Parul® University

COMPILER DESIGN
SUBJECT CODE: 203105351

Kapil Raghuwanshi, Assistant Professor
Computer Science & Engineering

CHAPTER-3

Top Down Parsing

Conten

- Introduction to YACC
- Error recover by YACC
- Example of YACC Specific

Parul® University

Introduction to YACC

- A parser generator is a program that takes as input a specification of a syntax, and produces as output a procedure for recognizing that language. They are also known as compiler-compilers.

- YACC (Yet another compiler-compiler) is a LALR(1) parser generator.

- It provides a tool to produce a parser for a given grammar LALR (1) grammar.
- It is used to produce source code of syntactic analyser of the language by LALR(1) grammar.

Introduction to YACC

- It generates c code of syntax analyzer of parserr.
- YACC (Yet another compiler-compiler) is a LALR(1) parser generator.
- It provides a tool to produce a parser for a given grammar LALR (1) grammar.
- It is used to produce source code of syntactic analyzer of the language by LALR(1) grammar.

Introduction to YACC

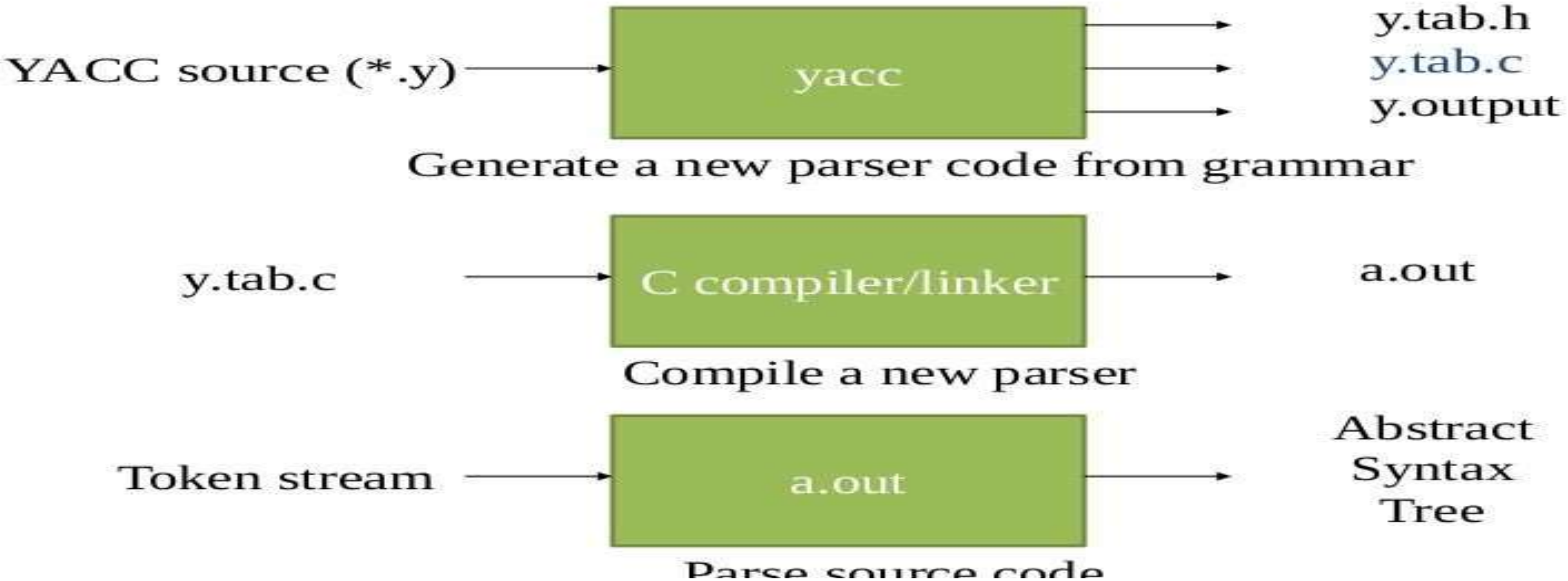
These are some points about YACC:

- **Input: A CFG- file.y**
- **Output: A parser y.tab.c (yacc)**
- The output file "file.output" contains the parsing tables.
- The file "file.tab.h" contains declarations.
- The parser called the yyparse ().
- Parser expects to use a function called yylex () to get tokens.

Parul® University

DIGITAL LEARNING CONTENT

How does YACC work?



Parul® University

How does YACC work?

- The tool yacc can be used to generate automatically an LALR parser.
- **Input File:**
  YACC input file is divided in three parts.

1. Definition

1. Rules

1. Subroutines

How does YACC work?

- The definition part includes information about the tokens used in the syntax definition:

```
%token NUMBER
%token ID
```

Yacc automatically assigns numbers for tokens, but it can be overridden by

```
%token NUMBER 621
```

The definition part can include C code external to the definition of the parser and variable declarations, within **%{** and **%}** in the first column

How does YACC work?

Input File: Rule Part:

- The rules part contains grammar definition in a modified BNF form.
- Actions is C code in { } and can be embedded inside (Translation schemes).

How does YACC work?

Input File: Auxiliary Routines Part:

- The auxiliary routines part is only C code.
- It includes function definitions for every function needed in rules part.
- It can also contain the main() function definition if the parser is going to be run as a program.
- The main() function must call the function yyparse().

How does YACC work?

Output Files::

- The output of YACC is a file named **y.tab.c**
- If it contains the **main()** definition, it must be compiled to be executable.
- Otherwise, the code can be an external function definition for the function **int yyparse()**
- If called with the **–d** option in the command line, Yacc produces as output a header file **y.tab.h** with all its specific definition (particularly important are token definitions to be included, for example, in a Lex input file).
- If called with the **–v** option, Yacc produces as output a file **y.output** containing a textual description of the LALR(1) parsing table used by the parser. This is useful for tracking down how the parser solves conflicts.

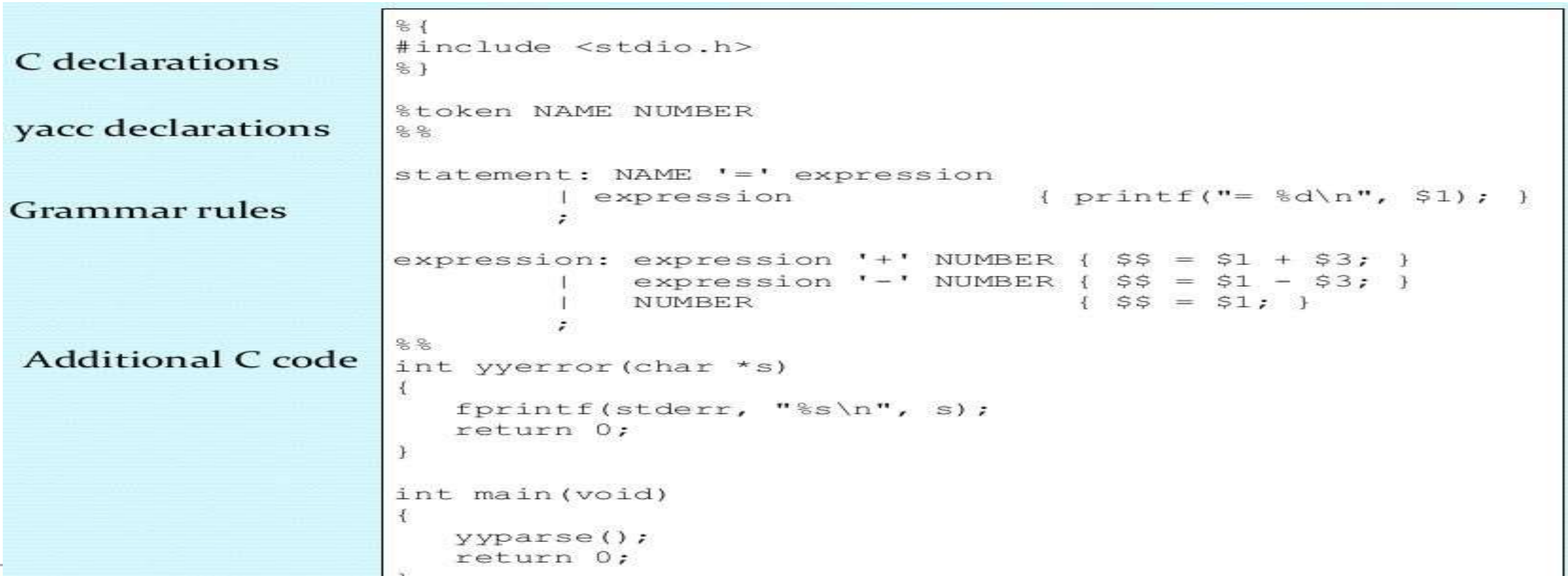
YAAC Format

```
%{
        C declarations
%}

        yaac declarations
%%

        Grammar rules
%%
         Additional c code
```

Parul® University

DIGITAL LEARNING CONTENT

Example of YAAC



C declarations

yacc declarations

Grammar rules

Additional C code

```
%{
#include <stdio.h>
%}

%token NAME NUMBER
%%

statement: NAME '=' expression
         | expression                { printf("= %d\n", $1); }
         ;

expression: expression '+' NUMBER { $$ = $1 + $3; }
          |  expression '-' NUMBER { $$ = $1 - $3; }
          |  NUMBER                { $$ = $1; }
          ;
%%
int yyerror(char *s)
{
   fprintf(stderr, "%s\n", s);
   return 0;
}

int main(void)
{
   yyparse();
   return 0;
```

YAAC Example

Example:

Yacc File (.y)

```
%{
#include <ctype.h>
#include <stdio.h>
#define YYSTYPE double /* double type for yacc stack */
%}

%%
Lines : Lines S '\n' { printf("OK \n"); }
             | S '\n’
             | error '\n' {yyerror("Error: reenter last line:");

             yyerrok; };
S             : '(' S ')’
             | '[' S ']’
             | /* empty */ ;
%%
```

```
#include "lex.yy.c"

void yyerror(char * s)
/* yacc error handler */
{
fprintf (stderr, "%s\n", s);
}

int main(void)
{
return yyparse();
}
```