Introduction to Formal Language Theory — day 1

Formal Complexity of Natural Languages; Languages, Grammars, Chomsky Hierarchy

Wiebke Petersen & Kata Balogh

(Heinrich-Heine-Universität Düsseldorf)

NASSLLI 2014
University of Maryland, College Park

About this course

- introduction to the theory of formal languages, grammars and automatons from a linguistic point of view
- core question: "How complex are natural languages?"
- topics:
  - modeling natural languages as formal languages
  - the Chomsky hierarchy and the properties of its language classes
  - grammars and automatons for language generation and acceptance
  - decision problems and the notion of reducibility
- lecturers:
  - Wiebke Petersen (`petersen@phil.uni-duesseldorf.de`)
    Kata Balogh (`katalin.balogh@hhu.de`)
  - Heinrich-Heine-Universität Düsseldorf
    `www.phil-fak.uni-duesseldorf.de/en/computational-linguistics/`
- course page: `http://user.phil.hhu.de/~petersen/NASSLLI2014/`

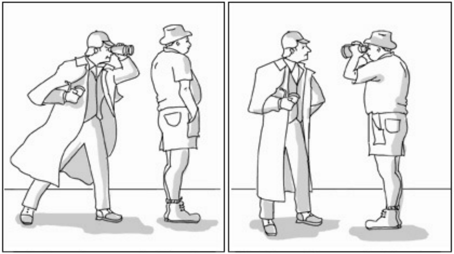
Outline

Formal complexity of natural languages

- Latvian, German, English, Chinese, . . .
- Prolog, Pascal, . . .
- Esperanto, Volapük, Interlingua, . . .
- proposition logic, predicate logic
- . . .

Formal complexity of natural languages

- Latvian, German, English, Chinese, . . .
- vague, ambiguous
  - lexical ambiguities
    - They passed the port at midnight.
  - structural ambiguities
    - Sherlock saw the man with the binoculars.



- only experts: humans
- natural languages develop

Formal complexity of natural languages

- difficult to learn (e.g. second language)
- complex phonology / morphology / syntax / . . .
- difficult to parse

Formal complexity of natural languages

- computational complexity
- structural complexity

Structural complexity

- Natural languages are modeled as abstract symbol systems with construction rules.
- Questions about the grammaticality of natural sentences correspond to questions about the syntactic correctness of programs or about the well-formedness of logic expressions.

What a grammar theory has to explain

- the cat chases dogs
- ~~the cat dogs chases~~
- ~~the dogs cat chases~~
- ~~the dogs chases cat~~
- ~~the chases dogs cat~~
- ~~the chases cat dogs~~
- ~~cat chases dogs the~~
- cat chases the dogs
- ~~cat dogs the chases~~
- ~~cat dogs chases the~~
- ~~cat the dogs chases~~
- ~~cat the chases dogs~~
- dogs the cat chases
- ~~dogs the chases cat~~
- ~~dogs chases cat the~~
- ~~dogs chases the cat~~
- ~~dogs cat chases the~~
- ~~dogs cat the chases~~
- ~~chases dogs cat the~~
- ~~chases dogs the cat~~
- chases the cat dogs
- ~~chases the dogs cat~~
- ~~chases cat the dogs~~
- ~~chases cat dogs the~~

The number of grammatical sentences is small compared to all possible word sequences.

How complex are English sentences?

1. Anne sees Peter.
2. Anne sees Peter in the garden with the binoculars.
3. Anne who dances sees Peter whom she met yesterday in the garden with the binoculars.
4. Anne sees Peter and Hans and Sabine and Joachim and Elfriede and Johanna and Maria and Jochen and Thomas and Andrea.

The length of a sentence influences the processing complexity, but it is not a sign of structural complexity!

Natural Language Theories vs. Formal Language Theory

Natural Language Theories

- grammar theories
- explain language data
- are language specific (Latvian, German, ...)

Formal Language Theory

- a theory about the structure of symbol strings
- not language specific
- allows statements about the mechanisms for generating and recognizing sets of symbol strings

Natural Languages and Formal Languages

- Generative Grammar (linguistics): from a finite number of words + finite number of rules → infinite number of sentences
- Standard (GG) Assumptions: (about any natural language)
  - The length of any sentence is finite. (whether letters, phonemes, morphemes, or words)
  - There is no longest sentence. (because of recursion)
- from these two assumptions it follows that the cardinality of the set of sentences in any natural language is infinite

Natural Languages and Formal Languages

- modeling any natural language as a set of strings (made of words, morphemes etc.)
- the set of possible strings formed from a vocabulary can be grammatical or ungrammatical
- language: the set of all grammatical strings
- grammar: determines the set of all grammatical strings

Alphabets and words

Definition

- **alphabet** Σ: nonempty, finite set of **symbols**
- **word** w: a finite string $x_1 \ldots x_n$ of symbols; $(x_1 ... x_n \in \Sigma)$
- **length** of a word $|w|$: number of symbols of a word w (example: $|abbaca| = 6$)
- **empty word** ϵ: the word of length 0
- Σ^* is the set of all words over Σ; $(\epsilon \in \Sigma^*)$
- Σ^+ is the set of all nonempty words over Σ $(\Sigma^+ = \Sigma^* \setminus \{\epsilon\})$

Blank symbol, empty word, and empty set

Be careful!

The blank symbol $\square$ can be a *symbol* of the alphabet and thus a word of length 1 (we do not distinguish in our notation between symbols and words of length 1).

The empty word ϵ is a *word* of length 0.

The empty set $\emptyset$ is a *set*.

Concatenation

Definition

The **concatenation** of two words $w = a_1 a_2 \dots a_n$ and $v = b_1 b_2 \dots b_m$ with $n, m \geq 0$ is

$$w \frown v = a_1 \dots a_n b_1 \dots b_m$$

The concatenation $\frown$ is a function $\frown : \Sigma^* \times \Sigma^* \to \Sigma^*$, which assigns strings to pairs of strings.

We often write uv instead of $u \frown v$.

$$w \frown \epsilon = \epsilon \frown w = w \qquad \text{neutral element}$$

$$u \frown (v \frown w) = (u \frown v) \frown w \qquad \text{associativity}$$

$(\Sigma^*, \frown)$ is a semi-group with neutral element (monoid).

Exponents, Kleene star, and reversals

Exponents

- w^n: w concatenated n-times with itself (e.g.: $w^3 = w \frown w \frown w$);
- $w^0 = \epsilon$; $w^* = \{w^0, w^1, w^2, w^3, ...\}$

The exponent of a word is a word.

Kleene star

- $w^* = \bigcup_{n \geq 0} \{w^n\}$ (the set of all words of the form w^n).
- Note: $\epsilon \in w^*$ for any word w ($\epsilon = w^0$).

The Kleene star of a word is a set of words.

Reversals

- The reversal of a word w is denoted w^R (e.g.: $(abcd)^R = dcba$).
- A word w with $w = w^R$ is called a **palindrome** (e.g.: madam, mum, otto, anna, . . .).

Formal language

Definition

A **formal language** L is a set of words over an alphabet Σ, i.e. $L \subseteq \Sigma^*$.

Examples:

- language L_{pal} over the Latin alphabet of the palindromes in English $L_{pal} = \{\text{mum, madam, } \dots\}$
- language L_{Mors} over the alphabet $\{-, \cdot\}$ of the letters of the Latin alphabet encoded in Morse's code: $L_{Mors} = \{\cdot -, - \cdot \cdot\cdot, \dots, - - \cdot\cdot\}$
- the empty set
- the set of words of length 13 over the alphabet $\{a, b, c\}$
- English?

Operations on formal languages

Definition

- If $L \subseteq \Sigma^*$ and $K \subseteq \Sigma^*$ are two formal languages over an alphabet Σ, then

$$K \cup L,\ K \cap L,\ K \setminus L$$

  are languages over Σ too.

- The **concatenation** of two formal languages K and L is

$$K \frown L := \{v \frown w \in \Sigma^* \mid v \in K, w \in L\}$$

- $L^n = \underbrace{L \frown L \frown L \ldots \frown L}_{n\text{-times}}$
- $L^* := \bigcup_{n \geq 0} L^n$. Note: $\epsilon \in L^*$ for any language L.

Examples: operations on formal languages

Example

$K = \{abb, a\}$ and $L = \{bbb, a\}$

- $K \setminus L = \{abb\}$
- $K \cup L = \{abb, a, bbb\}$
- $K \cap L = \{a\}$
- $K \frown L = \{abbbbb, abba, abbb, aa\}$
- $L \frown K = \{bbbabb, bbba, aabb, aa\}$
- $K^2 = \{abbabb, abba, aabb, aa\}$
- $K \frown \emptyset = \emptyset$
- $K \frown \{\epsilon\} = K = \{\epsilon\} \frown K$

Enumerating all elements of a language

- Peter says that Mary is fallen off the tree.
- Oskar says that Peter says that Mary is fallen off the tree.
- Lisa says that Oskar says that Peter says that Mary is fallen off the tree.
- ...

Enumerating all strings of a language is a bad idea, as

- the set of strings of a natural language is infinite
- the enumeration does not gather any generalizations about the language

Grammars

Grammar

- A formal grammar is a **generating device** which can generate (and analyze) strings/words.
- Grammars are finite rule systems.
- The set of all strings generated by a grammar is a formal language (= generated language).

- Example grammar:

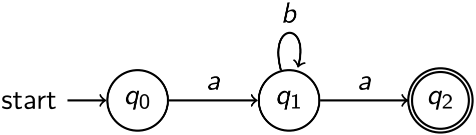
  $\mathsf{S} \rightarrow \mathsf{NP\ VP},\ \ \mathsf{VP} \rightarrow \mathsf{V},\ \ \mathsf{NP} \rightarrow \mathsf{DET\ N},\ \ \mathsf{NP} \rightarrow \mathsf{PN},$
  $\mathsf{DET} \rightarrow \mathsf{the},\ \ \mathsf{N} \rightarrow \mathsf{cat},\ \ \mathsf{V} \rightarrow \mathsf{sleeps},\ \ \mathsf{PN} \rightarrow \mathsf{Mia}$
- generates the sentences (strings of words):

  the cat sleeps, Mia sleeps

Automata

Automaton

- An automaton is a **recognizing device** which accepts strings/words.
- The set of all strings accepted by an automaton is a formal language (= accepted language).



accepts: $L(ab^\star a)$

Formal grammar

Definition

A **formal grammar** (also Type0-grammar) is a 4-tuple $G = (N, T, S, R)$ with

- an alphabet of nonterminals N,
- an alphabet of terminals T with $N \cap T = \emptyset$,
- a start symbol $S \in N$,
- a finite set of rules/productions $R \subseteq \{\langle \alpha, \beta \rangle \mid \alpha, \beta \in (N \cup T)^* \text{ and } \alpha \notin T^*\}$.

Instead of $\langle \alpha, \beta \rangle$ we often write $\alpha \to \beta$.

Formal grammar

Terminology

Let $G = (N, T, S, R)$ be a grammar and $v, w \in (T \cup N)^*$:

- v is **directly derived** from w (or w directly generates v), $w \Rightarrow v$ if $w = w_1 \alpha w_2$ and $v = w_1 \beta w_2$ such that $\langle \alpha, \beta \rangle \in R$
- v is **derived** from w (or w **generates** v), $w \Rightarrow^* v$
  if there exists $w_0, w_1, \ldots w_k \in (T \cup N)^*$ $(k \geq 0)$
  such that $w = w_0$, $w_k = v$ and $w_{i-1} \Rightarrow w_i$ for all $k \geq i \geq 0$
- $\Rightarrow^*$ denotes the reflexive, transitive closure of $\Rightarrow$
- $L(G) = \{w \in T^* | S \Rightarrow^* w\}$ is the **formal language generated by the grammar** G
- Two grammars G_1 and G_2 are **weakly equivalent** if and only if (iff) they generate the same language, i.e. $L(G_1) = L(G_2)$.

Example

$G_1 = (\{\text{S,NP,VP,N,V,D,N,PN}\}, \{\text{the, cat, peter, chases}\}, \text{S}, R)$

$$R = \left\{ \begin{array}{rclrclrcl} \text{S} & \rightarrow & \text{NP VP} & \text{VP} & \rightarrow & \text{V NP} & \text{NP} & \rightarrow & \text{D N} \\ \text{NP} & \rightarrow & \text{PN} & \text{D} & \rightarrow & \text{the} & \text{N} & \rightarrow & \text{cat} \\ \text{PN} & \rightarrow & \text{peter} & \text{V} & \rightarrow & \text{chases} & & & \end{array} \right\}$$

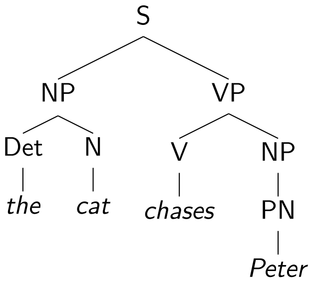
$$L(G_1) = \left\{ \begin{array}{cc} \text{the cat chases peter} & \text{peter chases the cat} \\ \text{peter chases peter} & \text{the cat chases the cat} \end{array} \right\}$$

"the cat chases peter" can be derived from S by:

$$\begin{array}{llll} \text{S} & \Rightarrow \text{NP VP} & \Rightarrow \text{NP V NP} & \Rightarrow \text{NP V PN} \\ & \Rightarrow \text{NP V peter} & \Rightarrow \text{NP chases peter} & \Rightarrow \text{D N chases peter} \\ & \Rightarrow \text{D cat chases peter} & \Rightarrow \text{the cat chases peter} & \end{array}$$

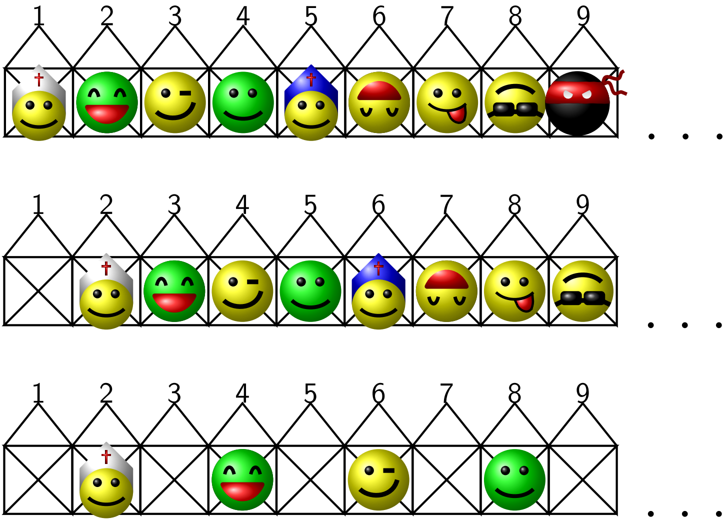
Derivation tree

S ⇒ NP VP ⇒ NP V NP ⇒ NP V PN
⇒ NP V peter ⇒ NP chases peter ⇒ D N chases peter
⇒ D cat chases peter ⇒ the cat chases peter



One derivation determines one derivation tree, but
the same derivation tree can result from different derivations.

Excursus: Hilbert's hotel – countable and uncountable sets



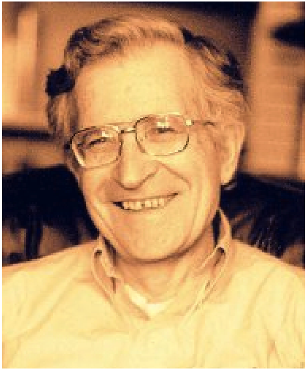
Not all formal languages are derivable from a formal grammar

- The set of all formal languages over an alphabet $\Sigma = \{a\}$ is $\mathcal{POW}(\Sigma^*)$; hence, the set is uncountable (infinite).
- The set of grammars generating formal languages over Σ with finite sets of productions is countable (infinite).
- Hence, the set of formal languages generated by a formal grammar is a strict subset of the set of all formal languages.

Chomsky-hierarchy

- The Chomsky-hierarchy is a hierarchy over structure conditions on the productions.
- Constraining the structure of the productions results in a restricted set of languages.
- The language classes correspond to conditions on the right- and left-hand sides of the productions.
- The Chomsky-hierarchy reflects a special form of complexity, other criteria are possible and result in different hierarchies.
- Linguists benefit from the rule-focussed definition of the Chomsky-hierarchy.

Noam Chomsky



Noam Chomsky
(∗ 7.12.1928, Philadelphia)
Noam Chomsky, *Three Models for the Description of Language*, (1956)

Chomsky-hierarchy

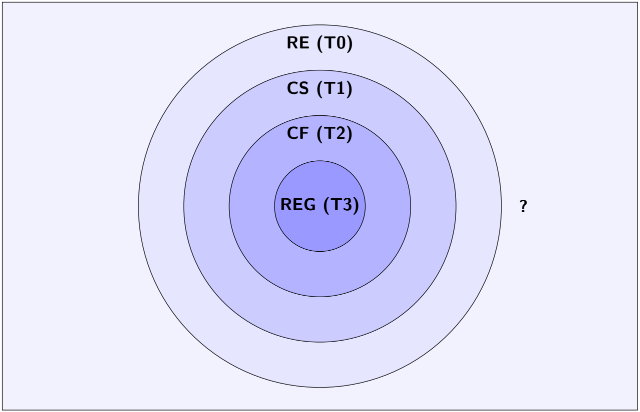
A grammar (N, T, S, R) is a

- Type 0 or unrestricted (phrase structure) grammar iff every production is of the form $\alpha \to \beta$ with $\alpha \in (N \cup T)^* \setminus T^*$ and $\beta \in (N \cup T)^*$;
generates a recursively enumerable language (RE).
- Type 1 or context-sensitive grammar iff every production is of the form $\gamma A \delta \to \gamma \beta \delta$ with $\gamma, \delta, \beta \in (N \cup T)^*, A \in N$ and $\beta \neq \epsilon$;
generates a context-sensitive language (CS).
- Type 2 or context-free grammar iff every production is of the form $A \to \beta$ with $A \in N$ and $\beta \in (N \cup T)^* \setminus \{\epsilon\}$;
generates a context-free language (CF).
- Type 3 or right-linear grammar iff every production is of the form $A \to \beta B$ or $A \to \beta$ with $A, B \in N$ and $\beta \in T^* \setminus \{\epsilon\}$;
generates a regular language (REG).

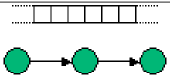
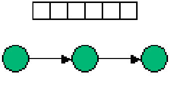
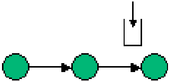
For Type 1-3 languages a rule $S \to \epsilon$ is allowed if S does not occur in any rule's right-hand side.

Chomsky-hierarchy: main theorem

$$\textbf{REG} \subset \textbf{CF} \subset \textbf{CS} \subset \textbf{RE}$$



Chomsky-hierarchy: overview

type	grammar	rules	machine	idea	word problem
RE	phrase struc- ture	$\alpha \to \beta$	Turing machine		undecidable
CS	context- sensitive	$\gamma A \delta \to \gamma \beta \delta$	linearly restricted automaton		exponential
CF	context- free	$A \to \beta$	pushdown- automaton		cubic
REG	right- linear	$A \to aB \mid b$	finite-state automaton		linear

Which is the class of natural languages?

Why is the formal formal complexity of natural languages interesting?

- It gives information about the general structure of natural language
- It allows to draw conclusions about the adequacy of grammar formalisms
- It determines a lower bound for the computational complexity of natural language processing tasks

Which is the class of natural languages?

Which idealizations about NL are necessary?

1. The family of natural languages exists.
2. Language = set of strings over an alphabet:
3. Natural languages are generated by finite rule systems (grammars)
4. Each NL consists of an *infinite* set of strings

About the idealizations

The family of natural languages exists:

- all natural languages are structurally similar
- all natural languages have a similar generative capacity

Arguments:

- all NLs serve for the same tasks
- children can learn each NL as their native language (within a similar period of time)

$\Rightarrow$ No evidence for a principal structural difference

About the idealizations (cont.)

Language = infinite set of strings over an alphabet:

- native speakers have full competence
- consistent grammaticality judgements

Arguments:

- all mistakes are due to performance not competence
- Mathews (1979) counter examples:
  - The canoe floated down the river sank.
  - The editor authors the newspaper hired liked laughed.
  - The man (that was) thrown down the stairs died.
  - The editor (whom) the authors the newspaper hired liked laughed.

About the idealizations (cont.)

Natural languages are generated by finite rule systems (grammars):

Arguments:

If a language is infinite, a finite set of rules can explain

- how a language can be learned
- how we understand each others sentences

About the idealizations (cont.)

Each NL consists of an *infinite* set of strings

Arguments:

- Recursion in NL:
  - John likes Peter
  - John likes Peter and Mary
  - John likes Peter and Mary and Sue
  - John likes Peter and Mary and Sue and Otto and . . .
- (Donaudampfschiffskapitänsmützenschirm . . .)

However:

- The set of all English sentences that have been used so far and that will be used in the time of mankind is finite.

Tomorrow

- bottom of the Chomsky-hierarchy
- Type 3 languages and grammars
- finite-state automaton
- regular expressions