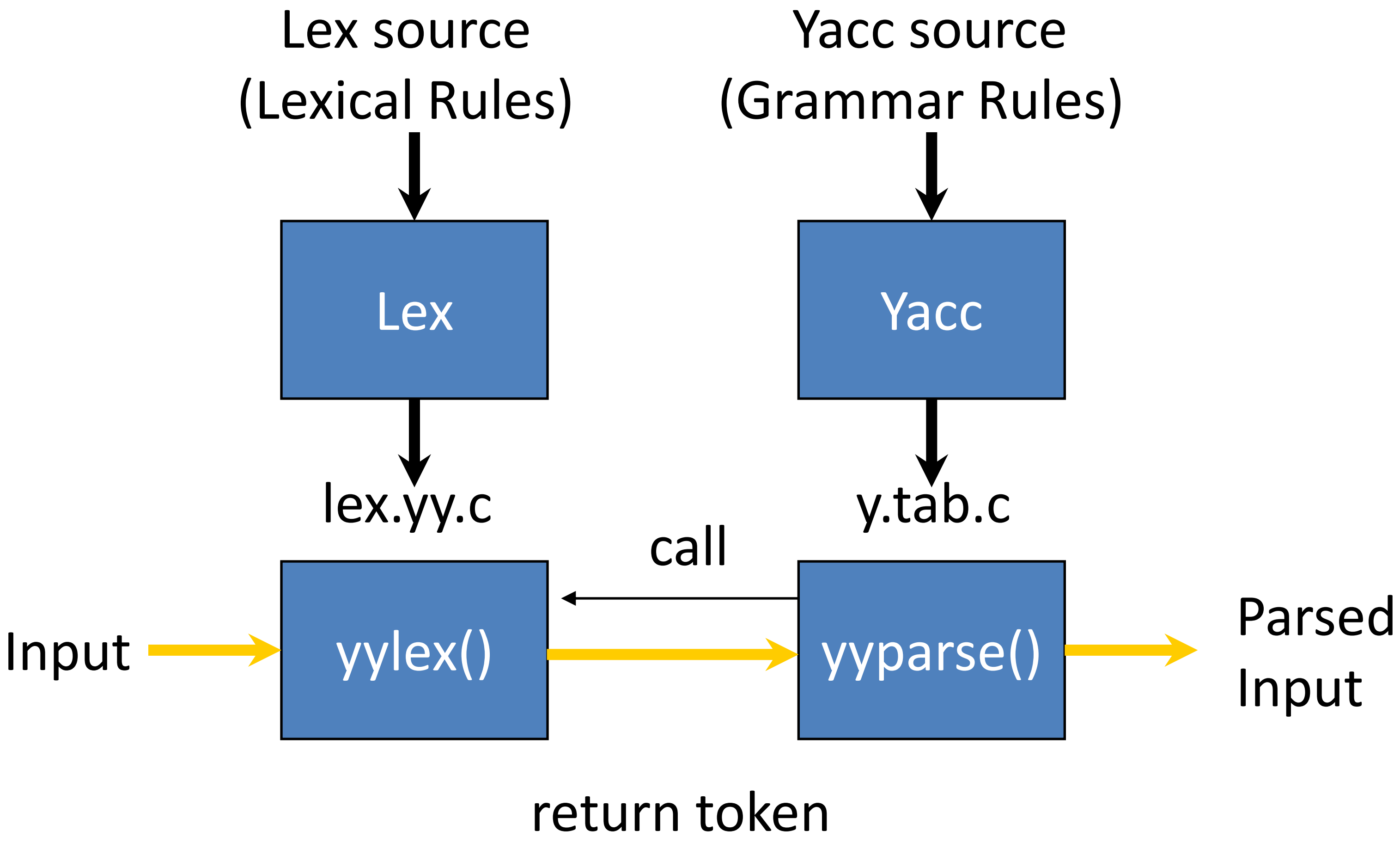
Yacc - Yet Another Compiler-Compiler

Introduction

- What is **YACC** ?
  - **Tool which will produce a parser for a given grammar**.
  - YACC (Yet Another Compiler Compiler) is a program designed to compile a LALR(1) grammar and to produce the source code of the syntactic analyzer of the language produced by this grammar.

Lex with Yacc



Lex source
(Lexical Rules)

Yacc source
(Grammar Rules)

Lex

Yacc

lex.yy.c

y.tab.c

call

Input

yylex()

yyparse()

Parsed
Input

return token

YACC File Format

%{

C declarations

%}

yacc declarations

%%

Grammar rules

%%

Additional C code

- **Comments enclosed in /* ... */ may appear in any of the sections.**

Definitions Section

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM
%start expr
```

It is a terminal

Start Symbol

- The first non-terminal specified in the grammar specification section.
- To overwrite it with %start declaraction.

```
%start  non-terminal
```

Rules Section

- This section defines grammar
- Example

```
expr : expr '+' term | term;
term : term '*' factor | factor;
factor : '(' expr ')' | ID | NUM;
```

Rules Section

- Normally written like this
- Example:

```
expr   : expr '+' term
       | term
       ;
term   : term '*' factor
       | factor
       ;
factor : '(' expr ')'
       | ID
       | NUM
       ;
```

The Position of Rules

```
expr : expr '+' term    { $$ = $1 + $3; }
     | term             { $$ = $1; }
     ;
term : term '*' factor  { $$ = $1 * $3; }
     | factor           { $$ = $1; }
     ;
factor : '(' expr ')'   { $$ = $2; }
       | ID
       | NUM
       ;
```

YACC Declaration Summary

`%start'

Specify the grammar's start symbol

`%union'

Declare the collection of data types that semantic values may have

`%token'

Declare a terminal symbol (token type name) with no precedence or associativity specified

`%type'

Declare the type of semantic values for a nonterminal symbol

YACC Declaration Summary

`%right'

Declare a terminal symbol (token type name) that is right-associative

`%left'

Declare a terminal symbol (token type name) that is left-associative

`%nonassoc'

Declare a terminal symbol (token type name) that is nonassociative (using it in a way that would be associative is a syntax error,
ex: x *op*. y *op*. z is syntax error)