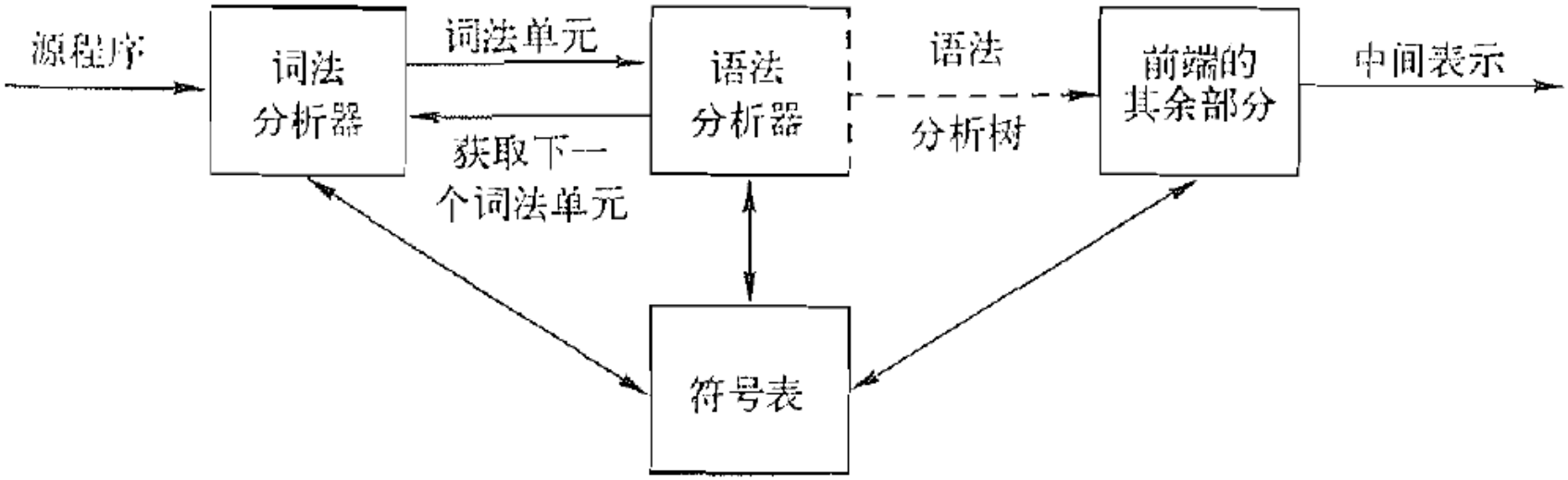
语法分析

魏恒峰

hfwei@nju.edu.cn

2020 年 11 月 24 日

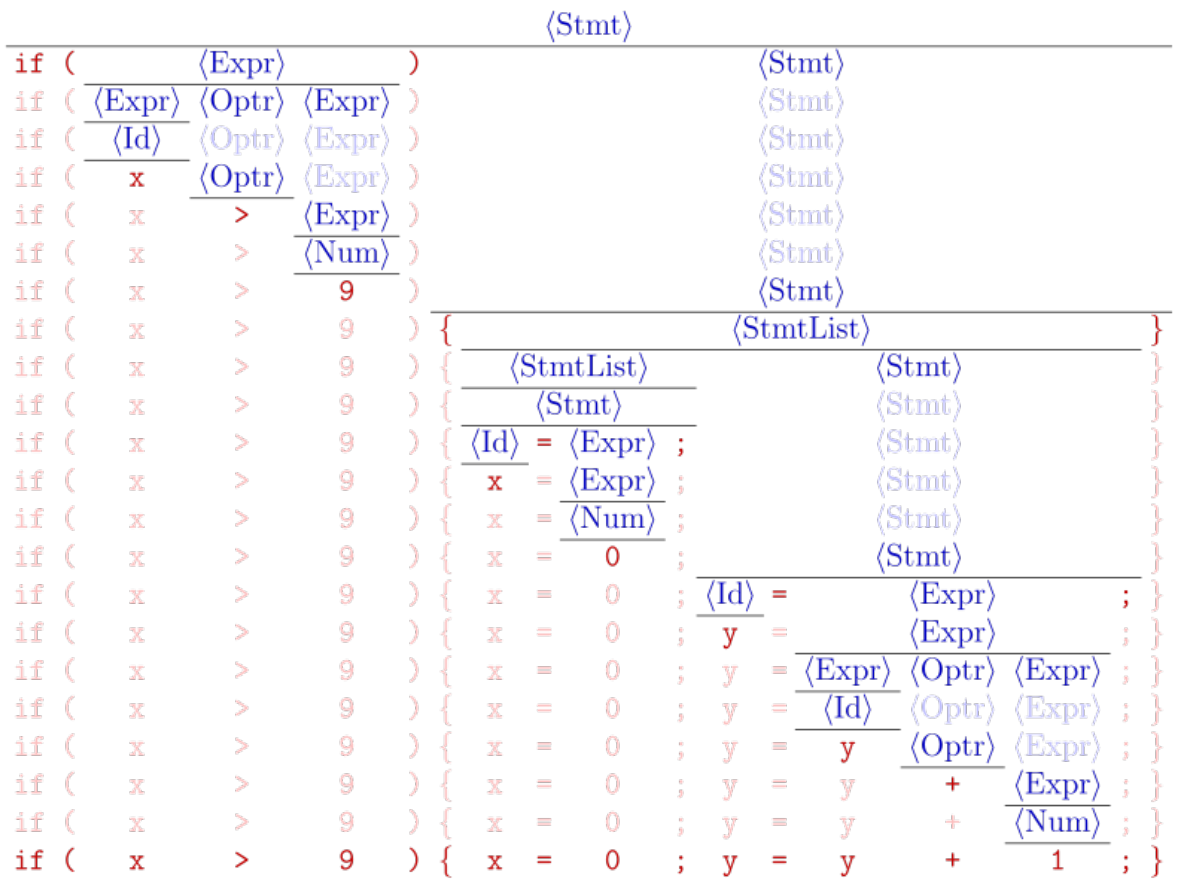
输入: 词法单元流 & **语言的语法规则**



输出: 语法分析树 (Parse Tree)

语法分析举例

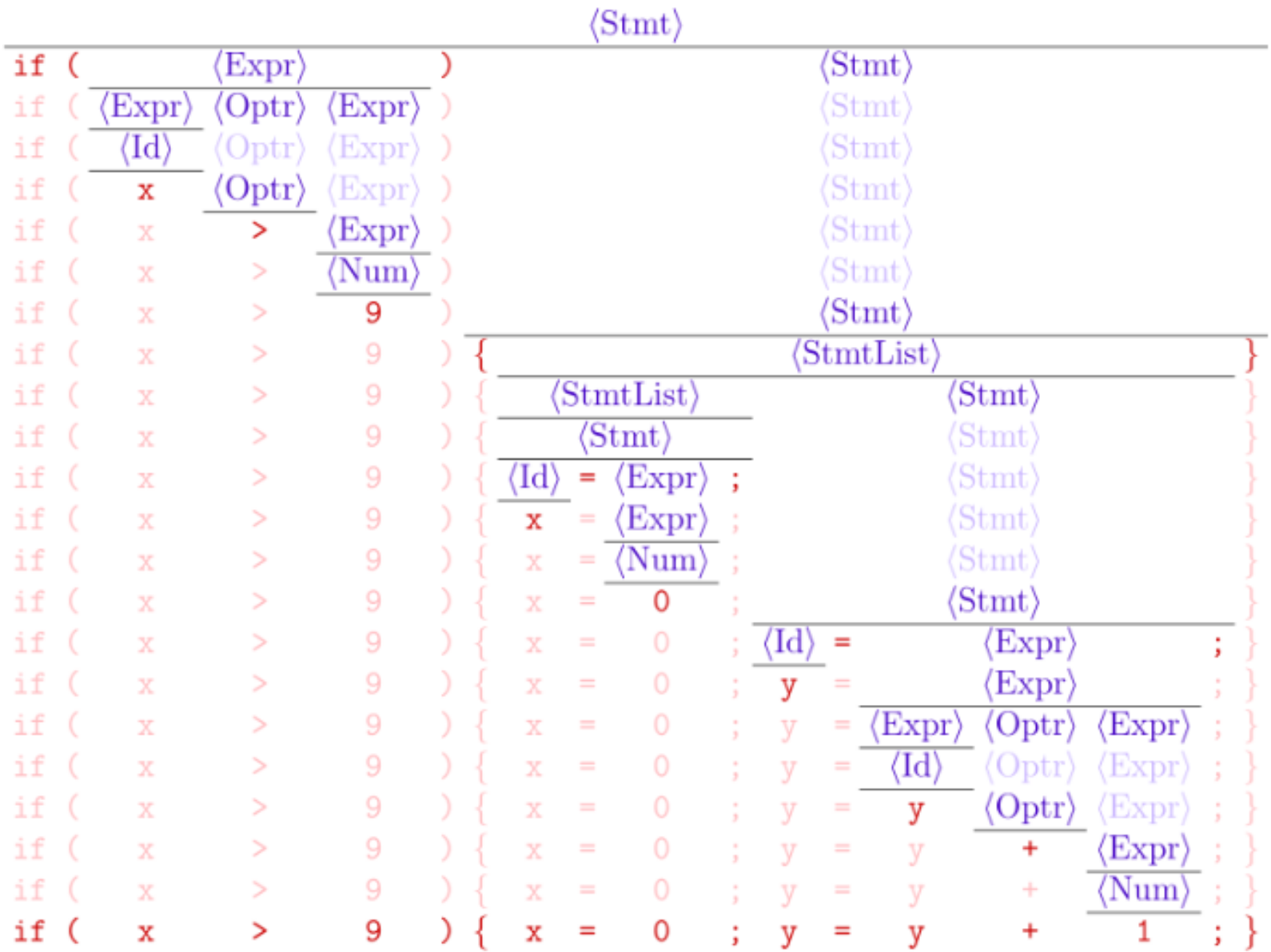
⟨Stmt⟩ → ⟨Id⟩ = ⟨Expr⟩ ;
⟨Stmt⟩ → { ⟨StmtList⟩ }
⟨Stmt⟩ → if (⟨Expr⟩) ⟨Stmt⟩
⟨StmtList⟩ → ⟨Stmt⟩
⟨StmtList⟩ → ⟨StmtList⟩ ⟨Stmt⟩
⟨Expr⟩ → ⟨Id⟩
⟨Expr⟩ → ⟨Num⟩
⟨Expr⟩ → ⟨Expr⟩ ⟨Optr⟩ ⟨Expr⟩
⟨Id⟩ → x
⟨Id⟩ → y
⟨Num⟩ → 0
⟨Num⟩ → 1
⟨Num⟩ → 9
⟨Optr⟩ → >
⟨Optr⟩ → +



语法分析阶段的主题之一：**上下文无关文法**

⟨Stmt⟩ → ⟨Id⟩ = ⟨Expr⟩ ;
⟨Stmt⟩ → { ⟨StmtList⟩ }
⟨Stmt⟩ → if (⟨Expr⟩) ⟨Stmt⟩
⟨StmtList⟩ → ⟨Stmt⟩
⟨StmtList⟩ → ⟨StmtList⟩ ⟨Stmt⟩
⟨Expr⟩ → ⟨Id⟩
⟨Expr⟩ → ⟨Num⟩
⟨Expr⟩ → ⟨Expr⟩ ⟨Optr⟩ ⟨Expr⟩
⟨Id⟩ → x
⟨Id⟩ → y
⟨Num⟩ → 0
⟨Num⟩ → 1
⟨Num⟩ → 9
⟨Optr⟩ → >
⟨Optr⟩ → +

语法分析阶段的主题之二: **构建语法分析树**



语法分析阶段的主题之三: **错误恢复**

报错、**恢复**、继续分析


"TALK IS CHEAP
SHOW ME THE CODE"
- Linus Torvalds

$S \rightarrow \text{if } E \text{ then } S \text{ else } S$

$S \rightarrow \text{begin } S\ L$

$S \rightarrow \text{print } E$

$L \rightarrow \text{end}$

$L \rightarrow ;\ S\ L$

$E \rightarrow \text{num} = \text{num}$

顺序语句、条件语句、打印语句

$S \rightarrow$ if E then S else S

$S \rightarrow$ begin S L

$S \rightarrow$ print E

$L \rightarrow$ end

$L \rightarrow$; S L

$E \rightarrow$ num = num

仅根据**当前字符**, 就可以确定应该使用哪条产生式

$S \rightarrow$ if E then S else S

$S \rightarrow$ begin S L

$S \rightarrow$ print E

$L \rightarrow$ end

$L \rightarrow$; S L

$E \rightarrow$ num = num

```
enum token {IF, THEN, ELSE, BEGIN, END, PRINT, SEMI, NUM, EQ};
extern enum token getToken(void);
```

getToken: 语法分析器**按需**向词法分析器索要下一个词法单元

$S \rightarrow$ if E then S else S

$S \rightarrow$ begin S L

$S \rightarrow$ print E

$L \rightarrow$ end

$L \rightarrow$; S L

$E \rightarrow$ num = num

```
enum token tok;
void advance() {tok=getToken();}
void eat(enum token t) {if (tok==t) advance(); else error();}
```

$eat(t)$: 根据当前的产生式, **预期**的词法单元应该是 t

匹配 t, 继续试图匹配下一个词法单元; 否则, 报错

$S \rightarrow$ if E then S else S

$S \rightarrow$ begin S L

$S \rightarrow$ print E

$L \rightarrow$ end

$L \rightarrow$; S L

$E \rightarrow$ num = num

```
void S(void) {switch(tok) {
        case IF:    eat(IF); E(); eat(THEN); S();
                                  eat(ELSE); S(); break;
        case BEGIN: eat(BEGIN); S(); L(); break;
        case PRINT: eat(PRINT); E(); break;
        default:    error();
      }}
void L(void) {switch(tok) {
        case END:   eat(END); break;
        case SEMI:  eat(SEMI); S(); L(); break;
        default:    error();
      }}
void E(void) {  eat(NUM); eat(EQ); eat(NUM); }
```

为每个**非终结符**写一个**递归函数**

对于每个**产生式**, 写一个 case **分支语句**

if $1 = 2$ **then begin print** $2 = 1$; **end else** $1 = 1$

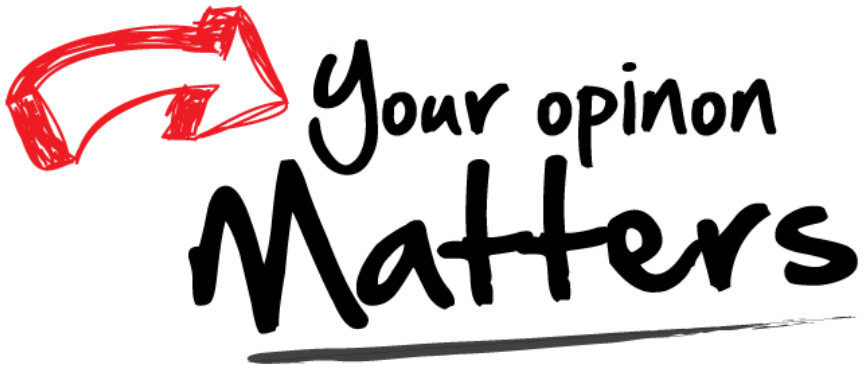
板书演示语法分析器的工作过程

```
enum token {IF, THEN, ELSE, BEGIN, END, PRINT, SEMI, NUM, EQ};
extern enum token getToken(void);

enum token tok;
void advance() {tok=getToken();}
void eat(enum token t) {if (tok==t) advance(); else error();}

void S(void) {switch(tok) {
        case IF:    eat(IF); E(); eat(THEN); S();
                                  eat(ELSE); S(); break;
        case BEGIN: eat(BEGIN); S(); L(); break;
        case PRINT: eat(PRINT); E(); break;
        default:    error();
       }}
void L(void) {switch(tok) {
        case END:   eat(END); break;
        case SEMI:  eat(SEMI); S(); L(); break;
        default:    error();
       }}
void E(void) {  eat(NUM); eat(EQ); eat(NUM); }
```


Thank
You!



Office 926

hfwei@nju.edu.cn