# Theory of Computer Science

## C1. Formal Languages and Grammars

Gabriele Röger

University of Basel

March 19, 2018

# Introduction

# Course Contents

Parts of the course:

A. background ✓
   - ▷ mathematical foundations and proof techniques

B. logic (Logik) ✓
   - ▷ How can knowledge be represented?
     How can reasoning be automated?

C. automata theory and formal languages (Automatentheorie und formale Sprachen)
   - ▷ What is a computation?

D. computability theory (Berechenbarkeitstheorie)
   - ▷ What can be computed at all?

E. complexity theory (Komplexitätstheorie)
   - ▷ What can be computed efficiently?

# Course Contents

Parts of the course:

A. background ✓
   - ▷ mathematical foundations and proof techniques

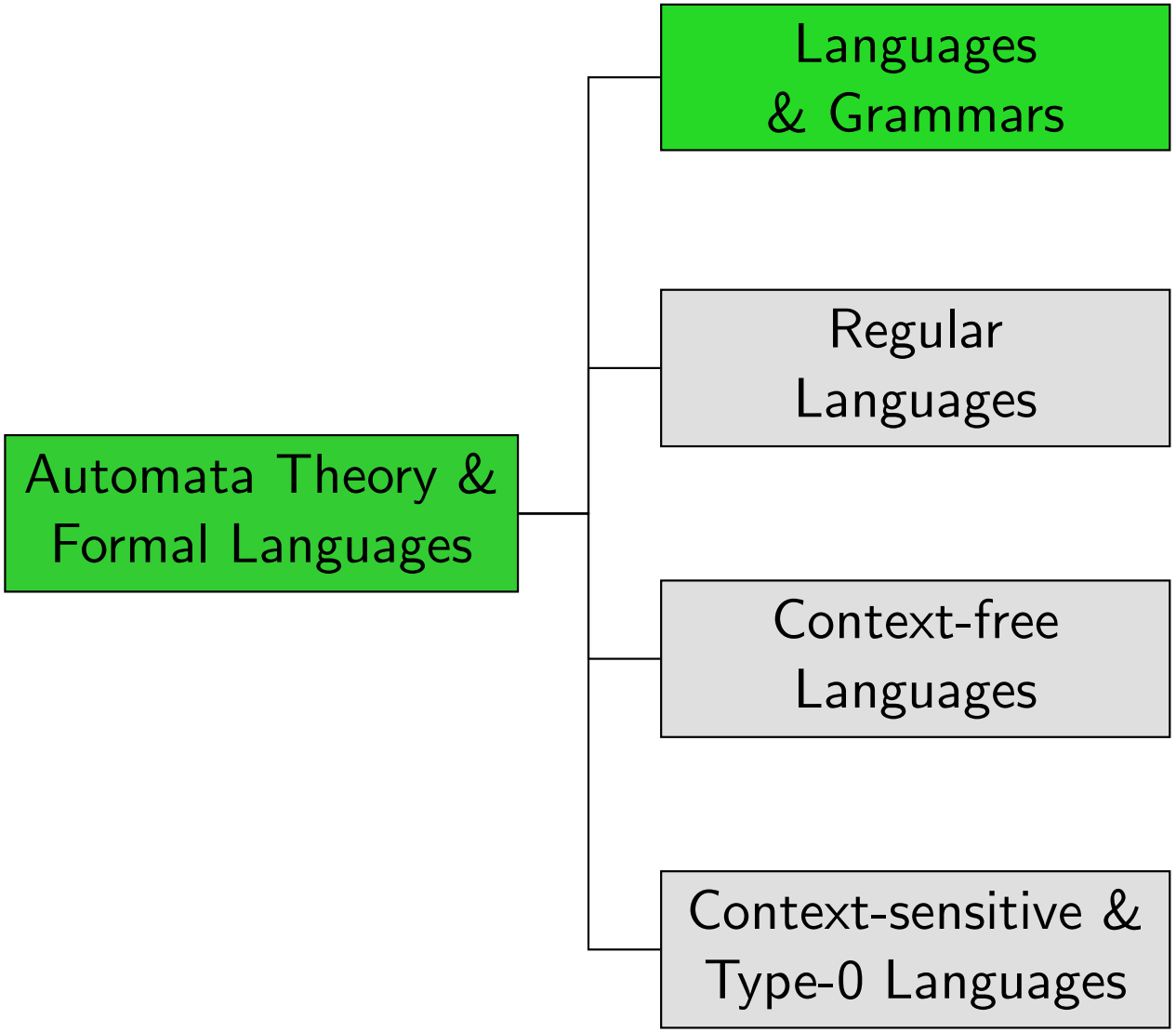
B. logic (Logik) ✓
   - ▷ How can knowledge be represented?
     How can reasoning be automated?

C. automata theory and formal languages (Automatentheorie und formale Sprachen)
   - ▷ What is a computation?

D. computability theory (Berechenbarkeitstheorie)
   - ▷ What can be computed at all?

E. complexity theory (Komplexitätstheorie)
   - ▷ What can be computed efficiently?

# Part C: Overview



# Example: Propositional Formulas

from the logic part:

**Definition (Syntax of Propositional Logic)**

Let $A$ be a set of atomic propositions. The set of propositional formulas (over $A$) is inductively defined as follows:

- Every atom $a \in A$ is a propositional formula over $A$.
- If $\varphi$ is a propositional formula over $A$, then so is its negation $\neg\varphi$.
- If $\varphi$ and $\psi$ are propositional formulas over $A$, then so is the conjunction $(\varphi \wedge \psi)$.
- If $\varphi$ and $\psi$ are propositional formulas over $A$, then so is the disjunction $(\varphi \vee \psi)$.

# Example: Propositional Formulas

Let $S_A$ be the set of all propositional formulas over $A$.

Such sets of symbol sequences (or words) are called languages.

Sought: General concepts to define such (often infinite) languages with finite descriptions.

- today: grammars
- later: automata

# Example: Propositional Formulas

**Example (Grammar for $S_{\{\mathtt{a},\mathtt{b},\mathtt{c}\}}$)**

Grammar variables $\{\mathsf{F}, \mathsf{A}, \mathsf{N}, \mathsf{C}, \mathsf{D}\}$ with start variable $\mathsf{F}$, terminal symbols $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \neg, \wedge, \vee, (, )\}$ and rules

$$\begin{array}{lll} \mathsf{F} \to \mathsf{A} & \mathsf{A} \to \mathtt{a} & \mathsf{N} \to \neg\mathsf{F} \\ \mathsf{F} \to \mathsf{N} & \mathsf{A} \to \mathtt{b} & \mathsf{C} \to (\mathsf{F} \wedge \mathsf{F}) \\ \mathsf{F} \to \mathsf{C} & \mathsf{A} \to \mathtt{c} & \mathsf{D} \to (\mathsf{F} \vee \mathsf{F}) \\ \mathsf{F} \to \mathsf{D} & & \end{array}$$

Start with $\mathsf{F}$. In each step, replace a left-hand side of a rule with its right-hand side until no more variables are left:

$$\mathsf{F} \Rightarrow \mathsf{N} \Rightarrow \neg\mathsf{F} \Rightarrow \neg\mathsf{D} \Rightarrow \neg(\mathsf{F} \vee \mathsf{F}) \Rightarrow \neg(\mathsf{A} \vee \mathsf{F}) \Rightarrow \neg(\mathtt{b} \vee \mathsf{F}) \Rightarrow \neg(\mathtt{b} \vee \mathsf{A}) \Rightarrow \neg(\mathtt{b} \vee \mathtt{c})$$

# Alphabets and Formal Languages

# Alphabets and Formal Languages

**Definition (Alphabets, Words and Formal Languages)**

An alphabet $\Sigma$ is a finite non-empty set of symbols.

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

**Example**

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

# Alphabets and Formal Languages

## Definition (Alphabets, Words and Formal Languages)

An alphabet $\Sigma$ is a finite non-empty set of symbols.

A word over $\Sigma$ is a finite sequence of elements from $\Sigma$.
The empty word (the empty sequence of elements) is denoted by $\varepsilon$.
$\Sigma^*$ denotes the set of all words over $\Sigma$.

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

## Example

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

# Alphabets and Formal Languages

## Definition (Alphabets, Words and Formal Languages)

An alphabet $\Sigma$ is a finite non-empty set of symbols.

A word over $\Sigma$ is a finite sequence of elements from $\Sigma$.
The empty word (the empty sequence of elements) is denoted by $\varepsilon$.
$\Sigma^*$ denotes the set of all words over $\Sigma$.

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

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

## Example

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$
$\Sigma^* = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \dots\}$
$|\mathtt{aba}| = 3, |\mathtt{b}| = 1, |\varepsilon| = 0$

# Alphabets and Formal Languages

## Definition (Alphabets, Words and Formal Languages)

An alphabet $\Sigma$ is a finite non-empty set of symbols.

A word over $\Sigma$ is a finite sequence of elements from $\Sigma$.
The empty word (the empty sequence of elements) is denoted by $\varepsilon$.
$\Sigma^*$ denotes the set of all words over $\Sigma$.

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

A formal language (over alphabet $\Sigma$) is a subset of $\Sigma^*$.

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

## Example

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$
$\Sigma^* = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \dots\}$
$|\mathtt{aba}| = 3, |\mathtt{b}| = 1, |\varepsilon| = 0$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$
- $S_2 = \Sigma^*$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$
- $S_2 = \Sigma^*$
- $S_3 = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \dots\}$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$
- $S_2 = \Sigma^*$
- $S_3 = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \dots\}$
- $S_4 = \{\varepsilon\}$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$
- $S_2 = \Sigma^*$
- $S_3 = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \dots\}$
- $S_4 = \{\varepsilon\}$
- $S_5 = \emptyset$

# Languages: Examples

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

- $S_1 = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \dots\}$
- $S_2 = \Sigma^*$
- $S_3 = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \dots\}$
- $S_4 = \{\varepsilon\}$
- $S_5 = \emptyset$
- $S_6 = \{w \in \Sigma^* \mid w \text{ contains twice as many as as bs}\}$
  $= \{\varepsilon, \mathtt{aab}, \mathtt{aba}, \mathtt{baa}, \dots\}$

# Languages: Examples

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

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

# Questions

Questions?

# Grammars

# Grammars

## Definition (Grammars)

A grammar is a 4-tuple $\langle \Sigma, V, P, S \rangle$ with:

1. $\Sigma$ finite alphabet of terminal symbols
2. $V$ finite set of variables (nonterminal symbols) with $V \cap \Sigma = \emptyset$
3. $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ finite set of rules (or productions)
4. $S \in V$ start variable

German: Grammatik, Terminalalphabet, Variablen, Regeln/Produktionen, Startvariable

# Rule Sets

What exactly does $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ mean?

- $(V \cup \Sigma)^*$: all words over $(V \cup \Sigma)$
- $(V \cup \Sigma)^+$: all non-empty words over $(V \cup \Sigma)$
  in general, for set $X$: $X^+ = X^* \setminus \{\varepsilon\}$
- $\times$: Cartesian product
- $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$: set of all pairs $\langle x, y \rangle$, where $x$ non-empty word over $(V \cup \Sigma)$ and $y$ word over $(V \cup \Sigma)$
- Instead of $\langle x, y \rangle$ we usually write rules in the form $x \to y$.

# Rules: Examples

## Example

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $V = \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$.

Some examples of rules in $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$:

$$\begin{aligned} \mathrm{X} &\to \mathrm{XaY} \\ \mathrm{Yb} &\to \mathtt{a} \\ \mathrm{XY} &\to \varepsilon \\ \mathrm{XYZ} &\to \mathtt{abc} \\ \mathtt{abc} &\to \mathrm{XYZ} \end{aligned}$$

# Derivations

## Definition (Derivations)

Let $\langle \Sigma, V, P, S \rangle$ be a grammar. A word $v \in (V \cup \Sigma)^*$ can be derived from word $u \in (V \cup \Sigma)^+$ (written as $u \Rightarrow v$) if

1. $u = xyz$, $v = xy'z$ with $x, z \in (V \cup \Sigma)^*$ and
2. there is a rule $y \to y' \in P$.

We write: $u \Rightarrow^* v$ if $v$ can be derived from $u$ in finitely many steps (i. e., by using $n$ derivations for $n \in \mathbb{N}_0$).

German: Ableitung

# Language Generated by a Grammar

**Definition (Languages)**

The language generated by a grammar $G = \langle \Sigma, V, P, S \rangle$

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

is the set of all words from $\Sigma^*$ that can be derived from $S$ with finitely many rule applications.

German: erzeugte Sprache

# Grammars

Examples: blackboard

# Questions

Questions?

# Chomsky Hierarchy

# Chomsky Hierarchy

Grammars are organized into the Chomsky hierarchy.

**Definition (Chomsky Hierarchy)**

- Every grammar is of type 0 (all rules allowed).
- Grammar is of type 1 (context-sensitive) if all rules $w_1 \to w_2$ satisfy $|w_1| \leq |w_2|$.
- Grammar is of type 2 (context-free) if additionally $w_1 \in V$ (single variable) in all rules $w_1 \to w_2$.
- Grammar is of type 3 (regular) if additionally $w_2 \in \Sigma \cup \Sigma V$ in all rules $w_1 \to w_2$.

special case: rule $S \to \varepsilon$ is always allowed if $S$ is the start variable and never occurs on the right-hand side of any rule.

German: Chomsky-Hierarchie, Typ 0, Typ 1 (kontextsensitiv), Typ 2 (kontextfrei), Typ 3 (regulär)

# Chomsky Hierarchy

Examples: blackboard

# Chomsky Hierarchy

**Definition (Type 0–3 Languages)**

A language $L \subseteq \Sigma^*$ is of type 0 (type 1, type 2, type 3) if there exists a type-0 (type-1, type-2, type-3) grammar $G$ with $\mathcal{L}(G) = L$.

# Type $k$ Language: Example

## Example

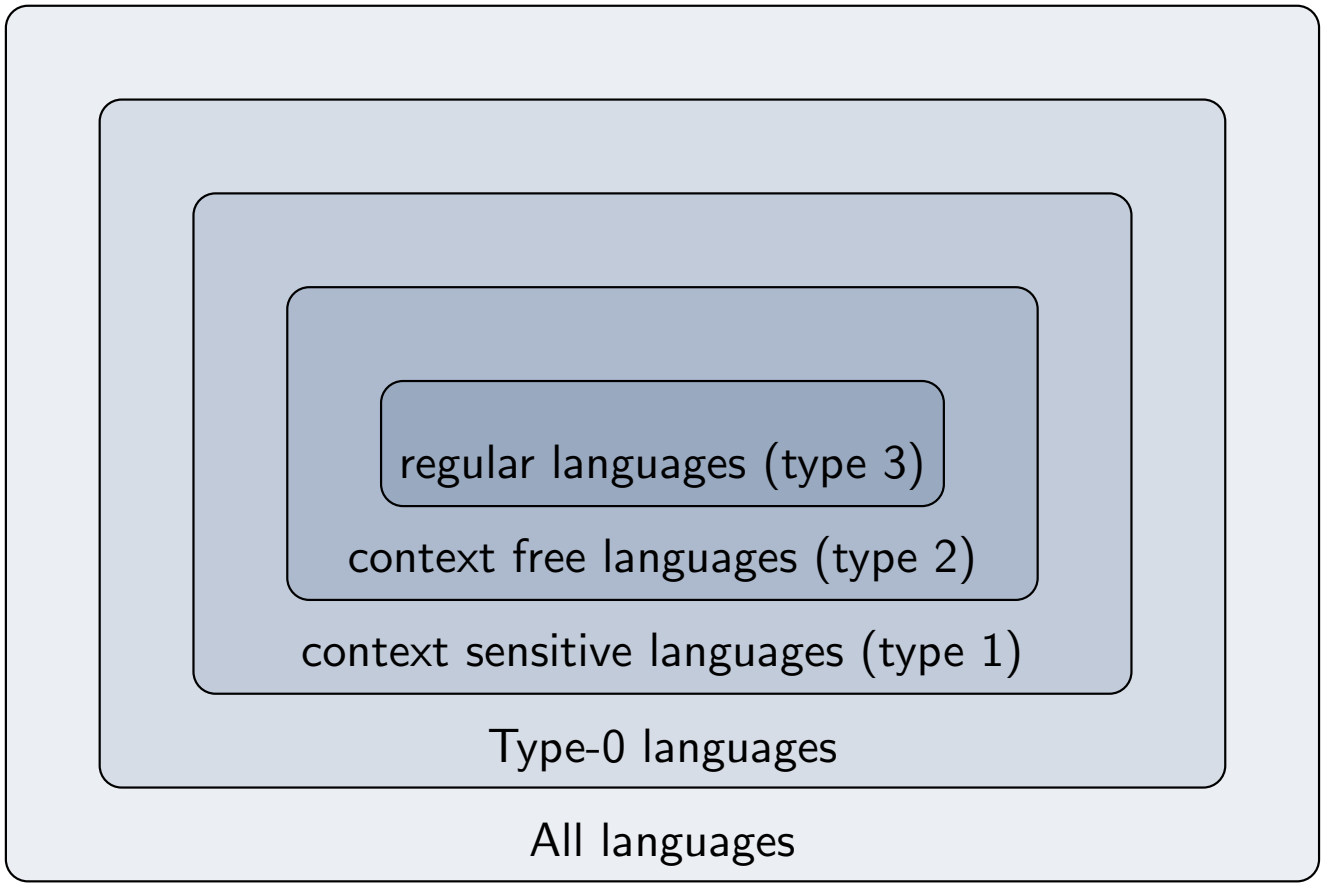
Consider the language $L$ generated by the grammar
$\langle \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \neg, \wedge, \vee, (, )\}, \{\mathrm{F}, \mathrm{A}, \mathrm{N}, \mathrm{C}, \mathrm{D}\}, P, \mathrm{F}\rangle$
with the following rules $P$:

$$\begin{array}{lll} \mathrm{F} \to \mathrm{A} & \mathrm{A} \to \mathtt{a} & \mathrm{N} \to \neg\mathrm{F} \\ \mathrm{F} \to \mathrm{N} & \mathrm{A} \to \mathtt{b} & \mathrm{C} \to (\mathrm{F} \wedge \mathrm{F}) \\ \mathrm{F} \to \mathrm{C} & \mathrm{A} \to \mathtt{c} & \mathrm{D} \to (\mathrm{F} \vee \mathrm{F}) \\ \mathrm{F} \to \mathrm{D} & & \end{array}$$

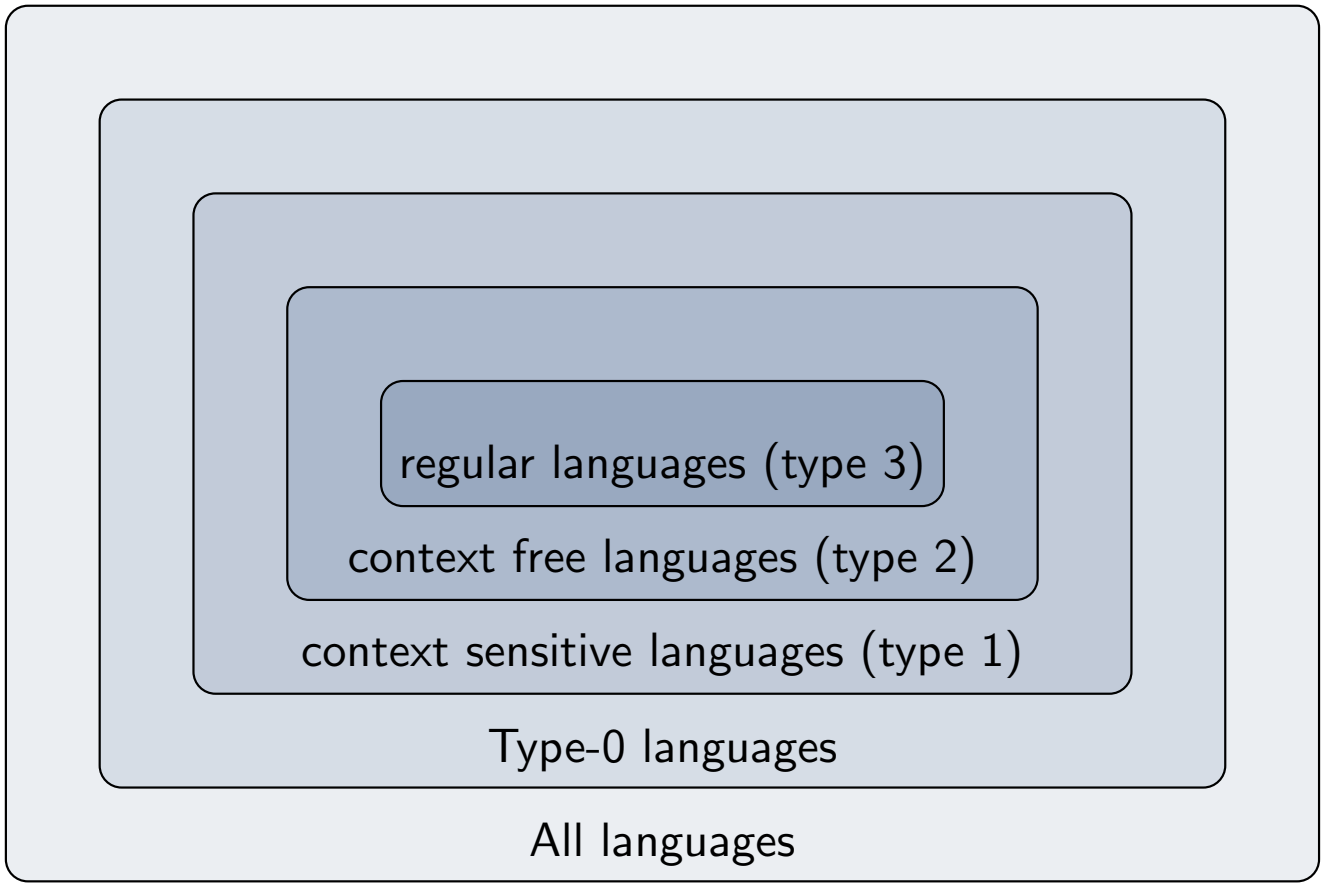
Questions:

- Is $L$ a type-0 language?
- Is $L$ a type-1 language?
- Is $L$ a type-2 language?
- Is $L$ a type-3 language?

# Chomsky Hierarchy



# Chomsky Hierarchy




Note: Not all languages can be described by grammars. (Proof?)

# Questions

Questions?

# Summary

# Summary

- Languages are sets of symbol sequences.
- Grammars are one possible way to specify languages.
- Language generated by a grammar is the set of all words (of terminal symbols) derivable from the start symbol.
- Chomsky hierarchy distinguishes between languages at different levels of expressiveness.

# Summary

- Languages are sets of symbol sequences.
- Grammars are one possible way to specify languages.
- Language generated by a grammar is the set of all words (of terminal symbols) derivable from the start symbol.
- Chomsky hierarchy distinguishes between languages at different levels of expressiveness.

following chapters:

- more about regular languages
- automata as alternative representation of languages