中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS290

Compilation Principle

编 译 原 理

第五章 语法制导翻译（1）

郑馥丹

zhengfd5@mail.sysu.edu.cn

CONTENTS

目 录

1. 语法制导翻译[Syntax-Directed Translation]

- 覆盖2个编译过程：
  - 语义分析
  - 中间代码生成
- 内容
  - 语法制导翻译的基本概念和框架
  - 这些概念和框架在语义分析和中间代码生成上的应用

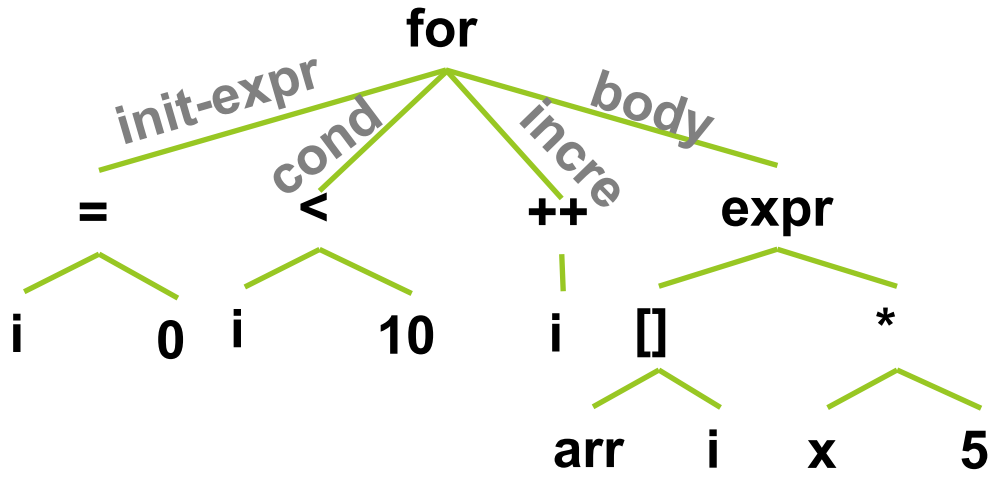
1. 语法制导翻译[Syntax-Directed Translation]

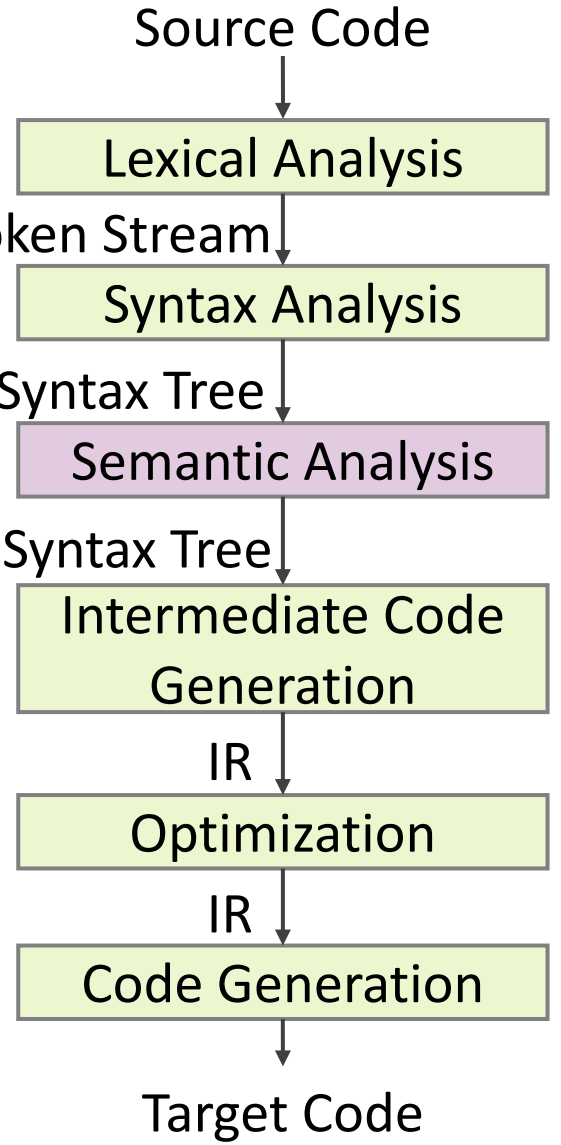
Semantic Analysis[语义分析]

- 基于语法结果进一步分析语义
  - 输入：语法树，输出：语法树+符号表
  - 收集标识符的属性信息（type, scope等）
  - 输入程序是否符合语义规则?
    - ✓ 变量未声明即使用；重复声明
    - ✓ int x; y = x(3);

```
void main()
{
    int arr[10], i, x = 1;

    for (i = 0; i < 10; i++)
        arr[i] = x * 5;
}
```





1. 语法制导翻译[Syntax-Directed Translation]

Semantic Analysis[语义分析]

- 基于语法结果进一步分析语义
  - 输入：语法树，输出：语法树+符号表
  - 收集标识符的属性信息（type，scope等）
  - 输入程序是否符合语义规则？ 数组下标越界

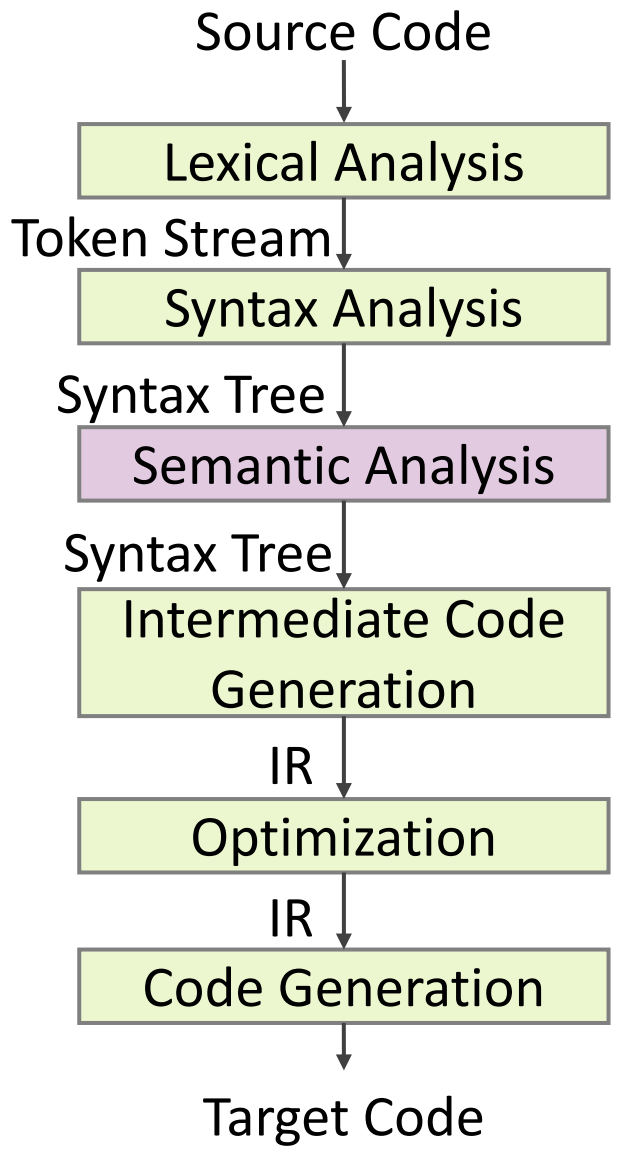
    声明和使用的函数没有定义

    零作除数

```
#include<iostream.h>

void main(){
    cout<<"Hello world!"<<endl;
    int a=10/0;
}
```

```
error C2124: divide or mod by zero
```



1. 语法制导翻译[Syntax-Directed Translation]

Semantic Analysis[语义分析]

- 主要是类型相容检查，有以下几种：
  - 各种条件表达式的类型是不是boolean型？
  - 运算符的分量类型是否相容？
  - 赋值语句的左右部的类型是否相容？
  - 形参和实参的类型是否相容？
  - 下标表达式的类型是否为所允许的类型？
  - 函数说明中的函数类型和返回值的类型是否一致？

1. 语法制导翻译[Syntax-Directed Translation]

Semantic Analysis[语义分析]

- 其它语义检查：
  - V[E]中的V是不是变量，而且是数组类型？
  - V.i中的V是不是变量，而且是记录类型？i是不是该记录的域名？
  - x+f(…)中的f是不是函数名？形参个数和实参个数是否一致？
  - 每个使用性标识符是否都有声明？有无标识符的重复声明？

1. 语法制导翻译[Syntax-Directed Translation]

Semantic Analysis[语义分析]

- 在语义分析同时产生中间代码，在这种模式下，语义分析的主要功能如下：
  - 语义审查
  - 在扫描声明部分时构造标识符的符号表
  - 在扫描语句部分时产生中间代码
- 语义分析方法
  - **语法制导翻译方法**
  - 使用**属性文法**为工具来说明程序设计语言的语义

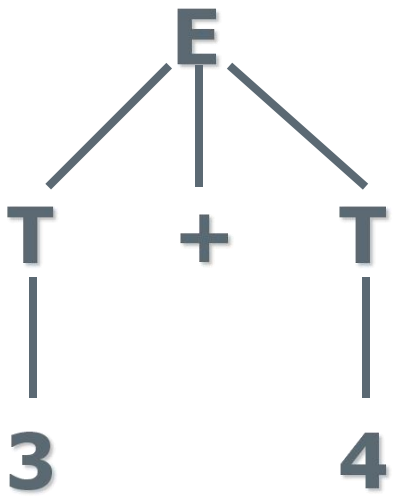
2. 语法制导定义

- 语法制导定义[Syntax-Directed Definition,**SDD**]是一个上下文无关文法和属性及规则的结合
- 语法制导定义也称**属性文法**[Attribute Grammar]
- 属性：
  - 对文法的每一个符号，引进一些属性，来代表与文法符号相关的信息，如类型、值、存储位置等。
  - 如果X是一个符号，**a是X的一个属性**，则用**X.a**来表示a在某个符号为X的分析树结点上的值

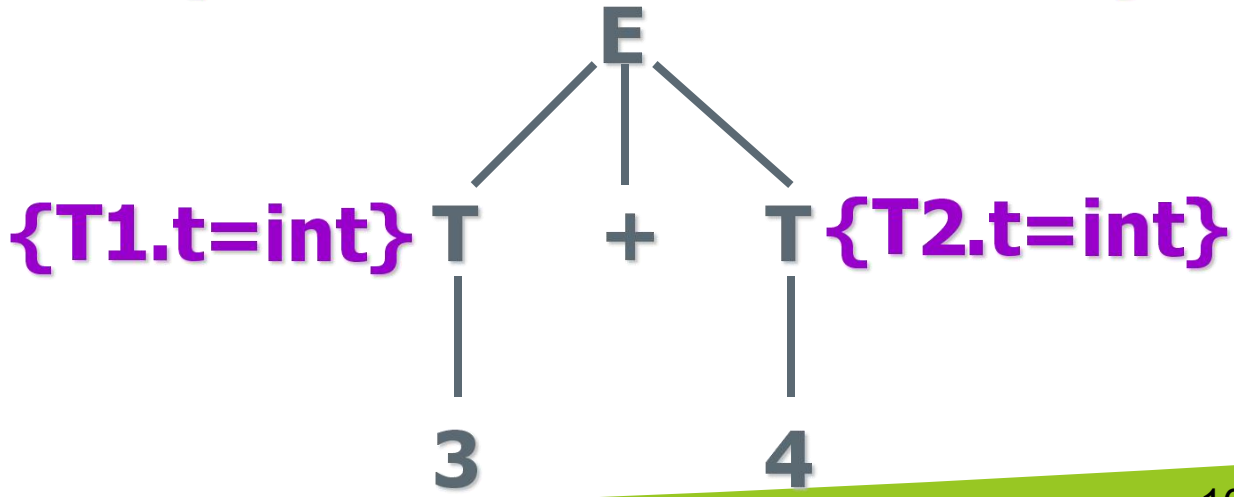
2. 语法制导定义

- 例1：类型检查的属性文法
  - E→T1+T2　　{T1.t = int AND T2.t = int}
  - E→T1 or T2　{T1.t = bool AND T2.t = bool}
  - T→num　　{T.t = int}
  - T→true　　{T.t = bool}
  - T→false　　{T.t = bool}

对输入串3+4的语法树：



{T1.t=int AND T2.t=int}



2. 语法制导定义

- 属性分类
  - **综合属性[synthesized attribute]:**
    - ✓从语法树的角度来看，如果一个结点的某一属性值是由该结点的**子结点**的属性值计算来的，则称该属性为综合属性
    - ✓用于“**自下而上**”传递信息
  - **继承属性[inherited attribute]:**
    - ✓从语法树的角度来看，若一个结点的某一属性值是由该结点的**兄弟结点和（或）父结点**的属性值计算来的，则称该属性为继承属性
    - ✓用于“**自上而下**”传递信息

2. 语法制导定义

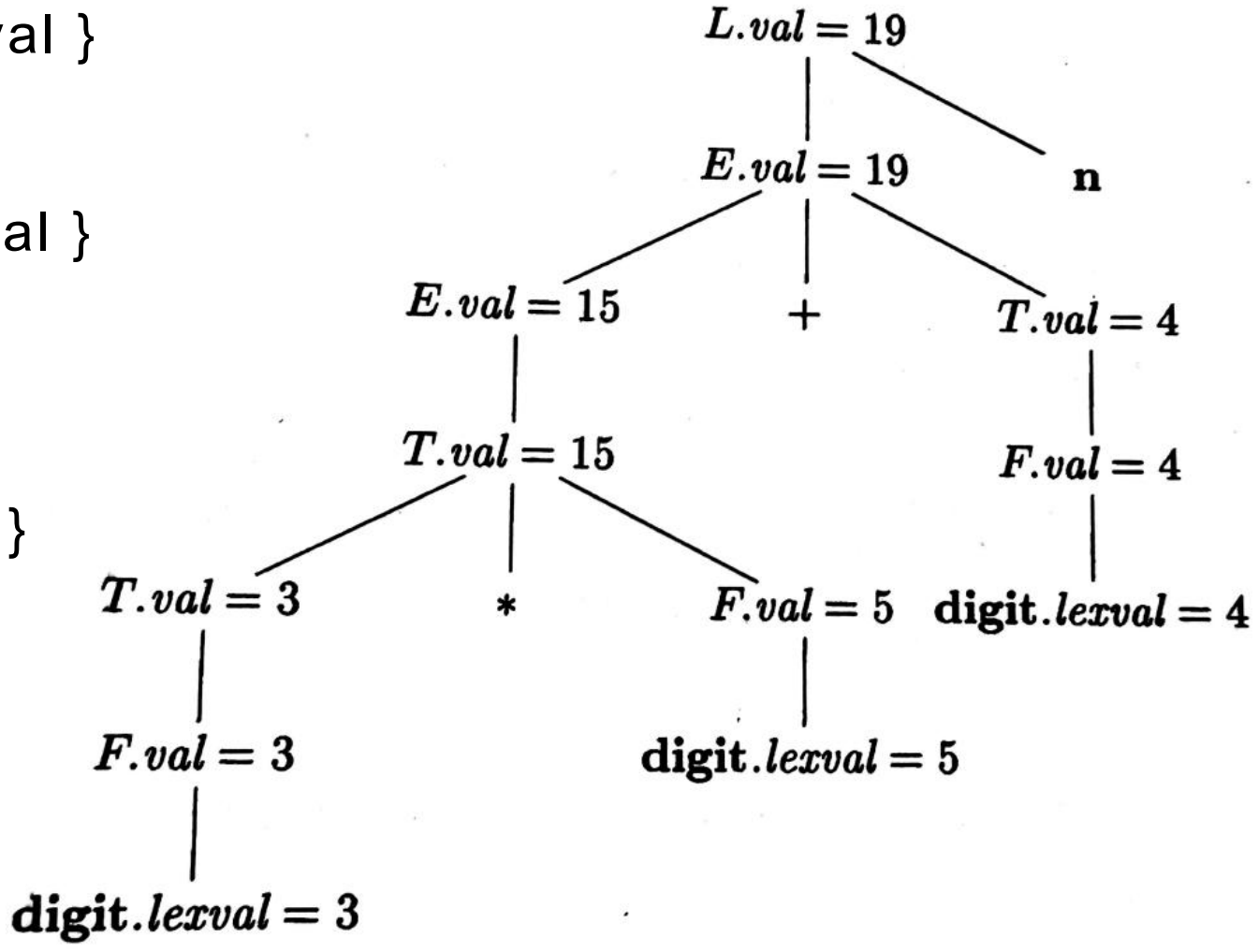
- 例2：简单算术表达式求值的属性文法
  - L→E **n**　　{ L.val = E.val, Print(L.val) } (**n**表示表达式的结尾标记)
  - E→E1+T　　{ E.val = E1.val +T.val }
  - E→T　　{ E.val = T.val }
  - T→T1*F　　{ T.val = T1.val * F.val }
  - T→F　　{ T.val = F.val }
  - F→(E)　　{ F.val = E.val }
  - F→digit　　{ F.val = digit.lexval }

E.val、T.val、F.val的计算都来自它右部的非终结符（子节点）——综合属性

2. 语法制导定义

- 例：简单算术表达式求值的属性文法
  - L→E **n**　　{ L.val = E.val, Print(L.val) } (**n**表示表达式的结尾标记)
  - E→ E1+T　　{ E.val = E1.val +T.val }
  - E→T　　{ E.val = T.val }
  - T→T1*F　　{ T.val = T1.val * F.val }
  - T→F　　{ T.val = F.val }
  - F→(E)　　{ F.val = E.val }
  - F→digit　　{ F.val = digit.lexval }

对应的3*5+4**n**的注释语法分析树：



2. 语法制导定义

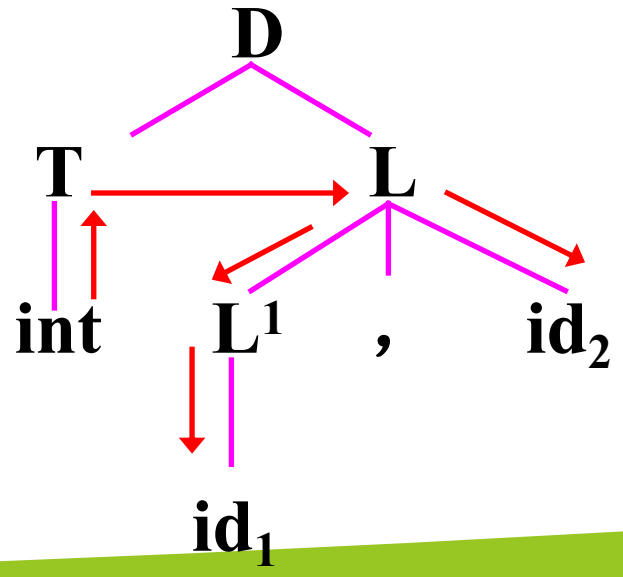
- 例3：描述变量类型说明的属性文法
  - D→TL　　{ L.in = T.type }
  - T→int　　{ T.type = int }
  - T→real　　{ T.type = real }
  - L→L1,id　　{ L1.in = L.in; addtype(id.entry, L.in)}
  - L→id　　{ addtype(id.entry, L.in)}

T.type是综合属性（依赖于子节点）

L.in是继承属性（其属性值的计算依赖于其兄弟或父节点）

int id_1,id_2的语法树:

用→表示属性的传递情况



2. 语法制导定义

- 属性分类
  - **终结符只有综合属性**，它们由词法分析器提供
  - 非终结符既有综合属性也有继承属性，但**文法开始符没有继承属性**

随堂练习（1）

- 对以下简单算术表达式求值的属性文法，给出(9+8*(7+6)+5)*4n的注释语法分析树：
  - L→E **n**　　{ L.val = E.val, Print(L.val) } (**n**表示表达式的结尾标记)
  - E→ E1+T　　{ E.val = E1.val +T.val }
  - E→T　　{ E.val = T.val }
  - T→T1*F　　{ T.val = T1.val * F.val }
  - T→F　　{ T.val = F.val }
  - F→(E)　　{ F.val = E.val }
  - F→digit　　{ F.val = digit.lexval }