Compiler Construction

Lecture 7: Syntactic Analysis II (Top-Down Parsing)

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/

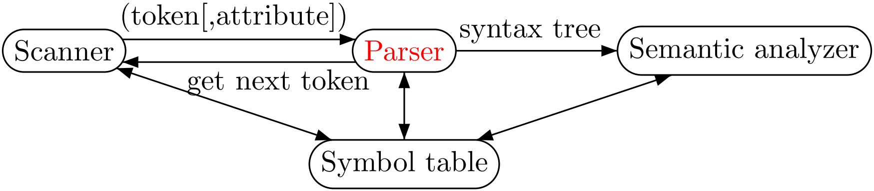
Winter semester 2010/11

Outline

Definition

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.

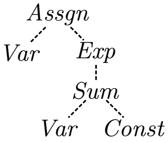
The corresponding program is called a parser:



Example: `...␣x1␣:=y2+␣1␣;␣...`

$\downarrow$ Scanner

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



Context-Free Grammars I

Definition (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 (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,

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

Outline

The Word Problem for Context-Free Languages

Problem 7.1 (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 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)

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 leafs, representation as leftmost derivation

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

Leftmost/Rightmost Analysis I

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

Definition 7.2 (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.

Leftmost/Rightmost Analysis

Example 7.3

Grammar for arithmetic expressions:

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

Leftmost derivation of `(a)*b`:

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

$\Longrightarrow$ leftmost analysis: 23452467

Rightmost derivation of `(a)*b`:

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

$\Longrightarrow$ rightmost analysis: 23745246

Reducedness of Context-Free Grammars

General assumption in the following: every grammar is reduced

Definition 7.4 (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

$$\begin{aligned} S &\Rightarrow^* \alpha A \beta && (A \text{ reachable}) \text{ and} \\ A &\Rightarrow^* w && (A \text{ productive}). \end{aligned}$$

Outline

1. Repetition: Context-Free Grammars and Languages
2. Parsing Context-Free Languages
3. 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: 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

The Nondeterministic Top-Down Automaton I

Definition 7.5 (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 7.6

Grammar for arithmetic expressions (cf. Example 7.3):

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

Leftmost analysis of `(a)*b`:

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

The Nondeterministic Top-Down Automaton III

Theorem 7.7 (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* $z \in [p]^*$,

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

Proof.

$\Longrightarrow$ (soundness): see exercises

$\Longleftarrow$ (completeness): on the board

□