

Syntax and Semantics

Hui Chen

Computer Science
Virginia State University
Petersburg, Virginia

January 20, 2016

Outline

- ▶ Backus-Naur Form
 - ▶ derivations, parse trees, ambiguity, descriptions of operator precedence and associativity, and extended Backus-Naur Form.
- ▶ Attribute grammars
- ▶ Operational axiomatic and denotational semantics

Chomsky Hierarchy

- ▶ Also called Chomsky-Schützenberger Hierarchy (Noam Chomsky, 1956)

Class	Grammar	Language	Automaton
Type-0	Unrestricted	Recursively enumerable	Turing machine (TM)
Type-1	Context-sensitive	Context-sensitive	Linear-bounded automaton (LBA)
Type-2	Context-free	Context-free	Pushdown automaton (PDA)
Type-3	Regular	Regular	Deterministic finite automaton (DFA)

- ▶ A strictly nested sets of classes of formal grammars, i.e.,

$$\text{Type-0} \supset \text{Type-1} \supset \text{Type-2} \supset \text{Type-3}$$

- ▶ Context-free and regular grammars are of our primary concern

Context-Free Grammar (CFG)

- ▶ A CFG is a quadruple, $G = (V, T, P, S)$ where
 - ▶ V : the set of variables or non-terminals
 - ▶ T : the set of terminals
 - ▶ P : the set of productions of the form $A \rightarrow \gamma$ where A is a single variable, i.e., $A \in V$ and γ is string of terminals and variables, i.e., $\gamma \in (V \cup T)^*$
 - ▶ S : the start symbol and $S \in V$
- ▶ To describe the grammar of a programming language,
 - ▶ Terminals are lexemes or tokens

Example: A Simple Programming Language¹

- ▶ Operators: $+$ and $*$ represent addition and multiplication, respectively
- ▶ Arguments are identifiers consisting *only* of letters a , b , and digits 0, 1
- ▶ An example statement in the language,

$$(a + b) * (a + b + 1)$$

¹This is an example given in [Hopcroft et al., 2006]

CFG of the Simple Language

- ▶ The language can be specified using a CFG as,

$$G = (\{E, I\}, T, P, E)$$

where

- ▶ E and I are the two variables, and E is the start symbol
- ▶ T , the terminals are the set of symbols $\{+, *, (,), a, b, 0, 1\}$
- ▶ P is the productions, i.e.,

$$1 \quad E \rightarrow I$$

$$2 \quad E \rightarrow E + E$$

$$3 \quad E \rightarrow E * E$$

$$4 \quad E \rightarrow (E)$$

$$5 \quad I \rightarrow a$$

$$6 \quad I \rightarrow a$$

$$7 \quad I \rightarrow Ia$$

$$8 \quad I \rightarrow Ib$$

$$9 \quad I \rightarrow I0$$

$$10 \quad I \rightarrow I1$$

Backus-Naur Form (BNF)

- ▶ John Backus (1959) and Peter Naur (1960) developed to describe syntax of ALGOL 58 and 60
- ▶ BNF is equivalent to context-free grammars
- ▶ Widely used today for describing syntax of programming languages

Production Rules in BNF

- ▶ Nonterminals (or variables in CFG, called *abstractions*) are often enclosed in angle brackets
- ▶ A start symbol is a special element of the nonterminals of a grammar
- ▶ Grammar: a finite non-empty set of rules
- ▶ Examples of BNF rules:

`<ident_list > → identifier`

`<ident_list > → identifier, <ident_list >`

`<if_stmt > → if <logic_expr> then <stmt >`

More than one RHS

- ▶ An abstraction (or a nonterminal symbol) can have more than one right-hand sides
- ▶ Example: applying this rule, we can rewrite,

$$\langle \text{ident_list} \rangle \rightarrow \text{identifier}$$

$$\langle \text{ident_list} \rangle \rightarrow \text{identifier}, \langle \text{ident_list} \rangle$$

as

$$\langle \text{ident_list} \rangle \rightarrow \text{identifier} \mid \text{identifier}, \langle \text{ident_list} \rangle$$

- ▶ Another example:

$$\langle \text{stmt} \rangle \rightarrow \langle \text{single_stmt} \rangle \mid \text{begin } \langle \text{stmt_list} \rangle \text{ end}$$

Lists

- ▶ Syntactic lists are described using recursion

$$\langle \text{ident_list} \rangle \rightarrow \text{ident} \mid \text{ident}, \langle \text{ident_list} \rangle$$

Derivation

- ▶ A repeated application of rules, starting with the start symbol and ending with a sentence (all terminal symbols)
 - ▶ Every string of symbols in a derivation is a sentential form
 - ▶ A sentence is a sentential form that has only terminal symbols
 - ▶ A leftmost derivation is one in which the leftmost nonterminal in each sentential form is the one that is expanded
 - ▶ A derivation may be neither leftmost nor rightmost

An Example of Derivation

- ▶ Given a grammar,

$$\langle \text{program} \rangle \rightarrow \langle \text{stmts} \rangle$$

$$\langle \text{stmts} \rangle \rightarrow \langle \text{stmt} \rangle \mid \langle \text{stmt} \rangle ; \langle \text{stmts} \rangle$$

$$\langle \text{stmt} \rangle \rightarrow \langle \text{var} \rangle = \langle \text{expr} \rangle$$

$$\langle \text{var} \rangle \rightarrow a \mid b \mid c \mid d$$

$$\langle \text{expr} \rangle \rightarrow \langle \text{term} \rangle + \langle \text{term} \rangle \mid \langle \text{term} \rangle - \langle \text{term} \rangle$$

$$\langle \text{term} \rangle \rightarrow \langle \text{var} \rangle \mid \text{const}$$

- ▶ we can have the following derivation,

$$\langle \text{program} \rangle \Rightarrow \langle \text{stmts} \rangle \Rightarrow \langle \text{stmt} \rangle \Rightarrow \langle \text{var} \rangle = \langle \text{expr} \rangle$$

$$\Rightarrow a = \langle \text{expr} \rangle \Rightarrow a = \langle \text{term} \rangle + \langle \text{term} \rangle$$

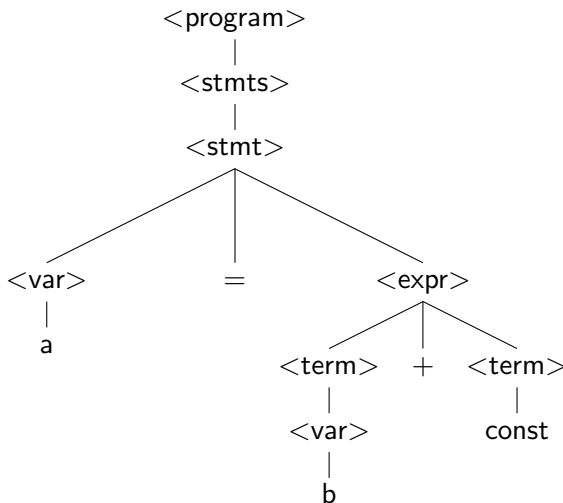
$$\Rightarrow a = \langle \text{var} \rangle + \langle \text{term} \rangle$$

$$\Rightarrow a = b + \langle \text{term} \rangle$$

$$\Rightarrow a = b + \text{const}$$

Parse Tree

- ▶ A parse tree is a hierarchical representation of a derivation
- ▶ Example:

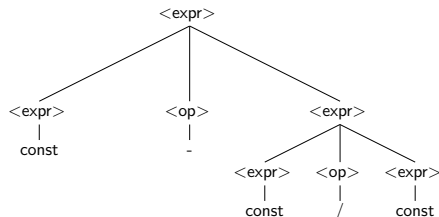
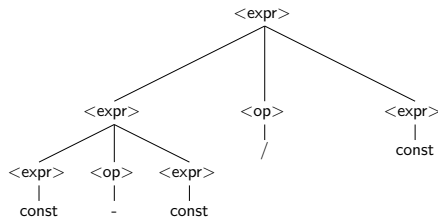


Ambiguity in Grammars

- ▶ A grammar is *ambiguous* if and only if it generates a sentential form that has two or more distinct parse trees

Example of Ambiguous Grammar and Parse Trees

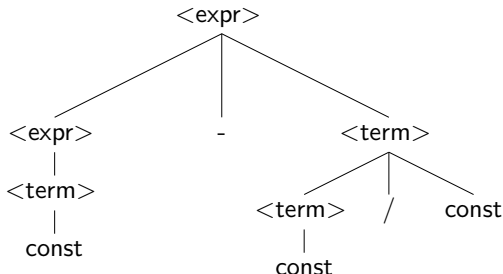
$$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \mid \text{const}$$

$$\langle \text{op} \rangle \rightarrow / \mid -$$


Unambiguous Grammar

- ▶ If we use the parse tree to indicate precedence levels of the operators, we cannot have ambiguity
- ▶ Example:

$$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle - \langle \text{term} \rangle \mid \langle \text{term} \rangle$$

$$\langle \text{term} \rangle \rightarrow \langle \text{term} \rangle / \text{const} \mid \text{const}$$


Associativity of Operators

- ▶ Operator associativity can also be indicated by a grammar
- ▶ Example: compare the following two grammars

1. Ambiguous grammar

$$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle + \langle \text{expr} \rangle \mid \text{const}$$

2. Unambiguous grammar

$$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle + \text{const} \mid \text{const}$$

Extended BNF (EBNF)

- ▶ The extensions *do not* enhance the descriptive power of BNF; they only increase its *readability* and *writability*
- ▶ Optional parts are placed in brackets [], e.g.,

$$\langle \text{proc_call} \rangle \rightarrow \text{ident} [(\langle \text{expr_list} \rangle)]$$

- ▶ Alternative parts of RHSs are placed inside () and separated via |, e.g.,

$$\langle \text{term} \rangle \rightarrow \langle \text{term} \rangle (+|-) \text{const}$$

- ▶ Repetitions (0 or more times) are placed inside { },

$$\langle \text{ident} \rangle \rightarrow \text{letter} \{ \text{letter} | \text{digit} \}$$

- ▶ Can you rewrite the above examples without using extensions?

Recent Variations in EBNF

- ▶ Alternative RHSs are put on separate lines
- ▶ Use of `a :` instead of \rightarrow
- ▶ Use of `opt` for optional parts
- ▶ Use of `oneof` for choices

Discussion on semantics to follow ...

To be continued.

Summary

- ▶ BNF and context-free grammars are equivalent meta-languages
 - ▶ Well-suited for describing the syntax of programming languages
- ▶ An attribute grammar is a descriptive formalism that can describe both the syntax and the semantics of a language
- ▶ Three primary methods of semantics description
 - ▶ Operation, axiomatic, denotational

References I



Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006).

Introduction to Automata Theory, Languages, and Computation (3rd Edition).

Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.