Script **generated by TTT**

Title: Petter: Compiler Construction (07.05.2020) - 09: Basics of Grammars

Date: Fri Apr 24 17:00:05 CEST 2020

Duration: 16:13 min

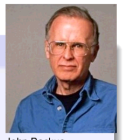
Pages: 7

Chapter 1:
Basics of Contextfree Grammars

Basics: Context-free Grammars

- Programs of programming languages can have arbitrary numbers of tokens, but only finitely many Token-classes.
- This is why we choose the set of Token-classes to be the finite alphabet of terminals T.
- The nested structure of program components can be described elegantly via context-free grammars...


Noam Chomsky


John Backus

Definition: Context-Free Grammar

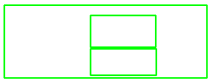
A context-free grammar (CFG) is a 4-tuple $G = (N, T, P, S)$ with:

- N the set of nonterminals,
- T the set of terminals,
- P the set of productions or rules, and
- $S \in N$ the start symbol

Conventions

The rules of context-free grammars take the following form:

$$A \to \alpha \quad \text{with} \quad A \in N\,, \quad \alpha \in (N \cup T)^*$$



Conventions

The rules of context-free grammars take the following form:

$$A \to \alpha \quad \text{with} \quad A \in N\,, \quad \alpha \in (N \cup T)^*$$

... for example:

$$\begin{array}{lcl} S & \to & a\,S\,b \\ S & \to & \epsilon \end{array}$$

Specified language: $\{a^n b^n \mid n \geq 0\}$

Conventions:

In examples, we specify nonterminals and terminals in general implicitely:

- nonterminals are: $A, B, C, \ldots, \langle \text{exp} \rangle, \langle \text{stmt} \rangle, \ldots$;
- terminals are: $a, b, c, \ldots$ int, name, ...;

... a practical example:

$$\begin{array}{lcl} S & \to & \langle \text{stmt} \rangle \\ \langle \text{stmt} \rangle & \to & \langle \text{if} \rangle \mid \langle \text{while} \rangle \mid \langle \text{rexp} \rangle\text{;} \\ \langle \text{if} \rangle & \to & \text{if (}\ \langle \text{rexp} \rangle\ \text{)}\ \langle \text{stmt} \rangle\ \text{else}\ \langle \text{stmt} \rangle \\ \langle \text{while} \rangle & \to & \text{while (}\ \langle \text{rexp} \rangle\ \text{)}\ \langle \text{stmt} \rangle \\ \langle \text{rexp} \rangle & \to & \text{int} \mid \langle \text{lexp} \rangle \mid \langle \text{lexp} \rangle = \langle \text{rexp} \rangle \mid \ldots \\ \langle \text{lexp} \rangle & \to & \text{name} \mid \ldots \end{array}$$

More conventions:

- For every nonterminal, we collect the right hand sides of rules and list them together.
- The j-th rule for A can be identified via the pair (A, j) (with $j \geq 0$).

... a practical example:

$$\begin{array}{lcl} S & \to & \langle \text{stmt} \rangle \\ \langle \text{stmt} \rangle & \to & \langle \text{if} \rangle \mid \langle \text{while} \rangle \mid \langle \text{rexp} \rangle\text{;} \\ \langle \text{if} \rangle & \to & \text{if (}\ \langle \text{rexp} \rangle\ \text{)}\ \langle \text{stmt} \rangle\ \text{else}\ \langle \text{stmt} \rangle \\ \langle \text{while} \rangle & \to & \text{while (}\ \langle \text{rexp} \rangle\ \text{)}\ \langle \text{stmt} \rangle \\ \langle \text{rexp} \rangle & \to & \text{int} \mid \langle \text{lexp} \rangle \mid \langle \text{lexp} \rangle = \langle \text{rexp} \rangle \mid \ldots \\ \langle \text{lexp} \rangle & \to & \text{name} \mid \ldots \end{array}$$

Pair of grammars:

$$\begin{array}{lcl} E & \to & E{+}E \mid E{*}E \mid (\ E\ ) \mid \text{name} \mid \text{int} \end{array}$$

$$\begin{array}{lcl} E & \to & E{+}T \mid T \\ T & \to & T{*}F \mid F \\ F & \to & (\ E\ ) \mid \text{name} \mid \text{int} \end{array}$$

Both grammars describe the same language