Syntactic Analysis

Sebastian Hack
(based on slides by Reinhard Wilhelm and Mooly Sagiv)

http://compilers.cs.uni-saarland.de

Compiler Construction Core Course 2017
Saarland University

Syntactic Analysis: Topics

- Introduction
  - The task of syntax analysis
  - Automatic generation
  - Error handling
- Context free grammars, derivations, and parse trees
- Grammar Flow Analysis
- Pushdown automata
- Top-down syntax analysis
- Bottom-up syntax analysis

Syntax Analysis (Parsing)

- Functionality

  Input Sequence of symbols (tokens)

  Output Parse tree

- Report syntax errors, e,g., unbalanced parentheses
- Create “‘pretty-printed” version of the program (sometimes)
- In some cases the tree need not be generated (one-pass compilers)

Handling Syntax Errors

- Report and locate the error (symptom)
- Diagnose the error
- Correct the error
- Recover from the error in order to discover more errors (without reporting errors caused by others)

Example

$$a := a * (b + c * d;$$

Error Diagnosis Data

- Line number (may be far from the actual error)
- The current symbol
- The symbols expected in the current parser state

Example Context Free Grammar (Section)

Stat	$\rightarrow$	If_Stat $\mid$
		While_Stat $\mid$
		Repeat_Stat $\mid$
		Proc_Call $\mid$
		Assignment
If_Stat	$\rightarrow$	**if** Cond **then** Stat_Seq **else** Stat_Seq **fi** $\mid$
		if Cond **then** Stat_Seq **fi**
While_Stat	$\rightarrow$	**while** Cond **do** Stat_Seq **od**
Repeat_Stat	$\rightarrow$	**repeat** Stat_Seq **until** Cond
Proc_Call	$\rightarrow$	Name **(** Expr_Seq **)**
Assignment	$\rightarrow$	Name := Expr
Stat_Seq	$\rightarrow$	Stat $\mid$
		Stat_Seq; Stat
Expr_Seq	$\rightarrow$	Expr $\mid$
		Expr_Seq, Expr

Context-Free-Grammar Definition

A context-free-grammar is a quadruple $G = (V_N, V_T, P, S)$ where:

- V_N — finite set of nonterminals
- V_T — finite set of terminals
- $P \subseteq V_N \times (V_N \cup V_T)^*$ — finite set of production rules
- $S \in V_n$ — the start nonterminal

Examples

$$G_0 = (\{E, T, F\}, \{+, *, (,), \mathbf{id}\}, P_0, E)$$

$$P_0 = \left\{ \begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (E) \mid \mathbf{id} \end{array} \right\}$$

$$G_1 = (\{E\}, \{+, *, (,), \mathbf{id}\}, P_1, E)$$

$$P_1 = \{E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}\}$$

Derivations

Given a context-free-grammar $G = (V_N, V_T, P, S)$

- $\varphi \implies \psi$

  if there exist $\varphi_1, \varphi_2 \in (V_N \cup V_T)^*$, $A \in V_N$
  - $\varphi \equiv \varphi_1 \, A \, \varphi_2$
  - $A \to \alpha \in P$
  - $\psi \equiv \varphi_1 \, \alpha \, \varphi_2$
- $\varphi \stackrel{*}{\implies} \psi$ reflexive transitive closure
- The language defined by G

$$L(G) = \{w \in V_T^* \mid S \stackrel{*}{\implies} w\}$$

Reduced and Extended Context Free Grammars

A nonterminal A is

reachable: There exist φ_1, φ_2 such that $S \overset{*}{\Longrightarrow} \varphi_1 A \varphi_2$

productive: There exists $w \in V_T^*$, $A \overset{*}{\Longrightarrow} w$

Removal of unreachable and non-productive nonterminals and the productions they occur in doesn't change the defined language.

A grammar is reduced if it has neither unreachable nor non-productive nonterminals.

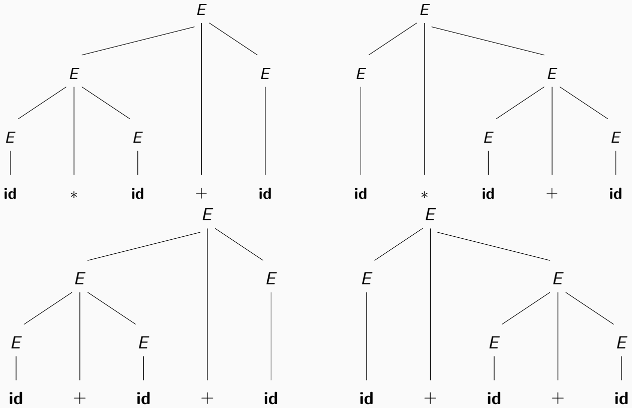
A grammar is extended if a new startsymbol S' and a new production $S' \rightarrow S$ are added to the grammar.

From now on, we only consider reduced and extended grammars.

Syntax Tree (Parse Tree)

- An ordered tree.
- Root is labeled with S.
- Internal nodes are labeled by nonterminals.
- Leaves are labeled by terminals or by ε.
- For internal nodes n:
  If n labeled by N and its children $n.1, \ldots, n.n_p$ are labeled by $N_1, \ldots, N_{n_p}$, then $N \rightarrow N_1, \ldots, N_{n_p} \in P$.

Examples



Leftmost (Rightmost) Derivations

Given a context-free grammar $G = (V_N, V_T, P, S)$

- $\varphi \underset{lm}{\Longrightarrow} \psi$ if there exist $\varphi_1 \in V_T^*$, $\varphi_2 \in (V_N \cup V_T)^*$, and $A \in V_N$
  - $\varphi \equiv \varphi_1\, A\, \varphi_2$
  - $A \to \alpha \in P$
  - $\psi \equiv \varphi_1\, \alpha\, \varphi_2$

replace leftmost nonterminal

- $\varphi \underset{rm}{\Longrightarrow} \psi$ if there exist $\varphi_2 \in V_T^*$, $\varphi_1 \in (V_N \cup V_T)^*$, and $A \in V_N$
  - $\varphi \equiv \varphi_1\, A\, \varphi_2$
  - $A \to \alpha \in P$
  - $\psi \equiv \varphi_1\, \alpha\, \varphi_2$

replace rightmost nonterminal

- $\varphi \overset{*}{\underset{lm}{\Longrightarrow}} \psi$, $\varphi \overset{*}{\underset{rm}{\Longrightarrow}} \psi$ are defined as usual

Ambiguous Grammars

- A grammar that has (equivalently)
  - two leftmost derivations for the same string,
  - two rightmost derivations for the same string,
  - two syntax trees for the same string.

  is called ambiguous.
- It is undecidable if a grammar is ambiguous or not
- There are unambiguous grammars (whose languages) cannot be accepted with a deterministic push-down automaton
- For parsing, we're interested in grammars that can be accepted with a deterministic push-down automaton