

Theory of Computer Science

B1. Formal Languages & Grammars

Gabriele Röger

University of Basel

March 2/4, 2026

1 / 37

Theory of Computer Science

March 2/4, 2026 — B1. Formal Languages & Grammars

B1.1 Introduction

B1.2 Formal Languages

B1.3 Grammars

B1.4 Chomsky Hierarchy

B1.5 Summary

2 / 37

B1. Formal Languages & Grammars

Introduction

B1.1 Introduction

3 / 37

B1. Formal Languages & Grammars

Introduction

Course Contents

Parts of the course:

A. background

▷ mathematical foundations and proof techniques

B. automata theory and formal languages

▷ What is a computation?

C. Turing computability ▷ What can be computed at all?

D. complexity theory ▷ What can be computed efficiently?

E. more computability theory ▷ Other models of computability

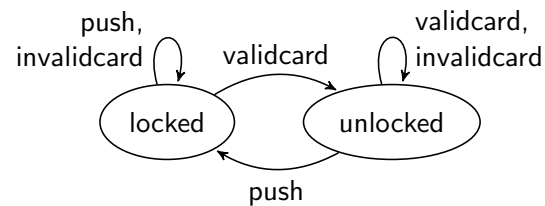
4 / 37

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



5 / 37

Turnstile Example: Decision Problem

Definition (Decision Problem for Turnstile Example)

- Given:** Sequence of actions from set {push, validcard, invalidcard}
- Question:** If the turnstile was initially locked, is it unlocked after the given sequence of actions? That is, does the input sequence contain an action validcard such that afterwards there is never an occurrence of push?

6 / 37

Decision Problems: Given-Question Form

Definition (Decision Problem, Given-Question Form)

- Given:** possible input
- Question:** does the given input have a certain property?

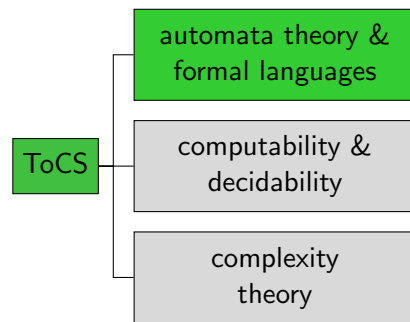
- ▶ often infinitely many instances (possible inputs).
- ▶ we want to characterize the set of all "Yes" instances
- ▶ **formal languages** are an alternative for representing such decision problems, using this set perspective instead of the given-question form.
- ▶ follow-up question: how can we characterize such a possibly infinite set with a final representation?

7 / 37

B1.2 Formal Languages

8 / 37

Content of the Course



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 .

$\Sigma^+ (= \Sigma^* \setminus \{\varepsilon\})$ 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^* .

Example

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

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

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

Languages: Examples

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

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

Languages: Turnstile Example

Example

$$\Sigma = \{\text{push, validcard, invalidcard}\}$$

$$\mathcal{L}_{\text{turnstile}} = \{w \in \Sigma^* \mid \text{there is an occurrence of validcard in } w \\ \text{and after the last occurrence of validcard} \\ \text{there is no occurrence of push}\}$$

Exercise (slido)

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

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



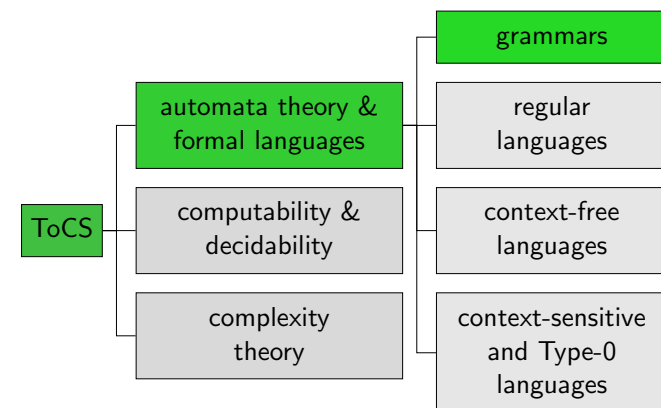
Ways to Specify Formal Languages?

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

- ▶ today: **grammars**
- ▶ later: automata, regular expressions, ...

B1.3 Grammars

Content of the Course



Grammar: Example

Variables $V = \{S, X, Y\}$

Alphabet $\Sigma = \{a, b, c\}$.

Production rules:

$$\begin{array}{lll} S \rightarrow \varepsilon & X \rightarrow aXYc & cY \rightarrow Yc \\ S \rightarrow X & X \rightarrow abc & bY \rightarrow bb \end{array}$$

You start from S and may in each step replace the left-hand side of a rule with the right-hand side of the same rule. This way, derive a word over Σ .

Short-hand Notation for Rule Sets

We abbreviate several rules with the same left-hand side in a single line, using “|” for separating the right-hand sides.

For example, we write

$$X \rightarrow \emptyset Y1 \mid XY$$

for:

$$X \rightarrow \emptyset Y1 \text{ and}$$

$$X \rightarrow XY$$

Exercise

Variables $V = \{S, X, Y\}$

Alphabet $\Sigma = \{a, b, c\}$.

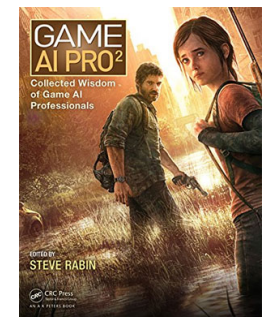
Production rules:

$$\begin{array}{l} S \rightarrow \varepsilon \mid X \\ X \rightarrow aXYc \mid abc \\ cY \rightarrow Yc \\ bY \rightarrow bb \end{array}$$


Derive word aabbcc starting from S .

Application: Content Generation in Games

- ▶ <http://www.gameapro.com/>
- ▶ GameAIPro 2, chapter 40
**Procedural Content Generation:
An Overview** by Gillian Smith



Grammars

Definition (Grammars)

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

- ▶ V finite set of **variables** (**nonterminal symbols**)
- ▶ Σ finite alphabet of **terminal symbols** with $V \cap \Sigma = \emptyset$
- ▶ $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ finite set of **rules**
- ▶ $S \in V$ **start variable**

A rule is sometimes also called a **production** or a **production rule**.

Rule Sets

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

- ▶ $(V \cup \Sigma)^*$: all words over $(V \cup \Sigma)$
- ▶ for languages L and L' , their **concatenation** is the language $LL' = \{xy \mid x \in L \text{ and } y \in L'\}$.
- ▶ $(V \cup \Sigma)^* V (V \cup \Sigma)^*$: words composed from
 - ▶ a word over $(V \cup \Sigma)$,
 - ▶ followed by a single variable symbol,
 - ▶ followed by a word over $(V \cup \Sigma)$
 → word over $(V \cup \Sigma)$ containing at least one variable symbol
- ▶ \times : Cartesian product
- ▶ $(V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$: set of all pairs $\langle x, y \rangle$, where x is a word over $(V \cup \Sigma)$ with at least one variable and y is a word over $(V \cup \Sigma)$
- ▶ Instead of $\langle x, y \rangle$ we usually write rules in the form $x \rightarrow y$.

Rules: Examples

Example

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

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

- $X \rightarrow XaY$
- $Yb \rightarrow a$
- $XY \rightarrow \varepsilon$
- $XYZ \rightarrow abc$
- $abXc \rightarrow XYZ$

Derivations

Definition (Derivations)

Let $\langle V, \Sigma, R, S \rangle$ 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 $y \rightarrow y' \in R$.

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

Language Generated by a Grammar

Definition (Languages)

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

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

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

Grammars

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

- ▶ $L_1 = \{a, aa, aaa, aaaa, \dots\} = \{a\}^+$
- ▶ $L_2 = \Sigma^*$
- ▶ $L_3 = \{a^n b^n \mid n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \dots\}$
- ▶ $L_4 = \{\varepsilon\}$
- ▶ $L_5 = \emptyset$
- ▶ $L_6 = \{w \in \Sigma^* \mid w \text{ contains twice as many } a\text{'s as } b\text{'s}\}$
 $= \{\varepsilon, aab, aba, baa, \dots\}$

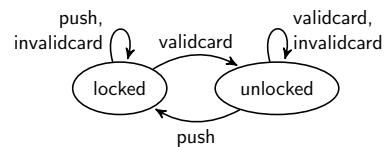
Example grammars: blackboard

Grammars

Example (Turnstile)

$G = \langle \{S, U\}, \{\text{push, validcard, invalidcard}\}, R, S \rangle$
 with the following production rules in R :

- $S \rightarrow \text{push } S$
- $S \rightarrow \text{invalidcard } S$
- $S \rightarrow \text{validcard } U$
- $U \rightarrow \text{invalidcard } U$
- $U \rightarrow \text{validcard } U$
- $U \rightarrow \varepsilon$
- $U \rightarrow \text{push } S$



$\mathcal{L}(G) = \mathcal{L}_{\text{turnstile}}$ from section "formal languages"

Exercise

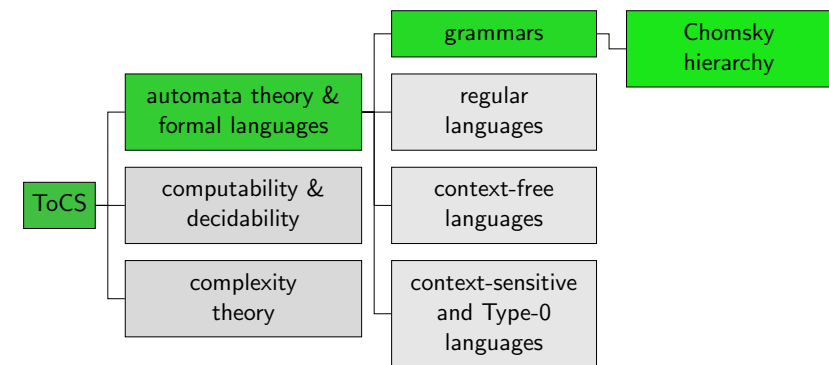
Specify a grammar that generates language

$$L = \{w \in \{a, b\}^* \mid |w| = 3\}.$$



B1.4 Chomsky Hierarchy

Content of the Course



Noam Chomsky

- ▶ Avram Noam Chomsky (*1928)
- ▶ "the father of modern linguistics"
- ▶ American linguist, philosopher, cognitive scientist, social critic, and political activist
- ▶ combined linguistics, cognitive science and computer science
- ▶ opponent of U.S. involvement in the Vietnam war
- ▶ there is a Wikipedia page solemnly on his political positions



CC BY 2.0 / Andrew Rusk

→ Organized grammars into the **Chomsky hierarchy**.

Chomsky Hierarchy

Definition (Chomsky Hierarchy)

- ▶ Every grammar is of **type 0** (all rules allowed).
- ▶ Grammar is of **type 1 (context-sensitive)** if all rules are of the form $\alpha B \gamma \rightarrow \alpha \beta \gamma$ with $B \in V$ and $\alpha, \gamma \in (V \cup \Sigma)^*$ and $\beta \in (V \cup \Sigma)^+$
- ▶ Grammar is of **type 2 (context-free)** if all rules are of the form $A \rightarrow w$, where $A \in V$ and $w \in (V \cup \Sigma)^+$.
- ▶ Grammar is of **type 3 (regular)** if all rules are of the form $A \rightarrow w$, where $A \in V$ and $w \in \Sigma \cup \Sigma V$.

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

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 (slido)

Example

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

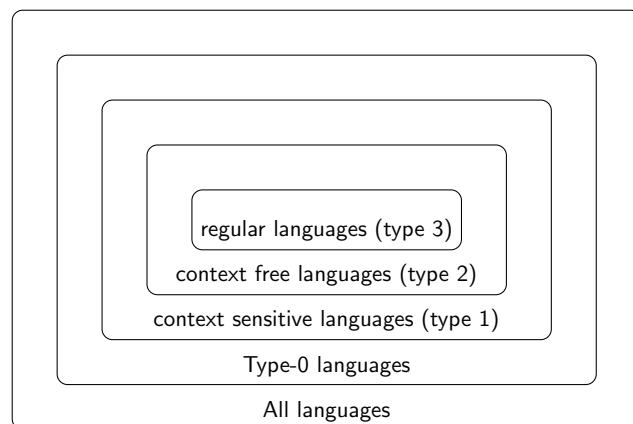
$$\begin{array}{lll} F \rightarrow A & A \rightarrow a & N \rightarrow \neg F \\ F \rightarrow N & A \rightarrow b & C \rightarrow (F \wedge F) \\ F \rightarrow C & A \rightarrow c & D \rightarrow (F \vee F) \\ F \rightarrow D & & \end{array}$$

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



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

B1.5 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.