Compiler Construction

Lecture 5: Syntax Analysis I (Introduction)

Thomas Noll

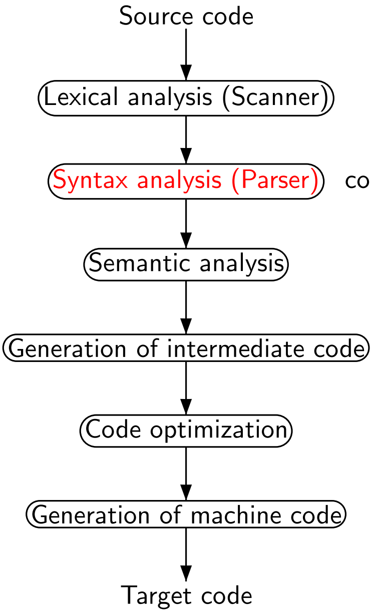
Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

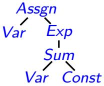
http://moves.rwth-aachen.de/teaching/ss-14/cc14/

Summer Semester 2014



(id, x1)(gets,)(id, y2)(plus,)(int, 1)

context-free grammars/pushdown automata



Outline

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)

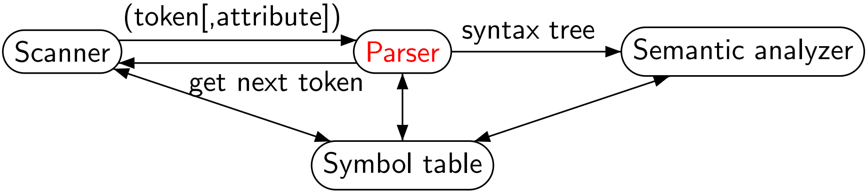
- **Starting point:** sequence of symbols as produced by the scanner
  Here: ignore attribute information
  - Σ (finite) set of tokens (= syntactic atoms; terminals)
    (e.g., $\{\mathsf{id}, \mathsf{if}, \mathsf{int}, \ldots\}$)
  - $w \in \Sigma^*$ token sequence
    (of course, not every $w \in \Sigma^*$ 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

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.

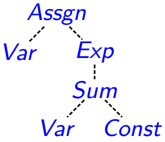
The corresponding program is called a parser:



Example: `...␣x1␣:=y2+␣1␣;␣...`

↓ Scanner

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



Outline

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 $A \to \alpha$ 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
- $u, v, w, x, y, \ldots \in \Sigma^*$ for terminal words
- $\alpha, \beta, \gamma, \ldots \in X^*$ for sentences

Context-Free Grammars II

Context-free grammars generate context-free languages:

Definition 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

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

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

Example 5.4

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

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

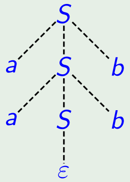
generates the context-free (and non-regular) language

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

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

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)

Ambiguity of CFGs and CFLs

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.

Outline

The Word Problem for Context-Free Languages

Problem 5.8 (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:

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

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

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 5.9 (Leftmost/rightmost analysis)

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

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

Example 5.10

Grammar for arithmetic expressions:

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

Leftmost derivation of $\texttt{(a)*b}$:

$$
\begin{array}{lllllllll}
E & \overset{2}{\Rightarrow}_l & T & \overset{3}{\Rightarrow}_l & T\texttt{*}F & \overset{4}{\Rightarrow}_l & F\texttt{*}F & \overset{5}{\Rightarrow}_l & \texttt{(}E\texttt{)*}F \\
& \overset{2}{\Rightarrow}_l & \texttt{(}T\texttt{)*}F & \overset{4}{\Rightarrow}_l & \texttt{(}F\texttt{)*}F & \overset{6}{\Rightarrow}_l & \texttt{(a)*}F & \overset{7}{\Rightarrow}_l & \texttt{(a)*b}
\end{array}
$$

$\Longrightarrow$ leftmost analysis: 23452467

Rightmost derivation of $\texttt{(a)*b}$:

$$
\begin{array}{lllllllll}
E & \overset{2}{\Rightarrow}_r & T & \overset{3}{\Rightarrow}_r & T\texttt{*}F & \overset{7}{\Rightarrow}_r & T\texttt{*b} & \overset{4}{\Rightarrow}_r & F\texttt{*b} \\
& \overset{5}{\Rightarrow}_r & \texttt{(}E\texttt{)*b} & \overset{2}{\Rightarrow}_r & \texttt{(}T\texttt{)*b} & \overset{4}{\Rightarrow}_r & \texttt{(}F\texttt{)*b} & \overset{6}{\Rightarrow}_r & \texttt{(a)*b}
\end{array}
$$

$\Longrightarrow$ rightmost analysis: 23745246

Reducedness of Context-Free Grammars

General assumption in the following: every grammar is reduced

Definition 5.11 (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

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

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

Outline

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 allowing lookahead on the input: $G \in LL(k)$ iff $L(G)$ recognizable by deterministic PDA with lookahead of k symbols

The Nondeterministic Top-Down Automaton I

Definition 5.12 (Nondeterministic top-down parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$. The nondeterministic top-down parsing automaton of G, $\mathrm{NTA}(G)$, 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 $\pi_i = A \to \beta$, 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: $\mathrm{NTA}(G)$ is nondeterministic iff G contains $A \to \beta \mid \gamma$

The Nondeterministic Top-Down Automaton II

Example 5.13

Grammar for arithmetic expressions (cf. Example 5.10):

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

Leftmost analysis of `(a)*b`:

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