construction step by step of an DFA \( M' \) that accepts the same language as \( M \)

Before we define the proof construction we discuss some examples and some general automata properties
EXAMPLES and QUESTIONS
Examples

Example 1
Here is a **diagram** of NDFA $M1$ - Class Definition

\[ L(M1) = (ab \cup aba)^* \]
Examples

Example 2
Here is a diagram of NDFA M2 - Book Definition

Observe that M2 is not deterministic (even if we add "plus trap states) because Δ is not a function as (q₁, b, q₀) ∈ Δ and (q₁, b, q₂) ∈ Δ

\[ L(M2) = (ab \cup aba)^* \]
Example 3
Here is a diagram of NDFA M3 - Book Definition

Observe that M2 is not deterministic $(q_1, e, q_0) \in \Delta$

$L(M3) = (ab \cup aba)^*$
Question 1

All automata in Examples 1-3 accept the same language, hence by definition, they are equivalent nondeterministic automata, i.e.

\[ M_1 \approx M_2 \approx M_3 \]

Question 1

Construct a deterministic automaton \( M_4 \) such that

\[ M_1 \approx M_2 \approx M_3 \approx M_4 \]
Question 1 Solution

Here is a **diagram** of **deterministic** DFA $M_4$

Observe that $q_4$ is a **trap state**

$L(M_4) = (ab \cup aba)^*$
Question 2

Given an alphabet

\[ \Sigma = \{a_1, a_2, \ldots, a_n\} \quad \text{for} \quad n \geq 2 \]

Question 2

Construct a nondeterministic automaton \( M \) such that

\[ L = \{ w \in \Sigma^* : \text{at least one letter from } \Sigma \text{ is missing in } w \} \]

Take \( n = 4 \), i.e. \( \Sigma = \{a_1, a_2, a_3, a_4\} \)

Some words in \( L \) are:

\[ e \in L, \quad a_1 \in L, \quad a_1a_2a_3 \in L, \quad a_1a_2a_2a_3a_3 \in L \quad a_1a_4a_1a_2 \in L, \ldots \]
Question 2 Solution

Here is solution for $n = 3$, i.e. $\Sigma = \{a_1, a_2, a_3\}$

Write a solution for $n = 4$
Question 2 Solution

Here is the solution for $n = 4$, i.e. $\Sigma = \{a_1, a_2, a_3, a_4\}$

Write a general form of solution for $n \geq 2$
Question 2 Solution

General case

\[ M = (K, \Sigma, \Delta, s, F) \] for \( \Sigma = \{a_1, a_2, \ldots, a_n\} \) and \( n \geq 2 \),
\( K = \{s = q_0, q_1, \ldots, q_n\} \), \( F = K - \{q_0\} \), or \( F = K \) and

\[ \Delta = \bigcup_{i=1}^{n} \{(q_0, e, q_i)\} \cup \bigcup_{i,j=1}^{n} \{(q_i, a_j, q_i) : i \neq j\} \]

\( i \neq j \) means that \( a_i \) is missing in the loop at state \( q_i \)
PROPERTIES
Equivalence of Two Definitions
Equivalence of Two Definitions

Book Definition (BD)

\[ \Delta \subseteq K \times (\Sigma \cup \{e\}) \times K \]

Class Definition (CD)

\[ \Delta \text{ is a finite set and} \]

\[ \Delta \subseteq K \times \Sigma^* \times K \]

Fact 1

Any (BD) automaton \( M \) is a (CD) automaton \( M \)

Proof

The (BD) of \( \Delta \) is a particular case of the (CD) as

\[ \Sigma \cup \{e\} \subseteq \Sigma^* \]
Equivalence of Two Definitions

Fact 2
Any (CD) automaton $M$ can be transformed into an equivalent (BD) automaton $M'$.

Proof
We use a “stretching” technique.
For any $w \neq e, \ w \in \Sigma^*$ and (CD) transition $(q, w, q') \in \Delta$, we transform it into a sequence of (BD) transactions each reading only $\sigma \in \Sigma$ that will at the end read the whole word $w \in \Sigma^*$.
We leave the transactions $(q, e, q') \in \Delta$ unchanged.
Stretching Process

Consider \( w = \sigma_1, \sigma_2, \ldots, \sigma_n \) and a transaction \((q, w, q) \in \Delta\) as depicted on the diagram.

We construct \( \Delta' \) in \( M' \) by replacing the transaction \((q, \sigma_1, \sigma_2, \ldots, \sigma_n, q)\) by

\[
(q, \sigma_1, p_1), (p_1, \sigma_2, p_2), \ldots, (p_{n-1}, \sigma_n, q)
\]

and adding new states \( p_1, p_2, \ldots, p_{n-1} \) to the set \( K \) of \( M \) making at this stage

\[
K' = K \cup \{p_1, p_2, \ldots, p_{n-1}\}
\]
Stretching Process

This transformation is depicted on the diagram below.

We proceed in a similar way in a case of \( w = \sigma_1, \sigma_2, \ldots \sigma_n \) and a transaction \((q, w, q') \in \Delta \)
Equivalent $M'$

We proceed to do the "stretching" for all $(q, w, q') \in \Delta$ for $w \neq e$ and take as

$$K' = K \cup P$$

where $P = \{p : p \text{ added by stretching for all } (q, w, q') \in \Delta\}$

We take as

$$\Delta = \Delta^\Sigma \cup \{(q, \sigma_i, p) : p \in P, w = \sigma_1, \ldots \sigma_n, (q, w, q') \in \Delta\}$$

where

$$\Delta^\Sigma = \{(q, \sigma, q') \in \Delta : \sigma \in (\Sigma \cup \{e\}), q, q' \in K\}$$
Proof of Equivalency of DFA and NDFA
Equivalency of DFA and NDFA

Let’s now go back now to the **Equivalency Statement** that consists of the following two equivalency theorems

**Equivalency Theorem 1**
For any DFA $M$, there is is a NDFA $M'$, such that $M \approx M'$, i.e. such that

$$L(M) = L(M')$$

This is already **proved**

**Equivalency Theorem 2**
For any NDFA $M$, there is a DFA $M'$, such that $M \approx M'$, i.e. such that

$$L(M) = L(M')$$

This is **to be proved**
Equivalency Theorem

Our goal now is to prove the following

**Equivalency Theorem 2**

For any nondeterministic automaton

\[ M = (K, \Sigma, \Delta, s, F) \]

there is, i.e. we give an algorithm for its construction a deterministic automaton

\[ M' = (K', \Sigma, \delta = \Delta', s', F') \]

such that

\[ M \approx M' \]

i.e.

\[ L(M) = L(M') \]
General Remark

We base the proof of the equivalency of DFA and NDFA automata on the Book Definition of NDFA.

Let's now explore some ideas laying behind the main points of the proof. They are based on two differences between the DFA and NDF automata.

We discuss now these differences and basic ideas how to overcome them, i.e. how to "make" a deterministic automaton out of a non-deterministic one.
NDFA and DFA Differences

Difference 1

DFA transition function $\delta$ even if expressed as a relation

$$\delta \subseteq K \times \Sigma \times K$$

must be a function, while the NDFA transition relation $\Delta$

$$\Delta \subseteq K \times (\Sigma \cup \{e\}) \times K$$

may not be a function
NDFA and DFA Differences

Difference 2

DFA transition function $\delta$ domain is the set

$$K \times \Sigma$$

while NDFA transition relation $\Delta$ domain is the set

$$K \times \Sigma \cup \{e\}$$

Observe that the NDFA transition relation $\Delta$ may contain a configuration $(q, e, q')$ that allows a nondeterministic automaton to read the empty word $e$, what is not allowed in the deterministic case.

In order to transform a nondeterministic $M$ into an equivalent deterministic $M'$ we have to eliminate the both Differences 1 and 2.
Example

Let’s look first at the following

Example

\[ M = (\{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \Delta, s = q_0, F = \{q_2\}) \]

\[ \Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0)\} \]

Diagram of \( M \)
Example

The **non-function** part of the diagram is

![Diagram](image)

**Question**

How to transform it into a FUNCTION???

**IDEA 1**: make the states of $M'$ as some **SETS** made out of states of $M$ and put in this case

$$\delta(\{q_1\}, b) = \{q_0, q_2\}$$
IDEA ONE

IDEA 1: we make the states of $M'$ as some SETS made out of states of $M$

We read other transformation from the Diagram of $M$

\[
\delta(\{q_0\}, a) = \{q_1\}, \quad \delta(\{q_2\}, a) = \{q_0\} \text{ and of course } \delta(\{q_1\}, b) = \{q_0, q_2\}
\]

We make the state $\{q_0\}$ the initial state of $M'$ as $q_0$ was the initial state of $M$ and we make the states $\{q_0, q_2\}$ and $\{q_2\}$ final states of $M'$ and as $q_2$ was a final state of $M$
Example

We have constructed a part of

\[ M' = (K', \Sigma, \delta = \Delta', s', F') \]

The **Unfinished Diagram** is

There will be many **trap states**
Example Revisited

In the case of our **Example** we had \( K = \{ q_0, q_1, q_2 \} \)

\( K' = 2^k \) has \( 2^3 \) states

The portion of the **unfinished diagram** of \( M' \) is

It is obvious that even the finished diagram will have A LOT of trap states
Difference 2 and Idea Two

Difference 2 and Idea Two - how to eliminate the \( e \) transitions

Example 1
Consider \( M_1 \)

Observe that we can go from \( q_0 \) to \( q_1 \) reading only \( e \), i.e. without reading any input symbol \( \sigma \in \Sigma \)

\[ L(M_1) = a \]
Examples

Example 2
Consider $M_2$

Observe that we can go from $q_1$ to $q_2$ reading only $e$, i.e. without reading any input symbol $\sigma \in \Sigma$

$$L(M_2) = a$$
Examples

Example 3
Consider $M_3$

Observe that we can go from $q_2$ to $q_3$ and from $q_1$ to $q_3$ without reading any input.

$L(M_3) = a \cup b$
Idea Two - Sets $E(q)$

The definition of the transition function $\delta$ of $M'$ uses the following

**Idea Two:** a move of $M'$ on reading an input symbol $\sigma \in \Sigma$ imitates a move of $M$ on input symbol $\sigma$, possibly followed by any number of e-moves of $M$

To formalize this idea we need a special definition

**Definition of $E(q)$**

For any state $q \in K$, let $E(q)$ be the set of all states in $M$ they are reachable from state $q$ without reading any input, i.e.

$$E(q) = \{ p \in K : (q, e) \vdash_{M^*} (p, e) \}$$
Fact 1
For any state \( q \in K \) we have that \( q \in E(q) \)

Proof
By definition

\[
E(q) = \{ p \in K : (q, e) \vdash_{M^*} (p, e) \}
\]

and by the definition of reflexive, transitive closure \( \vdash_{M^*} \) the trivial path (case \( n=1 \)) always exists, hence

\[
(q, e) \vdash_{M^*} (q, e)
\]

what proves that \( q \in E(q) \)
Sets E(q)

Observe that by definitions of $\mathcal{HM}^*$ and E(q) we have that

Fact 2
1. E(q) is a closure of the set \{q\} under the relation

\[\{(p, r) : \text{there is a transition } (p, e, r) \in \Delta\}\]

2. E(q) can be computed by the following Algorithm

Initially set $E(q) := \{q\}$

while there is $(p, e, r) \in \Delta$ with $p \in E(q)$ and $r \notin E(q)$
do: $E(q) := E(q) \cup \{r\}$
Example

We go back to the Example 1, i.e.
Consider M1

We evaluate

\[ E(q_0) = \{q_0, q_1\}, \quad E(q_1) = \{q_1\}, \quad E(q_2) = \{q_2\} \]

Remember that always \( q \in E(q) \)
Definition of $M'$

Given a **nondeterministic** automaton $M = (K, \Sigma, \Delta, s, F)$ we define the **deterministic** automaton $M'$ equivalent to $M$ as

$$M' = (K', \Sigma, \delta', s', F')$$

where

$$K' = 2^K, \quad s' = \{s\}$$

$$F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$$

$\delta' : 2^K \times \Sigma \rightarrow 2^K$ is such that and for each $Q \subseteq K$ and for each $\sigma \in \Sigma$

$$\delta'(Q, \sigma) = \bigcup \{E(p) : p \in K \text{ and } (q, \sigma, p) \in \Delta \text{ for some } q \in Q\}$$
Definition of $\delta'$

**Definition** of $\delta'$

We re-write the definition of $\delta'$ in a following form that is easier to use.

$\delta' : 2^K \times \Sigma \rightarrow 2^K$ is such that and for each $Q \subseteq K$ and for each $\sigma \in \Sigma$

$$\delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : (q, \sigma, p) \in \Delta \text{ for some } q \in Q \}$$

or we write it in a more clear form as

$$\delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q (q, \sigma, p) \in \Delta \}$$
Construction of $M'$

Given a **nondeterministic** automaton $M = (K, \Sigma, \Delta, s, F)$

Here are the **STAGES** to follow when constructing $M'$

**STAGE 1**

1. For all $q \in K$, evaluate $E(q)$

   \[ E(q) = \{ p \in K : (q, e) \vdash_{M^*} (p, e) \} \]

2. Evaluate initial and final states: $s' = E(s)$ and $F' = \{ Q \subseteq K : Q \cap F \neq \emptyset \}$

**STAGE 2**

Evaluate $\delta'(Q, \sigma)$ for $\sigma \in \Sigma$, $Q \in 2^K$

\[ \delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q \ (q, \sigma, p) \in \Delta \} \]
Evaluation of $\delta'$

Observe that domain of $\delta'$ is $2^K \times \Sigma$ and can be very large.

We will evaluate $\delta'$ only on states that are relevant to the operation of $M'$ and making all other states trap states. We do so to assure that

$$M' \approx M$$

i.e. to be able to prove that

$$L(M) = L(M')$$

Having this in mind we adopt the following definition.
Evaluation of $\delta'$

**Definition**

We say that a state $Q \in 2^K$ is **relevant** to the operation of $M'$ and to the language $L(M')$ if it can be **reached** from the **initial state** $s' = E(s)$ by reading some input string.

Obviously, any state $Q \in 2^K$ that is **not reachable** from the **initial state** $s'$ is **irrelevant** to the operation of $M'$ and to the language $L(M')$. 
Construction of of M’ Example

Example
Let M be defined by the following diagram

STAGE 1
1. For all $q \in K$, evaluate $E(q)$
M does not have $e$-transitions so we get $E(q_0) = \{q_0\}$, $E(q_1) = \{q_1\}$, $E(q_2) = \{q_2\}$
2. Evaluate initial and some final states: $s' = E(q_0) = \{q_0\}$ and $\{q_2\} \in F'$
δ′ Evaluation

STAGE 2
Here is a General Procedure for δ′ evaluation
Evaluate δ′(Q, σ) only for relevant Q ∈ 2^K, i.e. follow the steps below

Step 1 Evaluate δ′(s′, σ) for all σ ∈ Σ, i.e. all states directly reachable from s′

Step (n+1)
Evaluate δ′ on all states that result from the Step n, i.e. on all states already reachable from s′

Remember

\[ δ′(Q, σ) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q (q, σ, p) \in \Delta \} \]
Example STAGE 2

### Diagram

![Diagram](image)

### STAGE 2

\[ \delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q (q, \sigma, p) \in \Delta \} \]

#### Step 1
We evaluate \( \delta'(\{q_0\}, a) \) and \( \delta'(\{q_0\}, b) \)

We look for the transitions from \( q_0 \)

We have only one \( (q_0, a, q_1) \in \Delta \) so we get

\[ \delta'(\{q_0\}, a) = E(q_1) = \{q_1\} \]

**There is no** transition \( (q_0, b, p) \in \Delta \) for any \( p \in K \), so we get

\[ \delta'(\{q_0\}, b) = E(p) = \emptyset \]
Example STAGE 2

By the Step 1 we have that all states directly reachable from $s'$ are $\{q_2\}$ and $\emptyset$

Step 2 Evaluate $\delta'$ on all states that result from the Step 1; i.e. on states $\{q_1\}$ and $\emptyset$

Obviously $\delta'(\emptyset, a) = \emptyset$ and $\delta'(\emptyset, b) = \emptyset$

To evaluate $\delta'(\{q_1\}, a), \delta'(\{q_1\}, b)$ we first look at all transitions $(q_1, a, p) \in \Delta$ on the diagram

There is no transition $(q_1, a, p) \in \Delta$ for any $p \in K$, so

$\delta'(\{q_1\}, a) = \emptyset$ and $\delta'(\emptyset, a) = \emptyset, \delta'(\emptyset, b) = \emptyset$
Step 2  To evaluate $\delta'(\{q_1\}, b)$ we now look at all transitions $(q_1, b, p) \in \Delta$ on the diagram

Here they are:  $(q_1, b, q_2), (q_1, b, q_0)$

$\delta'(Q, \sigma) = \bigcup_{p \in K} \{E(p) : \exists q \in Q (q, \sigma, p) \in \Delta\}$

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

We evaluated

\[\delta'(\{q_1\}, b) = \{q_0, q_2\}, \quad \delta'(\{q_1\}, a) = \emptyset\]

We also have that the state $\{q_0, q_2\} \in F'$
Example STAGE 2

**Step 3** Evaluate $\delta'$ on all states that result from the Step 2; i.e. on states $\{q_0, q_2\}$, $\emptyset$

Obviously $\delta'(\emptyset, a) = \emptyset$ and $\delta'(\emptyset, b) = \emptyset$

To evaluate $\delta'(\{q_0, q_2\}, a)$ we look at all transitions $(q_0, a, p)$ and $(q_2, a, p)$ on the diagram.

Here they are: $(q_0, a, q_1)$, $(q_2, a, q_0)$

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

Similarly $\delta'(\{q_0, q_2\}, b) = \emptyset$
Here is the **Diagram** of $M'$ after finishing STAGE 1 and **Steps 1-3** of the STAGE 2
Example STAGE 2

Step 4  Evaluate $\delta'$ on all states that result from the Step 3; i.e. on states $\{q_0, q_1\}, \emptyset$

Obviously $\delta'(\emptyset, a) = \emptyset$ and $\delta'(\emptyset, b) = \emptyset$

To evaluate $\delta'(\{q_0, q_1\}, a)$ we look at all transitions $(q_0, a, p)$ and $(q_1, a, p)$ on the diagram

Here there is one $(q_0, a, q_1)$, and there is no transition $(q_1, a, p)$ for any $p \in K$, so

$$\delta'(\{q_0, q_1\}, a) = E(q_1) \cup \emptyset = \{q_1\}$$

Similarly

$$\delta'(\{q_0, q_1\}, b) = \{q_0, q_2\}$$
Example STAGE 2

**Step 5** Evaluate \( \delta' \) on all states that result from the **Step 4**; i.e. on states \( \{q_1\} \) and \( \{q_0, q_2\} \)

**Observe** that we have already evaluated \( \delta'(\{q_1\}, \sigma) \) for all \( \sigma \in \Sigma \) in **Step 2** and \( \delta'(\{q_0, q_2\}, \sigma) \) in **Step 3**

The process of defining \( \delta'(Q, \sigma) \) for relevant \( Q \in 2^K \) is hence **terminated**

All other states are **trap states**
Diagram of of M’

Here is the **Diagram** of the **Relevant Part** of M’

and here is its **short pattern diagram** version
Book Example

Here is the nondeterministic $M$ from book page 70

**Exercise**  Read the example and re-write it as an exercise stage by stage as we did in class - it means follow the previous example

**Diagram** of $M$
Book Example

STAGE 1

\[ E(q_0) = \{ q_0, q_1, q_2, q_3 \} \]
\[ E(q_1) = \{ q_1, q_3, q_2 \} \]
\[ E(q_2) = \{ q_2 \} \]
\[ E(q_3) = \{ q_3 \} \]
\[ E(q_4) = \{ q_3, q_4 \} \in F \]

\[ H^0 \text{ has } 2^7 = 32 \text{ states} \]

We compute \[ \delta^0 \text{ on relevant states only} \]

STAGE 2 evaluation are on page 72
Evaluate them independently of the book
Diagram of $M'$
Book Example

Some **book computations**

\[
\delta'(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}, \\
\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}, \\
\delta'(\{q_2, q_3, q_4\}, a) = E(q_4) = \{q_3, q_4\}, \\
\delta'(\{q_2, q_3, q_4\}, b) = E(q_4) = \{q_3, q_4\}. \\
\delta'(\{q_3, q_4\}, a) = E(q_4) = \{q_3, q_4\}, \\
\delta'(\{q_3, q_4\}, b) = \emptyset, \\
\delta'(\emptyset, a) = \delta'(\emptyset, b) = \emptyset.
\]

Book Diagram
NDFA and DFA Differences Revisited

Difference 1  Revisited
DFA transition function $\delta$ even if expressed as a relation $\delta \subseteq K \times \Sigma \times K$

**must be a function**, while the NDFA transition relation $\Delta$
$\Delta \subseteq K \times (\Sigma \cup \{e\}) \times K$

**may not be a function**

Difference 2  Revisited
DFA transition function $\delta$ **domain** is the set $K \times \Sigma$ while
It is obvious that the definition of $\delta'$ solves the **Difference 2**
Difference 1

Given a **non-function diagram** of $M$

Proposed **IDEA** of $f$ solving the **Difference 1** was to make the states of $M'$ as some **subsets** of the set of states of $M$ and put in this case

$$\delta'(\{q_0\}, b) = \{q_1, q_2, q_3\}$$
Exercise

Given the **diagram** of $M$

![Diagram of M](image)

Exercise

**Show** that the definition of $\delta'$

$$
\delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q \ (q, \sigma, p) \in \Delta \}
$$

does exactly what we have proposed, i.e. show that

$$
\delta'({q_0}, b) = \{q_1, q_2, q_3\}
$$
Proof of Equivalency Theorem

**Equivalency Theorem**

For any **nondeterministic** automaton

\[ M = (K, \Sigma, \Delta, s, F) \]

there is (we have given an algorithm for its construction) a **deterministic** automaton

\[ M' = (K', \Sigma, \delta = \Delta', s', F') \]

such that

\[ M \approx M' \quad \text{i.e.} \quad L(M) = L(M') \]

**Proof**

\( M' \) is deterministic directly from the definition because the formula

\[ \delta'(Q, \sigma) = \bigcup_{p \in K} \{ E(p) : \exists q \in Q \ (q, \sigma, p) \in \Delta \} \]

defines a function and is well defined for all \( Q \in 2^K \) and \( \sigma \in \Sigma \).
Proof of Equivalency Theorem

We now claim that the following Lemma holds and we will prove equivalency $M \approx M'$ from the Lemma

**Lemma**

For any word $w \in \Sigma^*$ and any states $p, q \in K$

$$(q, w) \vdash_{M^*} (p, e) \quad \text{if and only if} \quad (E(q), w) \vdash_{M^*} (P, e)$$

for some set $P$ such that $p \in P$

We carry the proof of the Lemma by induction on the length $|w|$ of $w$

**Base Step** $|w| = 0$; this is possible only when $t \ w = e$ and we must show

$$(q, e) \vdash_{M^*} (p, e) \quad \text{if and only if} \quad (E(q), e) \vdash_{M^*} (P, e)$$

for some $P$ such that $p \in P$
Proof of Lemma

**Base Step** We must show that

\[(q, e) \vdash_{M^*} (p, e) \text{ if and only if } \exists P (p \in P \cap (E(q), e) \vdash_{M'}^* (P, e)))\]

**Observe** that \((q, e) \vdash_{M^*} (p, e)\) just says that \(p \in E(q)\) and the right side of statement holds for \(P = E(q)\)

Since \(M'\) is deterministic the statement

\[\exists P (p \in P \cap (E(q), e) \vdash_{M'}^* (P, e)))\]

is equivalent to saying that \(P = E(q)\) and since \(p \in P\) we get \(p \in E(q)\) what is equivalent to the left side

This completes the proof of the basic step

Inductive step is similar and is given as in the book page 71
Proof of The Theorem

We have just proved that for any $w \in \Sigma^*$ and any states $p, q \in K$

$$(q, w) \vdash_M^* (p, e) \quad \text{if and only if} \quad (E(q), w) \vdash_{M'}^* (P, e)$$

for some set $P$ such that $p \in P$

The proof of the Equivalency Theorem continues now as follows
Proof of The Theorem

We have to prove that \( L(M) = L(M') \)

Let’s take a word \( w \in \Sigma^* \)

We have (by definition of \( L(M) \)) that \( w \in L(M) \)
if and only if \((s, w) \vdash_M^* (f, e) \) for \( f \in F \)

if and only if \((E(s), w) \vdash_M^* (Q, e) \) for some \( Q \) such that \( f \in Q \)
(by the Lemma)

if and only if \((s', w) \vdash_M^* (Q, e) \) for some \( Q \in F \) (by definition of \( M' \))

if and only if \( w \in L(M') \)

Hence \( L(M) = L(M') \)

This end the proof of the Equivalency Theorem
Finite Automata

We have proved that the class *(CD)* and book *(BD)* definitions of a nondeterministic automaton are equivalent.

Hence by the **Equivalency Theorem** deterministic and nondeterministic automata defined by any of the both ways are equivalent.

We will use now a name

**FINITE AUTOMATA**

when we talk about **deterministic** or **nondeterministic** automata.