

# *Syntax Analysis*

Amitabha Sanyal

([www.cse.iitb.ac.in/~as](http://www.cse.iitb.ac.in/~as))

Department of Computer Science and Engineering,  
Indian Institute of Technology, Bombay

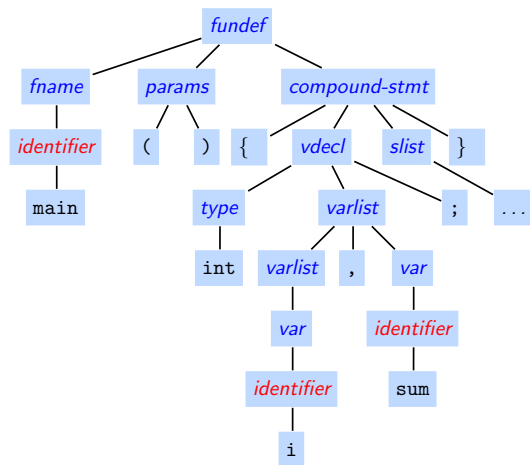


January 2014

# Syntax analysis – example

Syntax analysis discovers the larger structures in a program.

```
main ()  
{  
    int i,sum;  
    sum = 0;  
    for (i=1; i<=10; i++)  
        sum = sum + i;  
    printf("%d\n",sum);  
}
```



# Parsing

A **syntax analyzer** or **parser**

- Ensures that the input program is well-formed by attempting to group tokens according to certain rules. This is **syntax checking**.

# Parsing

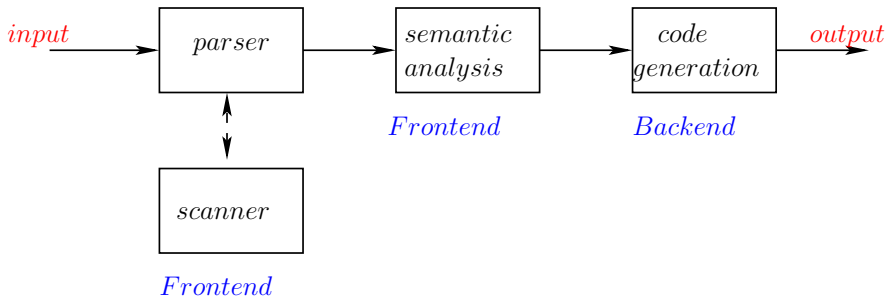
A **syntax analyzer** or **parser**

- Ensures that the input program is well-formed by attempting to group tokens according to certain rules. This is **syntax checking**.
- - May also create the hierarchical structure that arises out of such grouping.
  - The tree like representation of the structure is called a **parse tree**.
  - This information is required by subsequent phases.

# Place of a parser in a compiler organization

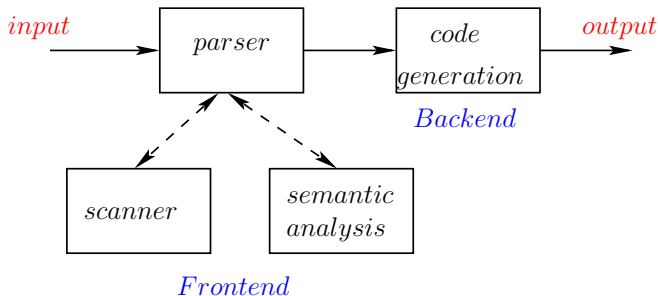
Where is the place of the parser in the overall organization of the compiler?

1. **Parser driven syntax tree creation.** The parser creates the entire syntax tree and passes control to the later stages.



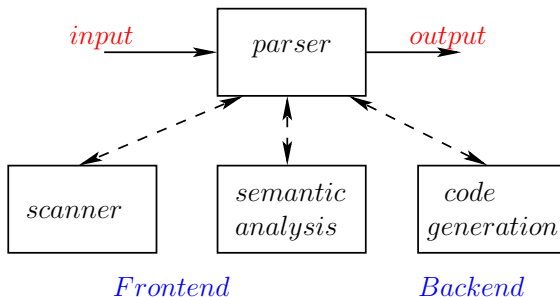
# Place of a parser in a compiler organization

2. **Parser driven front-end.** The parser also does the semantic analysis along with parsing.



# Place of a parser in a compiler organization

3. **Parser driven compilation.** The entire compilation is interleaved along with parsing.



# Parser Construction

How are parsers constructed ?

- Till early seventies, parsers (in fact the entire compiler) were written manually.
- A better understanding of parsing algorithms has resulted in tools that can automatically generate parsers.
- Examples of parser generating tools:
  - Yacc/Bison: Bottom-up (LALR) parser generator
  - Antlr: Top-down (LL) scanner cum parser generator. (Terence Parr)
  - PCCTS: Precursor of Antlr (Terence Parr)
  - COCO/R: Lexer and Parser Generators in various languages, generates recursive descent parsers (Hanspeter Mossenbock).
  - Java Compiler Compiler (JavaCC)
  - ...

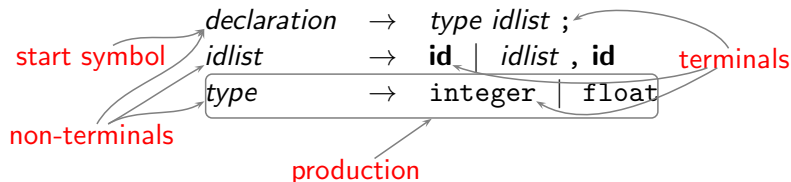


# Specification of syntax

- To **check** whether a program is well-formed requires a **specification** of what is a well-formed program:
  - 1 The specification should be **unambiguous**.
  - 2 The specification should be **correct** and **complete**. Must cover all the syntactic details of the language
  - 3 the specification must be **convenient** to use by both language designer and the implementer

A **context free grammar** meets these requirements.

# Context Free Grammar (CFG)



A CFG  $G$  is a 4-tuple  $(N, T, S, P)$ , where :

- 1  $N$  is a finite set of nonterminals.
- 2  $T$  is a finite set of terminals.
- 3  $S$  is a special nonterminal (from  $N$ ) called the *start* symbol.
- 4  $P$  is a finite set of production rules of the form such as  $A \rightarrow \alpha$ , where  $A$  is from  $N$  and  $\alpha$  from  $(N \cup T)^*$