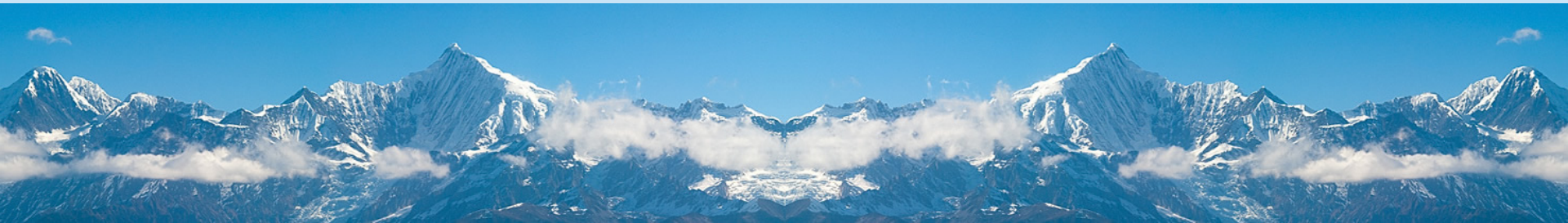
离散数学2016秋季

课堂讨论-1215

陈斌

北京大学地球与空间科学学院

gischen@pku.edu.cn



形式语言与自动机：基本概念

- 语言及研究方向：自然语言的音形义用
- 形式语言：语言的基本概念和字符串运算；正则表达式
- 短语结构语法：四元组；产生式关系
- 语言及语法表达：导出树；语言的差异
- 形式语法分类：Chomsky语法分类及对应自动机
- 语法分析：分析句子，得到其对应的构造过程
- BNF范式：上下文无关文法和正则文法的一种描述
- 语法图：用图来表示语法
- 正则语法：正则表达式/正则集合-正则语法/正则语言

Python语言语法的BNF描述

- 词法

```
keyword ::=
             "and"      | "del"      | "from"     | "not"      | "while"
           | "as"       | "elif"     | "global"   | "or"       | "with"
           | "assert"   | "else"     | "if"       | "pass"     | "yield"
           | "break"    | "except"   | "import"   | "print"
           | "class"    | "exec"     | "in"       | "raise"
           | "continue" | "finally"  | "is"       | "return"
           | "def"      | "for"      | "lambda"   | "try"

identifier ::=
             (letter|"_") (letter | digit | "_")*

    (where the matched string is not a keyword)

letter ::=
             lowercase | uppercase

lowercase ::=
             "a"|"b"|...|"z"

uppercase ::=
             "A"|"B"|...|"Z"

digit ::=
             "0"|"1"|...|"9"

stringliteralpiece ::=
             [stringprefix] (shortstring | longstring)
             | rawstringprefix (rawshortstring | rawlongstring)

stringprefix ::=
             "u" | "U"

rawstringprefix ::=
             "r" | "ur" | "R" | "UR" | "Ur" | "uR"
```

Python语言语法的BN

- 语法

```
pass_stmt ::=
             "pass"

del_stmt ::=
             "del" target_list

print_stmt ::=
             "print" (  [expression ("," expression)* [","]]
                      | ">>" expression [("," expression)+ [","]] )

return_stmt ::=
             "return" [expression_list]

yield_stmt ::=
             yield_expression

raise_stmt ::=
             "raise" [expression ["," expression
              ["," expression]]]

break_stmt ::=
             "break"

continue_stmt ::=
             "continue"

import_stmt ::=
             "import" module ["as" name]
                ( "," module ["as" name] )*
                | "from" relative_module "import" identifier
                    ["as" name]
                  ( "," identifier ["as" name] )*
                | "from" relative_module "import" "("
                    identifier ["as" name]
                  ( "," identifier ["as" name] )* [","] ")"
                | "from" module "import" "*"
```

自动进行语法分析(lex/yacc)

- 自定义语言的语法分析和解释
- Lex做词法分析，利用正则语法描述，提取词（Token）
- Yacc做语法分析，利用上下文无关文法(BNF)描述，以及Lex提取出来的Token，对语言进行解释和执行
- 可以通过书写规则，让Lex/Yacc自动生成C语言源代码
- 编译后即可执行

实现计算器的lex/yacc文件

```
%{
#include<string.h>
#include "y.tab.h"
extern int yylval;
%}
numbers ([0-9])+
plus "+"
minus "-"
times "*"
divide "/"
lp "("
rp ")"
delim [ /n/t]
ws {delim}*
%%
{numbers} {
                sscanf(yytext, "%d", &yylval);
                  return INTEGER;
}
{plus} {return PLUS;}
{minus} {return MINUS;}
{times} {return TIMES;}
{divide} {return DIVIDE;}
{lp} {return LP;}
{rp} {return RP;}
{ws}        ;
. {printf("Error");exit(1);}
%%
```

```
%{
#include <stdio.h>
#include "lex.yy.c"
#define YYSTYPE int
%}
%token INTEGER PLUS MINUS TIMES DIVIDE LP RP
%%
command : exp {printf("%d/n",$1);}
exp: exp PLUS term {$$ = $1 + $3;}
    |exp MINUS term {$$ = $1 - $3;}
    |term {$$ = $1;}
    ;
term : term TIMES factor {$$ = $1 * $3;}
    |term DIVIDE factor {$$ = $1/$3;}
    |factor {$$ = $1;}
    ;
factor : INTEGER {$$ = $1;}
    | LP exp RP {$$ = $2;}
    ;
%%
int main()
{
    return yyparse();
}
void yyerror(char* s)
{
    fprintf(stderr,"%s",s);
}
int yywrap()
{
    return 1;
}
```