Grammar

Definition

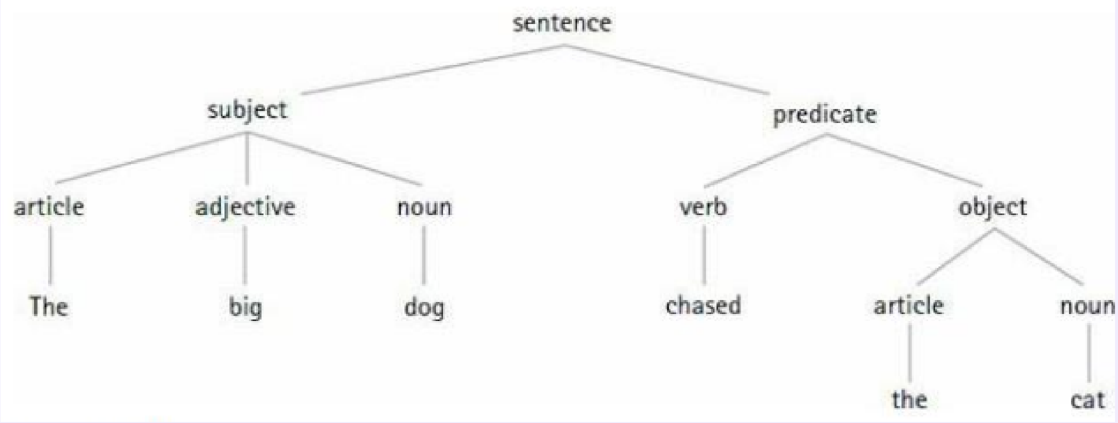
1. An alphabet N of grammar symbols called *nonterminals*
2. An alphabet T of symbols called *terminals*
    - The terminals are distinct from the nonterminals
3. A specific nonterminal S, called the *start symbol*
4. A finite set of productions of the form $\alpha \to \beta$, where α and β are strings over the alphabet $N \cup T$ with the restriction that α is not the empty string
    - There is at least one production with only the start symbol S on the left side
    - Each nonterminal must appear on the left side of some production

English Grammar

- We can think of an English sentence as a string of characters if we agree to let the alphabet consist of the usual letters together with blank characters, periods, commas, and so one
- To parse a sentence means to break it up into parts that conform to grammar conventions

Example

"The big dog chased the cat"

- Subject = The big dog
- Predicate = chased the cat
- 

- To denote that fact that a sentence consists of a subject followed by a predicate, we have the following rule
    - Sentence -> subject predicate

Structure

- Let L be a language over an alphabet A
- Then a grammar for L consists of set of grammar rules of the form where α and β denote strings of symbols taken from A and from a set of grammar symbols disjoint from A

Production

$\alpha \to \beta$ is called a *production*, and can be read in several different ways

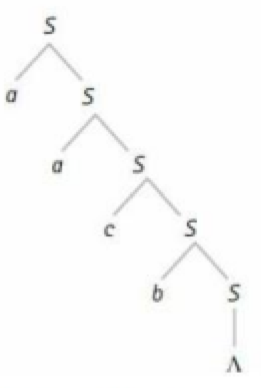
- Replace α by β
- α produces β
- α rewrites to β
- α reduces to β

Start Symbol

- Every grammar has a special grammar symbol called the *start symbol*
- There must be at least one production with the left side consisting of only the start symbol

> **Example**
> If S is the start symbol for a grammar, then there must be at least one production of the form
>
> $$S \to \beta$$

> **Example**
>
> Let $A = \{a, b, c\}$
> A grammar for the language A^* can be described by the following 4 productions
>
> - $S \to \Lambda$
> - $S \to aS$
> - $S \to bS$
> - $S \to cS$
>   Derivations
> - $S \Rightarrow aS$
> - $S \Rightarrow aS \Rightarrow aaS$
> - $S \Rightarrow aS \Rightarrow aaS \Rightarrow aacS$
> - $S \Rightarrow aS \Rightarrow aaS \Rightarrow aacS \Rightarrow aacbS$

- $S \Rightarrow aS \Rightarrow aaS \Rightarrow aacS \Rightarrow aacbS \Rightarrow aacb\Lambda = aacb$
- 

Derivation

Definition

If x and y are sentential forms and $\alpha \rightarrow \beta$ is a production, then the replacement of α by β in $x\alpha y$ is called a *derivation step*, which we denote by writing

$$x\alpha y \Rightarrow x\beta y$$

A *derivation* is a sequence of derivation steps

Language of a Grammar

Definition

If G is a grammar with a start symbol S and the set of terminals T, then the language of G is the set

$$L(G) = \{s|s \in T^* andS \Rightarrow^+ s\}$$

Combining Grammars

- Suppose M and N are languages whose grammars have disjoint sets of nonterminal
- Suppose that the start symbols are A and B respectively

Union Rule

The language $M \cup N$ starts with two productions

- $S \rightarrow A|B$

Product Rule

The language MN starts with one production

- $S \rightarrow AB$

Closure Rule

The language M^* starts with two productions

- $S \rightarrow AS|A$

Meaning and Ambiguity

Ambiguous Grammar

Definition

When its language contains some string that has two different parse trees

Syntax

Definition

The *syntax* of a programming language is a precise description of all grammatically correct programs

History

- Formal methods for defining syntax have been used since the emergence of Algol in the early 1960s

Types

Lexical Syntax

Definition

Defines the rules for basic symbols including identifiers, literals, operators, and punctuation

Phases

Scanning Phase

- Translator collects character sequences from the input program and forms tokens

Parsing Phase

- Translator processes tokens to determine syntactic structure

Tokens

- Several categories
- Described by regular expressions (regex)

Reserved Words / Keywords

- Things like `if` or `while`

Literals / Constants

- Things such as `42` (numeric literal) or `hello` (string literal)

Special Symbols

- Things like `:`, `,`, or `+`

Identifiers

- Things like `x24`, `monthly_balance`, or put `char`

Regex

Example

$(a|b) * c$ is a regex indicating 0 or more repetitions (*repetition*) of either the characters a or b (*choice*), followed by a single character c (*concatenation*)

Concatenation

Repetition

Choice / Selection

Concrete Syntax

Definition

Refers to the actual representation of its programs using lexical symbols as its alphabet

Abstract Syntax

Definition

Carries only the essential program information, without concern for syntactic idiosyncrasies like punctuation or parenthesis

Context Free Grammar

Definition

Has a set of productions P, set of terminal symbols T, and set of nonterminal symbols N, one of which S, is distinguished as the *start symbol*

Backus-Naur Form (BNF)

- Has been widely used to define the syntax of programming languages

Ambiguity

- Either the grammar must be revised to remove the ambiguity or a **disambiguating rule** must be stated to establish which structure is meant
- The usual way to revise a grammar is to write a new grammar rule (called a "term") that establishes a "precedence cascade" to force the matching of the ”*” at a lower point in the parse tree

Example

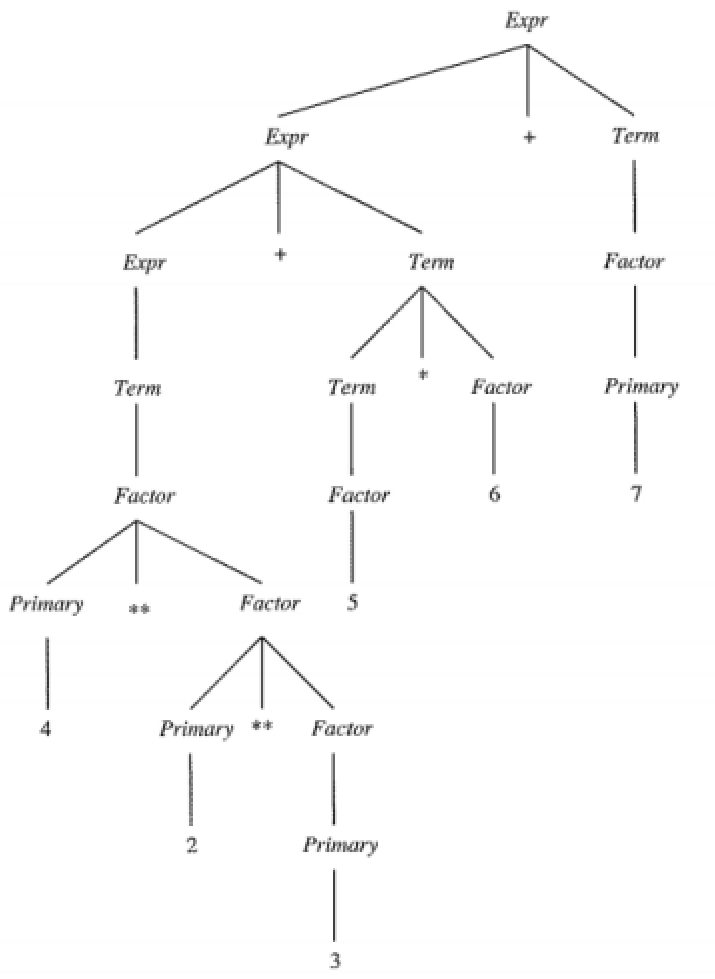
- 3 + 4 * 5
- We would choose the tree that keeps 4 * 5 together

Rules

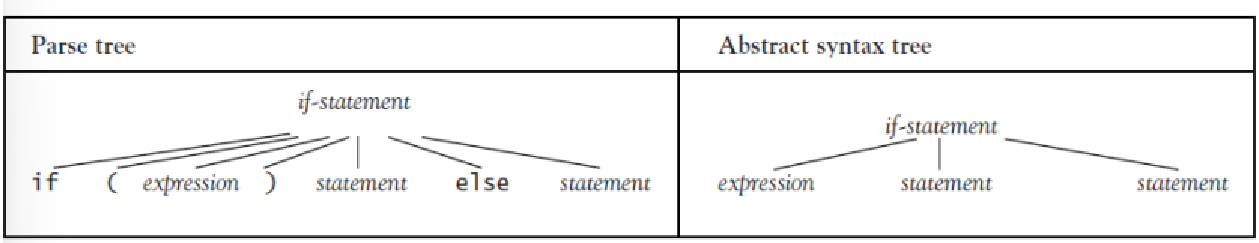
Expressions

- Expr -> expr + term | expr - term | term
- Term => 0 | ... | 9 | (Expr)

Example



If Statements

- Rule: if-statement -> if (*expression*) *statement* else *statement*
- 

Parsing Techniques and Tools

Recognizer

- A program that accepts or rejects strings, based on whether they are legal strings in the language

Parser Generator

- Both top-down and bottom-up parsing can be automated by a program