Information Science and Technology College of
Northeast Normal University

Compiling and Running of Program

Dr. Zheng Xiaojuan
Professor

December. 2019

Main Content of Chapter 6

§ 6. Semantic Analysis

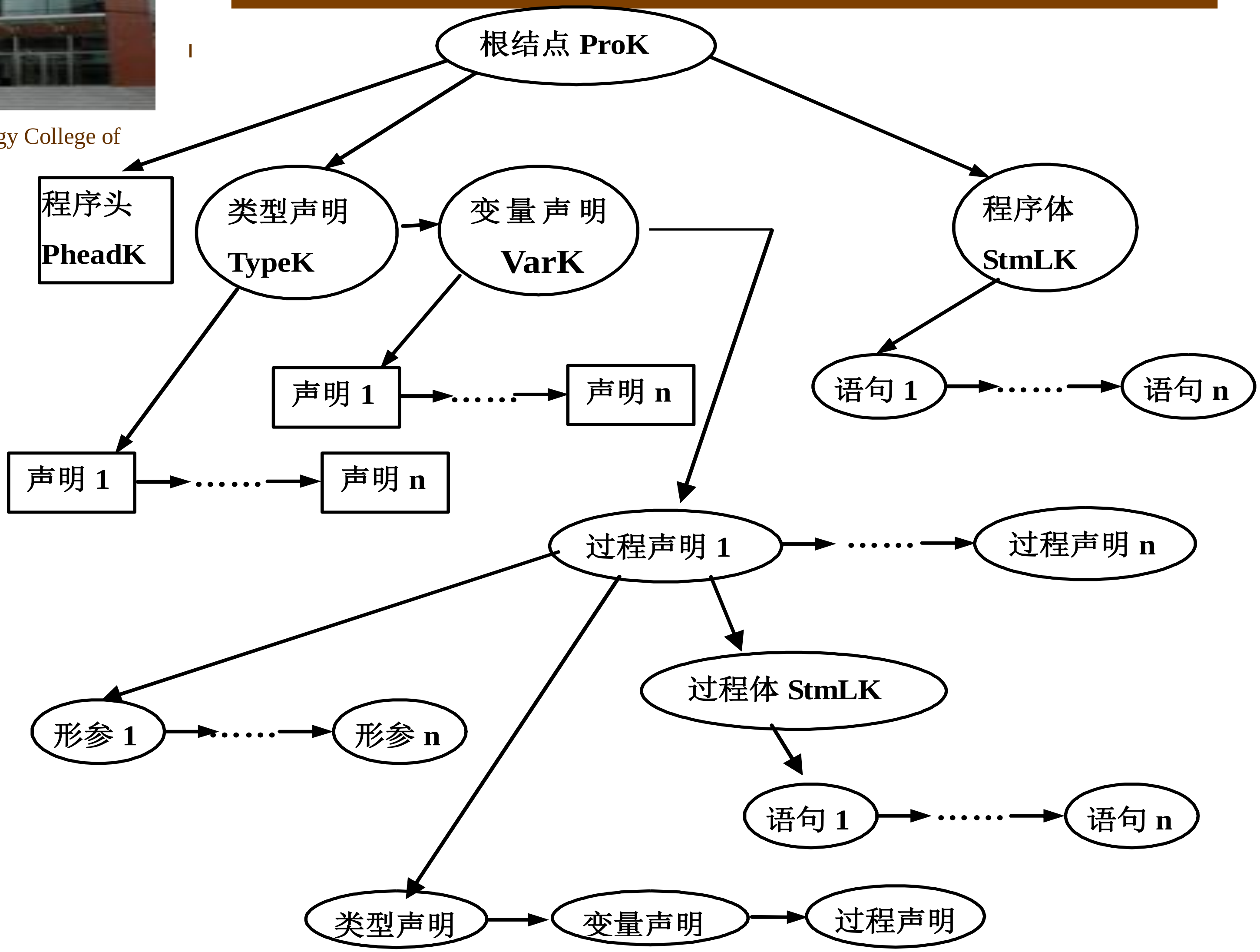
- 6.1 Overview of Semantic Analysis
- 6.2 Symbol Table
- 6.3 Semantic Analysis of Types
- 6.4 Semantic Analysis of Declaration
- 6.5 Semantic Analysis of Body
- 6.6 Attribute Grammar and Action Grammar

Semantic Analysis (review)

- **What?**
  - **Establish symbol table;**
  - **Check semantic errors;**
- **Where?**
  - **Declaration --- establish symbol table; check “repeat declaration” error;**
  - **Body -- search symbol table for attributes, check “use without declaration” error and type-related errors;**
- **How?**
  - **Attribute Grammar**
  - **Action Grammar**
  - **Parse Tree based semantic analysis;**

Information Science and Technology College of
Northeast Normal University

类PASCAL语言



6.3 Semantic Analysis of Types

- **Some Concepts**
  - **类型分析**
  - **类型等价性**
  - **类型相容性**
- **What?**
- **Where?**
- **How?**

6.3 Semantic Analysis of Types

- **类型分析：**

分解出类型表达式内部的各种信息。构造类型的内部表示。这些信息要直接从类型表达式文本结构直接得到是不容易的。

定义性出现时，把标识符与某个类型属性相关联，使用性出现时，取到与标识符相关联的类型属性。

6.3 Semantic Analysis of Types

- 类型等价性
  - 按名等价（比较指针值）
  - 按结构等价（复杂：需要定义一个判定两个类型是否等价的子程序）
- 类型相容性
  - 具体的编译器定义不同；
  - 常见的三种相容性
    - 操作分量相容
    - 赋值相容性
    - 形参和值参相容性

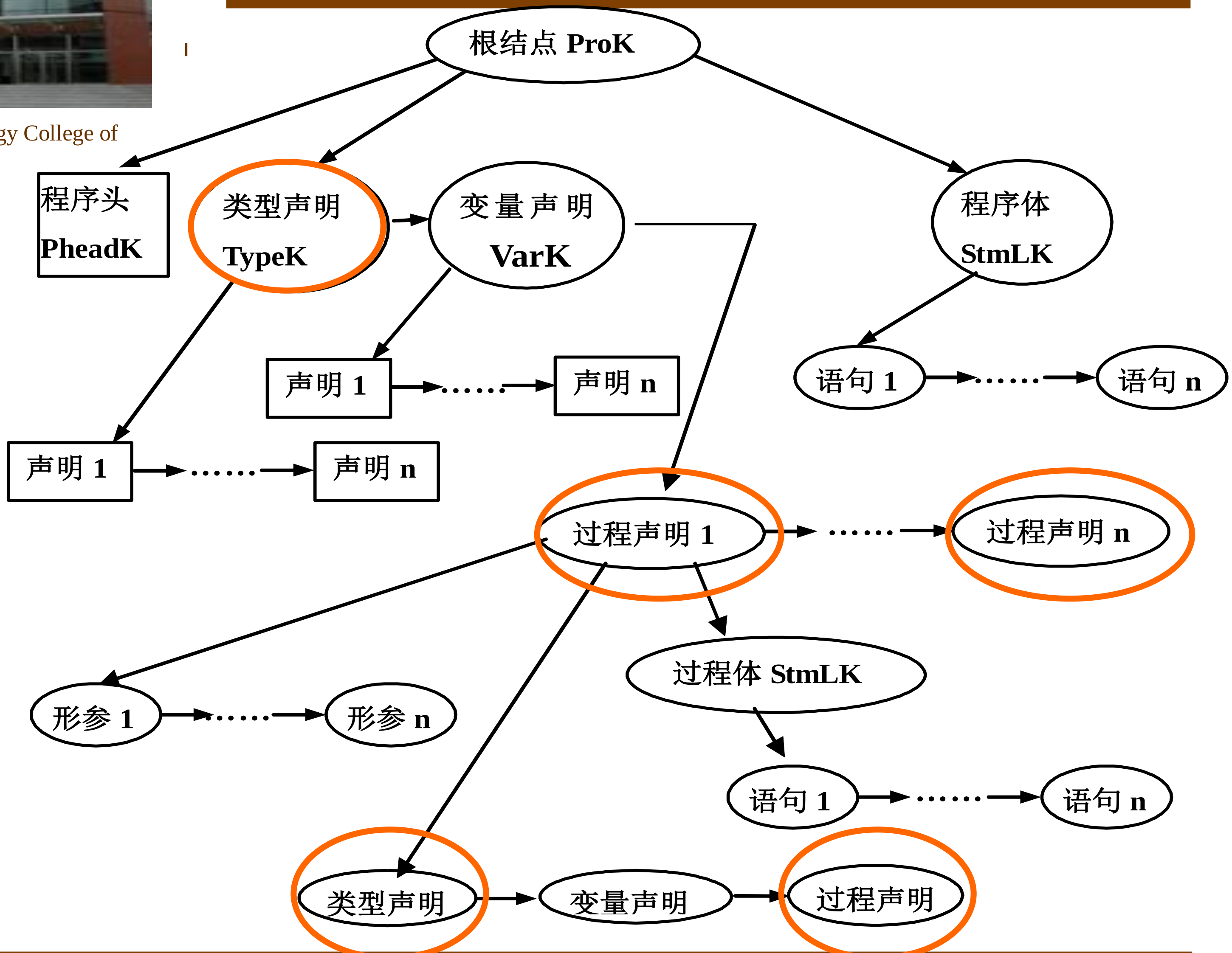
```
typedef  int  T1[10];
typedef  int  T2[10];

T1  a, b;
T2  c, d;
```

6.3 Semantic Analysis of Types

- **What?**
  - **Establish internal representation of types;**
  - **Check semantic errors in type definition;**
- **Where?**
  - **Type Declaration**
  - **Variable declaration**
  - **Parameter declaration**

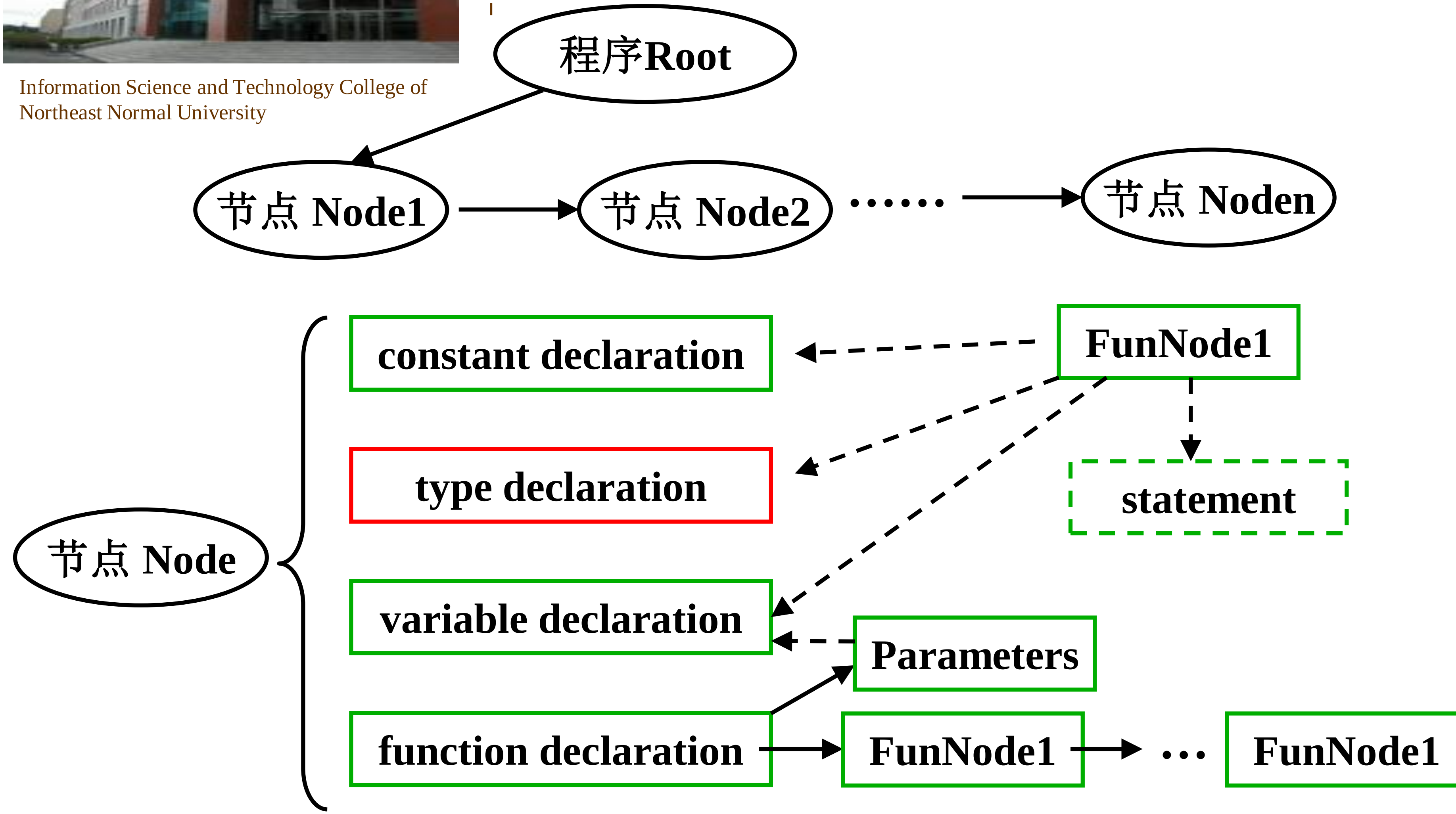
Modified Parse Tree

类PASCAL语言



Information Science and Technology College of
Northeast Normal University

Modified Parse Tree (C语言)



Semantic Analysis of Types

- **How?**
  - **According to different types, establish their internal representations;**
    - **类型名 ；**
    - **基本类型;**
    - **枚举类型;**
    - **数组类型;**
    - **结构类型;**
    - **联合类型;**
    - **指针类型;**
    - **递归类型**

6.3 Semantic Analysis of Types

- **类型名**
  - **search symbol table for pointer to the internal representation;**
- **基本类型**
  - **predefined & stored in symbol table**

Name	Kind	TypePtr
int	typeKind	intPtr
bool	typeKind	boolPtr
char	typeKind	charPtr
real	typeKind	realPtr

Size	Kind
intSize	intTy
boolSize	boolTy
charSize	charTy
realSize	realTy

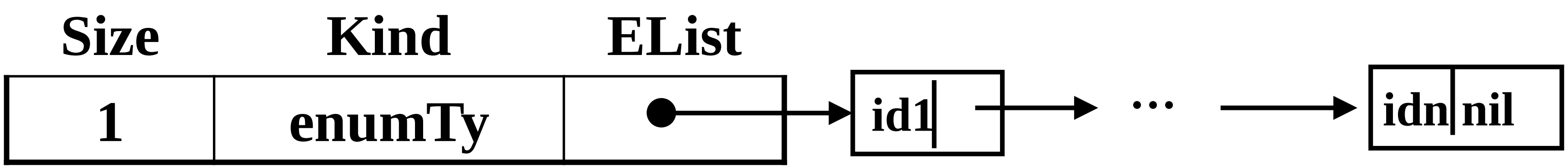
6.3 Semantic Analysis of Types

- **枚举类型**
  - **General form: (id1, …, idn)**
  - **Constant identifiers:**

    (id1, intType, constKind, 0)

    … …

    (idn, intType, constKind, n-1)
  - **Establish list of constant identifiers;**
  - **Establish internal representation of enumeration;**



6.3 Semantic Analysis of Types

- **数组类型**
  - **Get low and up;**
  - **Get or establish an pointer for internal representation of element type;**
  - **Calculate size: size = sizeof(ElemTyp)× (Up-Low+1)**
  - **Establish internal representation of array;**

Size	Kind	Low	Up	ElemTy
	arrayTy			

Information Science and Technology College of
Northeast Normal University

6.3 Semantic Analysis of Types

- **Structure**
  - **For each field identifier, establish the internal representation of its type;**
  - **Link pointers of all field to form a chain for Body;**
  - **Set the size as the sum of each field's size;**
  - **Establish the internal representation of structure;**

Size	Kind	Body
	structTy	

6.3 Semantic Analysis of Types

- **Union**
  - **For each field identifier, establish the internal representation of its type (*<u>the offset of each field is 0</u>*);**
  - **Link pointers of all field to form a chain for Body;**
  - **Set the size as *<u>the maximum of all fields' size</u>*;**
  - **Establish the internal representation of structure;**

Size	Kind	Body
	unionTy	

6.3 Semantic Analysis of Types

- **指针类型**
  - **Get the pointer to the base type;**
  - **Establish the internal representation of pointer;**
- **递归类型**
  - **Typedef struct treenode**

    ```
    { int val;
      struct treenode * left;
      struct treenode * right;} TreeNode;
    ```
  - **Solution: 回填;**

Information Science and Technology College of Northeast Normal University

Main Content of Chapter 6

§ 6. Semantic Analysis

- 6.1 Overview of Semantic Analysis
- 6.2 Symbol Table
- 6.3 Semantic Analysis of Types
- 6.4 Semantic Analysis of Declaration
- 6.5 Semantic Analysis of Body
- 6.6 Attribute Grammar and Action Grammar

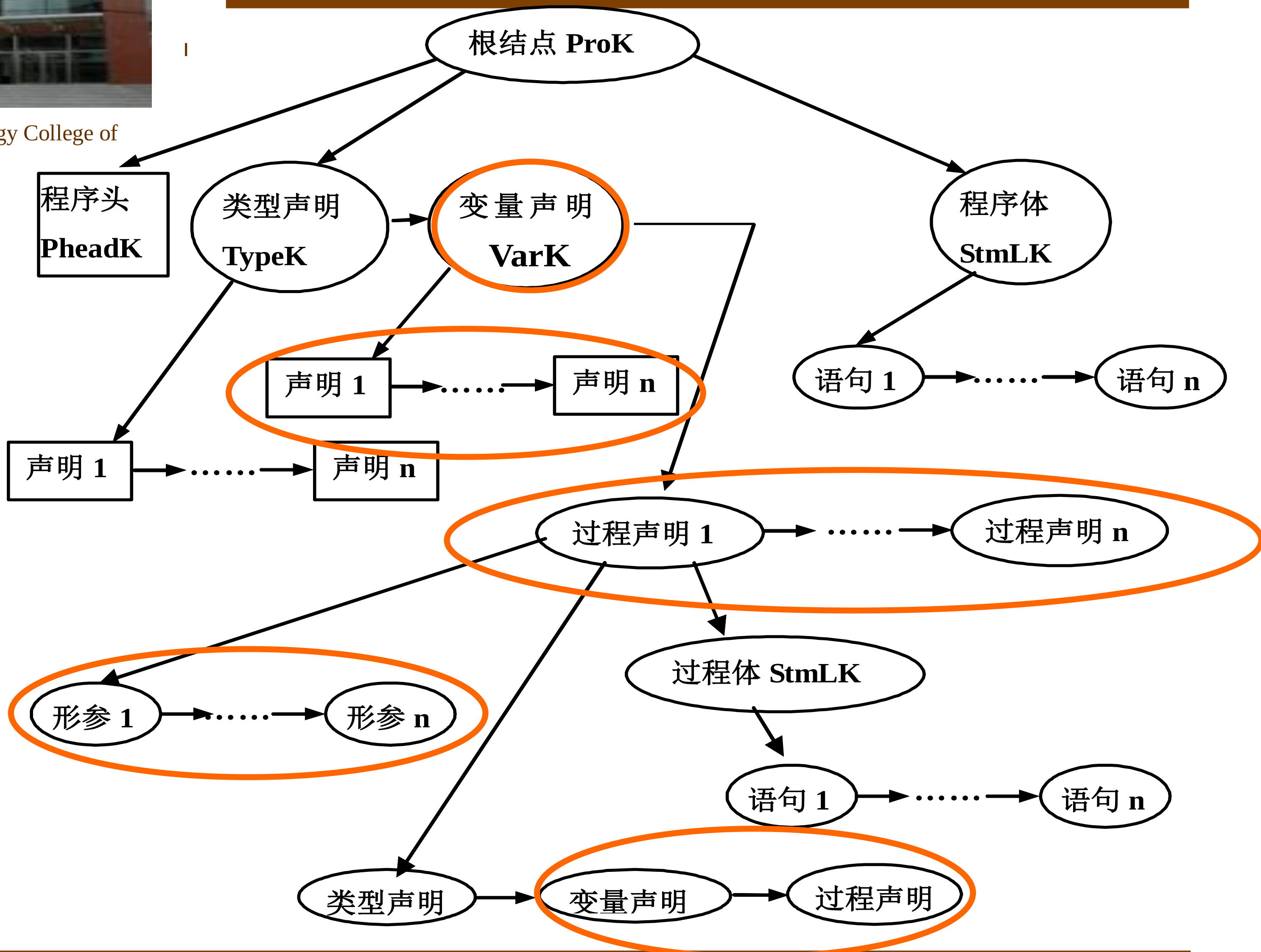
6.4 Semantic Analysis of Declaration

- 分析任务：扫描声明部分的程序段，并且做语法分析的同时建立定义性标识符的符号表，其间通过调用类型处理子程序检查类型的正确性。
- **General Process**
  - ***Collect*** **attributes of identifiers declared;**
  - ***Establish*** **internal representation of identifiers declared;**
  - ***Check*** **"repeat declaration" error through searching symbol table;**
  - ***Insert*** **into symbol table;**
- **Declaration**
  - **Constant declaration**
  - **Type declaration**
  - **Variable declaration**
  - **Function declaration**
  - **Label declaration (optional)**

Information Science and Technology College of Northeast Normal University

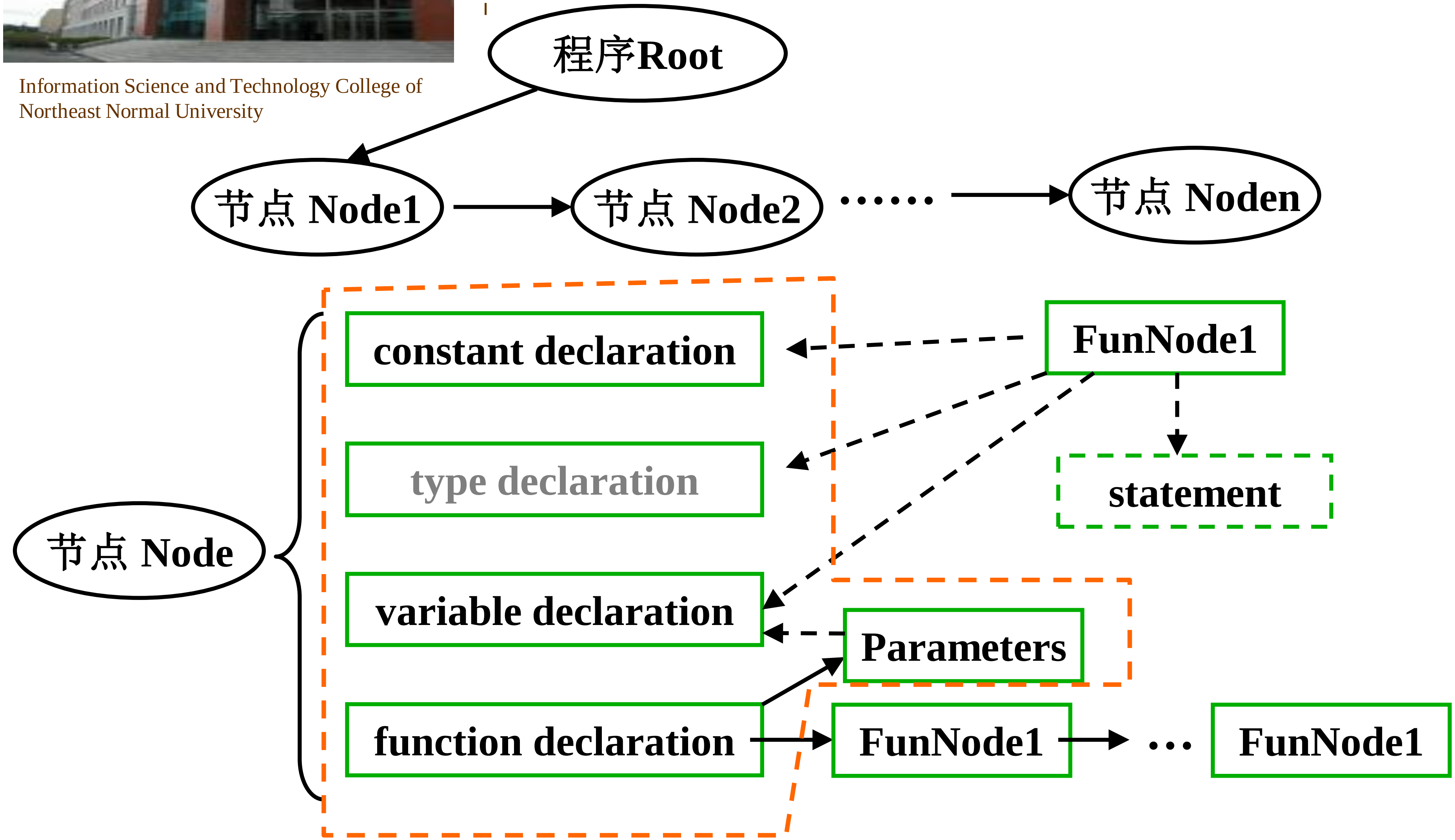
Modified Parse Tree

类PASCAL语言



Modified Parse Tree (C语言)

Information Science and Technology College of
Northeast Normal University



6.4 Semantic Analysis of Declaration

- **Constant declaration**
  - **General form: *id* = *number*;**
  - **Internal representation:**
    - **Establish the internal representation of *number*;**
    - **Establish the internal representation of the type of *number*;**
    - **Establish the internal representation of *id*;**

TypePtr	Kind	Value
	constKind	

6.4 Semantic Analysis of Declaration

- **Type declaration**
  - **General form: *id* = *type*;**
  - **Internal Representation:**
    - **Establish the internal representation of *type*;**
    - **Establish the internal representation of *id*;**

TypePtr	Kind
	typeKind

6.4 Semantic Analysis of Declaration

- **Variable declaration**
  - **General form:** ***type id*;**
  - **Internal Representation :**
    - **Establish the internal representation of *type*;**
    - **<CurrentLevel, CurrentOffset>**
    - **Access: dir/indir**
    - **Establish the internal representation of *id*;**
    - **CurrentOffset += sizeof(*type*)**

TypePtr	Kind	Access	Level	Offset
	varKind			

6.4 Semantic Analysis of Declaration

- **Function/Procedure declaration**
  - **General form: *type id* (params) { body }**
  - **Process of analysis:**
    - **Pointer to internal representation of *type*;**
    - **保存CurrentOffset， CurrentOffset = 0;**
    - **Establish internal representation of parameters (as variables, whose level is CurrentLevel+1)**
    - **Link all parameters;**
    - **Establish internal representation of *id* (whose level is CurrentLevel) ;**
    - **进入函数体，CurrentLevel ++;**
    - ***<u>Code</u>* will be set after codes have been created;**
    - ***<u>Size</u>* will be set after all variables declared in this function are handled;**
    - **恢复CurrentOffset；**

TypePtr	Kind	Class	Level	Param	Code	Size	Forward
	routKind	actual					

Information Science and Technology College of
Northeast Normal University

Main Content of Chapter 6

§ 6. Semantic Analysis

- 6.1 Overview of Semantic Analysis
- 6.2 Symbol Table
- 6.3 Semantic Analysis of Types
- 6.4 Semantic Analysis of Declaration
- 6.5 Semantic Analysis of Body
- 6.6 Attribute Grammar and Action Grammar

6.5 Semantic Analysis of Body

- **General checking**
  - **Whether identifiers in the statement are already declared;**
  - **Type-related semantic errors;**
- **Different elements in the Body**
  - **Assignment**
  - **Conditional statement**
  - **Loop statement**
  - **Function call**
  - **Expressions**
  - **Variables**

6.5 Semantic Analysis of Body

- **Assignment**
  - **General form: *<u>var</u>* = *<u>Expression</u>***
  - **General process:**
    - **Get the type of *<u>var</u>* ;**
    - **Get the result type of *<u>Expression</u>*;**
    - **Check whether the type of *<u>var</u>* and the type of *<u>Expression</u>* are compatible(赋值相容);**

6.5 Semantic Analysis of Body

- **Conditional statement**
  - **General form: if Exp S1 [else S2]**
  - **General Process:**
    - **Check whether the result type of Exp is boolean;**

6.5 Semantic Analysis of Body

- **Loop statement**
  - **Different loops**
    - **While Exp {S}**
      - **Check whether the result type of Exp is boolean;**
    - **For i=E1 to E2 {S}**
      - **Check whether the types of i, E1 and E2 are compatible;**

6.5 Semantic Analysis of Body

- **Function call**
  - **General form: *f* (E1, …., En)**
  - **General Process:**
    - **Search symbol table to get the attributes of *f*;**
    - **Check whether *f* is "routeKind";**
    - **Get the *result type* of Ei (i=1, …, n);**
    - **Check whether the number of actual parameters and the number of formal parameter of f is the same;**
    - **Check whether the type of i^{th} formal parameter and the i^{th} actual parameter is compatible; (类型和种类)**

6.5 Semantic Analysis of Body

- **Expressions**
  - Check “运算分量类型相容”；
  - 求表达式的结果类型；
  - 求表达式的种类
    - 数据
    - 地址（指针）
    - 过程/函数

x * (f(5) + a[i])

(1) Search x in Symbol Table
(2) x: (t1, varKind, L1, off1, dir)
(3) Search f in symbol table
(4) f: (t2, routKind, L2, paramL,…)
(5) Search a in Symbol Table
(6) a: (t3, varKind, L3, Off3, indir)
(7) Check whether t3 is arrayType;
(8) Check whether t2 and t3.elemTy are “+” compatible;
if yes, get the (result type t’, 种类);
(9) Check whether t1 and t’ are “*” compatible;
get (结果类型t，种类c)
(10) Return (t,c)

6.5 Semantic Analysis of Body

- **Variable**
  - **General form: V → id | V[Exp] | V.id | *V**
  - **标识符情形(id):**
    - **search symbol table to get its attributes, from which return (type, 种类);**
  - **数组下标变量V[Exp]:**
    - **check whether the result type of V is arrayKind,**
    - **whether the result type of Exp is correct,**
    - **returns (ElemType, 种类);**

6.5 Semantic Analysis of Body

- **Variable**
  - **结构/联合的域名V.id :**
    - **check whether the result type of V is structure or union,**
    - **whether id is one of its field,**
    - **return (fieldType, 种类)**
  - ***V:**
    - **check whether the result type of V is pointTy,**
    - **return (BaseType, 种类)**

Some Issues

- **Symbol Table**

Declaration	Action
constant declaration	Add constant identifier to symbol table
type declaration → enum	Add enum constant id to symbol table; Add enum type id to symbol table;
type declaration → other	Add type id to symbol table;
var declaration	Add variable id to symbol table;
Function declaration - function id	Add function id to symbol table;
- parameters	Add parameters(var id) to symbol table;
- declaration	Add id declared to symbol table;
- statements	Search for symbol table;

Some Issues

- **Level & Offset (L, Off)**

Declarations	(L, Off)
	(L, Off)
constant declaration	
	(L, Off)
type declaration	
	(L, Off)
var declaration	
type1 x1;	(L, Off+sizeof(type1))
…	
typen xn;	(L, Off+sizeof(type1))+…+sizeof(typen))
Function declaration	
- function id	(L, Off+sizeof(type1))+…+sizeof(typen))
- (ti a1, …, tm am)	(L+1, 0)
- declaration	(L+1, sizeof(t1)+…+sizeof(tm))
- statements	
	(L,Off+sizeof(type1))+…+sizeof(typen))

Exercise

- **Please write down (level,offset) at different red points of following program. If at point 0 it is (L, Off).**

```
    0
typedef struct {char name[30]; int age;} Student;
      1
Student Lee;
      2
 int i ;
   3
 int GetAge(Student S) 4 { return S.age;} 5
 void SetAge( int i) 6 {Lee.age=i; } 7
 void main() { student  Lee;  8 SetAge(10); 9
                          {int i = 20; 10 Lee.age = i;}  11
                           {int j = 30; 12 Lee.age = i;} 13 }
```

Main Content of Chapter 6

§ 6. Semantic Analysis

- 6.1 Overview of Semantic Analysis
- 6.2 Symbol Table
- 6.3 Semantic Analysis of Types
- 6.4 Semantic Analysis of Declaration
- 6.5 Semantic Analysis of Body
- 6.6 Attribute Grammar and Action Grammar

6.6 Attribute Grammar & Action Grammar

- **How to formally describe static semantics and implement static semantic analysis?**
- **Syntax directed methods(语法制导的方法)**
  - **Attribute Grammar = CFG + attributes, which was introduced by Knuth and used widely;**
  - **Action Grammar = CFG + actions, which was introduced in Prof. JIN's book;**
- **Attribute grammar and Action Grammar can be implemented automatically, the resulting program can be semantic analyzer;**

Attribute Grammar(属性文法)

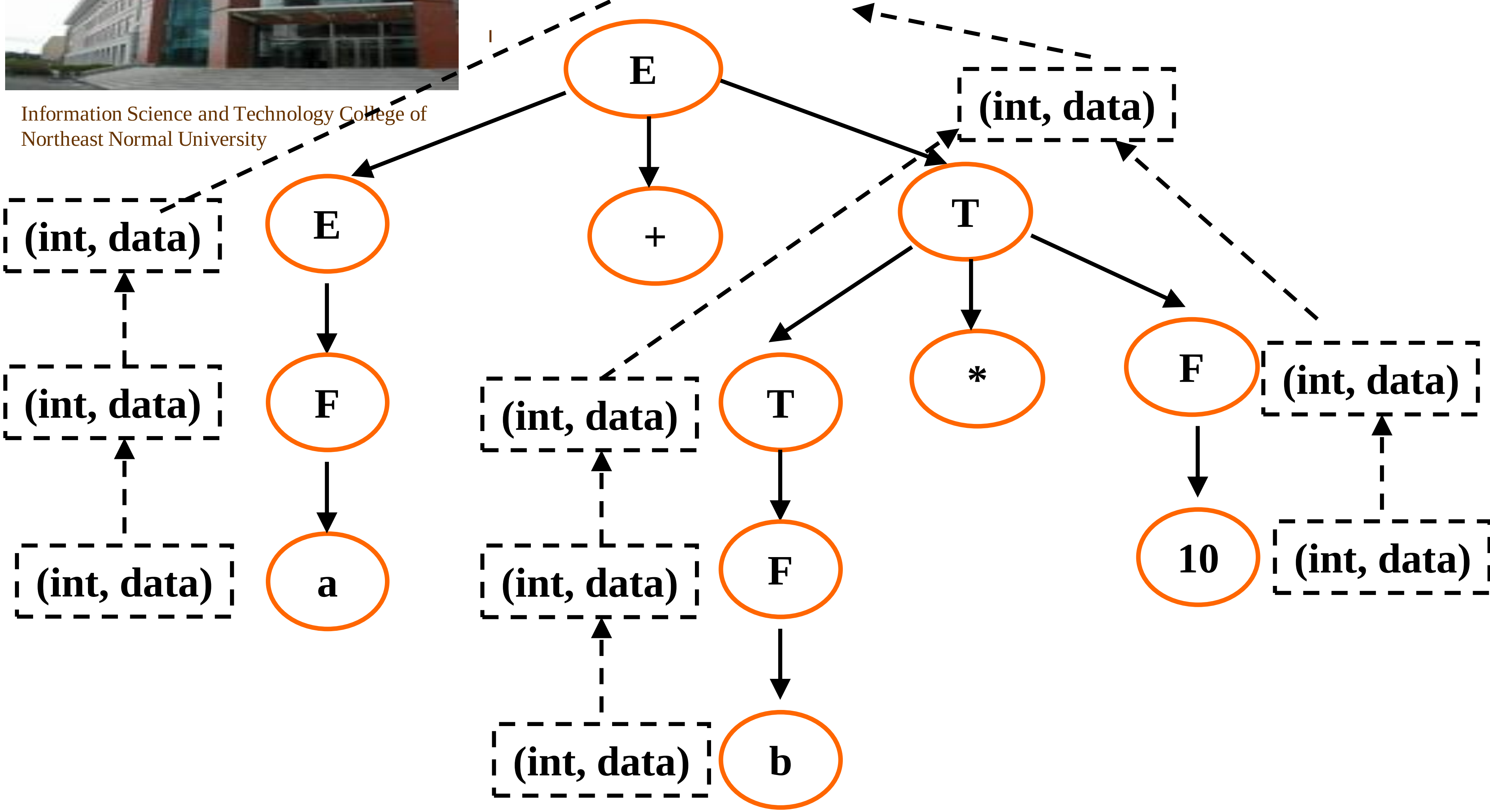
- **Overview**
  - 在CFG的基础上，描述静态语义(相关属性)；
  - 一种规范定义(specification)；
  - 广泛应用静态语义描述、程序转换 、程序分析、数据流分析等领域，是一个非常有用的抽象描述方式;
- **在编译器中的应用**
  - 语义检查
  - 代码生成
- **一个例子**

Attribute Grammar(属性文法)

- **Main idea**
  - **A given context free grammar *G***
  - **Define attributes for symbols (non-terminal or terminal)**
    - **Attributes could be of complex structure, representing value, type, parse tree, code, ……**
  - **For each production, define a rule for how to calculate the attributes of symbols in the production;**

Example of Attribute Grammar

符号	属性定义	产生式	属性计算规则
E	type, class	$E^1 \rightarrow E^2+T$	if (E^2.type=T.type) then {E^1.type= E^2.type; E^1.class = Data} else error;
T	type, class	$E \rightarrow T$	E.type= T.type; E.class = T.class
F	type, class	$T^1 \rightarrow T^2*F$	if (T^2.type=F.type) then {T^1.type= T^2.type; T^1.class = Data} else error;
n	type, class	$T \rightarrow F$	T.type= F.type; T.class = F.class
id	type, class	$F \rightarrow id$	F.type = id.type; F.class = id.class
+ *		$F \rightarrow n$	F.type = n.type; F.class = n.class
( )		$F \rightarrow (E)$	F.type = E.type; F.class = E.class


(int, data)
E
(int, data)
(int, data)
E
+
T
(int, data)
F
(int, data)
T
*
F
(int, data)
(int, data)
a
(int, data)
F
10
(int, data)
(int, data)
b

Action Grammar(动作文法)

- **Action Grammar**
  - **A CFG with action symbols appearing the right parts of productions;**
  - **each action symbol corresponds to a function or procedure;**
- **The definition of an action grammar depends on the parsing methods;**
- **Example**

Action Grammar for LL(1) Parsing

An action grammar for calculating the number of 1 and 0 in the input string

V_T = {0, 1, a}
V_N = {S, L}
S = S
P:
{
S → L
L → 0 L
L → 1 L
L → a
}

V_T = {0, 1, a}
V_N = {S, L}
S = S
P:
{
S → <init> L <print>
L → 0 <add0> L
L → 1 <add1> L
L → a
}

init()
{sum0 = 0; sum1 =0;}
print()
{printf(sum0, sum1);}
add0()
{sum0++;}
add1()
{sum1++;}

Working Process of Action Grammar

V_T = {0, 1, a}

V_N = {S, L}

S = S

P:

{

(1)S → <init> L <print>   {0,1}

(2)L → 0 <add0> L   {0}

(3)L → 1 <add1> L   {1}

(4)L → a   {a}

}

分析栈	输入串	动作
S	01a#	推导(1)
<init> L <print>	01a#	sum0=0; sum1=0;
L <print>	01a#	推导(2)
0 <add0> L <print>	01a#	匹配
<add0> L <print>	1a#	sum0=1;
L <print>	1a#	推导(3)
1 <add1> L <print>	1a#	匹配
<add1> L <print>	a#	sum1=1
L <print>	a#	推导(4)
a <print>	a#	匹配
<print>	#	“1,1”
	#	接受

Action Grammar for LR Parsing

V_T = {0, 1, a}
V_N = {S, L}
S = S
P:
{
  S → L
  L → 0 L
  L → 1 L
  L → a
}

V_T = {0, 1, a}
V_N = {S, L}
S = S
P:
{
  S → L <print>
  L → D L
  D → 1 <add1>
  D → 0 <add0>
  L → a
}

init()
{sum0 = 0; sum1 =1;}
print()
{printf(sum0, sum1);}
add0()
{sum0++;}
add1()
{sum1++;}

Working Process of Action Grammar

V_T = {0, 1, a}
V_N = {S, L}
S = S
P:
{
(1) S → L <print>
(2) L → D L
(3) D → 1 <add1>
(4) D → 0 <add0>
(5) L → a
}

符号栈	输入串	动作
<init>		sum0=0; sum1=0
	01a#	shift
0	1a#	Reduce (4)
D <add0>	1a#	sum0=1
D	1a#	Shift
D1	a#	Reduce (3)
DD<add1>	a#	sum1=1
DD	a#	Shift
DDa	#	Reduce(5)
DDL	#	Reduce(2)
DL	#	Reduce(2)
L<print>	#	Reduce(1) print
S	#	Accept

Summary of Chapter 6

- **Main function of Semantic Analysis?**
- **About Symbol table**
  - **What ?**
  - **The purpose of establishing symbol table;**
  - **Internal representation of identifiers, types & values;**
  - **Understanding (level, offset) for variables;**
  - **Understanding the scope for identifiers;**
  - **The organization of symbol table;**
- **How to establish symbol table during semantic analysis?**
  - **Add new item;**
  - **Check repeat declaration error;**
  - **Get attributes for identifiers;**
  - **Especially (level, offset)**

Summary of Chapter 6

- **The main task of semantic analysis for different syntactic structures of a program**
  - **Declaration**
    - **Constant**
    - **Type**
    - **Variable**
    - **Function/procedure**
  - **Body**
    - **Statements**
    - **Expressions**
    - **variables**

Summary of Chapter 6

- **The main idea of Attribute Grammar?**
- **The main idea of Action Grammar?**