



Compiler Construction

Lecture 5: Syntax Analysis I (Introduction)

Summer Semester 2016

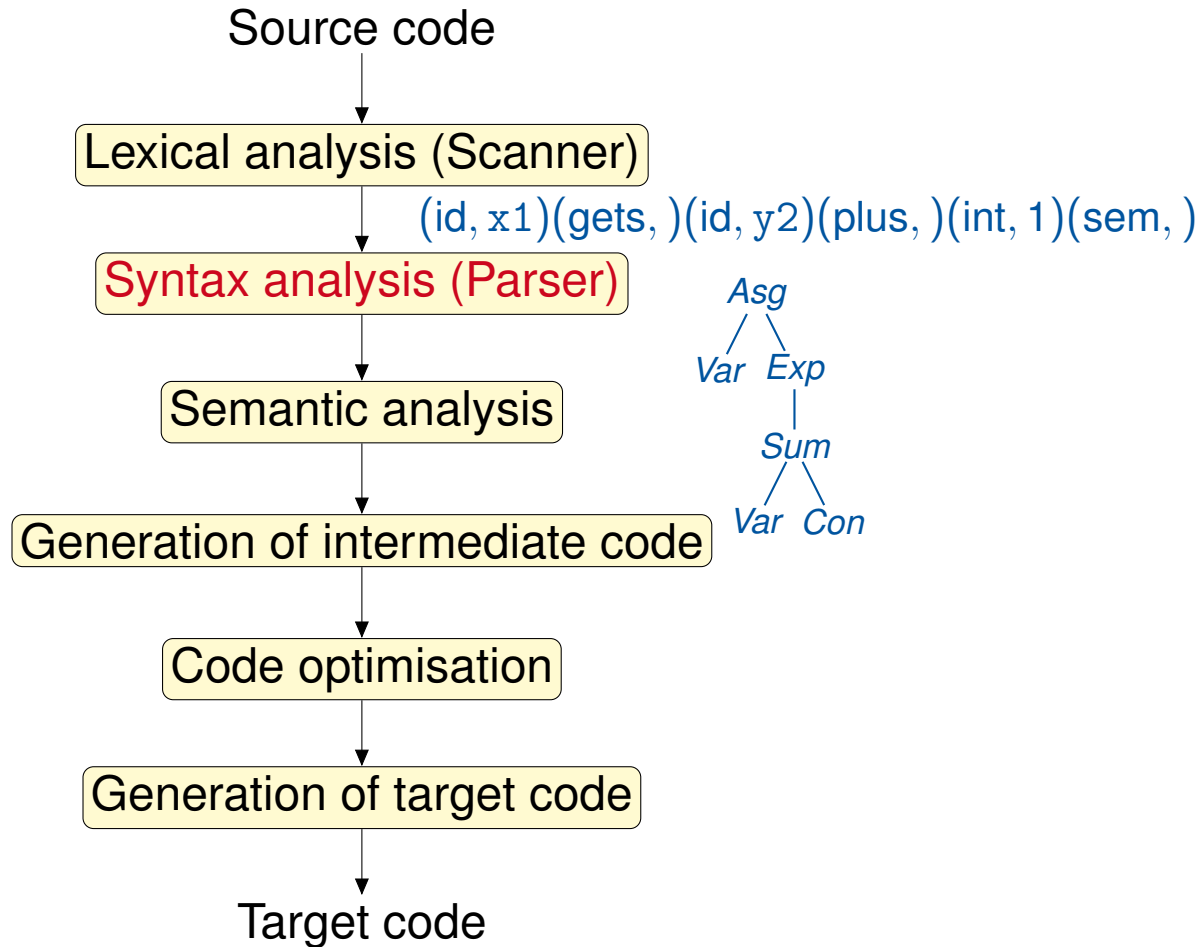
Thomas Noll

Software Modeling and Verification Group

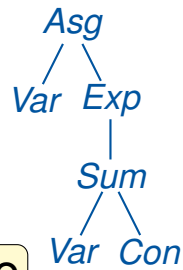
RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ss-16/cc/>

Conceptual Structure of a Compiler



context-free grammars/
pushdown automata



Problem Statement

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/**terminal symbols**, (e.g., {id, if, int, . . .}))
 - $w \in \Sigma^*$ **token sequence** (obviously, 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 (complex) syntactic units can be described by **context-free grammars**

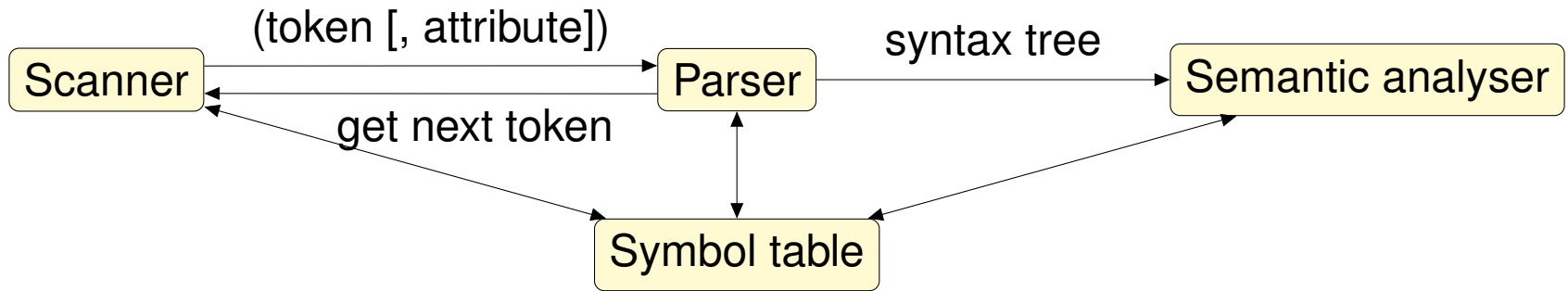
Problem Statement

Syntax Analysis

Definition 5.1

The goal of **syntax 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**:



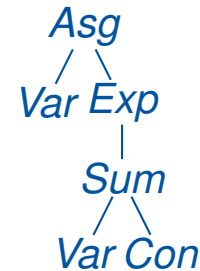
Example:

... $x_1 := y_2 + 1; \dots$

↓ Scanner

... (id, p_1)(gets,)(id, p_2)(plus,)(int, 1)(sem,) ...

Parser →



Context-Free Grammars and Languages

Context-Free Grammars I

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 $X := N \cup \Sigma$,
- $S \in N$ 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
- $u, v, w, x, y, \dots \in \Sigma^*$ for terminal words
- $\alpha, \beta, \gamma, \dots \in X^*$ for **sentences**

Context-Free Grammars and Languages

Context-Free Grammars II

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

Context-Free Grammars and Languages

Context-Free Languages

Example 5.4

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

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

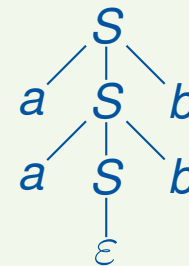
generates the context-free (and non-regular) language

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

The example derivation

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

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



Context-Free Grammars and Languages

Syntax Trees, Derivations, and Words

Observations:

1. Every syntax tree yields exactly one word (= concatenation of leaves).
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.

Thus: syntax trees are **uniquely** representable by leftmost/rightmost derivations

But: a word can have **several** syntax trees (see next slide)

Context-Free Grammars and Languages

Ambiguity of CFGs and CFLs I

Definition 5.5 (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 5.6

on the board

Corollary 5.7

*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.*

Context-Free Grammars and Languages

Ambiguity of CFGs and CFLs II

Theorem 5.8

It is generally *undecidable* whether a given CFG is ambiguous or not.

Proof (idea).

Reduction from **Post Correspondence Problem**: given instance (\vec{x}, \vec{y}) of PCP, construct CFG G with two “branches” $S \rightarrow X \mid Y$ that respectively enumerate all \vec{x}/\vec{y} -concatenations (plus some index information).

Result: G is ambiguous iff (\vec{x}, \vec{y}) has a solution
(see [Hopcroft/Motwani/Ullman 2001, Section 9.5.2] for details) □

Remark: *resolution* of ambiguities by parser (cf. bottom-up parsing using yacc):

- operator precedences
- operator associativities

Parsing Context-Free Languages

The Word Problem for Context-Free Languages

Problem 5.9 (Word problem for context-free languages)

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

This problem is **decidable** for arbitrary CFGs:

- (for CFGs in Chomsky Normal Form)
Using the **tabular method by Cocke, Younger, and Kasami**
("CYK Algorithm"; time/space complexity $\mathcal{O}(|w|^3)/\mathcal{O}(|w|^2)$)
- Using the **predecessor method**:

$$w \in L(G) \iff S \in \text{pre}^*(\{w\})$$

where $\text{pre}^*(M) := \{\alpha \in X^* \mid \alpha \Rightarrow^* \beta \text{ for some } \beta \in M\}$
(polynomial [non-linear] time complexity)

Parsing Context-Free Languages

Parsing Context-Free Languages

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 parsing: construction of syntax tree from the **root towards the leaves**, representation as **leftmost derivation**

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

Leftmost/Rightmost Analysis I

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

Definition 5.10 (Leftmost/rightmost analysis)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$ where $P = \{\pi_1, \dots, \pi_p\}$.

- If $i \in [p]$, $\pi_i = A \rightarrow \gamma$, $w \in \Sigma^*$, and $\alpha \in X^*$, then we write

$$wA\alpha \xRightarrow{i}_l w\gamma\alpha \quad \text{and} \quad \alpha Aw \xRightarrow{i}_r \alpha\gamma w.$$

- If $z = i_1 \dots i_n \in [p]^*$, we write $\alpha \xRightarrow{z}_l \beta$ if there exist $\alpha_0, \dots, \alpha_n \in X^*$ such that $\alpha_0 = \alpha$, $\alpha_n = \beta$, and $\alpha_{j-1} \xRightarrow{i_j}_l \alpha_j$ for every $j \in [n]$ (analogously for \xRightarrow{z}_r).
- An index sequence $z \in [p]^*$ is called a **leftmost analysis** (**rightmost analysis**) of α if $S \xRightarrow{z}_l \alpha$ ($S \xRightarrow{z}_r \alpha$), respectively.

Parsing Context-Free Languages

Leftmost/Rightmost Analysis II

Example 5.11

Grammar for arithmetic expressions: $G_{AE} : E \rightarrow E+T \mid T \quad (1, 2)$
 $T \rightarrow T*F \mid F \quad (3, 4)$
 $F \rightarrow (E) \mid a \mid b \quad (5, 6, 7)$

Leftmost derivation of $(a)*b$:

$$\begin{array}{ccccccc} E & \xrightarrow{2}_l & T & \xrightarrow{3}_l & T*F & \xrightarrow{4}_l & F*F & \xrightarrow{5}_l & (E)*F \\ & \xrightarrow{2}_l & (T)*F & \xrightarrow{4}_l & (F)*F & \xrightarrow{6}_l & (a)*F & \xrightarrow{7}_l & (a)*b \end{array}$$

\Rightarrow **leftmost analysis:** 23452467

Rightmost derivation of $(a)*b$:

$$\begin{array}{ccccccc} E & \xrightarrow{2}_r & T & \xrightarrow{3}_r & T*F & \xrightarrow{7}_r & T*b & \xrightarrow{4}_r & F*b \\ & \xrightarrow{5}_r & (E)*b & \xrightarrow{2}_r & (T)*b & \xrightarrow{4}_r & (F)*b & \xrightarrow{6}_r & (a)*b \end{array}$$

\Rightarrow **rightmost analysis:** 23745246

Parsing Context-Free Languages

Reducedness of Context-Free Grammars

General assumption in the following: every grammar is reduced

Definition 5.12 (Reduced CFG)

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

$$S \Rightarrow^* \alpha A \beta \quad (A \text{ reachable}) \text{ and}$$

$$A \Rightarrow^* w \quad (A \text{ productive}).$$

Nondeterministic Top-Down Parsing

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: not required
2. **Remove nondeterminism** by supporting **lookahead** on the input:
 $G \in LL(k)$ iff $L(G)$ recognisable by deterministic PDA with lookahead of k symbols

Nondeterministic Top-Down Parsing

The Nondeterministic Top-Down Automaton I

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 $A \rightarrow \beta \mid \gamma$

Nondeterministic Top-Down Parsing

The Nondeterministic Top-Down Automaton II

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 $(a)*b$:

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