PROGRAMMING LANGUAGES RECITATION

-Monika Dagar

12th September 2020

Agenda

- Lexical Analysis
- Syntax Analysis
- Ambiguous Grammar
- Precedence and Associativity
- Flex and Bison
- Calculator Example
- Resources

Lexical Analysis

temp = 1 ; ⇒ 4 tokens

- It is a process of converting a sequence of characters into a sequence of tokens.
- Tokens are the basic building blocks of a program. They are the shortest strings of characters with individual meaning. Examples include keywords, identifiers, symbols, constants and numbers.
- In order to specify tokens we use the notation of regular expressions.
- Lexer - A lexer is a component to take a sequence of characters (program code) and generates several tokens to represent those inputs. Lexer: Text ⟶ Tokens

Syntax analysis

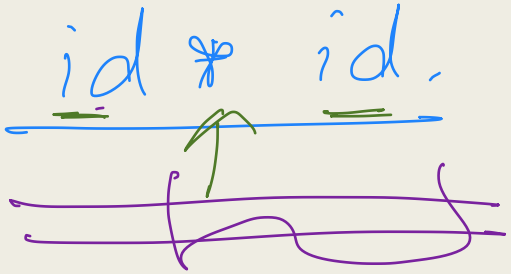
- It is a process of analyzing the input sequence of tokens and giving a structural representation of the input (usually represent by an abstract syntax tree or a parse tree).
- Parser - A parser is a component that takes the tokens produced by the lexer as input and builds a parse tree based on the input. Parser: Tokens → Parse Tree.

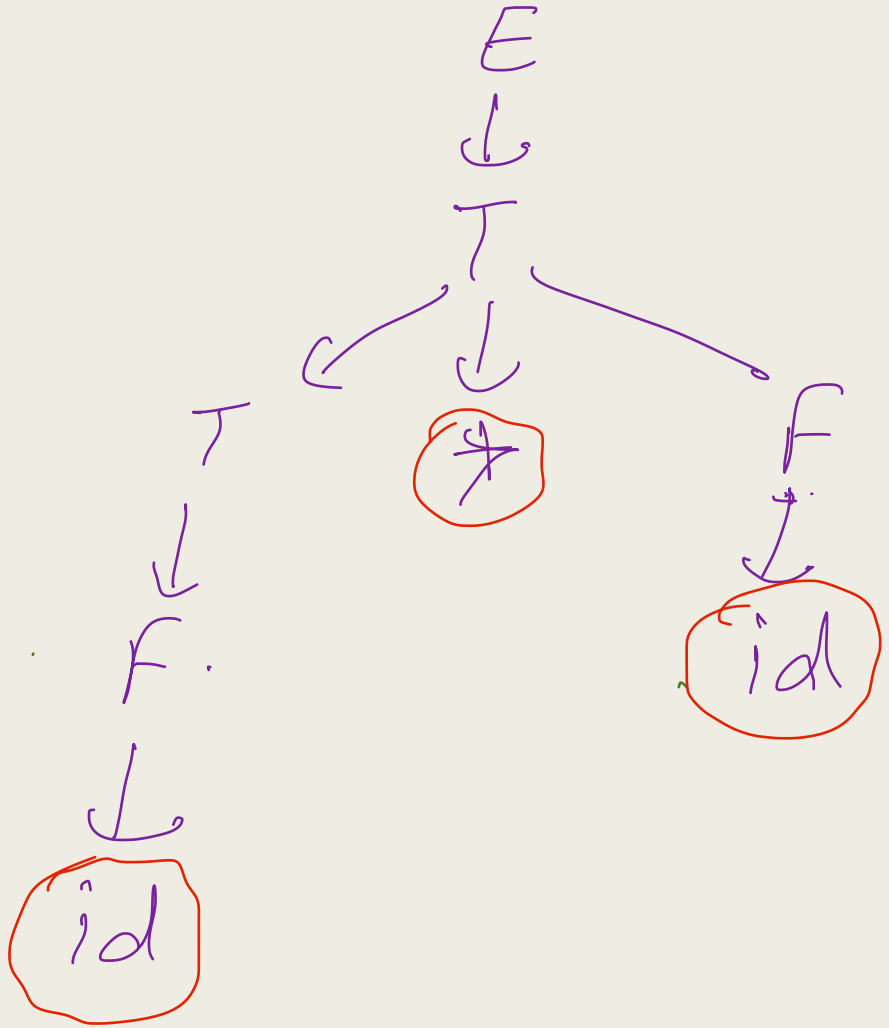
$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

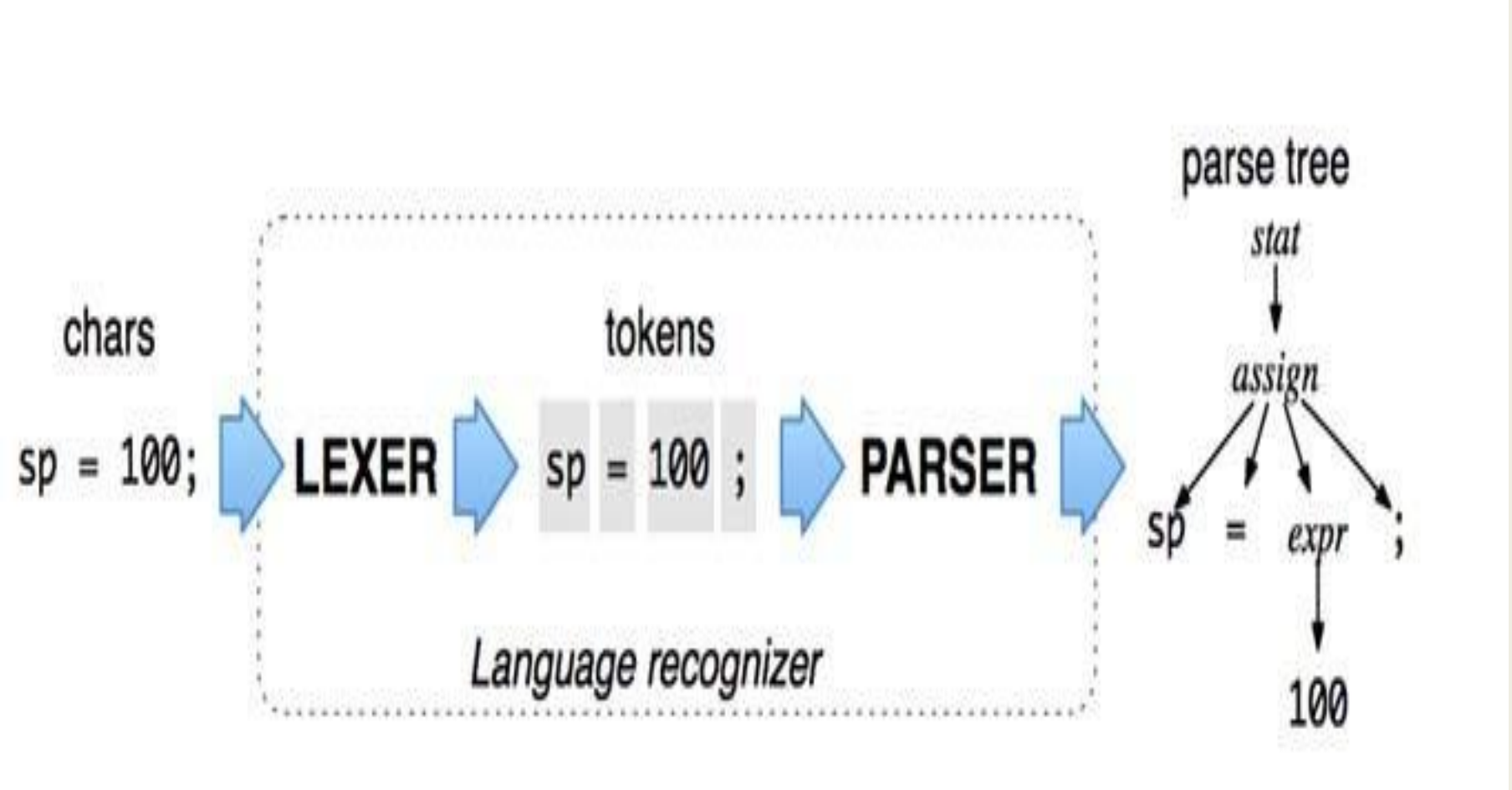
$F \rightarrow (E) \mid id$

$\Rightarrow$ id * id.



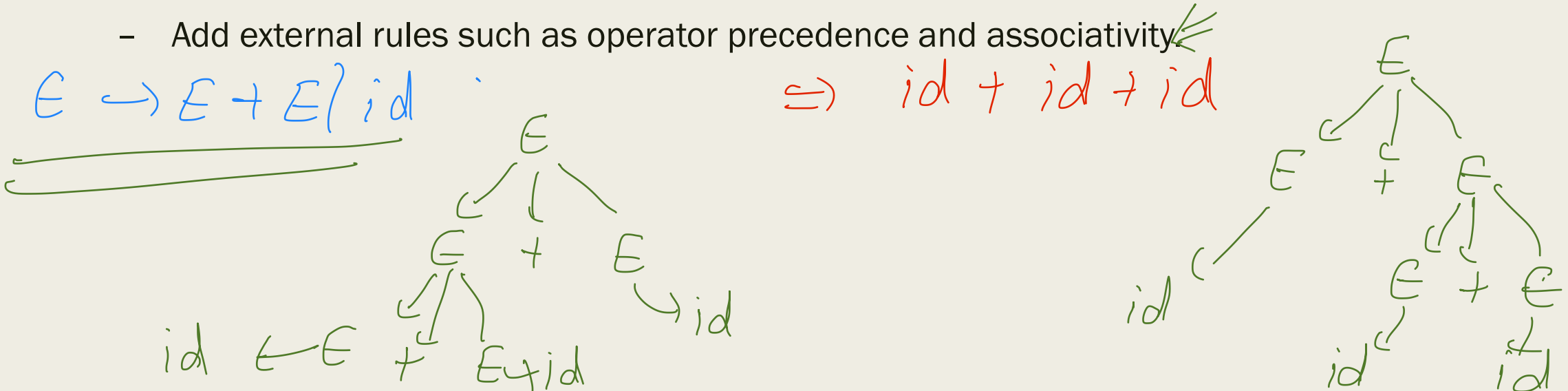


Summary


chars
sp = 100;
LEXER
tokens
sp = 100 ;
PARSER
parse tree
stat
assign
sp
=
expr
;
100
Language recognizer

Ambiguous Grammar

- A CFG is ambiguous if it has more than one parse tree for some strings i.e. there is more than 1 derivation for a string.
- Ambiguity is a property of Grammars, not Languages.
- Two ways to fix them:
  - Rewrite the grammar.
  - Add external rules such as operator precedence and associativity.



Precedence

- Consider the expression 5 + 2 * 3.
- We say operator * has precedence over operator +.
- Precedence can be specified in a couple of ways:
  - Write the precedence rules directly into the grammar. Rules for higher precedence operators will tend to be deeper in the parse tree than other rules.
  - Specify operator precedence separately. Parser generators like Bison offer this as a convenience. Ex. – Calc2

Associativity

- Consider the expression 5 + 2 + 3.
- We know that 5 + (2 + 3) yields the same mathematical result as (5+2) +3, but parser still needs to know which interpretation to choose.
- Associativity tells the parser what to do with operators at the same level of precedence.
- Some options:
  - Use Left associativity: ((5 + 2) + 3)
  - Use Right associativity: (5 + (2 + 3))
  - Leave the grammar ambiguous, since it doesn't matter which interpretation is used (not recommended)

Flex and Bison

- Lex (or Flex) is a lexical analyzer generator.

Input: rules containing regular expressions.

Output: C code. Can be compiled into a standalone lexical analyzer or integrated into a parser.

- Yacc (or Bison) is a parser generator.

Input: Context-free grammar and Lex generated source code(optional).

Output: An LR parser.

** Lecture -1 slide - 35

Flex and Bison Installation

- **Mac:**

brew install flex # to install flex

brew install bison # to install bison

To install homebrew, please do this following code in your terminal:

/usr/bin/ruby -e "$(curl -fsSLhttps://raw.githubusercontent.com/Homebrew/install/master/install)”

- **Windows:** Tutorial.
- **Ubuntu Linux:** Tutorial

Flex Scanner (*.l files)

- Skeleton (structure for a flex file):

%{

C/C++ declarations

%}

Flex declarations

%%

Token rules(Regular expression i.e. Regex)

%%

Additional C/C++ code

Bison Parser (*.y files)

- Skeleton (structure for a bison file):

%{

C/C++ declarations

%}

Bison declarations

%%

Grammar rules(BNF form)

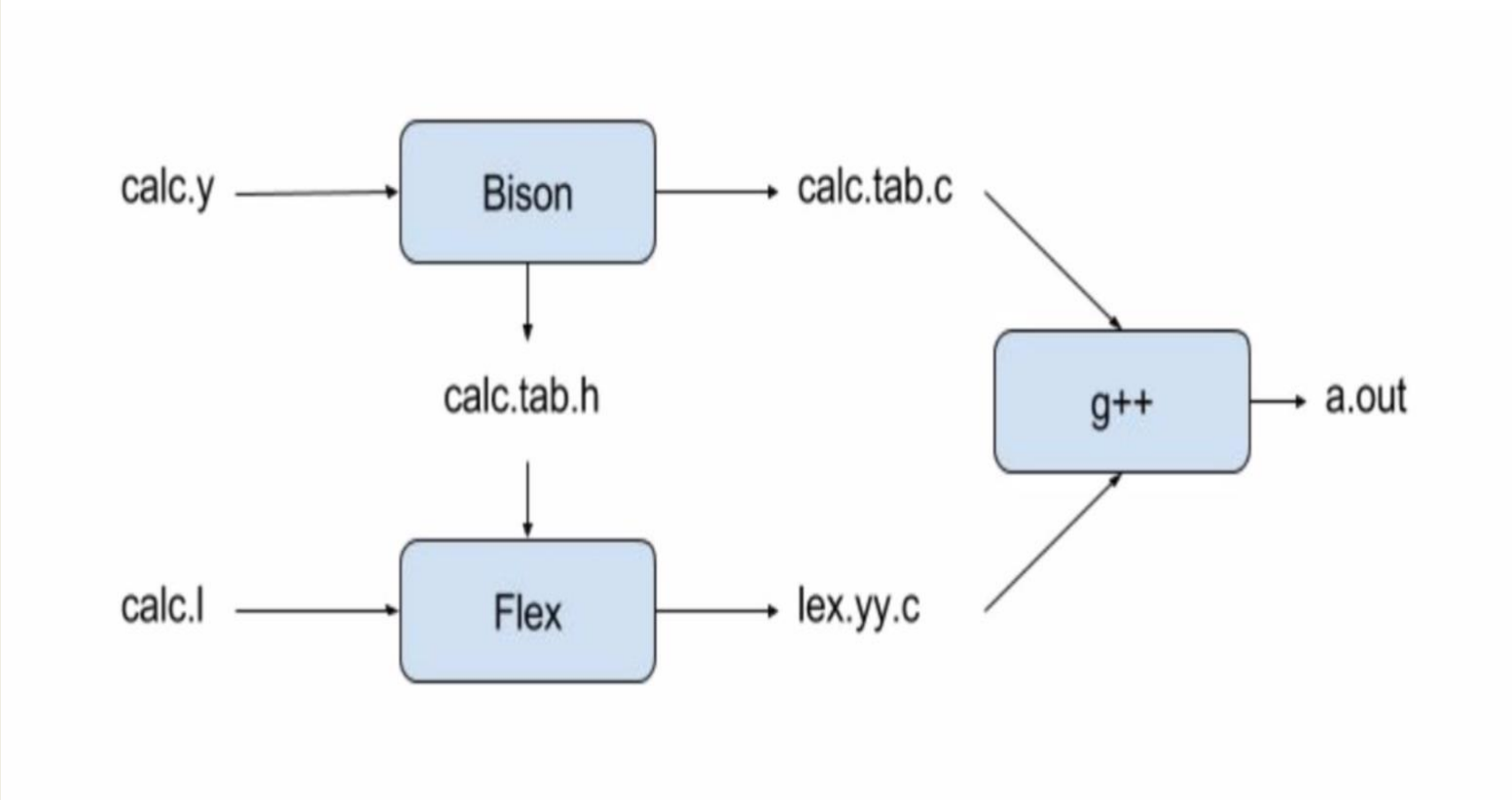
%%

Additional C/C++ code

Running the example

- bison –d filename.y
- flex filename.l
- g++ lex.yy.c filename.tab.c
- ./a.out (on mac)

- exprs, enter, ctrl + D
- ex: 4+4, enter, ctrl + D



Resources

- Flex and bison manual.
- To test the correctness of CFG – Here.
- Tutorial to design bison grammar rules.
- To test the correctness of regular expression – Online Tool.