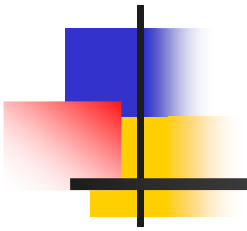
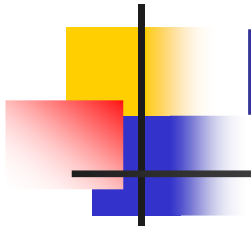


Lexical Syntax & Semantic Analyzer

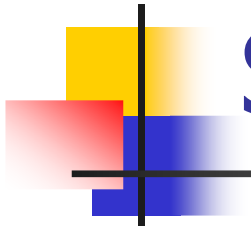
Asad Ali

GIK Institute of Engineering Sciences & Technology



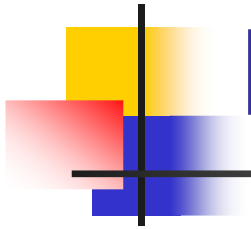
Lexical Analysis

- Identification of lexemes to tokens with additional information like type, LOD and LOR maintained in the Symbol Table.
- Symbol table implemented using chaining method of hashing.
- Panic mode used for error recovery.
- Forward feed / character pre-fetch.



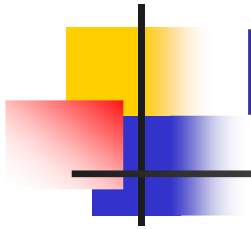
Syntax Analysis

- Finite state automata for the following constructs have been implemented:
  - For, while, assignment Stmt, pre-processor, Declarations, functions and comments.
- Syntax of assignment statement has also been verified using the right recursive grammar with non recursive predictive parsing.



Program Flow

```
Start:
    perform init and load tables.
    open source file
    while (not end of file)
    {
        token = Get Next Token
        if valid token
            Insert Into Symbol table
        if the token is EQ followed by identifier
            perform non recursive predictive parsing using
            first set
        if no syntax error
            check semantics of assignment statement
    }
```



Lexical Analysis

- Identification of lexemes to tokens with additional information like type, LOD and LOR maintained in the Symbol Table.
- Symbol table implemented using chaining method of hashing.
- Static Table of List of Known Lexemes and their tokens

  String Tokens[40], Lexemes[40]
- Implemented using FSA (Regular Grammar)

Known Keywords

Lexemes	Tokens
Lexeme[0]="float"; Lexeme[1]="int"; Lexeme[2]="double"; Lexeme[3]="char"; Lexeme[4]="include"; Lexeme[5]="iostream.h"; Lexeme[6]="conio.h"; Lexeme[7]="void"; Lexeme[8]="main"; Lexeme[9]="for"; Lexeme[10]="if"; Lexeme[11]="while"; Lexeme[12]="struct"; Lexeme[13]="do";	Token[0]="DEC"; Token[1]="DEC"; Token[2]="DEC"; Token[3]="DEC"; Token[4]="RW"; Token[5]="LIB"; Token[6]="LIB"; Token[7]="DEC"; Token[8]="main"; Token[9]="for"; Token[10]="if"; Token[11]="while"; Token[12]="struct"; Token[13]="do";

Known Keywords(2)

Lexemes

```
Lexeme[14]="long";
Lexeme[15]="stdio.h";
Lexeme[16]="+";
Lexeme[17]="-";
Lexeme[18]="*";
Lexeme[19]="/";
Lexeme[20]="=";
Lexeme[21]=";";
Lexeme[22]="{";
Lexeme[23]="}";
Lexeme[24]="[";
Lexeme[25]="]";
Lexeme[26]="(";
```

Tokens

```
Token[14]="DEC";
Token[15]="LIB";
Token[16]="ADD";
Token[17]="SUB";
Token[18]="MUL";
Token[19]="DIV";
Token[20]="EQ";
Token[21]="SC";
Token[22]="CBO";
Token[23]="CBC";
Token[24]="SBO";
Token[25]="SBC";
Token[26]="BRO";
```

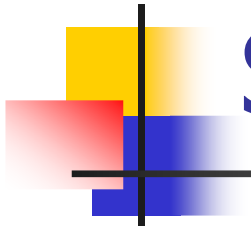
Known Keywords(3)

Lexemes

```
Lexeme[27]=")";
Lexeme[28]="<=";
Lexeme[29]=">=";
Lexeme[30]="<";
Lexeme[31]=">";
Lexeme[32]="+=";
Lexeme[33]="-=";
Lexeme[34]="*=";
Lexeme[35]="/=";
Lexeme[36]="||";
Lexeme[37]="&&";
Lexeme[38]="!=";
Lexeme[39]=",";
```

Tokens

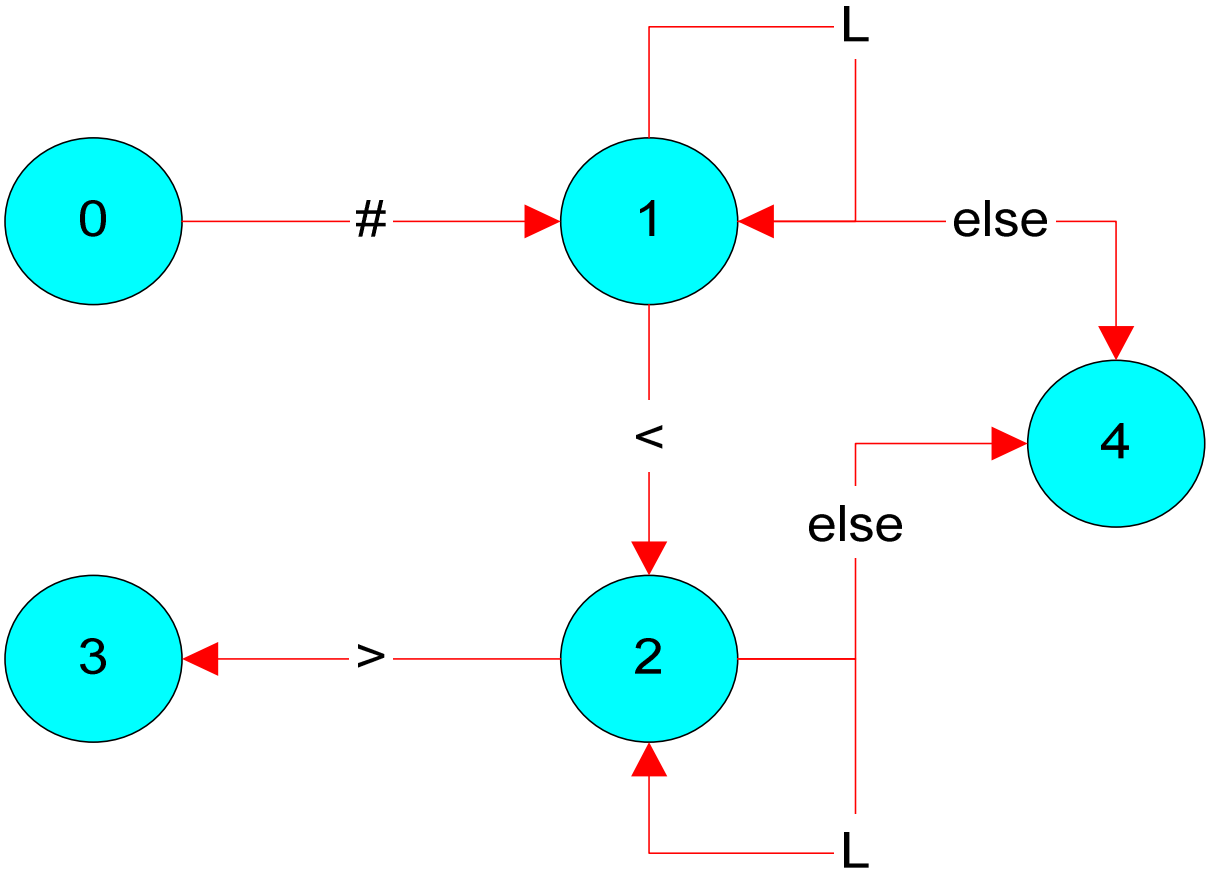
```
Token[27]="BRC";
Token[28]="LE";
Token[29]="GE";
Token[30]="LT";
Token[31]="GT";
Token[32]="PEQ";
Token[33]="SEQ";
Token[34]="MEQ";
Token[35]="DEQ";
Token[36]="OR";
Token[37]="AND";
Token[38]="NEQ";
Token[39]="COMMA";
```



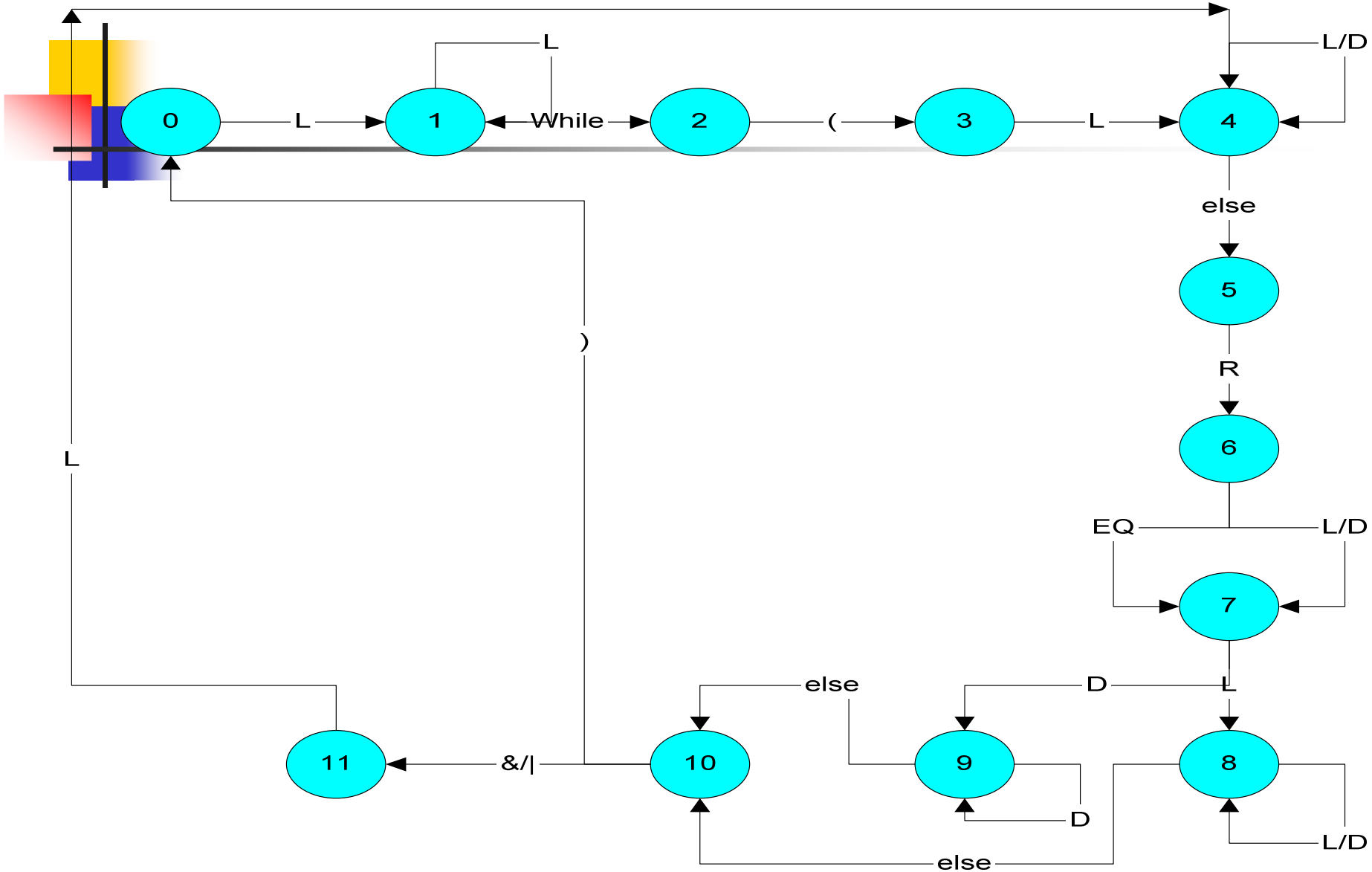
Syntax

- Function Declaration
- Preprocessor Directive
- While Loop
- For Loop
- Declaration Statement
- Assignment Statement
- Comments

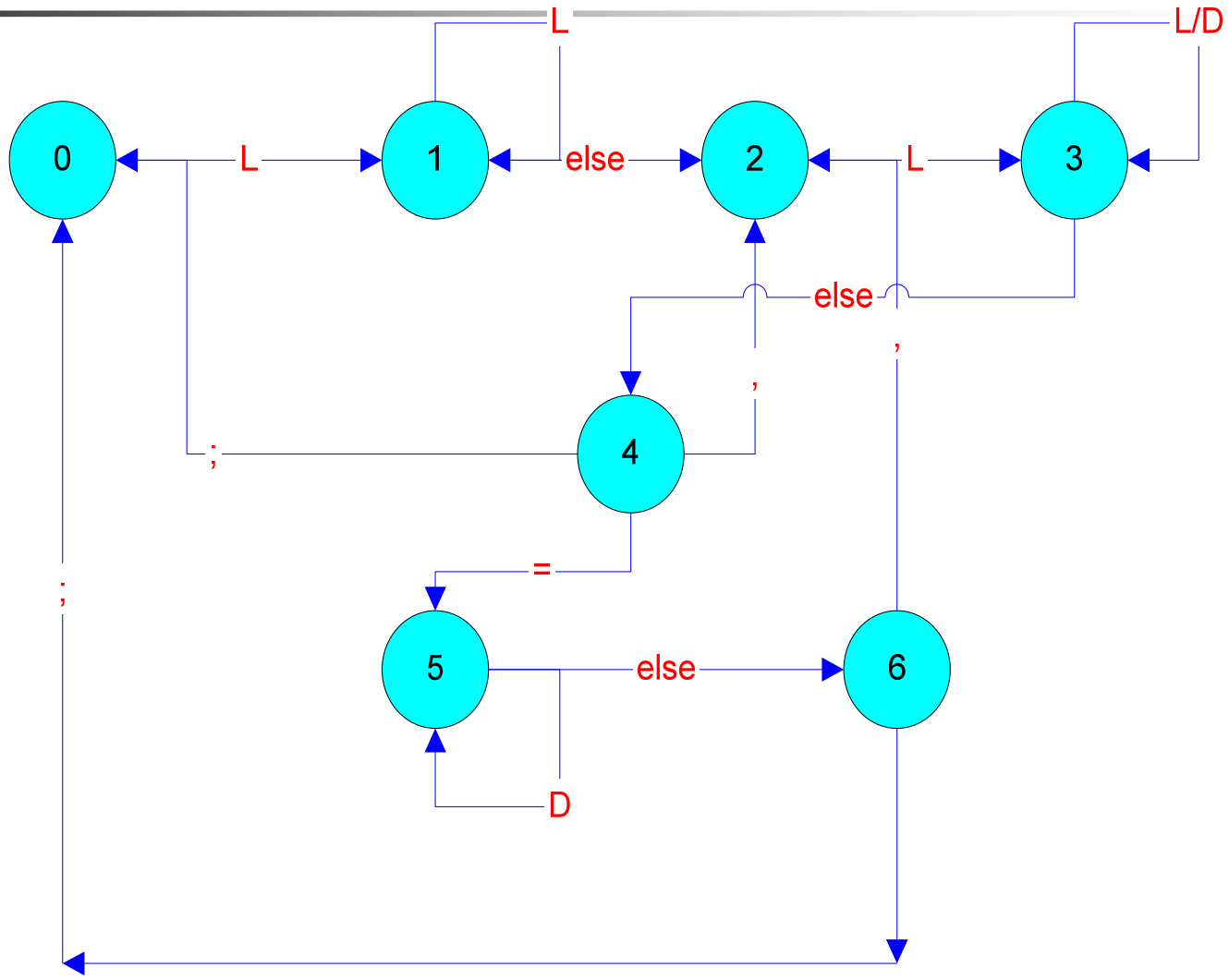
Include directive



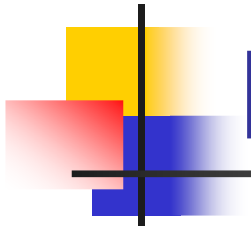
While



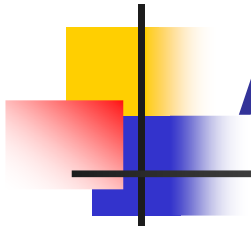
Declaration



Non-Recursive Predictive parsing



- Syntax of assignment statement also verified using right recursive grammar
- First set is constructed from the grammar which identifies for each terminal the corresponding productions to use during parsing.



Algorithm

While not inserted and stmt index less than stmt length

extract non-terminal from the stmt

search for the corresponding entry in 0^{th} column to locate row

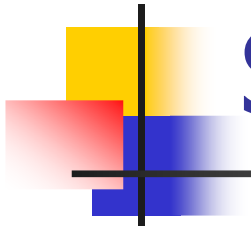
if found then compare input lexeme type with 0^{th} row to locate the corresponding column

if matched then replace the non-terminal with production

if input terminal symbol inserted **set inserted**

If not inserted move stmt index ahead

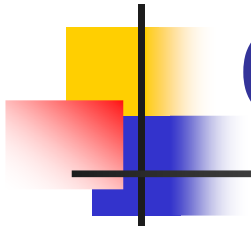
Setting previous production to N (NULL)



Semantic

- Undeclared Variables
- Type Compatibility
  - Operation Compatibility
  - Assignment Compatibility

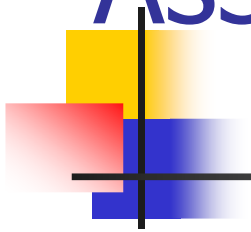
Any undeclared variable present in any assignment statement will cause declaration error



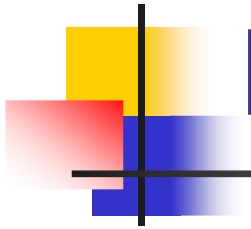
Operations Compatibility Rules

Type1	Type2	Result
Long	Long	Long
Long	Float	Long
Long	Int	Long
Long	Char	Error
Float	Long	Long
Float	Float	Float
Float	Int	Float
Float	Char	Error
Int	Long	Long
Int	Float	Float
Int	Int	Int
Char	Long	Error
Char	Float	Error
Char	Int	Error
Char	Char	char

Assignment Compatibility Rules



Type1	Type2	Result
Long	Long	Valid
Long	Float	Valid
Long	Int	Valid
Long	Char	Error
Float	Long	Error
Float	Float	Valid
Float	Int	Valid
Float	Char	Error
Int	Long	Error
Int	Float	Error
Int	Int	Valid
Char	Long	Error
Char	Float	Error
Char	Int	Error
Char	Char	Valid



Examples

- int a,b;
- float c;
- char d;
- a=a+b (Valid)
- b=a+c (Invalid)
- c=a+b (Valid)
- c=a+b+c (Valid)
- a=b+e (InValid)
- a=b+d (Invalid)