

Theory of Computer Science

B1. Finite Automata

Gabriele Röger

University of Basel

March 8, 2021

Theory of Computer Science

March 8, 2021 — B1. Finite Automata

B1.1 Introduction

B1.2 Alphabets and Formal Languages

B1.3 DFAs

B1.4 NFAs

B1.5 DFAs vs. NFAs

B1.6 Summary

B1.1 Introduction

Course Contents

Parts of the course:

A. background

▷ mathematical foundations and proof techniques

B. automata theory and formal languages
(Automatentheorie und formale Sprachen)

▷ What is a computation?

C. Turing computability (Turing-Berechenbarkeit)

▷ What can be computed at all?

D. complexity theory (Komplexitätstheorie)

▷ What can be computed efficiently?

E. more computability theory (mehr Berechenbarkeitstheorie)

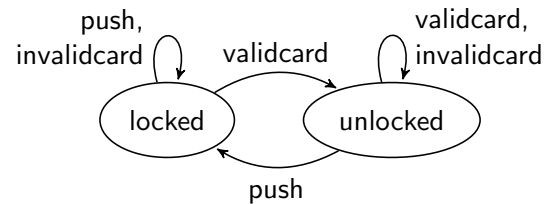
▷ Other models of computability

A Controller for a Turnstile



CC BY-SA 3.0, author: Stolbovsky

- ▶ simple access control
- ▶ card reader and push sensor
- ▶ card can either be valid or invalid



- ▶ Finite automata are a good model for computers with very limited memory.
Where can the turnstile controller store information about what it has seen in the past?
- ▶ We will not consider automata that run forever but that process a **finite input sequence** and then classify it as **accepted** or not.
- ▶ Before we get into the details, we need some background on **formal languages** to formalize what is a valid input sequence.

B1.2 Alphabets and Formal Languages

Alphabets and Formal Languages

Definition (Alphabets, Words and Formal Languages)

An **alphabet** Σ is a finite non-empty set of **symbols**.

A **word over** Σ is a finite sequence of elements from Σ .

The **empty word** (the empty sequence of elements) is denoted by ε .

Σ^* denotes the set of all words over Σ .

Σ^+ ($= \Sigma^* \setminus \{\varepsilon\}$) denotes the set of all non-empty words over Σ .

We write $|w|$ for the **length** of a word w .

A **formal language** (over alphabet Σ) is a subset of Σ^* .

German: Alphabet, Zeichen/Symbole, leeres Wort, formale Sprache

Example

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$$

$$|aba| = 3, |b| = 1, |\varepsilon| = 0$$

Languages: Examples

Example (Languages over $\Sigma = \{a, b\}$)

- ▶ $S_1 = \{a, aa, aaa, aaaa, \dots\} = \{a\}^+$
- ▶ $S_2 = \Sigma^*$
- ▶ $S_3 = \{a^n b^n \mid n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \dots\}$
- ▶ $S_4 = \{\varepsilon\}$
- ▶ $S_5 = \emptyset$
- ▶ $S_6 = \{w \in \Sigma^* \mid w \text{ contains twice as many } a\text{'s as } b\text{'s}\}$
 $= \{\varepsilon, aab, aba, baa, \dots\}$
- ▶ $S_7 = \{w \in \Sigma^* \mid |w| = 3\}$
 $= \{aaa, aab, aba, baa, bba, bab, abb, bbb\}$

Exercise (slido)

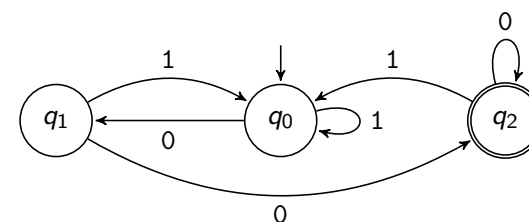
Consider $\Sigma = \{\text{push, validcard}\}$.

What is $|\text{pushvalidcard}|$?



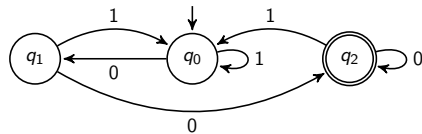
B1.3 DFAs

Finite Automaton: Example



When reading the input 01100 the automaton visits the states $q_0, q_1, q_0, q_0, q_1, q_2$.

Finite Automata: Terminology and Notation



- ▶ states $Q = \{q_0, q_1, q_2\}$
 - ▶ input alphabet $\Sigma = \{0, 1\}$
 - ▶ transition function δ
 - ▶ start state q_0
 - ▶ accept states $\{q_2\}$
- $$\delta(q_0, 0) = q_1$$
- $$\delta(q_0, 1) = q_0$$
- $$\delta(q_1, 0) = q_2$$
- $$\delta(q_1, 1) = q_0$$
- $$\delta(q_2, 0) = q_2$$
- $$\delta(q_2, 1) = q_0$$

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

table form of δ

Deterministic Finite Automaton: Definition

Definition (Deterministic Finite Automata)

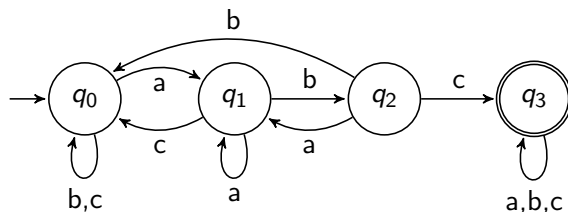
A **deterministic finite automaton (DFA)** is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where

- ▶ Q is the finite set of **states**
- ▶ Σ is the **input alphabet**
- ▶ $\delta : Q \times \Sigma \rightarrow Q$ is the **transition function**
- ▶ $q_0 \in Q$ is the **start state**
- ▶ $F \subseteq Q$ is the set of **accept states** (or **final states**)

German: deterministischer endlicher Automat, Zustände, Eingabealphabet, Überführungs-/Übergangsfunktion, Startzustand, Endzustände

Exercise

Give a formal definition of the following DFA (for the transition function, only exemplarily specify the transitions for state q_0):



DFA: Accepted Words

Intuitively, a DFA **accepts a word** if its computation terminates in an **accept state**.

Definition (Words Accepted by a DFA)

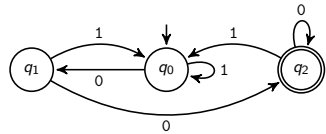
DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ **accepts the word** $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

- 1 $q'_0 = q_0$,
- 2 $\delta(q'_{i-1}, a_i) = q'_i$ for all $i \in \{1, \dots, n\}$ and
- 3 $q'_n \in F$.

German: DFA akzeptiert das Wort

Example

Example



accepts:

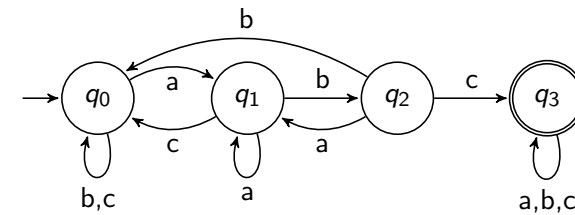
00
10010100
01000

does not accept:

ϵ
1001010
010001

Exercise (slido)

Consider again the following DFA:



Which of the following words does it accept?

- ▶ abc
- ▶ ababcb
- ▶ babbc



DFA: Recognized Language

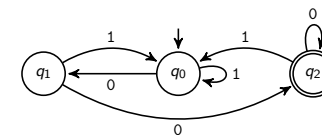
Definition (Language Recognized by a DFA)

Let M be a deterministic finite automaton.

The **language recognized by M** is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}.$$

Example



The DFA recognizes the language
 $\{w \in \{0, 1\}^* \mid w \text{ ends with } 00\}$.

Exercise

Specify a DFA with input alphabet $\Sigma = \{0, 1\}$
that recognizes the following language:

$L = \{w \in \{0, 1\}^* \mid \text{every } 0 \text{ in } w \text{ is directly followed by a } 1\}$

E.g. $001 \notin L$, $11 \in L$, $101 \in L$



A Note on Terminology

- ▶ In the literature, “accept” and “recognize” are sometimes used synonymously or the other way around.
DFA recognizes a word or accepts a language.
- ▶ We try to stay consistent using the previous definitions (following the text book by Sipser).

B1.4 NFAs

Nondeterministic Finite Automata

Why are DFAs called
deterministic automata? What are
nondeterministic automata,
then?

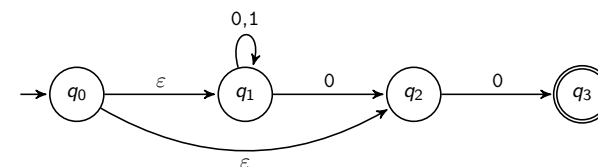


Picture courtesy of stockimages / FreeDigitalPhotos.net

In what Sense is a DFA Deterministic?

- ▶ A DFA has a single fixed state from which the computation starts.
- ▶ When a DFA is in a specific state and reads an input symbol, we know what the next state will be.
- ▶ For a given input, the entire computation is determined.
- ▶ This is a **deterministic** computation.

Nondeterministic Finite Automata: Example



differences to DFAs:

- ▶ transition function δ can lead to **zero** or **more** successor states for the **same** $a \in \Sigma$
- ▶ **ϵ -transitions** can be taken without “consuming” a symbol from the input
- ▶ the automaton accepts a word if there is **at least one** accepting sequence of states

Nondeterministic Finite Automaton: Definition

Definition (Nondeterministic Finite Automata)

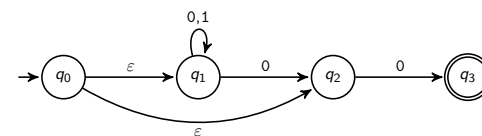
A **nondeterministic finite automaton (NFA)** is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where

- ▶ Q is the finite set of **states**
- ▶ Σ is the **input alphabet**
- ▶ $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ is the transition function (mapping to the **power set** of Q)
- ▶ $q_0 \in Q$ is the **start state**
- ▶ $F \subseteq Q$ is the set of **accept states**

German: nichtdeterministischer endlicher Automat

DFAs are (essentially) a special case of NFAs.

Accepting Computation: Example



$w = 0100$

\rightsquigarrow computation tree on blackboard

ϵ -closure of a State

For a state $q \in Q$, we write $E(q)$ to denote the set of states that are reachable from q via ϵ -transitions in δ .

Definition (ϵ -closure)

For NFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ and state $q \in Q$, state p is in the ϵ -closure $E(q)$ of q iff there is a sequence of states q'_0, \dots, q'_n with

- 1 $q'_0 = q$,
- 2 $q'_i \in \delta(q'_{i-1}, \epsilon)$ for all $i \in \{1, \dots, n\}$ and
- 3 $q'_n = p$.

$q \in E(q)$ for every state q

NFA: Accepted Words

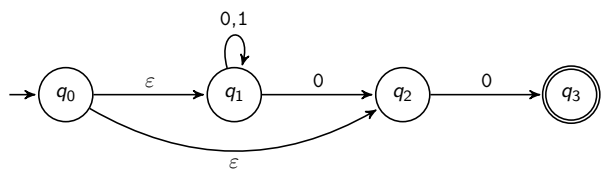
Definition (Words Accepted by an NFA)

NFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ **accepts the word** $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

- 1 $q'_0 \in E(q_0)$,
- 2 $q'_i \in \bigcup_{q \in \delta(q'_{i-1}, a_i)} E(q)$ for all $i \in \{1, \dots, n\}$ and
- 3 $q'_n \in F$.

Example: Accepted Words

Example



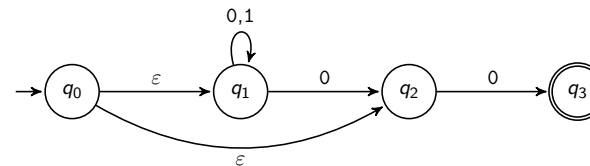
accepts:

0
10010100
01000

does not accept:

ϵ
1001010
010001

Exercise (slido)



Does this NFA accept input 01010?



NFA: Recognized Language

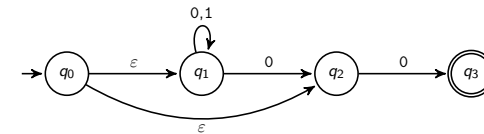
Definition (Language Recognized by an NFA)

Let M be an NFA with input alphabet Σ .

The **language recognized by M** is defined as
 $\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$.

Example: Recognized Language

Example



The NFA recognizes the language
 $\{w \in \{0, 1\}^* \mid w = 0 \text{ or } w \text{ ends with } 00\}$.

B1.5 DFAs vs. NFAs

DFAs are No More Powerful than NFAs

Observation

Every language recognized by a DFA is also recognized by an NFA.

We can transform a DFA into an NFA by replacing every transition
 $\delta(q, a) = q'$ with $\delta(q, a) = \{q'\}$.

Question



DFAs are
no more powerful than NFAs.
But are there languages
that can be recognized
by an NFA but not by a DFA?

Picture courtesy of imagerymajestic / FreeDigitalPhotos.net

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language recognized by an NFA is also recognized by a DFA.

The proof of the theorem is constructive and shows how we can convert an NFA to an equivalent DFA. Let's first have a look at the idea by means of an example (on the blackboard).

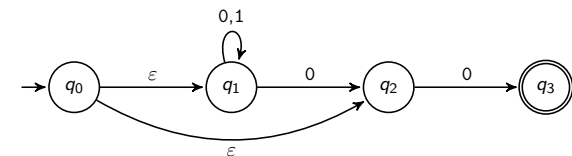
NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language recognized by an NFA is also recognized by a DFA.

The proof of the theorem is constructive and shows how we can convert an NFA to an equivalent DFA. Let's first have a look at the idea by means of an example (on the blackboard).

Conversion of an NFA to an Equivalent DFA: Example



NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language recognized by an NFA is also recognized by a DFA.

Proof.

For every NFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ we can construct a DFA $M' = \langle Q', \Sigma, \delta', q'_0, F' \rangle$ with $\mathcal{L}(M) = \mathcal{L}(M')$.

Here M' is defined as follows:

- ▶ $Q' := \mathcal{P}(Q)$ (the power set of Q)
- ▶ $q'_0 := E(q_0)$
- ▶ $F' := \{Q \subseteq Q \mid Q \cap F \neq \emptyset\}$
- ▶ For all $Q \in Q'$: $\delta'(Q, a) := \bigcup_{q \in Q} \bigcup_{q' \in \delta(q, a)} E(q')$

...

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language recognized by an NFA is also recognized by a DFA.

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$

iff there is a sequence of states p_0, p_1, \dots, p_n with

$p_0 \in E(q_0), p_n \in F$ and

$p_i \in \bigcup_{q \in \delta(p_{i-1}, a_i)} E(q)$ for all $i \in \{1, \dots, n\}$

iff there is a sequence of subsets Q_0, Q_1, \dots, Q_n with

$Q_0 = q'_0, Q_n \in F'$ and $\delta'(Q_{i-1}, a_i) = Q_i$ for all $i \in \{1, \dots, n\}$
iff $w \in \mathcal{L}(M')$ □

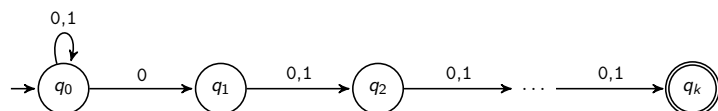
NFAs are More Compact than DFAs

Example

For $k \geq 1$ consider the language

$L_k = \{w \in \{0, 1\}^* \mid |w| \geq k \text{ and the } k\text{-th last symbol of } w \text{ is } 0\}$.

The language L_k can be accepted by an NFA with $k + 1$ states:



There is no DFA with less than 2^k states that accepts L_k
(without proof).

NFAs can often represent languages more compactly than DFAs.

B1.6 Summary

Summary

- ▶ **DFAs** are automata where **every state transition is uniquely determined**.
- ▶ **NFAs** can have zero, one or more transitions for a given state and input symbol.
- ▶ **NFAs** can have ϵ -transitions that can be taken without reading a symbol from the input.
- ▶ **NFAs** accept a word if there is **at least one accepting sequence of states**.
- ▶ **DFAs** and **NFAs** accept the same languages.