Theory of Computer Science
B1. Formal Languages & Grammars

Gabriele Röger

University of Basel

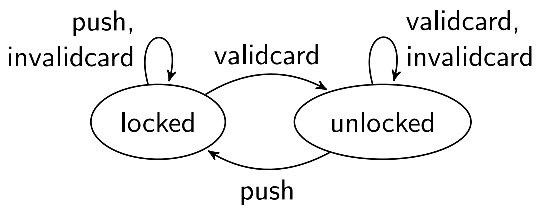
March 6, 2024

Theory of Computer Science
March 6, 2024 — B1. Formal Languages & Grammars

B1.1 Introduction

A Controller for a Turnstile

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



Turnstile Example: Decision Problem

Definition (Decision Problem for Turnstile Example)

Given: Sequence of actions from set $\{\text{push}, \text{validcard}, \text{invalidcard}\}$

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?

Decision Problems: Given-Question Form

Definition (Decision Problem, Given-Question Form)

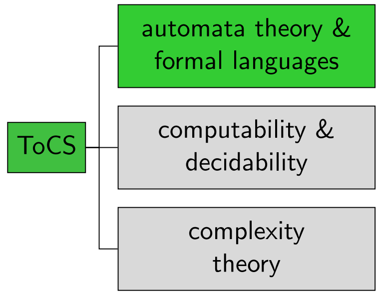
Given: possible input

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?

B1.2 Formal Languages

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.
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^*.

Example

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

Languages: Examples

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

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

Languages: Turnstile Example

Example

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

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

Exercise (slido)

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

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

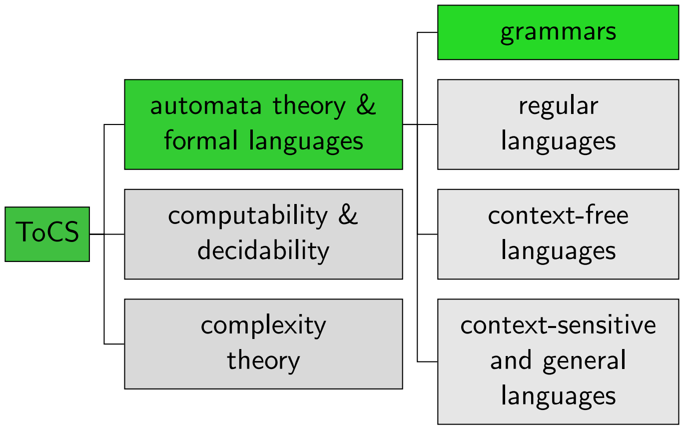
Ways to Specify Formal Languages?

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 $V = \{\mathrm{S}, \mathrm{X}, \mathrm{Y}\}$
Alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$.
Production rules:

$\mathrm{S} \to \varepsilon$	$\mathrm{X} \to \mathrm{aXYc}$	$\mathrm{cY} \to \mathrm{Yc}$
$\mathrm{S} \to \mathrm{abc}$	$\mathrm{X} \to \mathrm{abc}$	$\mathrm{bY} \to \mathrm{bb}$
$\mathrm{S} \to \mathrm{X}$		

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 variable in a single line, using "|" for separating the right-hand sides.

For example, we write

$\mathtt{X} \to \mathtt{0Y1} \mid \mathtt{XY}$

for:

$\mathtt{X} \to \mathtt{0Y1}$ and
$\mathtt{X} \to \mathtt{XY}$

Exercise

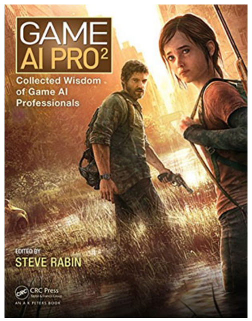
Variables $V = \{\mathtt{S}, \mathtt{X}, \mathtt{Y}\}$
Alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.
Production rules:

$$\begin{aligned} \mathtt{S} &\to \varepsilon \mid \mathtt{abc} \mid \mathtt{X} \\ \mathtt{X} &\to \mathtt{aXYc} \mid \mathtt{abc} \\ \mathtt{cY} &\to \mathtt{Yc} \\ \mathtt{bY} &\to \mathtt{bb} \end{aligned}$$

Derive word $\mathtt{aabbcc}$ starting from $\mathtt{S}$.

Application: Content Generation in Games

- `http://www.gameaipro.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 $(V \cup \Sigma)$

  $\rightarrow$ 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 word over $(V \cup \Sigma)$ with at least one variable and y 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 $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $V = \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$.

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

$$\begin{aligned} \mathrm{X} &\rightarrow \mathrm{X}\mathtt{a}\mathrm{Y} \\ \mathrm{Y}\mathtt{b} &\rightarrow \mathtt{a} \\ \mathrm{XY} &\rightarrow \varepsilon \\ \mathrm{XYZ} &\rightarrow \mathtt{abc} \\ \mathtt{ab}\mathrm{X}\mathtt{c} &\rightarrow \mathrm{XYZ} \end{aligned}$$

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

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

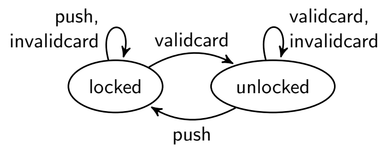
Example grammars: blackboard

Grammars

Example (Turnstile)

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

$$\begin{aligned}
\mathrm{S} &\to \mathtt{push}\,\mathrm{S}\\
\mathrm{S} &\to \mathtt{invalidcard}\,\mathrm{S}\\
\mathrm{S} &\to \mathtt{validcard}\,\mathrm{U}\\
\mathrm{U} &\to \mathtt{invalidcard}\,\mathrm{U}\\
\mathrm{U} &\to \mathtt{validcard}\,\mathrm{U}\\
\mathrm{U} &\to \varepsilon\\
\mathrm{U} &\to \mathtt{push}\,\mathrm{S}
\end{aligned}$$



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

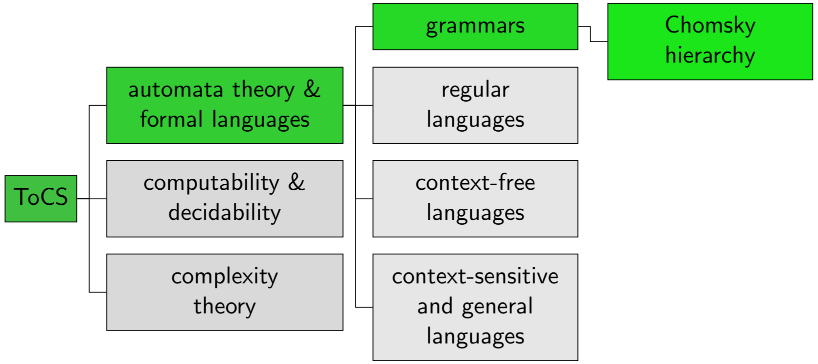
Exercise

Specify a grammar that generates language

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

B1.4 Chomsky Hierarchy

Content of the Course



Noam Chomsky

CC BY 2.0 / Andrew Rusk

- 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

$\rightarrow$ 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)

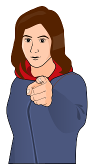
Example

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

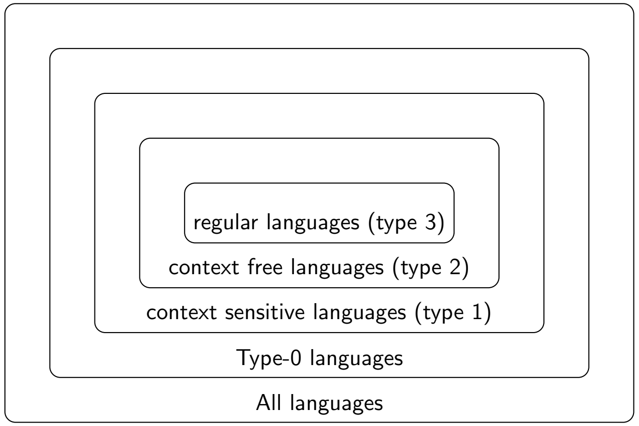
$\mathrm{F} \to \mathrm{A}$	$\mathrm{A} \to \mathtt{a}$	$\mathrm{N} \to \neg\mathrm{F}$
$\mathrm{F} \to \mathrm{N}$	$\mathrm{A} \to \mathtt{b}$	$\mathrm{C} \to (\mathrm{F} \wedge \mathrm{F})$
$\mathrm{F} \to \mathrm{C}$	$\mathrm{A} \to \mathtt{c}$	$\mathrm{D} \to (\mathrm{F} \vee \mathrm{F})$
$\mathrm{F}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

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.