GRAMMARS AND EXPRESSIONS

Comp3031 Lab 04
Fall 2014

SU Zhiyang, JIA Xiaoying

Natural Language

- Syntax
  - Anne plays tennis.
    - Subject: Anne
    - Predicate: plays
    - Object: tennis
- Semantics
  - Anne is a person
  - Tennis is a kind of sport

Computer Language

- Computer language also has syntax and semantics
- Syntax
  - Defined by a grammar(e.g. in BNF or in regular expression)
- Semantics
  - Specify what the language means

Context Free Grammar

- BNF example

Production

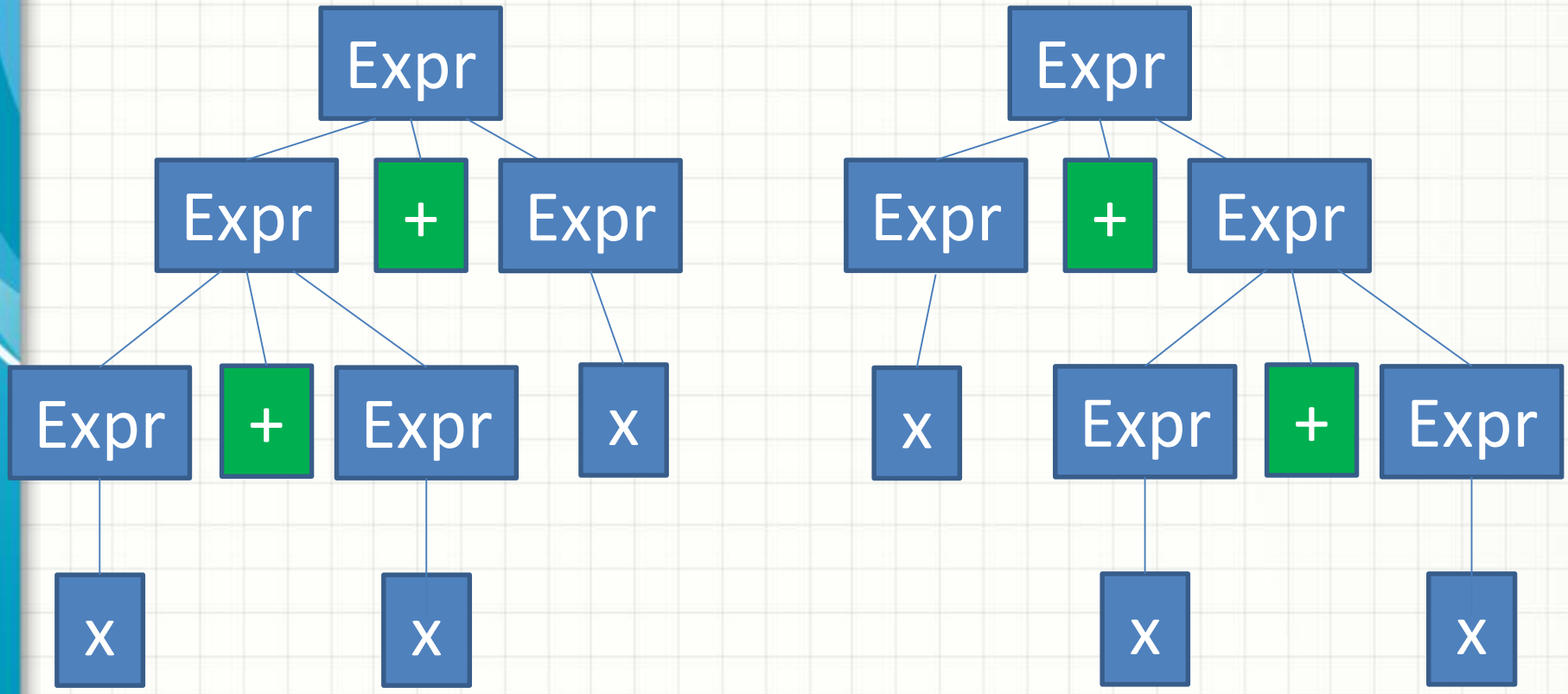
<S> ::= <Expr> + <Expr> | x

Start symbol Nonterminal Terminal

Ambiguous Grammar

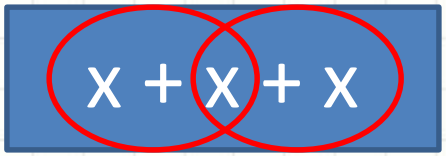
- Consider the grammar
  - <Expr> ::= <Expr> + <Expr> | x
- How many parse trees?
  - When a string has at least two different parsing trees, the grammar is ambiguous

Two Parse Trees for x+x+x



Eliminating Ambiguity in Grammar

- Ambiguity



- Specify the associativity of "+"
  - <Expr> ::= <Expr> + <Expr> | x
  - <Expr> ::= <Expr> + x | x

Enforcing Operator Precedence in Grammar

- Operators have different precedence in an expression
  - x + x * x
- How to make “*” having a higher priority?
- Introducing new non-terminals
  - <Expr> ::= <Expr> + <Term> | <Term>
  - <Term> ::= <Term> * x | x

Regular Grammars

- Left-linear grammars
  - Every production rule only has at most one non-terminal, which appears on the leftmost
  - <Ones> ::= <Ones>1 | 1
- Right-linear grammars
  - Similar with left-linear, just the non-terminal is on the rightmost
  - <Ones> ::= 1<Ones> | 1

Regular Expressions

- Another way to represent a regular grammar

Expression	Meaning
x	The single char “x”
r*	Repeat r 0 or more times
r+	Repeat r 1 or more times
r?	0 or 1 occurrence of r
rs	Concatenation of r and s
(r)s	Evaluate then concatenate s
r|s	r or s

Example Regular Expressions

- Examples
  - abc
  - a+b+ aa…abb…b
  - xy(abc)+ xyabcabc…abc

Example Exercise

- Suppose there is a language composed of all binary numbers each of which contains at least three consecutive **1**'s
  - Valid: 0001111100, 111001011
  - Invalid: 01010011, 0110101
- Write the corresponding regular expression
- Design an unambiguous grammar

Example Exercise Solution

- Regular expression
  - (1|0)*111(1|0)*
- Unambiguous grammar
  - <S> ::= 0<S> | 1<S1>
  - <S1> ::= 0<S> | 1<S2>
  - <S2> ::= 0<S> | 1<S3>
  - <S3> ::= 0<S3> | 1<S3> | <empty>

Exercise

- Consider the following grammar
  - <S>::= <A><S><A> |a
  - <A>::= a|b
- Generate all strings of length less than 4 using this grammar
- Determine whether string ”bababab” belongs to the language of the above grammar

Extra Exercise for Flex

- Flex is used to generate a lexical analyzer (scanner)

- The executable file is the generated scanner
  - Use the scanner to process text files

Extra Exercise for Flex

- Edit roundchar.lex

```
%option noyywrap

%{
#include <stdio.h>
%}
%%
[a-yA-Y] printf("%c", *yytext+1);
[zZ]     printf("%c", *yytext-25);
.        printf("%c", *yytext);
%%

int main(int argc, char **argv)
{
    yylex();
    return 0;
}
```

Extra Exercise for Flex

- Generate the scanner
  - flex -oroundchar.yy.c roundchar.lex
  - gcc -o roundchar roundchar.yy.c
  - ./roundchar

```
zsuab@ras1:~/flex$ flex -oroundchar.yy.c roundchar.lex
zsuab@ras1:~/flex$ g++ -o roundchar roundchar.yy.c
zsuab@ras1:~/flex$ ./roundchar
abcdefghijklmn
```

- Check the output