Introduction to theory of languages

Patryk Kiepas

MINES ParisTech
AGH University Of Science and Technology
THALES

February 26, 2017

Course plan

1. Saturday, 25th of February 2017 – lecture
   - Languages
   - Grammars
2. Saturday, 4th of March 2017 – lecture
   - Parsing
   - ANTLR
3. Saturday, 11th of March 2017 – exercises
   - Grammars and languages
   - ANTLR
4. Saturday, 25th of March 2017 – exercises & exam (???)
   - ANTLR

Additional informations

Any questions?

Ask by mail: `kiepas@agh.edu.pl`

Course web-page

`http://home.agh.edu.pl/~kiepas` → **Teaching** → **Introduction to theory of languages (2017)**

This lecture

- Linguistics
- Languages
- Grammar
- Hierarchy of grammars
- Notations
- Automaton

Introduction

Linguistics

Scientific study of languages. Involves analysis of language:

- *form* – language evolution and task
- *context* – environment of language usage
- *semantics* – the meaning of the language

Some important aspects

- Phonetics
- Articulation
- Perception
- Acoustic features
- Morphology
- Syntax

Language types

1. Natural languages
    - *Ordinary* – evolves naturally in humans without planning
    - *Controlled* – a restricted subset of natural language in order reduce or eliminate ambiguity and complexity
2. **Artificial languages**
    - *Constructed* (planned *a priori* or *a posteriori*)
        - Engineered languages – experiments in *logic*, *philosophy*, *linguistics*
        - Auxiliary languages – international communication (e.g. Esperanto, Ido, Interlingua)
        - Artistic languages – aesthetic pleasure or humorous effect (e.g. Klingon)
    - **Formal**
        - Computer programming languages (e.g. Java, Haskell, C, C++, Ruby)
        - Files and formats descriptions (e.g. YAML, JSON, XML)

Description of natural languages

A really small bit of history

- In the late 1950's Noam Chomsky tried to describe natural languages
- Important paper: *"Three models for the description of language"*, Noam Chomsky (1956).
- In a result of his research two disciplines originated:
  1. **Theory of formal grammars**
  2. *Generative (transformational) grammars*

Figure 1: Professor of Linguistics (Emeritus) at MIT, Cambridge

Description of natural languages

What we know now?

- Description of natural languages is **hard**
- Description of any natural languages might be **impossible**

Why this is important?

- Better understanding of language creation processes
- More insights into functioning of our brain
- **Natural language processing (NLP)**
  - Translations (e.g. Google Translator)
  - Synthesis (e.g. speech generation)
  - Perceiving (e.g. robots, voice-control)

Description of formal languages

Result

Description of natural languages help us describe an artificial (formal) ones

Programming languages

- Protocol for communication with the computer
- Performing operations and computations
- Interpretation and execution
- Compilation
- Static code analysis

Data formats

- Structured data
- Interchangeable model for communication and data transmission

Alphabet

Alphabet

A set Σ of available symbols, the simplest elements in the language

Examples

- binary alphabet $\{0, 1\}$
- decimal numbers $\{0, 1, 2, 3, ..., 9\}$
- Latin alphabet $\{a, b, c, d, ..., z\}$
- Cyrillic

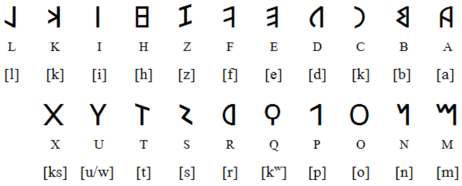

Figure 2: Ancient Latin alphabet

Word (I)

Word

Word w is a sequence of N symbols $w = x_1 x_2 ... x_N$ (e.g. 010110, $ABCDAAE$)

Length

Length of word w is a number of symbols it consists of $|w| = N$ (e.g. $|010110| = 6$, $|ABCDAAE| = 7$)

Empty word

Special word ϵ with length $|\epsilon| = 0$

Word (II)

Words examples

- $w = 010110$ word over alphabet $\Sigma = \{0, 1\}$
- $w = abc13dj3$ word over alphabet $\Sigma = \{a, b, ...z, 0, 1, ...9\}$
- $w = ACGTCCGGTA$ word over alphabet $\Sigma = \{A, C, G, T\}$

Closures

- Σ^* – set of all words over Σ
- Σ^+ – set of all nonempty words $\Sigma^+ = \Sigma^* \backslash \{\epsilon\}$

Closures examples

- if $\Sigma = \{a\}$ then $\Sigma^* = \{\epsilon, a, aa, aaa, aaaa, aaaaa, aaaaaa, ...\}$
- if $\Sigma = \{a, b\}$ then $\Sigma^+ = \{a, b, aa, bb, ab, ba, aaa, bbb, ...\}$
- if $\Sigma = \{a, b, ..., z\}$ then $\Sigma^+ = \{cat, dog, a, aa, aaa, ...\}$

Language

Definition

Formal language L is a set of words over an alphabet Σ, $L \subseteq \Sigma^*$

Examples

- Language L_1 of palindromes in English
  $L_1 = \{mum, hannah, madam, ...\}$
- Morse code with alphabet $\Sigma = \{\cdot, -\}$, $L_2 = \{\cdot -, - \cdot \cdot ..., - - \cdot\cdot\}$
- Empty language
- English language
- Language L_3 with the set of words with fixed-size of N

Grammar

Grammar

- Description of a language
- A recipe for composing elements into sentence
- Describes syntax of a language

Definition

Grammar is a system $G = (\Sigma, NT, P, S)$ where:

- Σ – alphabet (set of terminals)
- NT – set of nonterminals
- P – set of production rules
- S – marked nonterminal as *start symbol*

Grammar properties

- NT, P – are finite and nonempty
- Σ – usually nonempty (if empty we have an empty language)
- Σ, NT – are disjoint
- $P \subseteq (\Sigma \cup NT)^+ \times (\Sigma \cup NT)^*$
- $S \in NT$

Grammar and languages

- Sentence generated by some G is every $w \in \Sigma^*$ for each exists derivation from S
- Language $L(G)$ is generated by G and consists of sentences derivate using grammar G
- Two grammars G_1 and G_2 are *(weakly) equivalent* if $L(G_1) = L(G_2)$

Grammar example

Examples

Digits separated by plus or minus signs

$$\mathit{list} \rightarrow \mathit{list} + \mathit{list}$$
$$\mathit{list} \rightarrow \mathit{list} - \mathit{list}$$
$$\mathit{list} \rightarrow \ 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Chomsky's hierarchy

Hierarchy

- Describe the grammar expressiveness
- Describe the grammar hardness
- Tells us what "mechanical procedure" we need to use in order to:
  - Accept language
  - Generate language
- α, β – any sequence of terminals and nonterminals
- γ – any nonempty sequence of terminals and nonterminals
- A, B – nonterminals

Grammar	Language	Automaton	Production rules
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$
Type-1	Context-sensitive	Linear bounded ND TM	$\alpha A \beta \rightarrow \alpha\gamma\beta$
Type-2	Context-free	ND pushdown	$\alpha \rightarrow \gamma$
Type-3	Regular	Finite state	$A \rightarrow a$ and $A \rightarrow aB$

Backus-Naur form (BNF)

Backus-Naur form (BNF)

Notation technique for *context-free grammars*. Frequently used to describe syntax of *programming languages*, *document formats* etc.

Syntax

<term> ::= __expression__

- <term> is a *nonterminal*
- __expression__ is a sequence of one or more terminal and/or nonterminal symbols separated by vertical line |
- Terminal symbols: a, b, c, A, 0, 1, 2 etc.
- Nonterminal symbols: <digit>, <postal-code> etc.

Backus-Naur form (BNF)

Meta-symbols

- $::=$ – production rule definition
- $|$ – rule alternative
- $<>$ – nonterminals
- "" – literal
- $< EOL >$ – End Of Line

Examples

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<postal-code> ::= <digit> <digit> <digit> <digit> <digit>
```

BNF example : Palindrome

Palindrome grammar

```
<letter>     ::= a | b | c | ... | y | z
<palindrome> ::= <letter> |
<palindrome> ::= a <palindrome> a | b <palindrome> b |
                 c <palindrome> c | d <palindrome> d |
                 e <palindrome> e | ...
                                  | z <palindrome> z
```

Results

```
a
bb
bab
pop
hannah
```

BNF example : Postal address

Postal address grammar

```
<postal-address> ::= <name-part> <street-address> <zip-part>
<name-part> ::= <first-name> <last-name> <EOL>
<street-address> ::= <number> <street-name> <apt-num> <EOL>
<zip-part> ::= <postal-code> <town-name> <EOL>
<apt-num> ::= <number> | ""
```