Theory of Computer Science

C1. Formal Languages and Grammars

Malte Helmert

University of Basel

March 14, 2016

Theory of Computer Science

March 14, 2016 — C1. Formal Languages and Grammars

C1.1 Introduction

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 φ is a propositional formula over A, then so is its negation $\neg\varphi$.
- If φ and ψ are propositional formulas over A, then so is the conjunction $(\varphi \wedge \psi)$.
- If φ and ψ 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 $S_{\{a,b,c\}}$)

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

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

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

$F \Rightarrow N \Rightarrow \neg F \Rightarrow \neg D \Rightarrow \neg(F \vee F) \Rightarrow \neg(A \vee F) \Rightarrow \neg(b \vee F)$
$\Rightarrow \neg(b \vee A) \Rightarrow \neg(b \vee c)$

C1.2 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.

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

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

C1.3 Grammars

Grammars

Definition (Grammars)

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

1. Σ 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 $V = \{\mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$.

The following rules are in $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$:

$$\mathsf{X} \to \mathsf{X}\mathtt{a}\mathsf{Y}$$
$$\mathsf{Y}\mathtt{b} \to \mathtt{a}$$
$$\mathsf{XY} \to \varepsilon$$
$$\mathsf{XYZ} \to \mathtt{abc}$$
$$\mathtt{abc} \to \mathsf{XYZ}$$

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 rules 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 Σ^* that can be derived from S with finitely many rule applications.

German: erzeugte Sprache

Grammars

Examples: blackboard

C1.4 Chomsky Hierarchy

Chomsky Hierarchy

Grammars are ordered 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

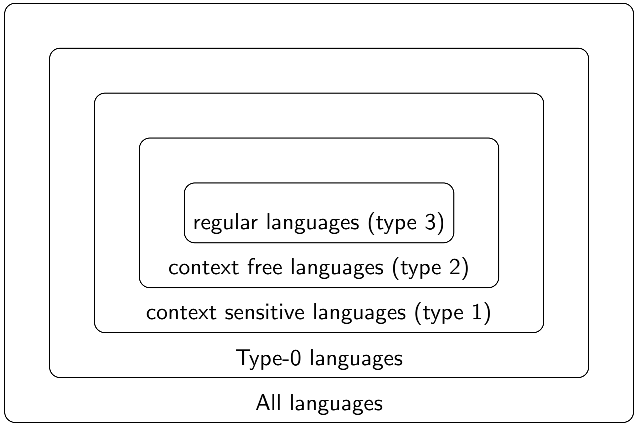
Consider the language L generated by the grammar
$\langle\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \neg, \wedge, \vee, (,)\}, \{\mathsf{F}, \mathsf{A}, \mathsf{N}, \mathsf{C}, \mathsf{D}\}, P, \mathsf{F}\rangle$
with the following rules P:

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

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?)

C1.5 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 nonterminal symbols) derivable from the start symbol.
- Chomsky hierarchy distinguishes between languages at different levels of expressiveness.

next chapters:

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