

Compiler Construction

Lecture 5: Syntactic Analysis I (Introduction)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

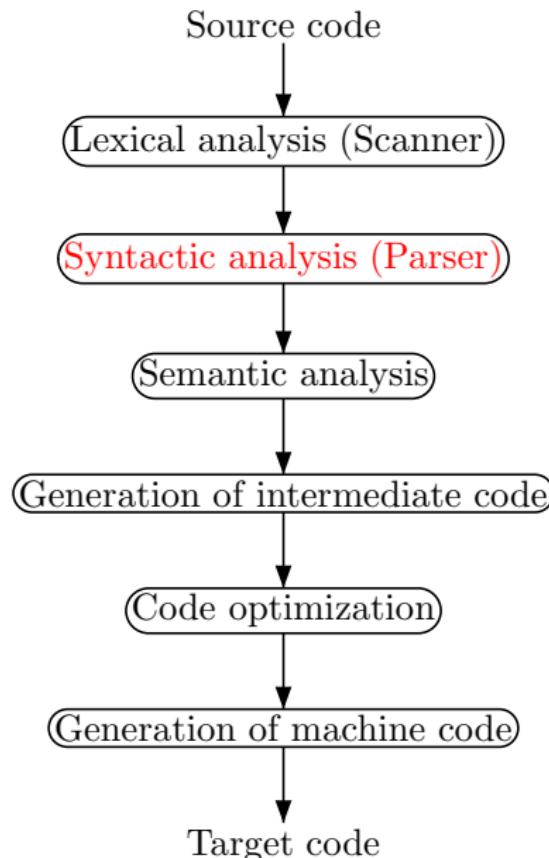
RWTH Aachen University

noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/cc08/>

Summer semester 2008

Conceptual Structure of a Compiler



- 1 Problem Statement
- 2 Context-Free Grammars and Languages
- 3 Parsing Context-Free Languages
- 4 Nondeterministic Top-Down Parsing

Starting point: sequence of symbols as produced by the scanner

Here: ignore attribute information

- Σ (finite) set of **tokens** (= syntactic atoms; **terminals**)
(e.g., {id, if, int, ...})
- $w \in \Sigma^*$ **token sequence**
(of course, not every $w \in \Sigma^*$ forms a valid program)

Syntactic units:

atomic: keywords, variable/type/procedure/... identifiers,
numerals, arithmetic/boolean operators, ...

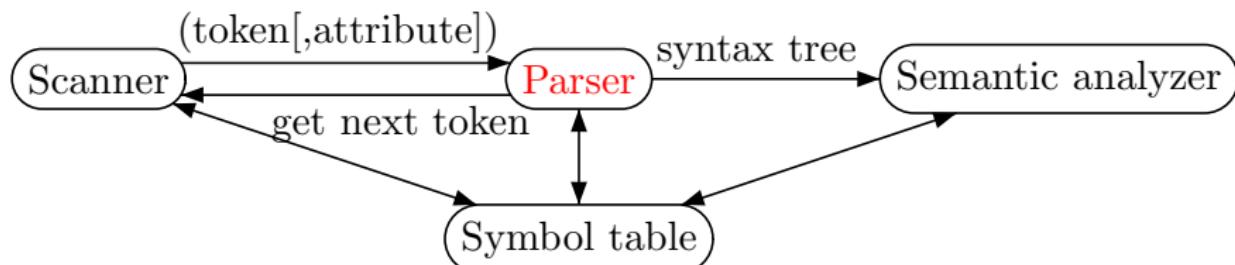
complex: declarations, arithmetic/boolean expressions, statements,
...

Observation: the hierarchical structure of syntactic units can be described by **context-free grammars**

Definition 5.1

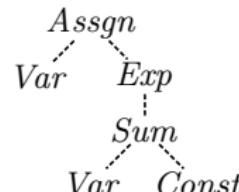
The goal of **syntactic analysis** is to determine the syntactic structure of a program, given by a token sequence, according to a context-free grammar.

The corresponding program is called a **parser**:



Example:

$\dots (\text{id}, p_1)(\text{gets},) (\text{id}, p_2)(\text{plus},) (\text{int}, 1)(\text{sem},) \dots \rightsquigarrow$



- 1 Problem Statement
- 2 Context-Free Grammars and Languages
- 3 Parsing Context-Free Languages
- 4 Nondeterministic Top-Down Parsing

Definition 5.2 (Syntax of context-free grammars)

A **context-free grammar (CFG)** (over Σ) is a quadruple

$$G = \langle N, \Sigma, P, S \rangle$$

where

- N is a finite set of **nonterminal symbols**,
- Σ is a (finite) alphabet of **terminal symbols** (disjoint from N),
- P is a finite set of **production rules** of the form $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in X^*$ for $X := N \cup \Sigma$, and
- $S \in N$ is a **start symbol**.

The set of all context-free grammars over Σ is denoted by CFG_Σ .

Remarks: as denotations we generally use

- $A, B, C, \dots \in N$ for nonterminal symbols
- $a, b, c, \dots \in \Sigma$ for terminal symbols
- $u, v, w, \dots \in \Sigma^*$ for terminal words
- $\alpha, \beta, \gamma, \dots \in X^*$ for **sentences**

Context-free grammars generate context-free languages:

Definition 5.3 (Semantics of context-free grammars)

Let $G = \langle N, \Sigma, P, S \rangle$ be a context-free grammar.

- The **derivation relation** $\Rightarrow \subseteq X^* \times X^*$ of G is defined by
$$\alpha \Rightarrow \beta \text{ iff there exist } \alpha_1, \alpha_2 \in X^*, A \rightarrow \gamma \in P \\ \text{such that } \alpha = \alpha_1 A \alpha_2 \text{ and } \beta = \alpha_1 \gamma \alpha_2.$$
- If in addition $\alpha_1 \in \Sigma^*$ or $\alpha_2 \in \Sigma^*$, then we write $\alpha \Rightarrow_l \beta$ or $\alpha \Rightarrow_r \beta$, respectively (**leftmost/rightmost** derivation).
- The **language generated by G** is given by
$$L(G) := \{w \in \Sigma^* \mid S \Rightarrow^* w\}.$$
- If a language $L \subseteq \Sigma^*$ is generated by some $G \in CFG_\Sigma$, then L is called **context free**. The set of all **context-free languages** over Σ is denoted by CFL_Σ .

Remark: obviously,

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_l^* w\} = \{w \in \Sigma^* \mid S \Rightarrow_r^* w\}$$

Example 5.4

The grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$ over $\Sigma := \{a, b\}$, given by the productions

$$S \rightarrow aSb \mid \varepsilon,$$

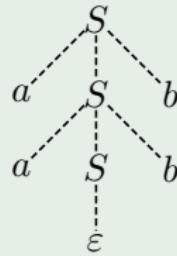
generates the context-free (and non-regular) language

$$L = \{a^n b^n \mid n \in \mathbb{N}\}.$$

The example derivation

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

can be represented by the following **syntax tree** for $aabb$:



Remark: the connection between derivations, syntax trees, and generated words is **not unique**

- ① A syntax tree generally represents several derivations.
- ② A derivation can generally be represented by several syntax trees.
- ③ A word can generally be produced by several derivations.
- ④ A word can have several syntax trees.

Example 5.5

on the board

However:

- ① Every syntax tree yields exactly one word
(= concatenation of leafs).
- ② Every syntax tree corresponds to exactly one leftmost derivation, and vice versa.
- ③ Every syntax tree corresponds to exactly one rightmost derivation, and vice versa.

Definition 5.6 (Ambiguity)

- A context-free grammar $G \in CFG_{\Sigma}$ is called **unambiguous** if every word $w \in L(G)$ has exactly one syntax tree. Otherwise it is called **ambiguous**.
- A context-free language $L \in CFL_{\Sigma}$ is called **inherently ambiguous** if every grammar $G \in CFG_{\Sigma}$ with $L(G) = L$ is ambiguous.

Example 5.7

on the board

Corollary 5.8

*A grammar $G \in CFG_{\Sigma}$ is unambiguous
iff every word $w \in L(G)$ has exactly one leftmost derivation
iff every word $w \in L(G)$ has exactly one rightmost derivation.*

- 1 Problem Statement
- 2 Context-Free Grammars and Languages
- 3 Parsing Context-Free Languages
- 4 Nondeterministic Top-Down Parsing

Problem 5.9 (Word problem for context-free languages)

Given $G \in \text{CFG}_{\Sigma}$ and $w \in \Sigma^*$, decide whether $w \in L(G)$ (and determine a corresponding syntax tree).

- Decidable for arbitrary CFGs (in Chomsky Normal Form) using the **tabular method by Cocke, Younger, and Kasami** (“CYK Algorithm”; space/time complexity $\mathcal{O}(|w|^2)/\mathcal{O}(|w|^3)$)
- **Goal:** exploit the special syntactic structures as present in programming languages (usually: no ambiguities) to devise parsing methods which are based on **deterministic pushdown automata with linear space and time complexity**

Two approaches:

Top-down analysis: construction of syntax tree from the **root towards the leafs**, representation as **leftmost derivation**

Bottom-up analysis: construction of syntax tree from the **leafs towards the root**, representation as (reversed) **rightmost derivation**

Goal: compact representation of left-/rightmost derivations by index sequences

Definition 5.10 (Leftmost/rightmost analysis)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$, $p := |P|$, and $\pi : [p] \rightarrow P$ a bijection.

- If $i \in [p]$, $\pi(i) = A \rightarrow \gamma$, $w \in \Sigma^*$, and $\alpha \in X^*$, then we write
$$wA\alpha \xrightarrow{i} w\gamma\alpha \quad \text{and} \quad \alpha A w \xrightarrow{i} \alpha\gamma w.$$
- If $z = i_1 \dots i_n \in [p]^*$, we write $\alpha \xrightarrow{z} \beta$ if there exist $\alpha_0, \dots, \alpha_n \in X^*$ such that $\alpha_0 = \alpha$, $\alpha_n = \beta$, and $\alpha_{j-1} \xrightarrow{i_j} \alpha_j$ for every $j \in [n]$ (analogously for \xrightarrow{z}).
- An index sequence $z \in [p]^*$ is called a **leftmost analysis** (**rightmost analysis**) of α if $S \xrightarrow{z} \alpha$ ($S \xrightarrow{z} \alpha$), respectively.

Leftmost/Rightmost Analysis

Example 5.11

Grammar for arithmetic expressions:

$$G_{AE} : \begin{array}{l} E \rightarrow E+T \mid T \quad (1, 2) \\ T \rightarrow T*F \mid F \quad (3, 4) \\ F \rightarrow (E) \mid a \mid b \quad (5, 6, 7) \end{array}$$

Leftmost derivation of $(a)*b$:

$$\begin{array}{lllllll} E & \xrightarrow[2]{l} & T & \xrightarrow[3]{l} & T*F & \xrightarrow[4]{l} & F*F & \xrightarrow[5]{l} & (E)*F \\ & \xrightarrow[2]{l} & (T)*F & \xrightarrow[4]{l} & (F)*F & \xrightarrow[6]{l} & (a)*F & \xrightarrow[7]{l} & (a)*b \end{array}$$

⇒ leftmost analysis: 23452467

Rightmost derivation of $(a)*b$:

$$\begin{array}{lllllll} E & \xrightarrow[2]{r} & T & \xrightarrow[3]{r} & T*F & \xrightarrow[7]{r} & T*b & \xrightarrow[4]{r} & F*b \\ & \xrightarrow[5]{r} & (E)*b & \xrightarrow[2]{r} & (T)*b & \xrightarrow[4]{r} & (F)*b & \xrightarrow[6]{r} & (a)*b \end{array}$$

⇒ rightmost analysis: 23745246

General assumption in the following: every grammar is reduced

Definition 5.12 (Reduced CFG)

A grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$ is called **reduced** if for every $A \in N$ there exist $\alpha, \beta \in X^*$ and $w \in \Sigma^*$ such that

$S \Rightarrow^* \alpha A \beta$ (A **reachable**) and

$A \Rightarrow^* w$ (A **productive**).

- 1 Problem Statement
- 2 Context-Free Grammars and Languages
- 3 Parsing Context-Free Languages
- 4 Nondeterministic Top-Down Parsing

Approach:

- ① Given $G \in CFG_{\Sigma}$, construct a **nondeterministic pushdown automaton** (PDA) which accepts $L(G)$ and which additionally computes corresponding leftmost derivations (similar to the proof of " $L(CFG_{\Sigma}) \subseteq L(PDA_{\Sigma})$ ")
 - input alphabet: Σ
 - pushdown alphabet: X
 - output alphabet: $[p]$
 - state set: omitted
- ② Remove nondeterminism by allowing **lookahead** on the input:
 $G \in LL(k)$ iff $L(G)$ recognizable by deterministic PDA with lookahead of k symbols

Definition 5.13 (Nondeterministic top-down parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$. The **nondeterministic top-down parsing automaton** of G , $NTA(G)$, is defined by the following components.

- **Input alphabet:** Σ
- **Pushdown alphabet:** X
- **Output alphabet:** $[p]$
- **Configurations:** $\Sigma^* \times X^* \times [p]^*$ (top of pushdown to the left)
- **Transitions** for $w \in \Sigma^*$, $\alpha \in X^*$, and $z \in [p]^*$:
 - expansion steps: if $\pi(i) = A \rightarrow \beta$, then $(w, A\alpha, z) \vdash (w, \beta\alpha, zi)$
 - matching steps: for every $a \in \Sigma$, $(aw, a\alpha, z) \vdash (w, \alpha, z)$
- **Initial configuration** for $w \in \Sigma^*$: (w, S, ε)
- **Final configurations**: $\{\varepsilon\} \times \{\varepsilon\} \times [p]^*$

Remark: $NTA(G)$ is nondeterministic iff G contains $A \rightarrow \beta \mid \gamma$

Example 5.14

Grammar for
arithmetic expressions
(cf. Example 5.11):

$$\begin{aligned}
 G_{AE} : E &\rightarrow E+T \mid T & (1, 2) \\
 T &\rightarrow T*F \mid F & (3, 4) \\
 F &\rightarrow (E) \mid a \mid b & (5, 6, 7)
 \end{aligned}$$

Leftmost analysis of $(a)*b$:

$$\begin{aligned}
 & ((a)*b, E, \varepsilon) \\
 \vdash & ((a)*b, T, 2) \\
 \vdash & ((a)*b, T*F, 23) \\
 \vdash & ((a)*b, F*F, 234) \\
 \vdash & ((a)*b, (E)*F, 2345) \\
 \vdash & ((a)*b, (E)*F, 2345) \\
 \vdash & ((a)*b, (T)*F, 23452) \\
 \vdash & ((a)*b, (F)*F, 234524) \\
 \vdash & ((a)*b, a)*F, 2345246) \\
 \vdash & (())*b,)*F, 2345246) \\
 \vdash & ((*b, *F, 2345246) \\
 \vdash & ((b, F, 2345246) \\
 \vdash & ((b, b, 23452467) \\
 \vdash & ((\varepsilon, \varepsilon, 23452467)
 \end{aligned}$$

Theorem 5.15 (Correctness of NTA(G))

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$ and NTA(G) as before. Then, for every $w \in \Sigma^*$ and $z \in [p]^*$,

$$(w, S, \varepsilon) \vdash^* (\varepsilon, \varepsilon, z) \quad \text{iff} \quad z \text{ is a leftmost analysis of } w$$

Proof.

\implies (soundness): see exercises

\impliedby (completeness): on the board

