2.3 语法制导翻译

- 用文法来指导对程序的翻译过程。
- 语义描述：给产生式附加一些语义规则或程序片段
- 两种附加程序片段的方法：（语法制导）翻译方案和语法制导定义。

语法制导定义

- 每个文法符号与一个属性集合关联
  - 属性：文法符号的一些特征。如：

    表达式的属性可以有类型、值等。
- 每个产生式与一组语义规则关联
  - 语义规则给出属性值的计算方法，或进行一些操作。

语法制导定义（例）

PRODUCTION	SEMANTIC RULES
$expr \rightarrow expr_1 + term$	$expr.t = expr_1.t \mid\mid term.t \mid\mid '+'$
$expr \rightarrow expr_1 - term$	$expr.t = expr_1.t \mid\mid term.t \mid\mid '-'$
$expr \rightarrow term$	$expr.t = term.t$
$term \rightarrow 0$	$term.t = '0'$
$term \rightarrow 1$	$term.t = '1'$
...	...
$term \rightarrow 9$	$term.t = '9'$

注：

(1) 在有必要区分一个文法符号的多次出现时，加下标

(2) || 表示连接两部分内容

语法制导定义（例2）

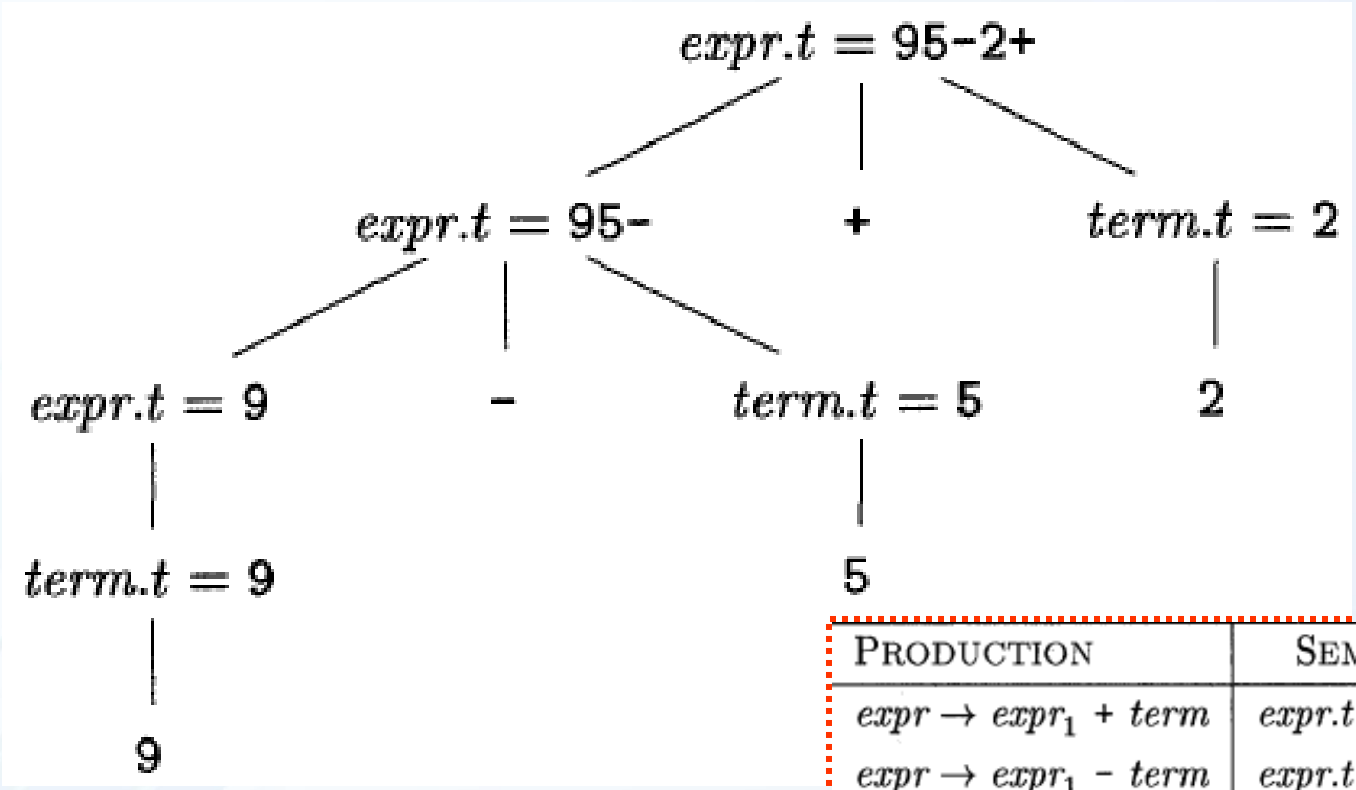
文法规则	语义规则
$expr \rightarrow expr_1 + term$	$expr.t = expr_1.t + term.t$
$expr \rightarrow expr_1 - term$	$expr.t = expr_1.t - term.t$
$expr \rightarrow term$	$expr.t = term.t$
$term \rightarrow 0$	$term.t = 0$
$term \rightarrow 1$	$term.t = 1$
...	...
$term \rightarrow 9$	$term.t = 9$

注释分析树

- 在结点上标记了属性值的分析树。
- 综合属性：如果某属性在结点N上的值是由N及其子结点的属性值确定，这个属性称为综合属性。
  - 注释方法：先构建分析树，然后注释(细分两步：先标记属性，再计算属性值）。

    如果只存在综合属性，则对分析树进行一次后序遍历，就可以完成注释（算出所有结点的各个属性值）。

注释分析树（例）



PRODUCTION	SEMANTIC RULES
$expr \rightarrow expr_1 + term$	$expr.t = expr_1.t \mid\mid term.t \mid\mid '+'$
$expr \rightarrow expr_1 - term$	$expr.t = expr_1.t \mid\mid term.t \mid\mid '-'$
$expr \rightarrow term$	$expr.t = term.t$
$term \rightarrow 0$	$term.t = '0'$
$term \rightarrow 1$	$term.t = '1'$
...	...
$term \rightarrow 9$	$term.t = '9'$

注释分析树如何变化?

文法规则	语义规则
$expr \rightarrow expr_1 + term$	$expr.t = expr_1.t + term.t$
$expr \rightarrow expr_1 - term$	$expr.t = expr_1.t - term.t$
$expr \rightarrow term$	$expr.t = term.t$
$term \rightarrow 0$	$term.t = 0$
$term \rightarrow 1$	$term.t = 1$
...	...
$term \rightarrow 9$	$term.t = 9$

2.4 语法分析

- 基于文法生成一个终结符号串（句子）的过程。
- 自顶向下分析：从根结点开始，逐步向叶结点方向构造
- 自底向上分析：从叶结点开始，逐步构造出根结点。
- 两种语法分析器构造路径
  - 手工构造：人工编程实现。支持自顶向下分析
  - 自动构造：首先形式化地描述文法，接着分析器构造程序自动地将形式化描述转为分析程序。支持自顶向下和自底向上分析

递归下降分析法

- 一般性的自顶向下分析方法
- 使用一组递归过程来处理输入，文法的每个非终结符都关联一个过程。
  - 需要回溯，本质上是一种试探过程，反复使用不同规则谋求匹配输入串
- 例：考虑文法：

  E→T | T+ E

  T→ int | int * T | (E)

给出递归下降法分析(int)的过程。

Recursive Descent

Choose the derivation that is a valid recursive descent parse for the string id + id in the given grammar. Moves that are followed by backtracking are given in red.

$E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

- E
  E'
  E' + E
  id + E
  id + E'
  id + id

- E
  E'
  -E'
  id
  (E)
  E' + E
  -E' + E
  id + E
  id + E'
  id + -E'
  id + id

- E
  E' + E
  id + E
  id + E'
  id + id

- E
  E'
  id
  E' + E
  id + E
  id + E'
  id + id

预测分析法

- 不需要回溯，通过向前看一个符号来确定当前操作
  - 分析输入串时出现的过程调用序列对应着先根遍历该输入串的一棵语法分析树

一个预测分析器的伪代码（例）

S → +SS | -SS | a

+a+aa

```
void s( ) {
    switch (lookahead) {
        case + :  match('+'); s( ); s( ); break;
        case - :   match('-'); s( );  s( ); break;
        case a: match ('a'); break;
        default: report ("syntax error");
    }
}
```

2.6 词法分析

- 从输入中读取字符，并将它们组织成词法单元（记号）
  - 词法单元是二元组，包括token-name（称为记号名或种别码，语法分析时称终结符）和attribute-value(属性值)。
- 主要工作：预读、剔除空白和注释、常量识别、关键字和标识符的识别。

预读(P613)

- 在返回某个词法单元之前，词法分析器可能要预先读入一些字符
  - 常常是预先读入一个字符

```
18)    void readch() throws IOException { peek = (char)System.in.read(); }
19)    boolean readch(char c) throws IOException {
20)       readch();
21)       if( peek != c ) return false;
22)       peek = ' ';
23)       return true;
24)    }
```

剔除空白和注释

空白的处理（代码P614，line26-30）

```
for ( ; ; peek = next input character ) {
    if ( peek is a blank or a tab ) do nothing;
      else if ( peek is a newline ) line = line+1;
            else break;
}
```

识别常量(P614 line45-58)

整型常量处理

```
if ( peek holds a digit ) {
    v = 0;
    do {
        v = v * 10 + integer value of digit peek
        peek = next input character;
    } while ( peek holds a digit ) ;
    return token<num, v>;
}
```

常量的种类

- 无符号数：152， 6.21，0x58A，‘%’
- 布尔常量：true, false
- 字符串常量：“Fuzhou”

识别关键字和标识符

- 关键字：也称保留字，是程序设计语言赋予特殊含义的字符串，如for, do, if等等。
  - 为了区分关键字和标识符，在初始化符号表时即加入所有关键字。
- 标识符：字母开头的字母数字串
  - 若匹配的串不是关键字，则其为标识符

识别关键字和标识符（P614 line59-70）

```
if ( peek holds a letter ) {
    collect letters or digits into a buffer b;
    s = string formed from the characters in b;
    w = token returned by words.get(s);
    if ( w is not null ) return w;
    else {
        Enter the key-value pair (s, <id, s>) into words
        return token <id, s>;
    }
}
```

词法分析器（P614 line25-70）

```
Token scan( ) {
    skip white space, as in Section 2.6.1;
    handle numbers, as in Section 2.6.3;
    handle reserved words and identifiers, as in Section 2.6.4;
    Token t = new Token(peek);
    peek = blank;
    return t;
}
```