



第6章 语义分析



李文生

2024年2月19日 星期一



北京邮电大学计算机学院(国家示范性软件学院)

SCHOOL OF COMPUTER SCIENCE(NATIONAL PILOT SOFTWARE ENGINEERING SCHOOL)BUPT

学习任务

■ 作业要求

- 掌握数据对象的类型表示
- 数据对象类型等价判断

语义分析考题示例

五、(10 分) 有如下 C 语言代码：

```
typedef int A[10];  
typedef int B[20];  
A a;  
typedef struct {int i;} S;  
typedef struct {int i;} T;  
S s;  
B * f() {return(&a);}  
T g() {return(s);}
```

(1) 写出这段代码中所定义的名字的类型表达式。

(2) C 语言对结构类型采用名字等价, 对其他类型采用结构等价。据此判断上述代码中是否存在类型错误。若存在, 请指明哪个语句中存在类型错误, 并说明产生类型错误的原因。

考点1：类型表达式的递归定义

(1) 基本类型是类型表达式

boolean、**char**、**integer**、和 **real**

错误类型 **type_error**、回避类型 **void**

```
typedef double real;  
real x, y;
```

```
typedef int arr[10];  
arr a, b;
```

(2) 类型名是类型表达式，因为类型表达式可以命名。

(3) 类型构造器作用于类型表达式的结果仍是类型表达式

① **数组**：如果T是类型表达式，那么 **array(I, T)** 是元素类型为T和下标集合为I的数组的类型表达式，I通常是一个整数域。

A:array[1..10] of integer;

A的类型表达式为：**array(1..10, integer)**

② **笛卡儿积**：如果 T_1 和 T_2 是类型表达式，那么它们的笛卡儿积 $T_1 \times T_2$ 也是类型表达式，假定 \times 是左结合的。

类型表达式的递归定义

③ 记录：把类型构造器 **record** 作用于记录中各域类型的笛卡儿积上，就形成了记录的类型表达式。

域类型是由域名和域的类型表达式组成的二元组。

```
B: record  
  i: integer;  
  c: char;  
  r: real  
end;
```

```
struct {  
  int i;  
  char c;  
  float r;  
} B;
```

```
typedef struct {  
  int age;  
  char name[10];  
} row;  
row table[10];
```

B的类型表达式: **record((i×integer)×(c×char)×(r×real))**

row 的类型表达式: **record((age×integer)×(name×array(0..9, char)))**

table 的类型表达式: **array(0..9, row)**

类型表达式的递归定义

④ 指针：如果T是类型表达式，那么 $\text{pointer}(T)$ 也是类型表达式，表示类型“指向类型T的对象的指针”。

如：p:↑row;

row *p;

P的类型表达式为： $\text{pointer}(\text{row})$

⑤ 函数：从定义域类型D到值域类型R的映射。类型表达式 $D \rightarrow R$

如：Pascal的内部函数 mod 的类型表达式： $\text{integer} \times \text{integer} \rightarrow \text{integer}$

用户定义的Pascal函数：function fun(a: char, b: integer): ↑integer;

函数 fun 的类型表达式： $\text{char} \times \text{integer} \rightarrow \text{pointer}(\text{integer})$

形参名字?

如：C 语言函数声明 int square(int x) { return x*x };

函数square的类型表达式为： $\text{integer} \rightarrow \text{integer}$

(4) 类型表达式中可以包含变量（称为类型变量），变量的值是类型表达式。

考点2.1 结构等价(structure type equivalence)

■ 两个类型表达式结构等价：

- 要么是同样的基本类型
- 要么是同样的构造器作用于结构等价的类型表达式。

■ 两个类型表达式结构等价当且仅当它们完全相同。

■ 例如：

- integer 仅等价于 integer
- pointer(integer) 仅等价于 pointer(integer)

■ 类型等价的变量，它们的类型具有相同的结构。

例：考虑如下Pascal声明

A: record i: integer f: real end;	B: record i: integer f: real end;	C: record f: real i: integer end;	D: record x: real y: integer end;
--	--	--	--

考虑如下C语言声明

struct recA { int i; char c; } a;	typedef struct { int i; char c; } recB; recB b;	struct { int i; char c; } c;
--	--	---

考点2.1 结构等价(structure type equivalence) (答案)

■ 两个类型表达式结构等价：

- 要么是同样的基本类型
- 要么是同样的构造器作用于结构等价的类型表达式。

■ 两个类型表达式结构等价当且仅当它们完全相同。

■ 例如：

- integer 仅等价于 integer
- pointer(integer) 仅等价于 pointer(integer)

■ 类型等价的变量，它们的类型具有相同的结构。

例：考虑如下Pascal声明

A: record i: integer f: real end;	B: record i: integer f: real end;	C: record f: real i: integer end;	D: record x: real y: integer end;
--	--	--	--

考虑如下C语言声明

struct recA { int i; char c; } a;	typedef struct { int i; char c; } recB; recB b;	struct { int i; char c; } c;
---	--	--

考点2.2 名字等价(name type equivalence)

- 多数语言(如Pascal、C等)允许用户定义类型名

```
typedef int int_var;  
int a;  
int_var b;
```

- C类型定义和变量声明 (6.1)

```
typedef struct {  
    int age;  
    char name[20];  
} recA;  
typedef recA *recP;  
recP a;  
recP b;  
recA *c, *d;  
recA *e;
```

- 问题：a、b、c、d、e是否具有相同的类型？

变量	类型表达式
a	recP
b	recP
c	pointer(recA)
d	pointer(recA)
e	pointer(recA)

- 关键：recP 和 pointer(recA) 是否等价？
- 回答：依赖于具体的实现系统

考点2.2 名字等价(name type equivalence)

- 名字等价把每个类型名看成是一个可区别的类型。
- 两个类型表达式名字等价，当且仅当它们完全相同。
- 名字等价的变量，或者它们在同一个声明语句中定义，或者使用相同类型名声明。
- 所有的名字被替换后，两个类型表达式成为结构等价的类型表达式，那么这两个表达式结构等价。
- 变量a、b、c、d、e的类型表达式：

变量	类型表达式
a	recP
b	recP
c	pointer(recA)
d	pointer(recA)
e	pointer(recA)

结构等价

名字等价

名字等价

名字不等价

类型等价实例：C语言

■ 使用名字等价和结构等价

□ 对于struct、enum和union的类型，使用名字等价

➤ 例外：在不同的文件中定义，使用结构等价

■ 有如下C语言声明：

```
struct {  
    short j;  
    char c;  
} x, y;  
struct {  
    short j;  
    char c;  
} b;
```

x、y 名字等价
x、y 与 b 名字不等价

Pascal 采用了与 C语言类似的规则：
类型构造器（如记录、数组、指针等）的
每次应用，都将产生一个新的内部名字。

```
struct rec1{  
    short j;  
    char c;  
};  
rec1 x, y;  
struct rec2{  
    short j;  
    char c;  
};  
rec2 b;
```

与声明6.1等效的声明6.2:

```
typedef struct {  
    int age;  
    char name[20];  
} recA;  
typedef recA *recP;  
recP a;  
recP b;  
recA *c, *d;  
recA *e;
```

```
typedef struct {  
    int age;  
    char name[20];  
} recA;  
typedef recA *recP;  
typedef recA *recD;  
typedef recA *recE;  
recP a;  
recP b;  
recD c, d;  
recE e;
```

■ 名字等价

- a 和 b 的类型: recP 等价
- c 和 d 的类型: recD 等价
- e 的类型: recE
- a、c、e 不等价



语义分析考题示例

五、(10 分) 有如下 C 语言代码：

```
typedef int A[10];  
typedef int B[20];  
A a;  
typedef struct {int i;} S;  
typedef struct {int i;} T;  
S s;  
B * f() {return(&a);}  
T g() {return(s);}
```

(1) 写出这段代码中所定义的名字的类型表达式。

(2) C 语言对结构类型采用名字等价, 对其他类型采用结构等价。据此判断上述代码中是否存在类型错误。若存在, 请指明哪个语句中存在类型错误, 并说明产生类型错误的原因。

语义分析考题示例（答案）

五、参考答案：

(1) 这段代码中所定义的名字及其类型表达式如下：

A: array(0..9, integer)

B: array(0..19, integer)

a: A

S: record(i * integer)

T: record(i * integer)

s: S

f: void \rightarrow pointer(B)

g: void \rightarrow T

(2) 函数 g 声明语句中存在类型错误。因为 g 返回值类型要求是 T, 而函数返回的 s 的类型是 S, 在 C 语言中, 对结构采取名字等价, 所以, S 与 T 类型不等价。

