Compiler Construction

Lecture 6: Syntactic Analysis I (Introduction)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

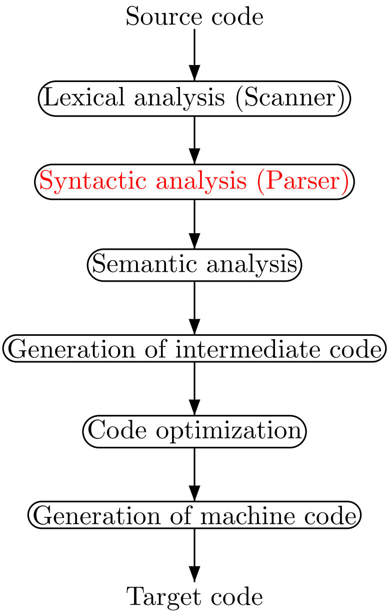
RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc10/

Winter semester 2010/11

Conceptual Structure of a Compiler



Outline

1 Problem Statement

2 Context-Free Grammars and Languages

Syntactic Structures

From Merriam-Webster's Online Dictionary

Syntax: the way in which linguistic elements (as words) are put together to form constituents (as phrases or clauses)

- **Starting point:** sequence of symbols as produced by the scanner
  Here: ignore attribute information
  - Σ (finite) set of tokens (= syntactic atoms; terminals)
    (e.g., $\{\mathsf{id}, \mathsf{if}, \mathsf{int}, \ldots\}$)
  - $w \in \Sigma^*$ 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, ...

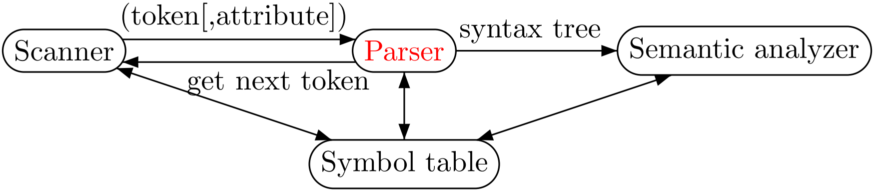
  complex: declarations, arithmetic/Boolean expressions, statements, ...
- **Observation:** the hierarchical structure of syntactic units can be described by context-free grammars

Syntactic Analysis

Definition 6.1

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.

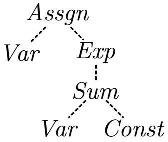
The corresponding program is called a parser:



Example: $\ldots$ `␣x1␣:=y2+␣1␣;␣` $\ldots$

$\downarrow$ Scanner

$\ldots(\mathsf{id}, p_1)(\mathsf{gets},)(\mathsf{id}, p_2)(\mathsf{plus},)(\mathsf{int}, 1)(\mathsf{sem},)\ldots \xrightarrow{\text{Parser}}$



Outline

1 Problem Statement

2 Context-Free Grammars and Languages

Context-Free Grammars I

Definition 6.2 (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 $A \to \alpha$ 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

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

Context-Free Grammars II

Context-free grammars generate context-free languages:

Definition 6.3 (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

$$\alpha \Rightarrow \beta \text{ iff there exist } \alpha_1, \alpha_2 \in X^*, A \to \gamma \in P \text{ such that } \alpha = \alpha_1 A \alpha_2 \text{ and } \beta = \alpha_1 \gamma \alpha_2.$$

- 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,

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

Context-Free Languages

Example 6.4

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

$$S \to aSb \mid \varepsilon,$$

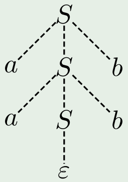
generates the context-free (and non-regular) language

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

The example derivation

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

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



Syntax Trees, Derivations, and Words

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 6.5

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.

(Un-)Ambiguity of CFGs and CFLs

Definition 6.6 (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 6.7

on the board

Corollary 6.8

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.