Automata & Complexity

Jörg Endrullis

Vrije Universiteit Amsterdam

2018

Information

Lecturer:

- **Jörg Endrullis**
- room: T437
- email: `j.endrullis@vu.nl`

Teaching assistant:

- **Christopher Esterhuyse**
- **Johannes van der Meer**
- **Petar Vukmirovic**

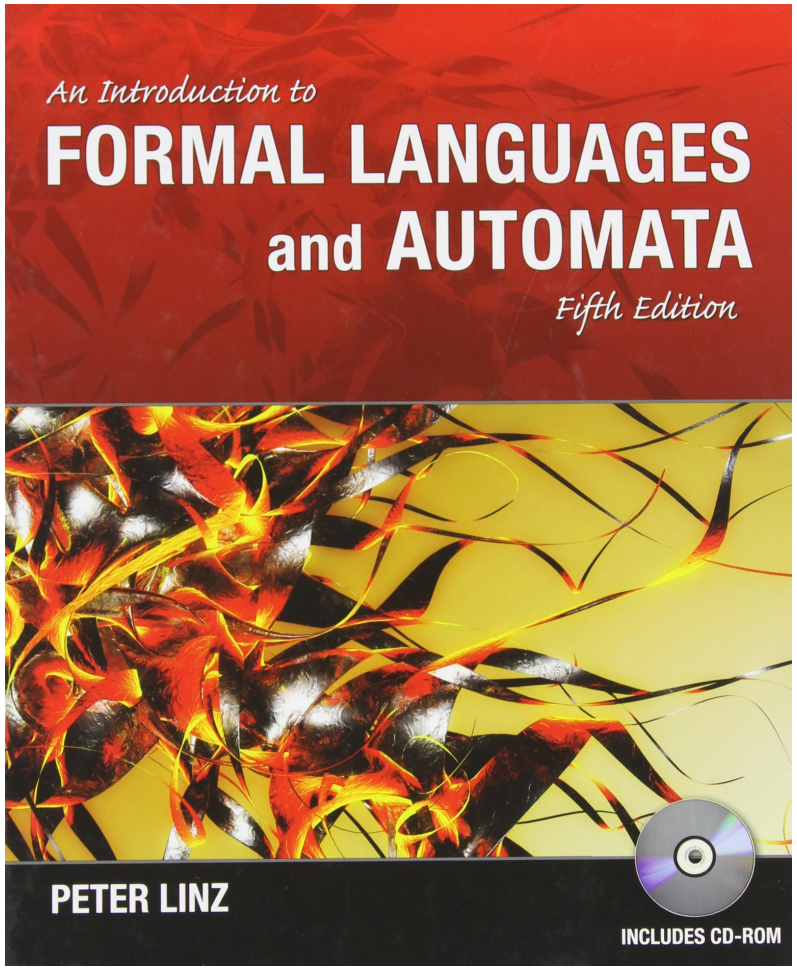
Course information:

- all relevant course information is on Canvas
- link to the **slides** on Canvas

Course Structure

- **2 lectures per week** (Tuesday and Wednesday)
- **2 exercise classes per week** (Wednesday and Thursday)
- **Homework**:
  - 80% of the homework in an online system
    (you will receive an email with activation link today)
  - 20% of the homework written assignments
    (one written assignment at the end of the course)
  - **50% of homework points to qualify for the exam**
  - **90% of homework points for 0.5 points bonus**
- **Written exam**

Material



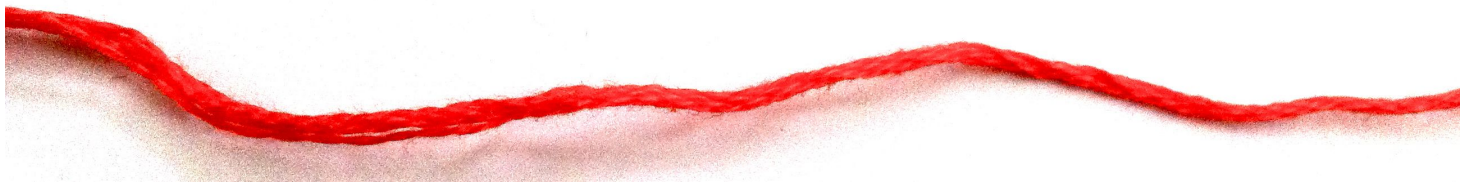
Peter Linz

An Introduction to Formal Languages and Automata (5th edition)

Jones and Bartlett

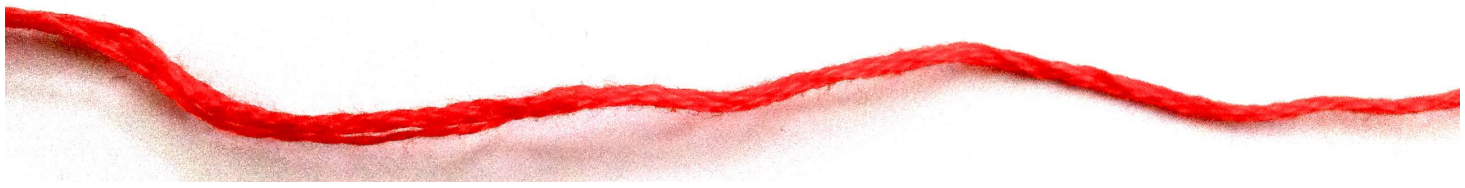
Course Motivation

Computer is based on a **universal computation mechanism**.



Course Motivation

Computer is based on a **universal computation mechanism**.



Different applications $\Longrightarrow$ different formalisms:

- **Pattern recognition**: regular languages
- **Parsing**: context-free languages, grammars
- **Computation**: Turing machines

Course Motivation

What can a computer do?

Course Motivation

What can a computer do?

Some (at first glance simple) problems are undecidable.

For example:

- program termination,
- Post correspondence problem,
- validity in predicate logic.

Course Motivation

What can a computer do?

Some (at first glance simple) problems are undecidable.

For example:

- program termination,
- Post correspondence problem,
- validity in predicate logic.

Some problems (NP-complete problems) are (probably) not efficiently solvable by a computer.

For example:

- travelling salesman problem,
- satisfiability in propositional logic.

Course Motivation

Typical questions that we will answer

- What is a (programming) **language**?
- How can languages be recognised by computers? (**automata**, **regular expressions**, **grammars**)
- What problems can be solved by what **types of automata**?
- **How much time/memory** is needed for solving a problem?

Course Motivation

Typical questions that we will answer

- What is a (programming) **language**?
- How can languages be recognised by computers? (**automata**, **regular expressions**, **grammars**)
- What problems can be solved by what **types of automata**?
- **How much time/memory** is needed for solving a problem?

Aspects of languages:

- **syntax**: the form of the words in the language
- **semantics**: the meaning of the words in the language

We will focus on the syntax.

Words

Words

Word = finite sequence of **symbols** from a finite **alphabet** Σ.

- notation for symbols: $a, b, c, \ldots$
- symbol from the alphabet: $a \in \Sigma$
- notation for words: u, v, w, x, y, z
- **empty word:** λ

Words

Words

Word = finite sequence of **symbols** from a finite **alphabet** Σ.

- notation for symbols: $a, b, c, \ldots$
- symbol from the alphabet: $a \in \Sigma$
- notation for words: u, v, w, x, y, z
- **empty word:** λ

Note that, in programming, words are often called **strings**.

Words

Words

Word = finite sequence of **symbols** from a finite **alphabet** Σ.

- notation for symbols: $a, b, c, \ldots$
- symbol from the alphabet: $a \in \Sigma$
- notation for words: u, v, w, x, y, z
- **empty word:** λ

Note that, in programming, words are often called **strings**.

A computer program:

- is itself a **word** (a finite sequence of bits/bytes)
- takes **input word**
- produces **output word**

Operations on Words (1)

Concatenation

If $v = a_1 \cdots a_n$ and $w = b_1 \cdots b_m$, then

$$vw = a_1 \cdots a_n b_1 \cdots b_m$$

Operations on Words (1)

Concatenation

If $v = a_1 \cdots a_n$ and $w = b_1 \cdots b_m$, then

$$vw = a_1 \cdots a_n b_1 \cdots b_m$$

Length

If $v = a_1 \cdots a_n$, then $|v| = n$.

The length can be defined inductively:

$$|\lambda| = 0$$
$$|va| = |v| + 1$$

Operations on Words (2)

Power

The power v^k consists of k concatenations of v's:

$$v^0 = \lambda$$

$$v^{k+1} = v^k v$$

Operations on Words (2)

Power

The power v^k consists of k concatenations of v's:

$$v^0 = \lambda$$
$$v^{k+1} = v^k v$$

Reverse

The reverse of $a_1 \cdots a_n$ is

$$(a_1 \cdots a_n)^R = a_n \cdots a_1$$

The reverse can be inductively defined

$$\lambda^R = \lambda$$
$$(va)^R = a(v^R)$$

Formal Languages

Formal language = **set** of **words**

Formal Languages

Formal language = **set** of **words**

Example

All parseable C programs form a language.

Formal Languages

Formal language = **set** of **words**

Example

All parseable C programs form a language.

More precisely:

- Σ^* is the set of all words over Σ.

Formal language

A **(formal) language** L is a subset of Σ^*, that is, $L \subseteq \Sigma^*$.

Examples of Formal Languages

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

Examples of Formal Languages

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

$\{ab,\ aab,\ bbaaabb\}$ is a finite language

Examples of Formal Languages

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

$\{\, ab,\ aab,\ bbaaabb \,\}$ is a finite language

$\{\, ab^n a \mid n \geq 1 \,\}$ is an infinite language:

$$\{\, aba,\ abba,\ abbba,\ abbbba,\ \ldots \,\}$$

Examples of Formal Languages

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

$\{ ab,\ aab,\ bbaaabb \}$ is a finite language

$\{ ab^n a \mid n \geq 1 \}$ is an infinite language:

$$\{ aba,\ abba,\ abbba,\ abbbba,\ \ldots \}$$

$\{a^n b^n \mid n \geq 0\}$ is an infinite language:

$$\{ \lambda,\ ab,\ aabb,\ aaabbb,\ aaaabbbb,\ \ldots \}$$

Operations on Languages (1)

A language is a **set** of words.

So the usual set operations have meaning for languages:

$$\in,\ \subseteq,\ \cap,\ \cup,\ \setminus,\ \ldots$$

Operations on Languages (1)

A language is a **set** of words.

So the usual set operations have meaning for languages:

$$\in,\ \subseteq,\ \cap,\ \cup,\ \setminus,\ \ldots$$

Complement

The complement $\overline{L}$ = all words that are not in the language L:

$$\overline{L} = \Sigma^* \setminus L$$

Operations on Languages (1)

A language is a **set** of words.

So the usual set operations have meaning for languages:

$$\in,\ \subseteq,\ \cap,\ \cup,\ \setminus,\ \ldots$$

Complement

The complement $\overline{L}$ = all words that are not in the language L:

$$\overline{L} = \Sigma^* \setminus L$$

Concatenation

The concatenation of languages L_1 and L_2 is defined as

$$L_1 L_2 = \{\, xy \mid x \in L_1 \wedge y \in L_2 \,\}$$

Operations on Languages (1)

A language is a **set** of words.

So the usual set operations have meaning for languages:

$$\in,\ \subseteq,\ \cap,\ \cup,\ \setminus,\ \ldots$$

Complement

The complement $\overline{L}$ = all words that are not in the language L:

$$\overline{L} = \Sigma^* \setminus L$$

Concatenation

The concatenation of languages L_1 and L_2 is defined as

$$L_1 L_2 = \{ xy \mid x \in L_1 \wedge y \in L_2 \}$$

Reverse

The reverse of a language L is

$$L^R = \{ x^R \mid x \in L \}$$

Operations on Languages (2)

Power

The n-th power of a language L is defined by induction on n:

$$L^0 = \{\lambda\}$$
$$L^{n+1} = L^n L \qquad (n \geq 0)$$

Operations on Languages (2)

Power

The n-th power of a language L is defined by induction on n:

$$L^0 = \{\lambda\}$$
$$L^{n+1} = L^n L \qquad (n \geq 0)$$

Attention: $L^2 = \{uv \mid u, v \in L\} \neq \{uu \mid u \in L\}$

Operations on Languages (2)

Power

The n-th power of a language L is defined by induction on n:

$$L^0 = \{\lambda\}$$
$$L^{n+1} = L^n L \qquad (n \geq 0)$$

Attention: $L^2 = \{uv \mid u, v \in L\} \neq \{uu \mid u \in L\}$

Kleene star

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$
$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup \cdots$$

Thus $L^* = L^+ \cup \{\lambda\}$.

Exercise

Let

- $\Sigma = \{a, b\}$
- $L_1 = \{a,\ bb\}$
- $L_2 = \{ab,\ ba\}$

(In groups of two, 2 minutes)

Describe the following languages as sets:

- $L_1 L_2 =$
- $L_1^2 =$
- $L_1^3 =$

Exercise

Let

- $\Sigma = \{a, b\}$
- $L_1 = \{a,\ bb\}$
- $L_2 = \{ab,\ ba\}$

(In groups of two, 2 minutes)

Describe the following languages as sets:

- $L_1 L_2 = \{aab,\ aba,\ bbab,\ bbba\}$
- $L_1^2 =$
- $L_1^3 =$

Exercise

Let

- $\Sigma = \{a, b\}$
- $L_1 = \{a,\ bb\}$
- $L_2 = \{ab,\ ba\}$

(In groups of two, 2 minutes)

Describe the following languages as sets:

- $L_1 L_2 = \{aab,\ aba,\ bbab,\ bbba\}$
- $L_1^2 = \{aa,\ abb,\ bba,\ bbbb\}$
- $L_1^3 =$

Exercise

Let

- $\Sigma = \{a, b\}$
- $L_1 = \{a,\ bb\}$
- $L_2 = \{ab,\ ba\}$

(In groups of two, 2 minutes)

Describe the following languages as sets:

- $L_1 L_2 = \{aab,\ aba,\ bbab,\ bbba\}$
- $L_1^2 = \{aa,\ abb,\ bba,\ bbbb\}$
- $L_1^3 =$ answered in the lecture

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R =$
- $\overline{L} =$
- $\overline{L}^R =$
- $\overline{L^R} =$

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R = \{ b^n a \mid n \geq 0 \}$
- $\overline{L} =$
- $\overline{L}^R =$
- $\overline{L^R} =$

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R = \{ b^n a \mid n \geq 0 \}$
- $\overline{L} = \{ \lambda \} \cup \{ bw \mid w \in \Sigma^* \} \cup \{ awau \mid w, u \in \Sigma^* \}$
- $\overline{L}^R =$
- $\overline{L^R} =$

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R = \{ b^n a \mid n \geq 0 \}$
- $\overline{L} = \{ \lambda \} \cup \{ bw \mid w \in \Sigma^* \} \cup \{ awau \mid w, u \in \Sigma^* \}$
- $\overline{L}^R =$ answered in the lecture
- $\overline{L^R} =$

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R = \{ b^n a \mid n \geq 0 \}$
- $\overline{L} = \{ \lambda \} \cup \{ bw \mid w \in \Sigma^* \} \cup \{ awau \mid w, u \in \Sigma^* \}$
- $\overline{L}^R =$ answered in the lecture
- $\overline{L^R} =$ answered in the lecture

Exercise

Let

- $\Sigma = \{ a, b \}$
- $L = \{ ab^n \mid n \geq 0 \}$

(In groups of two, 3 minutes)

Describe the following languages as sets:

- $L^R = \{ b^n a \mid n \geq 0 \}$
- $\overline{L} = \{ \lambda \} \cup \{ bw \mid w \in \Sigma^* \} \cup \{ awau \mid w, u \in \Sigma^* \}$
- $\overline{L}^R =$ answered in the lecture
- $\overline{L^R} =$ answered in the lecture

Conclusion

Sets are not ideal to describe languages.

Deterministic Finite Automata

A **deterministic finite automaton**, short **DFA**, consists of:

- a finite set Q of states
- a finite input alphabet Σ
- a transition function $\delta : Q \times \Sigma \to Q$
- a starting state $q_0 \in Q$
- a set $F \subseteq Q$ of final states

Understanding the transition function $\delta : Q \times \Sigma \to Q$

If the automaton in state q reads the symbol a, then the resulting state is $\delta(q, a)$.

Deterministic Finite Automata

A **deterministic finite automaton**, short **DFA**, consists of:

- a finite set Q of states
- a finite input alphabet Σ
- a transition function $\delta : Q \times \Sigma \to Q$
- a starting state $q_0 \in Q$
- a set $F \subseteq Q$ of final states

Understanding the transition function $\delta : Q \times \Sigma \to Q$

If the automaton in state q reads the symbol a, then the resulting state is $\delta(q, a)$.

Example DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_0, \lambda)$$

DFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We write

$$(q, aw) \vdash (q', w) \quad \text{if } \delta(q, a) = q'$$

Here, (q, w) indicates that M is in state q and reads word w.

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_0, \lambda)$$

We define $\vdash^*$ as the **reflexive transitive closure of** $\vdash$.

Continuing the above example, we have

$$(q_0, abba) \vdash^* (q_0, \lambda)$$

Transition Function in Table Notation

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Hint: transition function δ can be written in the form of a **table**:

δ	q_0	q_1
a	q_0	q_1
b	q_1	q_0

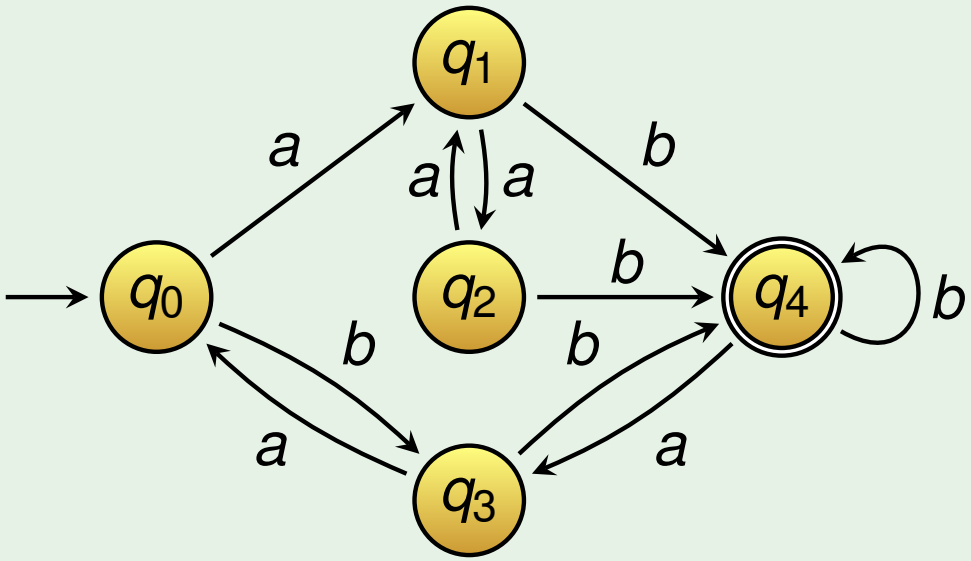
DFA's as Transition Graphs

A DFA can be visualised as a **transition graph**, consisting of:

- **states** are the **nodes** of the graph
- **arrows** with labels from Σ
- **starting state** indicated by an **extra incoming arrow**
- **final states** indicated by **double circle**

We have an arrow from q to q' with label a if $\delta(q, a) = q'$.

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



Exercise (1)

We define a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with

$$Q = \{q_0, q_1\} \qquad \Sigma = \{a, b\} \qquad F = \{q_0\}$$

and the transition function given by

δ	q_0	q_1
a	q_0	q_1
b	q_1	q_0

(In groups of two, 1 minute)

Visualise M as a transition graph.

Exercise (1)

We define a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with

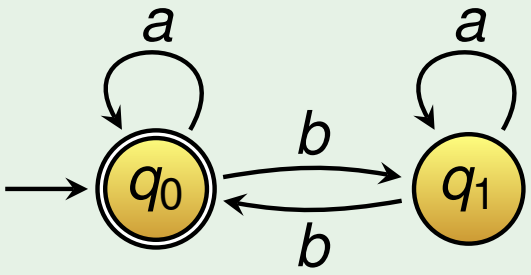
$$Q = \{q_0, q_1\} \qquad \Sigma = \{a, b\} \qquad F = \{q_0\}$$

and the transition function given by

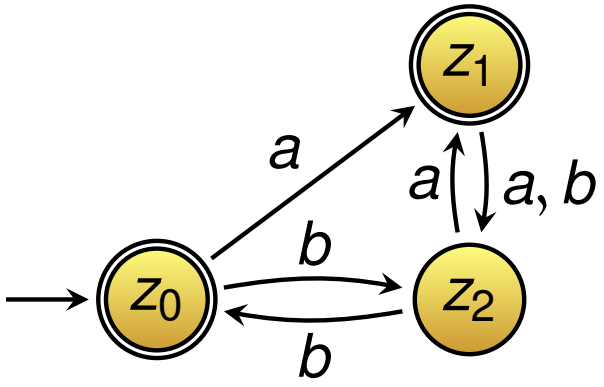
δ	q_0	q_1
a	q_0	q_1
b	q_1	q_0

(In groups of two, 1 minute)

Visualise M as a transition graph.



Exercise (2)



Note that the transition with label a, b is shorthand for two transitions: one with label a and one with label b.

(In groups of two, 1 minute)

Translate the graph to a deterministic finite automaton:

- states $Q = ?$
- alphabet $\Sigma = ?$
- transition function $\delta : Q \times \Sigma \to Q$ is given by

δ	z_0	z_1	z_2
a			
b			

- starting state ?
- final states $F = ?$

DFA's Reading Words continued...

If $(q, w) \vdash^* (q', \lambda)$, we also write

$$q \xrightarrow{w} q'$$

DFA's Reading Words continued...

If $(q, w) \vdash^* (q', \lambda)$, we also write

$$q \xrightarrow{w} q'$$

In other words,

$$q \xrightarrow{w} q'$$

means that there is a path from q to q' in the automaton with transitions labelled $a_1, \ldots, a_n$ such that $w = a_1 \cdots a_n$.

Regular Languages

A DFA defines (accepts) a language!

Regular Languages

A DFA defines (accepts) a language!

The **language accepted by** DFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$

$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

Regular Languages

A DFA defines (accepts) a language!

The **language accepted by** DFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_0, \lambda)$$

Regular Languages

A DFA defines (accepts) a language!

The **language accepted by** DFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_0, \lambda)$$

The word $abba$ is accepted by M, that is, $abba \in L(M)$.

Regular Languages

A DFA defines (accepts) a language!

The **language accepted by** DFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$,

$$\delta(q_0, a) = q_0 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_1 \qquad \delta(q_1, b) = q_0$$

Then we have

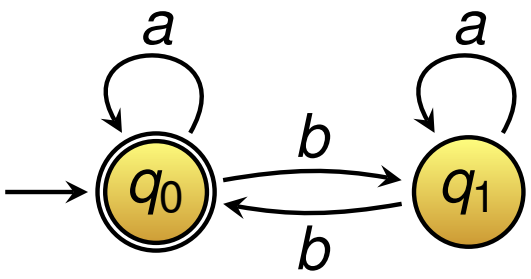
$$(q_0, abba) \vdash (q_0, bba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_0, \lambda)$$

The word $abba$ is accepted by M, that is, $abba \in L(M)$.

A language L is **regular** if there exists a DFA M with $L(M) = L$.

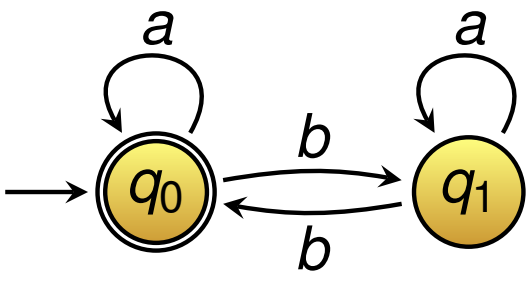
Exercise

Let M be the following DFA:



Exercise

Let M be the following DFA:

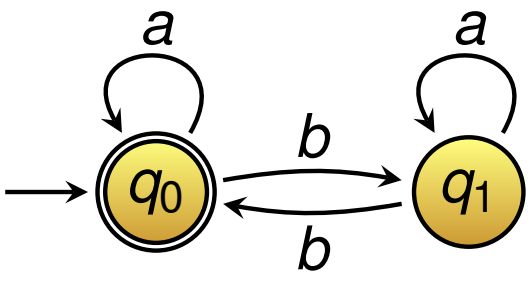


(In groups of two, 2 minutes)

What is the language accepted by M? (describe in words)

Exercise

Let M be the following DFA:



(In groups of two, 2 minutes)

What is the language accepted by M? (describe in words)

$L(M)$ consists of all words that contain an even number of b's.

Determinism

DFAs are deterministic

For every state $q \in Q$ and every symbol $a \in \Sigma$, the state q has **precisely one outgoing arrow** with label a.

Recall that δ is a function from $Q \times \Sigma$ to Q.

Determinism

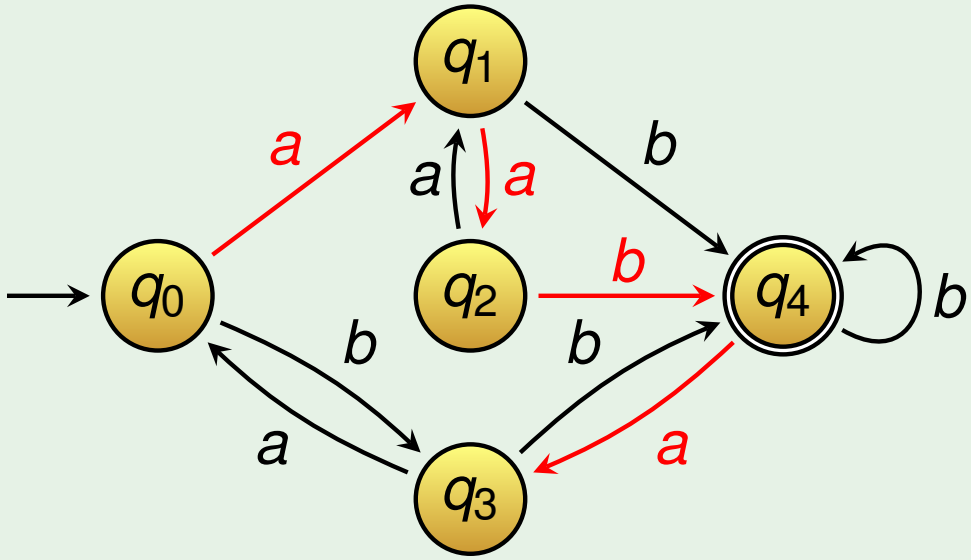
DFAs are deterministic

For every state $q \in Q$ and every symbol $a \in \Sigma$, the state q has **precisely one outgoing arrow** with label a.

Recall that δ is a function from $Q \times \Sigma$ to Q.

Hence, for every input word, there is precisely one path from the starting state through the transition graph.

The following picture shows the path for *aaba*:



Exercise (1)

(Individually, 10 seconds)

Show that the following language is regular.

Construct a deterministic finite automaton for the language:

$$\{\lambda\}$$

Exercise (2)

(Individually, 1 minutes)

Show that the following language is regular.

Construct a deterministic finite automaton for the language:

$$\{ a^n b \mid n \geq 0 \}$$

Exercise (3)

(Individually, 2 minutes)

Show that the following language is regular.

Construct a deterministic finite automaton for the language:

$$\{ a^{2n+1} \mid n \geq 0 \} \cup \{ b^{2n} \mid n \geq 0 \}$$

Exercise (4)

(Individually, 4 minutes)

Show that the following languages are regular.

Construct deterministic finite automata for the languages:

$$\{ w \in \{a, b\}^* \mid w \text{ contains the subword } bab \}$$

and

$$\{ w \in \{a, b\}^* \mid w \text{ does } \textbf{not} \text{ contain the subword } bab \}$$

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Then $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA with $L(N) = \overline{L}$.

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Then $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA with $L(N) = \overline{L}$.

Here it is important that for every input word w:

- There is precisely one path starting at q_0 labelled with w.

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Then $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA with $L(N) = \overline{L}$.

Here it is important that for every input word w:

- There is precisely one path starting at q_0 labelled with w.
- There is precisely one state q with $q_0 \xrightarrow{w} q$,

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Then $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA with $L(N) = \overline{L}$.

Here it is important that for every input word w:

- There is precisely one path starting at q_0 labelled with w.
- There is precisely one state q with $q_0 \xrightarrow{w} q$, and

$$w \in L \iff w \in L(M) \iff q \in F$$

Theorems about Regular Languages (1)

Theorem

If L is a regular language, then $\overline{L}$ is also regular.

Proof.

Let L be regular.

Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = L$.

Then $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA with $L(N) = \overline{L}$.

Here it is important that for every input word w:

- There is precisely one path starting at q_0 labelled with w.
- There is precisely one state q with $q_0 \xrightarrow{w} q$, and

$$w \in L \iff w \in L(M) \iff q \in F$$
$$w \in \overline{L} \iff w \in \overline{L(M)} \iff q \in (Q \setminus F) \iff w \in L(N)$$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q =$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) =$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$
- $q_0 =$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$
- $q_0 = (q_{0,1},\ q_{0,2})$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$
- $q_0 = (q_{0,1},\ q_{0,2})$
- $F =$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$
- $q_0 = (q_{0,1},\ q_{0,2})$
- $F = \{(q_1, q_2) \in Q \mid q_1 \in F_1 \text{ or } q_2 \in F_2\}$

Theorems about Regular Languages (2)

Theorem

If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Construction (Product)

There exists a DFAs

$$M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1) \qquad M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$$

such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

Idea: We run M_1 and M_2 in parallel.

We define a DFA $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2 \}$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a),\ \delta_2(q_2, a))$
- $q_0 = (q_{0,1},\ q_{0,2})$
- $F = \{(q_1, q_2) \in Q \mid q_1 \in F_1 \text{ or } q_2 \in F_2\}$

Then it follows that $L(N) = L(M_1) \cup L(M_2) = L_1 \cup L_2$.

Theorems about Regular Languages (3)

Question

Let L_1 and L_2 be regular languages.

How to change the product construction to show that

- $L_1 \cap L_2$ is regular, and
- $L_1 \setminus L_2$ is regular ?

Theorems about Regular Languages (3)

Question

Let L_1 and L_2 be regular languages.

How to change the product construction to show that

- $L_1 \cap L_2$ is regular, and
- $L_1 \setminus L_2$ is regular ?

Theorem

If L is regular, then L^R is also regular.

Theorems about Regular Languages (3)

Question

Let L_1 and L_2 be regular languages.

How to change the product construction to show that

- $L_1 \cap L_2$ is regular, and
- $L_1 \setminus L_2$ is regular ?

Theorem

If L is regular, then L^R is also regular.

This proof will be an exercise.

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

This language is not regular!

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

This language is not regular!

Intuition: a DFA has only a finite memory (the states).

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

This language is not regular!

Intuition: a DFA has only a finite memory (the states).

(Groups of two, 2 minutes)

Give a deterministic automaton with **infinitely** many states for

$$\{ a^n b^n \mid n \geq 0 \}$$

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

This language is not regular!

Intuition: a DFA has only a finite memory (the states).

(Groups of two, 2 minutes)

Give a deterministic automaton with **infinitely** many states for

$$\{ a^n b^n \mid n \geq 0 \}$$

Answer in the lecture.

Exercise

Is the following language regular?

$$\{ a^n b^n \mid n \geq 0 \}$$

This language is not regular!

Intuition: a DFA has only a finite memory (the states).

(Groups of two, 2 minutes)

Give a deterministic automaton with **infinitely** many states for

$$\{ a^n b^n \mid n \geq 0 \}$$

Answer in the lecture.

Question

Is every **finite** language regular ?

Nondeterministic Finite Automata

Nondeterministic finite automaton (NFA)

- A state can have **zero or more outgoing arrows** with the same label.
- Allows for **empty steps**: arrows with label λ that do not 'eat' a symbol from the input word.

Nondeterministic Finite Automata

Nondeterministic finite automaton (NFA)

- A state can have **zero or more outgoing arrows** with the same label.
- Allows for **empty steps**: arrows with label λ that do not 'eat' a symbol from the input word.

DFA's and NFA's are used everywhere:

- in software engineering,
- for modelling of hardware circuits,
- in compilers, and
- in network protocols.

Nondeterministic Finite Automata

NFA's are defined like DFA's, except for the transition function!

Nondeterministic Finite Automata

NFA's are defined like DFA's, except for the transition function!

A **nondeterministic finite automaton**, short **NFA**, consists of:

- a finite set Q of states
- a finite input alphabet Σ
- a transition function $\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$
- a starting state $q_0 \in Q$
- a set $F \subseteq Q$ of final states

Here 2^Q is the set of all subsets of Q:

$$2^Q = \{ S \mid S \subseteq Q \}$$

NFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA. We write

$$(q, \alpha w) \vdash (q', w) \quad \text{if } q' \in \delta(q, \alpha) \text{ with } \alpha \in \Sigma \cup \{\lambda\}$$

NFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA. We write

$$(q, \alpha w) \vdash (q', w) \quad \text{if } q' \in \delta(q, \alpha) \text{ with } \alpha \in \Sigma \cup \{\lambda\}$$

Note that if $\alpha = \lambda$, then

- the state changes (q to q'), but
- the input word stays the same ($\lambda w = w$).

NFA's Reading Words

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA. We write

$$(q, \alpha w) \vdash (q', w) \quad \text{if } q' \in \delta(q, \alpha) \text{ with } \alpha \in \Sigma \cup \{\lambda\}$$

Note that if $\alpha = \lambda$, then

- the state changes (q to q'), but
- the input word stays the same ($\lambda w = w$).

Again we write

$$q \xrightarrow{w} q'$$

whenever $(q, w) \vdash^* (q', \lambda)$.

It means that there is a path from q to q' with transitions labelled $\alpha_1, \ldots, \alpha_n \in (\Sigma \cup \{\lambda\})$ such that $w = \alpha_1 \cdots \alpha_n$.

NFA's Reading Words

The **language accepted by** NFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$

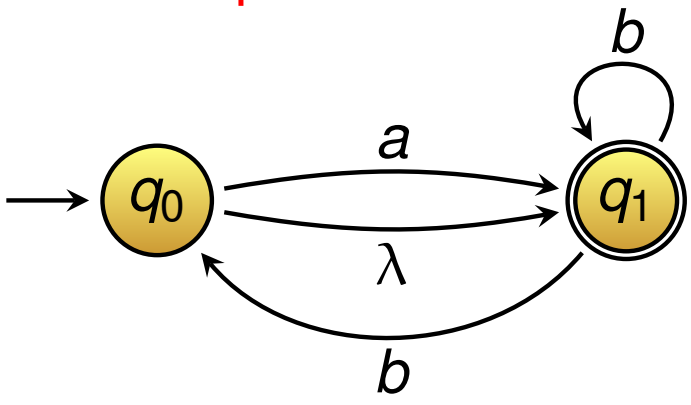
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

NFA's Reading Words

The **language accepted by** NFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

But now paths are not unique!



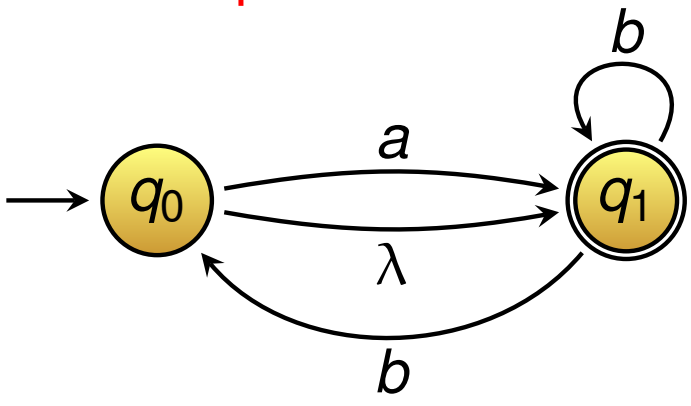
This automaton can accept the word *ab* in 2 different ways!

NFA's Reading Words

The **language accepted by** NFA $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \lambda) \text{ with } q \in F \}$$
$$= \{ w \in \Sigma^* \mid q_0 \xrightarrow{w} q \text{ with } q \in F \}$$

But now paths are not unique!



This automaton can accept the word *ab* in 2 different ways!

$$(q_0, ab) \vdash (q_1, b) \vdash (q_1, \lambda)$$
$$(q_0, ab) \vdash (q_1, b) \vdash (q_0, \lambda) \vdash (q_1, \lambda)$$

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

We construct a DFA $N_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ where

$$Q_D = 2^Q$$

$$\delta_D(X, a) =$$

$$q_{0D} =$$

$$F_D =$$

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

We construct a DFA $N_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ where

$$Q_D = 2^Q$$

$$\delta_D(X, a) = \{\, q' \in Q \mid (q, a) \vdash^* (q', \lambda) \text{ for some } q \in X \,\}$$

$$q_{0D} =$$

$$F_D =$$

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

We construct a DFA $N_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ where

$$
\begin{aligned}
Q_D &= 2^Q \\
\delta_D(X, a) &= \{\, q' \in Q \mid (q, a) \vdash^* (q', \lambda) \text{ for some } q \in X \,\} \\
q_{0D} &= \{\, q' \in Q \mid (q_0, \lambda) \vdash^* (q', \lambda) \,\} \\
F_D &=
\end{aligned}
$$

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

We construct a DFA $N_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ where

$$
\begin{aligned}
Q_D &= 2^Q \\
\delta_D(X, a) &= \{\, q' \in Q \mid (q, a) \vdash^* (q', \lambda) \text{ for some } q \in X \,\} \\
q_{0D} &= \{\, q' \in Q \mid (q_0, \lambda) \vdash^* (q', \lambda) \,\} \\
F_D &= \{\, X \subseteq Q \mid X \cap F \neq \emptyset \,\}
\end{aligned}
$$

DFA's and NFA's are Equally Expressive

Theorem

A language L is accepted by a NFA $\iff$ L is regular.

Construction (Powerset)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

Idea: state of DFA = set of all states the NFA can be in

We construct a DFA $N_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ where

$$
\begin{aligned}
Q_D &= 2^Q \\
\delta_D(X, a) &= \{\, q' \in Q \mid (q, a) \vdash^* (q', \lambda) \text{ for some } q \in X \,\} \\
q_{0D} &= \{\, q' \in Q \mid (q_0, \lambda) \vdash^* (q', \lambda) \,\} \\
F_D &= \{\, X \subseteq Q \mid X \cap F \neq \emptyset \,\}
\end{aligned}
$$

For every $w \in \Sigma^*$ and $q \in Q$ it holds that

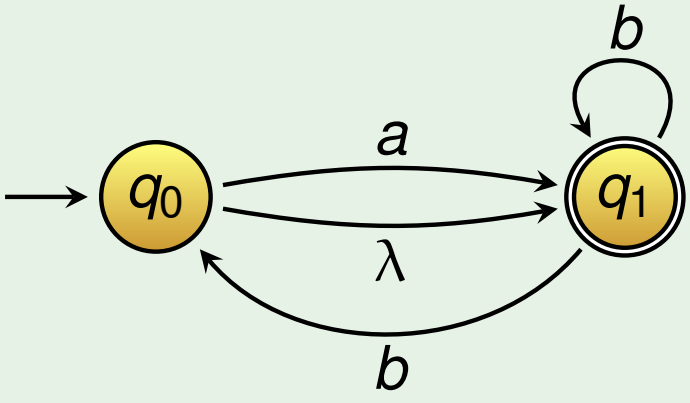
$$(q_0, w) \vdash^* (q, \lambda) \iff (q_{0D}, w) \vdash^* (X, \lambda) \text{ with } q \in X.$$

From this property it follows that $L(N_D) = L(M)$.

Exercise

(Groups of two, 2 minutes)

Given is the following NFA:



Construct a DFA that accepts the same language.

Looking Forward

Read:

- Linz 1.2–1.2, 2.1–2.3

Do the following exercises:

- Linz 1.2: 2, 4, 8, 10
- Linz 2.1: 1, 2d, 3, 7b, 9b,f, 11
- Linz 2.2: 12
- Linz 2.3: 2, 3, 6, 12

Following lecture:

- Alternative descriptions of regular languagse:
  - regular expressions
  - grammars