



中国科学技术大学  
University of Science and Technology of China


类型检查

《编译原理和技术》

张昱

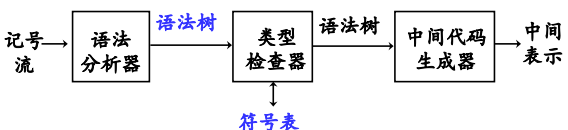
0551-63603804, yuzhang@ustc.edu.cn

中国科学技术大学  
计算机科学与技术学院



中国科学技术大学  
University of Science and Technology of China

本章内容



☐ 语义检查中最典型的部分——类型检查

☐ 其他的静态检查（不详细介绍）


■ 类型系统、类型检查、符号表的作用

■ 多态函数、重载

■ 控制流检查、唯一性检查、关联名字检查

张昱：《编译原理和技术》语法制导的翻译

2




中国科学技术大学  
University of Science and Technology of China

5.1 类型在编程语言中的作用

☐ 执行错误与安全语言

☐ 类型化语言与类型系统

☐ 类型的作用



中国科学技术大学  
University of Science and Technology of China

程序运行时的执行错误

☐ 会被捕获的错误（trapped error）

☐ 不会被捕获的错误(untrapped error)

■ 例：非法指令错误、非法内存访问、除数为零

■ 引起计算立即停止


■ 例：下标变量的访问越过了数组的末端；  
跳到一个错误的地址，该地址开始的内存正好代表一个指令序列

■ 错误可能会有一段时间未引起注意

希望可执行的程序不存在不会被捕获的错误

张昱：《编译原理和技术》语法制导的翻译

4



中国科学技术大学  
University of Science and Technology of China

安全语言

☐ 良行为的(well-behaved)程序

☐ 安全语言(safe language)

☐ 禁止错误(forbidden error)

■ 没有统一的定义

■ 如：良行为的程序定义为没有任何不会被捕获的程序

■ 定义：安全语言的任何合法程序都是良行为的

■ 设计类型系统,通过静态类型检查拒绝不会被捕获错误

■ 设计正好只拒绝不会被捕获错误的类型系统是困难的

■ 不会被捕获错误集合+ 会被捕获错误的一个子集

张昱：《编译原理和技术》语法制导的翻译

5



中国科学技术大学  
University of Science and Technology of China

类型化的语言

☐ 变量的类型

☐ 类型化的语言(typed language)

☐ 未类型化的语言(untyped language)

■ 限定了变量在程序执行期间的取值范围

■ 变量都被给定类型的语言

■ 表达式、语句等程序构造的类型都可以静态确定


例如，类型boolean的变量x在程序每次运行时的值只能是布尔值，not(x)总有意义

no static types

■ 不限制变量值范围的语言，如 LISP、JavaScript、Perl

张昱：《编译原理和技术》语法制导的翻译

6



中国科学技术大学


University of Science and Technology of China

## 类型化的语言

- 显式类型化语言
  - 类型是语法的一部分
- 隐式类型化的语言
  - 不存在隐式类型化的主流语言，但可能存在忽略类型信息的程序片段，如不需要程序员声明函数的参数类型

张昱：《编译原理和技术》语法制导的翻译

7



中国科学技术大学


University of Science and Technology of China

## 类型系统

- 语言的组成部分, 其构成成分是一组定型规则 (*typing rule*), 用来给各种程序构造指派类型
- 设计目的
  - 用静态检查的方式来保证合法程序在运行时的良行为
- 类型系统的形式化
  - 类型表达式、定型断言、定型规则
- 类型检查算法
  - 通常是静态地完成类型检查

张昱：《编译原理和技术》语法制导的翻译

8



中国科学技术大学

University of Science and Technology of China

## 类型可靠的语言

- 良类型的程序(*well-typed program*)
  - 没有类型错误的程序，也称**合法程序**
- 类型可靠 (*type sound*) 的语言
  - 所有良类型程序（合法程序）都是良行为的
  - 类型可靠的语言一定是安全的语言

若语言定义中，除类型系统外，没有用其它方式表示对程序的约束

语法的和静态的概念

动态的概念

类型化语言


安全语言

良类型程序

良行为的程序

张昱：《编译原理和技术》语法制导的翻译

9



中国科学技术大学


University of Science and Technology of China

## 类型检查

- 未类型化语言
  - 可以通过运行时的**类型推断和检查**来排除禁止错误
- 类型化语言
  - 类型检查也可以放在运行时完成，但影响效率
  - 一般都是静态检查，类型系统被用来支持静态检查
  - 通常也需要一些运行时的检查，如数组访问越界检查

张昱：《编译原理和技术》语法制导的翻译

10



中国科学技术大学

University of Science and Technology of China

## 一些实际的编程语言并不安全

禁止错误集合没有囊括所有不会被捕获的错误

例 C语言的共用体


```

union U { int u1; int *u2; } u;
int *p;
u.u1 = 10;
p = u.u2;
*p = 0;

```

张昱：《编译原理和技术》语法制导的翻译

11



中国科学技术大学

University of Science and Technology of China

## 一些实际的编程语言并不安全

- C语言
  - 有很多不安全但被广泛使用的特征，如：
    - 指针算术运算、类型强制、参数个数可变
  - 在语言设计的历史上，安全性考虑不足是因为当时强调代码的执行效率
- 在现代语言设计上，安全性的位置越来越重要
  - C的一些问题已经在C++中得以缓和
  - 更多一些问题在Java中已得到解决

张昱：《编译原理和技术》语法制导的翻译

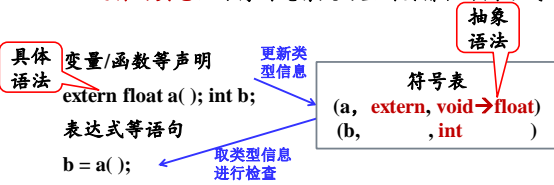
12



## 类型化语言的优点

### 从工程的观点看

- **开发的实惠**: 较早发现错误、类型信息具有文档作用
- **编译的实惠**: 程序模块可以相互独立地编译
- **运行的实惠**: 可得到更有效的空间安排和访问方式



张昱:《编译原理和技术》语法制导的翻译

13



## 5.2 描述类型系统的语言

- 类型系统的形式化
  - 断言、推理规则
- 类型检查和类型推断

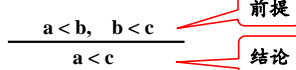


## 类型系统的形式化

### 类型系统是一种逻辑系统

有关自然数的逻辑系统

- 自然数表达式 (需要定义它的语法)  
 $a+b, 3$
- 良形公式 (逻辑断言, 需要定义它的语法)  
 $a+b=3, (d=3) \wedge (c<10)$
- 推理规则



张昱:《编译原理和技术》语法制导的翻译

15



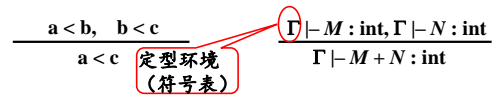
## 类型系统的形式化

### 类型系统是一种逻辑系统

有关自然数的逻辑系统

类型系统

- 自然数表达式  
 $a+b, 3$
- 良形公式  
 $a+b=3, (d=3) \wedge (c<10)$
- 推理规则
- 类型表达式  
 $\text{int}, \text{int} \rightarrow \text{int}$
- 定型断言 (typing assertion)  
 $x:\text{int} \vdash x+3 : \text{int}$
- 定型规则 (typing rules)



张昱:《编译原理和技术》语法制导的翻译

16



## 断言

### 断言的形式

$\Gamma \vdash S$  S的所有自由变量都声明在 $\Gamma$ 中  
其中

- $\Gamma$ 是一个静态定型环境 (编译器实现中的符号表), 如  $x_1:T_1, \dots, x_n:T_n$
- S的形式随断言形式的不同而不同
- 断言有三种具体形式

张昱:《编译原理和技术》语法制导的翻译

17



## 断言的种类

### 环境断言

$\Gamma \vdash \Diamond$  该断言表示 $\Gamma$ 是良形的环境

- 将用推理规则来定义环境的语法 (而不是用文法)

### 语法断言

$\Gamma \vdash \text{nat}$  在环境 $\Gamma$ 下, nat是类型表达式

- 将用推理规则来定义类型表达式的语法

### 定型断言

$\Gamma \vdash M : T$  在环境 $\Gamma$ 下, M具有类型T

例:  $\emptyset \vdash \text{true} : \text{boolean}$   $x : \text{nat} \vdash x+1 : \text{nat}$

- 将用推理规则来确定程序构造实例的类型

张昱:《编译原理和技术》语法制导的翻译

18

中国科学技术大学

University of Science and Technology of China

断言的有效性、推理规则

断言的有效性

有效断言(valid assertion)

$\Gamma \vdash \text{true} : \text{boolean}$

无效断言(invalid assertion)

$\Gamma \vdash \text{true} : \text{nat}$

推理规则(inference rules)

$\frac{\Gamma_1 \vdash S_1, \dots, \Gamma_n \vdash S_n}{\Gamma \vdash S}$

前提(premise)、结论(conclusion)

公理(axiom) (前提为空)、推理规则

张昱:《编译原理和技术》语法制导的翻译

19

中国科学技术大学

University of Science and Technology of China

推理规则

(规则名)

(注释)

推理规则

(注释)

环境规则

(Env  $\emptyset$ )

$\frac{}{\emptyset \vdash \diamond}$

空环境是良形的环境

语法规则

(Type Bool)

$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \text{boolean}}$

在环境 $\Gamma$ 下,  
 $M + N$ 是int类型

boolean是类型表达式

定型规则

(Val +)

$\frac{\Gamma \vdash M : \text{int}, \Gamma \vdash N : \text{int}}{\Gamma \vdash M + N : \text{int}}$

张昱:《编译原理和技术》语法制导的翻译

20

中国科学技术大学

University of Science and Technology of China

类型检查

类型检查(type checking)

用语法制导的方式, 根据上下文有关的**定型规则**来判断程序构造是否为良类型的程序构造的过程

可以边解析边检查, 也可以在访问AST时进行检查

类型推断(type inference)

类型信息不完全情况下的定型判定问题

例如:  $f(x : t) = E$  和  $f(x) = E$  的区别

张昱:《编译原理和技术》语法制导的翻译

21

中国科学技术大学

University of Science and Technology of China

5.3 简单类型检查器的说明

一个简单的语言及类型系统

类型检查

中国科学技术大学

University of Science and Technology of China

一个简单的语言

$P \rightarrow D ; S$   
 $D \rightarrow D ; D \mid \text{id} : T$   
 $T \rightarrow \text{boolean} \mid \text{integer} \mid \text{array} [\text{num}] \text{ of } T \mid \uparrow T \mid T \rightarrow T$   
 $S \rightarrow \text{id} := E \mid \text{if } E \text{ then } S \mid \text{while } E \text{ do } S \mid S ; S$   
 $E \rightarrow \text{truth} \mid \text{num} \mid \text{id} \mid E \text{ mod } E \mid E [ E ] \mid E \uparrow \mid E ( E )$

例

$i : \text{integer};$   
 $j : \text{integer};$   
 $j := i \text{ mod } 2000$

张昱:《编译原理和技术》语法制导的翻译

23

中国科学技术大学

University of Science and Technology of China

类型系统

环境规则

(Env  $\emptyset$ )

$\frac{}{\emptyset \vdash \diamond}$

(Decl Var)

$\frac{\Gamma \vdash T, \text{id} \notin \text{dom}(\Gamma)}{\Gamma, \text{id} : T \vdash \diamond}$

其中id:T是该简单语言的一个声明语句

遇到一个声明语句, 则向定型环境(符号表)中增加一个符号定型

张昱:《编译原理和技术》语法制导的翻译

24

中国科学技术大学

University of Science and Technology of China

## 类型系统

□ 语法规则

(Type Bool)

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{boolean}}$$

(Type Int)

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{integer}}$$

(Type Void)

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{void}}$$

void用于表示语句类型

编程语言和定型断言的类型表达式并非完全一致

张昱：《编译原理和技术》语法制导的翻译

25

中国科学技术大学

University of Science and Technology of China

## 类型系统

□ 语法规则

(Type Ref) ( $T \neq \text{void}$ )

$$\frac{\Gamma \vdash T}{\Gamma \vdash \text{pointer}(T)}$$

具体语法:  $\uparrow T$

(Type Array) ( $T \neq \text{void}$ )

$$\frac{\Gamma \vdash T, \Gamma \vdash N : \text{integer} \quad (N > 0)}{\Gamma \vdash \text{array}(N, T)}$$

具体语法:  $\text{array}[N] \text{ of } T$

(Type Function) ( $T_1, T_2 \neq \text{void}$ )

$$\frac{\Gamma \vdash T_1, \Gamma \vdash T_2}{\Gamma \vdash T_1 \rightarrow T_2}$$

定型断言中的类型表达式用的是抽象语法

张昱：《编译原理和技术》语法制导的翻译

26

中国科学技术大学

University of Science and Technology of China

## 类型系统 -- 定型规则

□ 定型规则——表达式

(Exp Truth)

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{truth} : \text{boolean}}$$

(Exp Num)

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{num} : \text{integer}}$$

(Exp Id)

$$\frac{\Gamma_1, \text{id} : T, \Gamma_2 \vdash \Diamond}{\Gamma_1, \text{id} : T, \Gamma_2 \vdash \text{id} : T}$$

张昱：《编译原理和技术》语法制导的翻译

27

中国科学技术大学

University of Science and Technology of China

## 类型系统 -- 定型规则

□ 定型规则——表达式

(Exp Mod)

$$\frac{\Gamma \vdash E_1 : \text{integer}, \Gamma \vdash E_2 : \text{integer}}{\Gamma \vdash E_1 \text{ mod } E_2 : \text{integer}}$$

(Exp Index)

$$\frac{\Gamma \vdash E_1 : \text{array}(N, T), \Gamma \vdash E_2 : \text{integer}}{\Gamma \vdash E_1[E_2] : T}$$

$(0 \leq E_2 \leq N-1)$

(Exp Deref)

$$\frac{\Gamma \vdash E : \text{pointer}(T)}{\Gamma \vdash \uparrow E : T}$$

(Exp FunCall)

$$\frac{\Gamma \vdash E_1 : T_1 \rightarrow T_2, \Gamma \vdash E_2 : T_1}{\Gamma \vdash E_1(E_2) : T_2}$$

张昱：《编译原理和技术》语法制导的翻译

28

中国科学技术大学

University of Science and Technology of China

## 类型系统 -- 定型规则

□ 定型规则——语句

(State Assign) ( $T = \text{boolean or } T = \text{integer}$ )

$$\frac{\Gamma \vdash \text{id} : T, \Gamma \vdash E : T}{\Gamma \vdash \text{id} := E : \text{void}}$$

(State If)

$$\frac{\Gamma \vdash E : \text{boolean}, \Gamma \vdash S : \text{void}}{\Gamma \vdash \text{if } E \text{ then } S : \text{void}}$$

(State While)

$$\frac{\Gamma \vdash E : \text{boolean}, \Gamma \vdash S : \text{void}}{\Gamma \vdash \text{while } E \text{ do } S : \text{void}}$$

(State Seq)

$$\frac{\Gamma \vdash S_1 : \text{void}, \Gamma \vdash S_2 : \text{void}}{\Gamma \vdash S_1; S_2 : \text{void}}$$

张昱：《编译原理和技术》语法制导的翻译

29

中国科学技术大学

University of Science and Technology of China

## 类型检查

□ 设计语法制导的类型检查器

■ 设计依据: 前面定义的类型系统

■ 类型环境  $\Gamma$  的信息进入符号表

■ 对类型表达式采用抽象语法

具体:  $\text{array}[N] \text{ of } T$

抽象:  $\text{array}(N, T)$


$\uparrow T$

$\text{pointer}(T)$

■ 考虑到报错的需要, 增加了类型 `type_error`

张昱：《编译原理和技术》语法制导的翻译

30



中国科学技术大学

University of Science and Technology of China

类型检查——声明语句

$$D \rightarrow D; D \quad //D1$$

$$D \rightarrow id : T \quad \{addtype(id.entry, T.type) \quad //D2$$

*addtype*: 把类型信息填入符号表

如果是在访问AST时进行类型检查, 该怎么做呢?

如, 可以在 *exitD2* (ast) 中增加对*addtype*的调用

如何表达多个声明*D1*呢? (lab1-3)


组织成list (可以用现成的表示线性表的容器类等)

如何处理多个声明*D1*呢?

对list 中元素的迭代访问 (可以用现成的Iterator等)

张昱: 《编译原理和技术》语法制导的翻译

31



中国科学技术大学

University of Science and Technology of China

类型检查——声明语句

$$D \rightarrow D; D$$

$$D \rightarrow id : T \quad \{addtype(id.entry, T.type)\}$$

$$T \rightarrow boolean \quad \{T.type = boolean\} \quad (Bool, --)$$

$$T \rightarrow integer \quad \{T.type = integer\} \quad (Int, --)$$

$$T \rightarrow \uparrow T_1 \quad \{T.type = pointer(T_1.type)\} \quad (Pointer, T1)$$

$$T \rightarrow array[num] of T_1 \quad \{T.type = array(num.val, T_1.type)\} \quad (Array, T1, num)$$


$$T \rightarrow T_1 ' \rightarrow ' T_2 \quad \{T.type = T_1.type \rightarrow T_2.type\} \quad (Fun, T1, T2)$$

如何表示不同的类型?

(类型类别, 该类别类型的其他信息)

张昱: 《编译原理和技术》语法制导的翻译

32



中国科学技术大学

University of Science and Technology of China

类型检查——表达式

$$E \rightarrow truth \quad \{E.type = boolean\}$$


$$E \rightarrow num \quad \{E.type = integer\}$$

$$E \rightarrow id \quad \{E.type = lookup(id.entry)\}$$

查符号表, 获取id的类型

张昱: 《编译原理和技术》语法制导的翻译

33



中国科学技术大学

University of Science and Technology of China

类型检查——表达式

$$E \rightarrow truth \quad \{E.type = boolean\}$$

$$E \rightarrow num \quad \{E.type = integer\}$$


$$E \rightarrow id \quad \{E.type = lookup(id.entry)\}$$

$$E \rightarrow E_1 \bmod E_2 \quad \{E.type = \text{if } E_1.type == integer \text{ and } E_2.type == integer \text{ then } integer \text{ else } type\_error\}$$

$$E \rightarrow E_1 [E_2] \quad \{E.type = \text{if } E_2.type == integer \text{ and } E_1.type == array(s, t) \text{ then } t \text{ else } type\_error\}$$

张昱: 《编译原理和技术》语法制导的翻译

34



中国科学技术大学

University of Science and Technology of China


类型检查——表达式

$$E \rightarrow E_1 \uparrow \{ E.type = \text{if } E_1.type == pointer(t) \text{ then } t \text{ else } type\_error \}$$

$$E \rightarrow E_1 (E_2) \{ E.type = \text{if } E_2.type == s \text{ and } E_1.type == s \rightarrow t \text{ then } t \text{ else } type\_error \}$$

张昱: 《编译原理和技术》语法制导的翻译

35



中国科学技术大学

University of Science and Technology of China


类型转换

$$E \rightarrow E_1 \text{ op } E_2$$

$$\{E.type = \text{if } E_1.type == integer \text{ and } E_2.type == integer \text{ then } integer \text{ else if } E_1.type == integer \text{ and } E_2.type == real \text{ then } real \text{ else if } E_1.type == real \text{ and } E_2.type == integer \text{ then } real \text{ else if } E_1.type == real \text{ and } E_2.type == real \text{ then } real \text{ else } type\_error \}$$

张昱: 《编译原理和技术》语法制导的翻译

36



中国科学技术大学  
University of Science and Technology of China

类型检查——语句


$S \rightarrow id := E \{ \text{if } (id.type == E.type \ \&\& \ E.type \in \{boolean, integer\}) \ S.type = void; \text{ else } S.type = type\_error; \}$

$S \rightarrow \text{if } E \text{ then } S_1 \{ S.type = \text{if } E.type == boolean \text{ then } S_1.type \text{ else } type\_error \}$

$S \rightarrow \text{while } E \text{ do } S_1 \{ S.type = \text{if } E.type == boolean \text{ then } S_1.type \text{ else } type\_error \}$

$S \rightarrow S_1; S_2 \quad \{ S.type = \text{if } S_1.type == void \text{ and } S_2.type == void \text{ then } void \text{ else } type\_error \}$

张昱：《编译原理和技术》语法制导的翻译37



中国科学技术大学  
University of Science and Technology of China

类型检查——程序

$P \rightarrow D; S \quad \{ P.type = \text{if } S.type == void \text{ then } void \text{ else } type\_error \}$

张昱：《编译原理和技术》语法制导的翻译38



中国科学技术大学  
University of Science and Technology of China

现代编译器的主流实现

□ [ParseTree]→ AST → 类型检查

□ 类型检查器的实现

■ 一般是对语法树进行类型检查

设计实现的关键：

■ 符号表的设计：如何表示不同的类型

■ 语法树的Visitor设计

回顾：ANTLR会生成与标签对应的语法结构的enter和exit方法

可以带标签(标签名, 后跟空格或换行)

antlr: e ::= e # Mult | e ::= e # Add | INT # int;

ANTLR为每个标签产生规则上下文类 XXXParserRuleContext

有何用处?

ANTLR会生成与标签对应的语法结构的enter和exit方法

public interface XXXListener extends ParserTreeListener {

void enterMult(XXXParserRuleContext ctx);

void exitMult(XXXParserRuleContext ctx);

.....

}

张昱：《编译原理和技术》40



中国科学技术大学  
University of Science and Technology of China

现代编译器的主流实现

□ [ParseTree]→ AST → 类型检查

□ 用ANTLR构造分析器

ParseTree→AST

■ syntax\_tree\_node

■ 访问者

syntax\_tree\_builder

antlrcpp: Any syntax\_tree\_builder::visitExpr(CIParser::ExpContext \*ctx) {

return visit(ctx);

if (result.iscsyntax\_tree\_node() result.ascsyntax\_tree\_node() >());

return ptrcsyntax\_tree\_node(result.ascsyntax\_tree\_node() >());

}

张昱：《编译原理和技术》语法制导的翻译40



中国科学技术大学  
University of Science and Technology of China

现代编译器的主流实现

□ [ParseTree]→ AST → 类型检查

□ AST的定义

■ syntax\_tree\_node

Public Member Functions

virtual void accept(syntax\_tree\_visitor &visitor)=0

Public Attributes

int line

int pos

■ syntax\_tree

□ 访问者

syntax\_tree\_visitor

可以带标签(标签名, 后跟空格或换行)

antlr: e ::= e # Mult | e ::= e # Add | INT # int;

ANTLR为每个标签产生规则上下文类 XXXParserRuleContext

有何用处?

ANTLR会生成与标签对应的语法结构的enter和exit方法

public interface XXXListener extends ParserTreeListener {

void enterMult(XXXParserRuleContext ctx);

void exitMult(XXXParserRuleContext ctx);

.....

}

张昱：《编译原理和技术》语法制导的翻译41



中国科学技术大学  
University of Science and Technology of China

例题 1

编译时的控制流检查的例子

```
main() {  
    printf("\n%d\n",gcd(4,12));  
    continue;  
}
```

编译时的报错如下:

continue.c: In function 'main':  
continue.c:3: continue statement not within a loop

张昱：《编译原理和技术》语法制导的翻译42

7



## 例题 2

编译时的唯一性检查的例子

```
main() {  
    int i;  
    switch(i){  
        case 10: printf("%d\n", 10); break;  
        case 20: printf("%d\n", 20); break;  
        case 10: printf("%d\n", 10); break;  
    }  
}
```

编译时的报错如下:  
switch.c: In function 'main':  
switch.c:6: duplicate case value  
switch.c:4: this is the first entry for that value

张昱:《编译原理和技术》语法制导的翻译

43



## 例题 3

C语言

- 称&为地址运算符, &a为变量a的地址
- 数组名代表数组第一个元素的地址

问题:

如果a是一个数组名, 那么表达式a和&a的值都是数组a第一个元素的地址, 它们的使用是否有区别?

用四个C文件的编译报错或运行结果来提示

张昱:《编译原理和技术》语法制导的翻译

44



## 例题 3

```
typedef int A[10][20];  
A a;
```

```
A *fun() {  
    return(a);  
}
```

该函数在Linux上用gcc编译, 报告的错误如下:

第5行: warning: return from incompatible pointer type

张昱:《编译原理和技术》语法制导的翻译

45



## 例题 3

```
typedef int A[10][20];  
A a;
```

```
A *fun() {  
    return(&a);  
}
```

该函数在Linux上用gcc编译时, 没有错误

张昱:《编译原理和技术》语法制导的翻译

46



## 例题 3

```
typedef int A[10][20];  
typedef int B[20];  
A a;
```

```
B *fun() {  
    return(a);  
}
```

该函数在Linux上用gcc编译时, 没有错误

张昱:《编译原理和技术》语法制导的翻译

47



## 例题 3

```
typedef int A[10][20];  
A a;
```

```
fun() { printf("%d,%d,%d\n", a, a+1, &a+1);}
```

```
main() { fun(); }
```


该程序的运行结果是:

134518112, 134518192, 134518912

张昱:《编译原理和技术》语法制导的翻译

48





中国科学技术大学

University of Science and Technology of China


例题 3

结论

对一个  $t$  类型的数组  $a[i_1][i_2] \dots [i_n]$  来说,  
 表达式  $a$  的类型是:  
 $\text{pointer}(\text{array}(0..i_2-1, \dots \text{array}(0..i_n-1, t) \dots))$   
  
 表达式  $\&a$  的类型是:  
 $\text{pointer}(\text{array}(0..i_1-1, \dots \text{array}(0..i_n-1, t) \dots))$

张昱:《编译原理和技术》语法制导的翻译

49



中国科学技术大学


University of Science and Technology of China

5.4 类型表达式的等价

□ 类型表达式的命名

□ 名字等价、结构等价

□ 记录类型的定义



中国科学技术大学

University of Science and Technology of China

类型表达式的等价

□ 对类型表达式命名= $\rightarrow$  如何解释类型表达式相同?

■ 结构等价、名字等价


■ 是类型表达式的一个语法规约, 而不是引入新的类型

```

typedef cell *link;
link next;
link last;
cell *p;
cell *q, *r;
          
```

张昱:《编译原理和技术》语法制导的翻译

51



中国科学技术大学

University of Science and Technology of China

结构等价

□ 结构等价

■ 无类型名时, 两个类型表达式完全相同

■ 有类型名时, 用类型名所定义的类型表达式代换它们, 所得表达式完全相同 (类型定义无环时)


```

typedef cell *link;
link next;
link last;
cell *p;
cell *q, *r;
          
```

next, last, p, q和r结构等价

张昱:《编译原理和技术》语法制导的翻译

52



中国科学技术大学

University of Science and Technology of China

结构等价测试

□  $\text{sequiv}(s, t)$  (无类型名时)

if  $s$  和  $t$  是相同的基本类型 then

return true

else if  $s == \text{array}(s_1, s_2)$  and  $t == \text{array}(t_1, t_2)$  then

return  $\text{sequiv}(s_1, t_1)$  and  $\text{sequiv}(s_2, t_2)$

else if  $s == s_1 \times s_2$  and  $t == t_1 \times t_2$  then

return  $\text{sequiv}(s_1, t_1)$  and  $\text{sequiv}(s_2, t_2)$

else if  $s == \text{pointer}(s_1)$  and  $t == \text{pointer}(t_1)$  then

return  $\text{sequiv}(s_1, t_1)$


else if  $s == s_1 \rightarrow s_2$  and  $t == t_1 \rightarrow t_2$  then

return  $\text{sequiv}(s_1, t_1)$  and  $\text{sequiv}(s_2, t_2)$

else return false

张昱:《编译原理和技术》语法制导的翻译

53



中国科学技术大学

University of Science and Technology of China

名字等价

□ 名字等价

■ 把每个类型名看成是一个可区分的类型

■ 两个类型表达式不做名字代换就结构等价

```

typedef cell *link;
link next;
link last;
cell *p;
cell *q, *r;
          
```

next和last名字等价

p, q和r名字等价

张昱:《编译原理和技术》语法制导的翻译

54

9



## 类型表达式的等价

Pascal语言的许多实现用隐含的类型名和每个声明的标识符联系起来

```
type link = ↑cell;      type link = ↑cell;
var  next : link;      np = ↑cell;
    last : link;      nqr = ↑cell;
    p : ↑cell;      var  next : link;
    q, r : ↑cell;    last : link;
                        p : np;
                        q : nqr;
                        r : nqr;
```

p与q和r不是名字等价

张昱：《编译原理和技术》语法制导的翻译

55



## 记录类型

### 记录类型

- 记录类型可看成其各个域类型的积类型
- 记录和积之间的主要区别是记录的域被命名

例如，C语言的记录类型

```
typedef struct {
    int address;
    char lexeme [15 ];
}row;
的类型表达式是
record(address : int, lexeme : array(15, char) )
```

张昱：《编译原理和技术》语法制导的翻译

56



## 记录类型

### 定型规则

(Type Record)  $(l_i \text{ 是有区别的})$

$$\frac{\Gamma \vdash T_1, \dots, \Gamma \vdash T_n}{\Gamma \vdash \text{record}(l_1:T_1, \dots, l_n:T_n)}$$

(Val Record)  $(l_i \text{ 是有区别的})$

$$\frac{\Gamma \vdash M_1:T_1, \dots, \Gamma \vdash M_n:T_n}{\Gamma \vdash \text{record}(l_1=M_1, \dots, l_n=M_n) : \text{record}(l_1:T_1, \dots, l_n:T_n)}$$

(Val Record Select)

$$\frac{\Gamma \vdash M : \text{record}(l_1:T_1, \dots, l_n:T_n)}{\Gamma \vdash M.l_j : T_j \quad (j \in 1..n)}$$

张昱：《编译原理和技术》语法制导的翻译

57



## 类型表示中的环

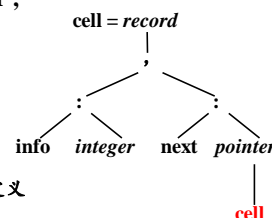
```
type link = ↑ cell ;
```

```
cell = record
```

```
    info : integer ;
```

```
    next : link
```

```
end;
```



引入环的话，递归定义  
的类型名可以替换掉

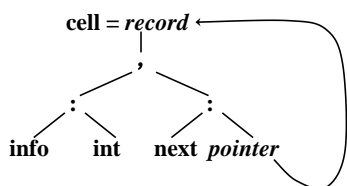
张昱：《编译原理和技术》语法制导的翻译

58



## 类型表示中的环

```
typedef struct cell {
    int info;
    struct cell *next;
} cell;
```



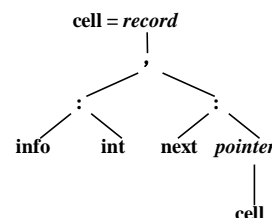
张昱：《编译原理和技术》语法制导的翻译

59



## 类型表示中的环

C语言对除记录（结构体）、共用体以外的所有类型使用结构等价，而对记录类型用的是名字等价，以避免类型图中的环。



张昱：《编译原理和技术》语法制导的翻译

60



## 例题 4

在X86/Linux机器上，编译器报告最后一行有错误：  
incompatible types in return

```
typedef int A1[10];      | A2 *fun1() {
typedef int A2[10];      |     return(&a);
A1 a;                    | }
typedef struct {int i;}S1; | S2 fun2() {
typedef struct {int i;}S2; |     return(s);
S1 s;                    | }
```

在C语言中，数组和结构体都是构造类型，为什么上面第2个函数有类型错误，而第1个函数却没有？

张昱：《编译原理和技术》语法制导的翻译

61



## 5.5 多态函数

- 参数化多态
- 类型系统的定义
- 类型检查



## 多态函数的引出

例 如何编写求表长的通用程序？

```
typedef struct {
    int info;
    link next;
} cell, *link;
```

unknown type name 'link'

张昱：《编译原理和技术》语法制导的翻译

63



## 多态函数的引出

例 如何编写求表长的通用程序？

```
typedef struct cell{      int length(link lptr) {
    int info;              int len = 0;
    struct cell *next;     link p = lptr;
} cell, *link;            while (list != NULL) {
                           len++;
                           p = p->next;
                           }
                           return len;
                           }
```

计算过程与表元的数据类型无关，但语言的类型系统使该函数不能通用

张昱：《编译原理和技术》语法制导的翻译

64



## 多态函数的引出

例 如何编写求表长的通用程序？

用ML语言很容易写出求表长的程序而不管表元的类型

```
fun length (lptr) =
  if null (lptr) then 0
  else length (tl (lptr)) + 1;
tl- 返回表尾    null-测试表是否为空
```

```
length ( ["sun", "mon", "tue"] )
length ( [10, 9, 8] )
都等于3
```

张昱：《编译原理和技术》语法制导的翻译

65



## 参数化多态

- 多态函数(polymorphic functions) 参数化多态
  - 允许函数参数的类型有多种不同的情况
  - 函数体中语句的执行能适应参数为不同类型的情况
- 多态算符(polymorphic operators) Ad-hoc多态
  - 例如：数组索引、函数应用、通过指针间接访问相应操作的代码段接受不同类型的数组、函数等
  - C语言手册中关于取地址算符&的论述是：如果运算对象的类型是‘...’，那么结果类型是指向‘...’的指针”

张昱：《编译原理和技术》语法制导的翻译

66



中国科学技术大学

University of Science and Technology of China

## 类型变量及其应用

□ 类型变量

length的类型可以写成 $\forall \alpha. list(\alpha) \rightarrow integer$

类型变量的引入便于讨论未知类型

如, 在不要求标识符的声明先于使用的语言中, 可以通过使用类型变量来确定程序变量的类型

function deref (p);

-- 对p的类型一无所知:  $\beta$

begin

return p↑

--  $\beta = pointer(\alpha)$

end;

deref 的类型是 $\forall \alpha. pointer(\alpha) \rightarrow \alpha$

张翌: 《编译原理和技术》语法制导的翻译

67



中国科学技术大学

University of Science and Technology of China

## 多态函数的类型系统

□ 一个含多态函数的语言

$P \rightarrow D; E$

$D \rightarrow D; D / id : Q$

$Q \rightarrow \forall type\_variable. Q$  多态函数

$/ T$

$T \rightarrow T' \rightarrow T$

$/ T \times T$  笛卡儿积类型

$/ unary\_constructor (T)$

$/ basic\_type$

$/ type\_variable$  引入类型变量

$/ (T)$

$E \rightarrow E (E) / E, E / id$

这是一个抽象语言, 忽略了函数定义的函数体

张翌: 《编译原理和技术》语法制导的翻译

68



中国科学技术大学

University of Science and Technology of China

## 多态函数的类型系统

□ 一个含多态函数的语言

$P \rightarrow D; E$

$D \rightarrow D; D / id : Q$

$Q \rightarrow \forall type\_variable. Q$

$/ T$

$T \rightarrow T' \rightarrow T$

$/ T \times T$

$/ unary\_constructor (T)$

$/ basic\_type$

$/ type\_variable$

$/ (T)$

$E \rightarrow E (E) / E, E / id$

一个程序:

deref :  $\forall \alpha. pointer(\alpha) \rightarrow \alpha$ ;

$q : pointer(pointer(integer));$

deref (deref (q))

张翌: 《编译原理和技术》语法制导的翻译

69



中国科学技术大学

University of Science and Technology of China

## 多态函数的类型系统

□ 类型系统中增加的推理规则

■ 环境规则

类型变量 $\alpha$ 加到定型环境中

(Env Var)

$$\frac{\Gamma \vdash \diamond, \alpha \notin dom(\Gamma)}{\Gamma, \alpha \vdash \diamond}$$

■ 语法规则

(Type Var)

$$\frac{\Gamma_1, \alpha, \Gamma_2 \vdash \diamond}{\Gamma_1, \alpha, \Gamma_2 \vdash \alpha}$$

(Type Product)

$$\frac{\Gamma \vdash T_1, \Gamma \vdash T_2}{\Gamma \vdash T_1 \times T_2}$$

张翌: 《编译原理和技术》语法制导的翻译

70



中国科学技术大学

University of Science and Technology of China

## 多态函数的类型系统

□ 类型系统中增加的推理规则

■ 语法规则

(Type Parenthesis)

$$\frac{\Gamma \vdash T}{\Gamma \vdash (T)}$$

(Type Forall)

$$\frac{\Gamma, \alpha \vdash T}{\Gamma \vdash \forall \alpha. T}$$

(Type Fresh) 类型变量换名 ( $\alpha_i$  不在 $\Gamma$ 中)

$$\frac{\Gamma \vdash \forall \alpha. T, \Gamma, \alpha_i \vdash \diamond}{\Gamma, \alpha_i \vdash [\alpha_i / \alpha] T}$$

张翌: 《编译原理和技术》语法制导的翻译

71



中国科学技术大学

University of Science and Technology of China

## 多态函数的类型系统

■ 定型规则

(Exp Pair)

$$\frac{\Gamma \vdash E_1 : T_1, \Gamma \vdash E_2 : T_2}{\Gamma \vdash E_1, E_2 : T_1 \times T_2}$$

(Exp FunCall)

$$\frac{\Gamma \vdash E_1 : T_1 \rightarrow T_2, \Gamma \vdash E_2 : T_3}{\Gamma \vdash E_1 (E_2) : S(T_2)}$$

(其中 $S$ 是 $T_1$ 和 $T_3$ 的最一般的合一代换)


代换: 类型表达式中的类型变量用其所代表的类型表达式去替换  $subst(t:type\_exp, Sv:type\_var \rightarrow type\_exp):type\_exp$

实例: 把 $subst$ 函数用于 $t$ 后所得的类型表达式是 $t$ 的一个实例, 用 $S(t)$ 表示

张翌: 《编译原理和技术》语法制导的翻译

72

12



中国科学技术大学

University of Science and Technology of China

代换和实例

```

function subst (t : type_exp, Sv: type_var → type_exp) :
    type_exp;
begin
    if t 是基本类型 then return t
    else if t 是类型变量 then return Sv(t)
    else if t 是  $t_1 \rightarrow t_2$  then return
        subst( $t_1$ , Sv) → subst( $t_2$ , Sv)
end

```

例子 ( $s < t$  表示  $s$  是  $t$  的实例,  $\alpha, \beta$  是类型变量)

$pointer(integer) < pointer(\alpha)$     $pointer(real) < pointer(\alpha)$

$integer \rightarrow integer < \alpha \rightarrow \alpha$     $pointer(\alpha) < \beta$

$\alpha < \beta$

张昱:《编译原理和技术》语法制导的翻译

73



中国科学技术大学

University of Science and Technology of China

不合法的实例

例 下面左边的类型表达式不是右边的实例

integer

real

代换不能用于基本类型

$integer \rightarrow real$

$\alpha \rightarrow \alpha$

$\alpha$  的代换不一致

$integer \rightarrow \alpha$

$\alpha \rightarrow \alpha$

$\alpha$  的所有出现都应该代换

张昱:《编译原理和技术》语法制导的翻译

74



中国科学技术大学

University of Science and Technology of China

合一

□ 合一(unify)

- 如果存在某个代换  $Sv$  使得  $S(t_1) = S(t_2)$ , 那么这两个表达式  $t_1$  和  $t_2$  能够合一

□ 最一般的合一代换(the most general unifier)  $S$

- $S(t_1) = S(t_2)$ ;
- 对任何其它满足  $S'(t_1) = S'(t_2)$  的代换  $Sv'$ , 代换  $S'(t_1)$  是  $S(t_1)$  的实例

张昱:《编译原理和技术》语法制导的翻译

75



中国科学技术大学

University of Science and Technology of China

多态函数的类型检查

□ 多态函数和普通函数在类型检查上的区别

- 同一多态函数的不同出现不要求变元/参数有相同类型
- 必须把类型相同的概念推广到类型合一
- 要记录类型表达式合一的结果

```


      apply:  $\alpha_o$ 
     /      \
derefo: $pointer(\alpha_o) \rightarrow \alpha_o$   apply:  $\alpha_i$ 
                             /      \
derefi: $pointer(\alpha_i) \rightarrow \alpha_i$   q:  $pointer(pointer(integer))$ 

```

deref(deref( $q$ ))的带标记的语法树

张昱:《编译原理和技术》语法制导的翻译

76



中国科学技术大学

University of Science and Technology of China

检查多态函数的翻译方案


```

E → E1 (E2)
    { p = mkleaf(newtypevar); // 返回类型
      unify (E1.type, mknode ( '→', E2.type, p ));
      E.type = p }
E → E1, E2
    { E.type = mknode ( '×', E1.type, E2.type ) }
E → id
    { E.type = fresh (lookup(id.entry)) } // 类型变量

```

张昱:《编译原理和技术》语法制导的翻译

77



中国科学技术大学

University of Science and Technology of China

例: 多态函数的检查

```

      apply:  $\alpha_o$ 
     /      \
derefo: $pointer(\alpha_o) \rightarrow \alpha_o$   apply:  $\alpha_i$ 
                             /      \
derefi: $pointer(\alpha_i) \rightarrow \alpha_i$   q:  $pointer(pointer(integer))$ 

```

表达式 : 类型	代换
$q : pointer(pointer(integer))$	
$deref_i : pointer(\alpha_i) \rightarrow \alpha_i$	
$deref_i(q) : pointer(integer)$	$\alpha_i = pointer(integer)$
$deref_o : pointer(\alpha_o) \rightarrow \alpha_o$	
$deref_o(deref_i(q)) : integer$	$\alpha_o = integer$

张昱:《编译原理和技术》语法制导的翻译

78

中国科学技术大学

University of Science and Technology of China

# 求表长的函数的检查

```

length :  $\beta$ ;      lptr :  $\gamma$ ;
if :  $\forall \alpha. \text{boolean} \times \alpha \times \alpha \rightarrow \alpha$ ;
null :  $\forall \alpha. \text{list}(\alpha) \rightarrow \text{boolean}$ ;
tl :  $\forall \alpha. \text{list}(\alpha) \rightarrow \text{list}(\alpha)$ ;
0 : integer; 1 : integer;
+ : integer  $\times$  integer  $\rightarrow$  integer;
match :  $\forall \alpha. \alpha \times \alpha \rightarrow \alpha$ ;

match (
    -- 表达式, 匹配length函数的
    length(lptr),      -- 函数头部和函数体的类型
    if (null(lptr), 0, length(tl(lptr)) + 1)
)

```

fun length(lptr) =

if null(lptr) then 0

else length(tl(lptr)) + 1;

类型声明部分

张翌:《编译原理和技术》语法制导的翻译

79

中国科学技术大学

University of Science and Technology of China

# 求表长的函数的检查

行	定型断言	代换	规则
(1)	$\text{lptr} : \gamma$		(Exp Id)
(2)	$\text{length} : \beta$		(Exp Id)
(3)	$\text{length}(\text{lptr}) : \delta$	$\beta = \gamma \rightarrow \delta$	(Exp FunCall)
(4)	$\text{lptr} : \gamma$		从(1)可得
(5)	$\text{null} : \text{list}(\alpha_n) \rightarrow \text{boolean}$		(Exp Id)和 (Type Fresh)
(6)	$\text{null}(\text{lptr}) : \text{boolean}$	$\gamma = \text{list}(\alpha_n)$	(Exp FunCall)
(7)	$0 : \text{integer}$		(Exp Num)
(8)	$\text{lptr} : \text{list}(\alpha_n)$		从(1)可得

张翌:《编译原理和技术》语法制导的翻译

80

中国科学技术大学

University of Science and Technology of China

# 求表长的函数的检查

行	定型断言	代换	规则
(9)	$\text{tl} : \text{list}(\alpha_i) \rightarrow \text{list}(\alpha_i)$		(Exp Id)和 (Type Fresh)
(10)	$\text{tl}(\text{lptr}) : \text{list}(\alpha_n)$	$\alpha_i = \alpha_n$	(Exp FunCall)
(11)	$\text{length} : \text{list}(\alpha_n) \rightarrow \delta$		从(2)可得
(12)	$\text{length}(\text{tl}(\text{lptr})) : \delta$		(Exp FunCall)
(13)	$1 : \text{integer}$		(Exp Num)
(14)	$+ : \text{integer} \times \text{integer} \rightarrow \text{integer}$		(Exp Id)

张翌:《编译原理和技术》语法制导的翻译

81

中国科学技术大学

University of Science and Technology of China

# 求表长的函数的检查

行	定型断言	代换	规则
(15)	$\text{length}(\text{tl}(\text{lptr})) + 1 : \text{integer}$	$\delta = \text{integer}$	(Exp FunCall)
(16)	$\text{if} : \text{boolean} \times \alpha_i \times \alpha_i \rightarrow \alpha_i$		(Exp Id)和 (Type Fresh)
(17)	$\text{if}(\dots) : \text{integer}$	$\alpha_i = \text{integer}$	(Exp FunCall)
(18)	$\text{match} : \alpha_m \times \alpha_m \rightarrow \alpha_m$		(Exp Id)和 (Type Fresh)
(19)	$\text{match}(\dots) : \text{integer}$	$\alpha_m = \text{integer}$	(Exp FunCall)

length函数的类型是  $\forall \alpha. \text{list}(\alpha) \rightarrow \text{integer}$

张翌:《编译原理和技术》语法制导的翻译

82

中国科学技术大学

University of Science and Technology of China

# 5.6 函数和算符重载

Ad-hoc多态

可能的类型集合及其缩小

附加:子类型关系引起的协变和逆变

中国科学技术大学

University of Science and Technology of China

# 重载

重载符号

有多个含义,但在每个引用点的含义都是唯一的

例如:

加法算符+可用于不同类型,“+”是多个函数的名字,而不是一个多态函数的名字

在Ada中,()是重载的,A(I)有不同含义

重载的消除

在重载符号的引用点,其含义能确定到唯一

张翌:《编译原理和技术》语法制导的翻译

84

中国科学技术大学

University of Science and Technology of China

表达式的可能类型集合

例 Ada语言

声明:

function “\*” (i, j : integer ) return complex;

function “\*” (x, y : complex ) return complex;

使得算符\*重载，可能的类型包括:

integer × integer → integer --这是预定义的类型

integer × integer → complex (3 \* 5) \* z (z:complex)

complex × complex → complex

E: {i}

3: {i}

E: {i, c}

\*

{i × i → i, i × i → c, c × c → c }

E: {i}

5: {i}

85

中国科学技术大学

University of Science and Technology of China

重载函数的应用

□ 缩小可能类型的集合

■  $E' \rightarrow E$   $E. unique = \text{if } E. types == \{ t \}$   
then  $t$  else  $type\_error$

■  $E \rightarrow id$   $E. types = lookup(id. entry)$

■  $E \rightarrow E_1(E_2)$   $E. types = \{ s' \mid E_2. types \text{ 中存在一个 } s, \text{ 使得 } s \rightarrow s' \text{ 属于 } E_1. types \}$

$t = E. unique$

$S = \{ s \mid s \in E_2. types \text{ and } s \rightarrow t \in E_1. types \}$

$E_2. unique = \text{if } S == \{ s \} \text{ then } s \text{ else } type\_error$

$E_1. unique = \text{if } S == \{ s \} \text{ then } s \rightarrow t \text{ else } type\_error$

张翌:《编译原理和技术》语法制导的翻译

86

中国科学技术大学

University of Science and Technology of China

附加：子类型 - 协变和逆变

□ 子类型关系 <

■ 类型上的偏序关系  $\tau$

■ 满足包含原理: 如果s是t的子类型，则需要类型为t的值时，都可以将类型为s的值提供给它

□ 协变 (covariant)  $t < t'$ , 则  $c(t) < c(t')$

■ 函数类型在值域上是协变的

假设  $e: \sigma \rightarrow \tau$ ,  $e1: \sigma$ , 则  $e(e1): \tau$  如果  $\tau < \tau'$ , 则  $e(e1): \tau'$ .

□ 逆变 (contravariant)  $t < t'$ , 则  $c(t') < c(t)$

■ 函数类型在定义域上是逆变的

假设  $e: \sigma \rightarrow \tau$ ,  $e1: \sigma'$ , 如果  $\sigma' < \sigma$ , 则  $e(e1): \tau$ .

张翌:《编译原理和技术》语法制导的翻译

87

中国科学技术大学

University of Science and Technology of China

例题 5

编译器和连接装配器未能发现下面的调用错误

long gcd (p, q) long p, q; /\*这是参数声明的传统形式\*/

/\*参数声明的现代形式是long gcd ( long p, long q) { \*/

if (p%q == 0)

return q;

else

return gcd (q, p%q);

}

main() {

printf(“%ld,%ld\n”, gcd(5), gcd(5,10,20));

}

张翌:《编译原理和技术》语法制导的翻译

88

15