数据类型

- 数据类型
  - 值的集合+操作的集合
  - 内部类型，用户自定义类型，抽象类型
    - 内部类型是基本位串的抽象
    - 用户自定义类型是内部类型（和其他用户自定义类型）的抽象
    - 抽象类型是用户自定义的一种（但满足俩个条件）
  - 六种数据类型聚合方式

笛卡尔积	记录、结构	递　　归	二叉树
有限映射	数组	判 定 或	联合、变体记录
序　　列	字符串、顺序文件	幂　　集	集合

数据类型

- 类型检查
  - 数据对象的**类型**和使用的**操作**是否**匹配**
  - 静态、动态
  - 延伸思考：语义翻译中的类型检查
- 类型转换
  - 某种类型的值转换为另一种类型的值
  - 延伸思考：语义翻译中的类型转换
- 类型等价
  - 名字等价：两个变量的类型名相同
  - 结构等价：两个变量的类型具有相同的结构
  - 用用户定义类型的定义来代替用户定义名，重复这一过程，直到没有用户定义类型名为止。
- 实现模型：数据用描述符和数据对象来表示

控制结构

- 语句级控制结构
  - 顺序
  - 重复(循环)
  - Goto选择(分支)
  - 存在的问题
- 控制结构的选择
- 顺序，选择(分支)，重复(循环)足够实现各种算法
- 增加一些控制结构的表达方式可提高提高程序的可写性和可读性

程序语言的设计

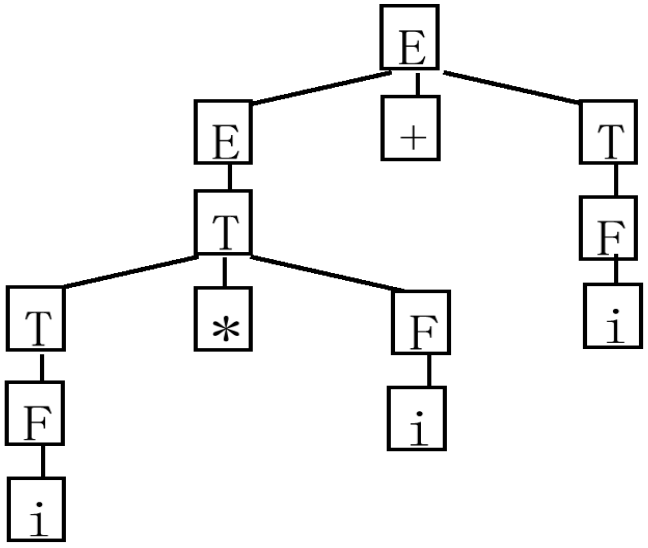
- 语言＝语法(规则)＋语义(规则)
- 字母表、符号、字汇表
- 词法规则、语法规则
- 定义语言：
  - 生成（文法）
  - 识别（语法图）
- 抽象机GAM
- 文法
  - 文法是描述语言的语法结构的形式规则，必须准确，易于理解，且描述能力强。

程序语言的设计

- 文法的分类
  - 0、1、2、3型文法
- 推导与规约
  - 最左推导与最右推导
- 文法与语言
  - 可能句子的集合
  - 有限规则描述无穷语言
- 句型与句子
  - 是否仅有终结符
- 短语、直接短语、句柄
  - $S \Rightarrow^* \alpha A \delta$ 且 $A \Rightarrow^+ \beta$

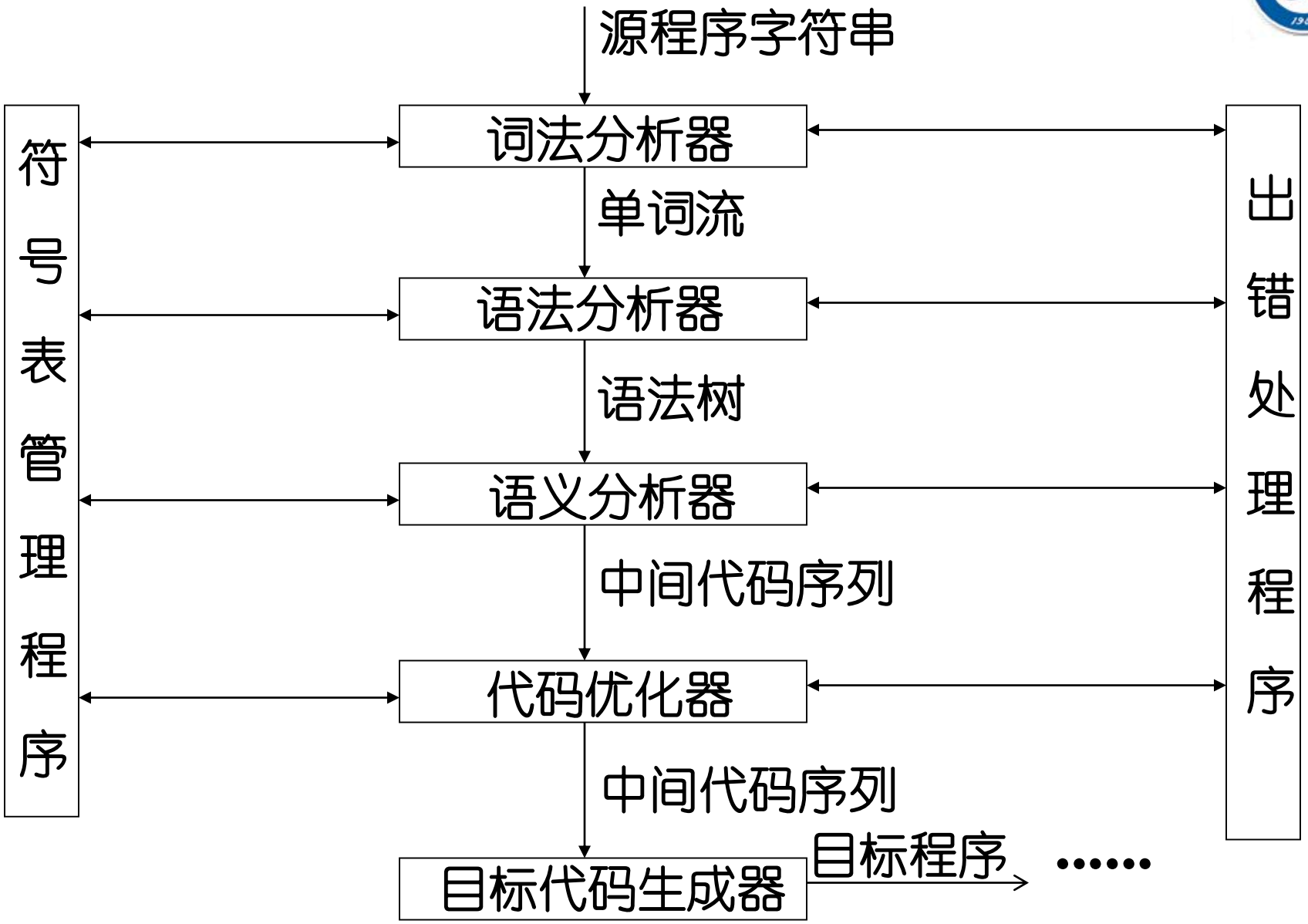
程序语言的设计

- 语法树-以图的方式表示推导过程
- n个内部节点 -- n棵子树
- n棵子树 -- n个短语
  - 每颗子树的叶结点从左至右排列组成一个短语
- m棵直接子树--只有父子两代--m个直接短语
- 最左直接子树——句柄



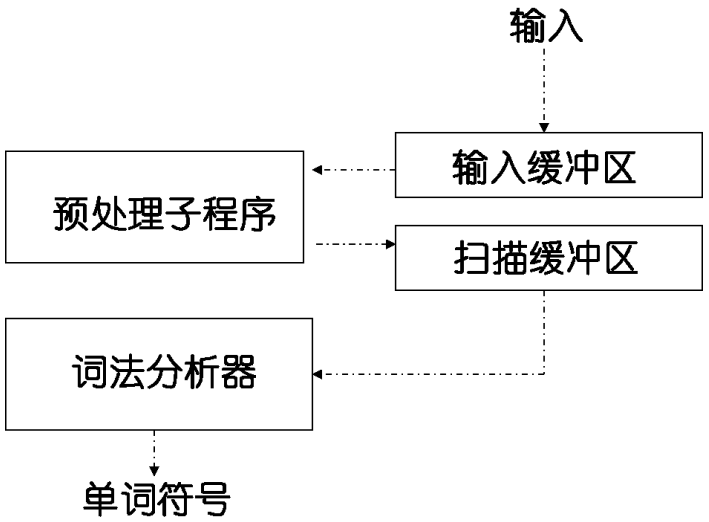
编译概述

- 编译的一些基本概念
  - 翻译、编译、宿主语言、宿主机、自驻留、交叉编译、自编译
  - 编译执行、解释执行
- 编译步骤
  - 词法分析：输入字符串，根据词法规则识别出单词符号
  - 语法分析：根据语法规则，将单词符号构成各类语法单位，并进行语法检查
  - 语义分析： 根据语义规则，进行初步编译
  - 优化：对中间代码进行等价变换，以使代码更有效
  - 目标代码生成：生成机器语言程序或汇编语言程序
  - 符号表管理和 出错处理


源程序字符串
词法分析器
单词流
语法分析器
语法树
语义分析器
中间代码序列
代码优化器
中间代码序列
目标代码生成器
目标程序
……
符号表管理程序
出错处理程序

词法分析

- 词法分析：读取源程序、识别单词
- 单词的输出形式：（单词类别，单词的属性）
- 单词类别的划分：基本字、运算符、界符；标识符；常数；
- 超前搜索、双缓冲区
- 状态转换图
  - 有限方向图
  - 通过状态转换图识别串
- 符号表相关概念



自上而下的语法分析

- 1. 语法分析功能
- 2. 两大类
  - 自上而下(自顶而下)、自下而上(自底而上)
- 3. 回溯分析
  - 公共左因子、左递归、空产生式
- 4. 解决回溯
  - 要么只有一个侯选式可用、要么没有侯选式可用
- 左递归的消除、递归下降分析器
- 预测分析法：
  - 栈顶符号与输入符号的匹配：X=a=#；X=a≠#；X≠a

第七章 自上而下的语法分析

- FIRST集：由α的所有可能推导的开头终结符及可能的ε组成的集合
- Follow集：考察A在产生式右端的出现情况，哪些终结符号可以跟随在A后面

```
S -> AB
A -> Ab | ε
B -> a | ε
```

	FIRST	FOLLOW
S	a b ε	#
A	b ε	a b #
B	a ε	#

自上而下的语法分析

- 预测分析表的构造

  对每个产生式A→α

  - 1.对∀a∈FIRST(α),将A→α记入M[A,a]中;
  - 2. 若 ε∈FIRST(α), 对 ∀ b∈FOLLOW(A), 将A→α记入M[A,b]中;
  - 3.凡未被定义的M[A,a]项中标以出错标志。
- LL(1)文法定义
- 非LL(1)文法的判断:
  - 方法一: 构造分析表
  - 方法二: FIRST集、FOLLOW集的交集是否为空

自下而上的语法分析

- 核心概念：短语、直接短语、句柄、素短语、最左素短语
- 自下而上分析关键问题
  - 移进与归约的冲突
  - 归约与归约的冲突
- 不同的语法分析方法-可归约串不同的定义
  - 算符优先分析法：最左素短语，结构规约
  - LR分析法：句柄
- 短语、直接短语、句柄、素短语
- 算符优先分析法
  - 算符文法定义：
    - P→ε 或 P→...QR...
    - 相邻的2个终结符之间的优先关系唯一
  - 结构规约：即长度一致，对应的终结符一致

自下而上的语法分析

- 算法优先分析法的实现：
  - 当其栈顶形成最左素短语时，就进行归约。
- 三种关系：
  - a=b，直观可知其相等关系；
  - a<b，FIRSTVT：所有可能推导的第一个终结符
  - a>b，LASTVT：所有可能推导的最后一个终结符
- FIRSTVT：
  - 若P→a…或P→Qa…，则a∈FIRSTVT(P)；
  - 若P→Q…，则FIRSTVT(Q)⊆FIRSTVT(P)；
- LASTVT：
  - 若P→…a或P→…aQ，则a∈LASTVT(P)
  - 若P→…Q，则LASTVT(Q)⊆LASTVT(P)
- 优先关系表的构造
  - 考察每一产生式每两个符号之间的关系
  - P→…aQ…的情况，横填，$\forall b \in FIRSTVT(X_{i+1})$ $X_i < b$；
  - P→…Qb…的情况，竖填，$\forall a \in LASTVT(X_i)$ $a > X_{i+1}$
  - P→…ab…及P→…aQb…的情况，关系“=”

自下而上的语法分析

- LR分析法
- 栈,控制程序,分析表,输入串
- LR分析表的使用：action和goto两个子表
  - action：在状态s下，当前输入符号为a时：s,r,acc,error
  - goto：规约后，上托|β|个状态出栈，在状态s下，针对归约后的符号A的入栈状态。
- LR(0)项目集规范族
  - 活前缀：目的是找句柄，所以不含句柄之后任何符号。活前缀与句柄之间的三种关系。
- 项目的概念、归约项目:形如A→α•、移进项目:形如A→α•aβ $a\in V_T$、待约项目:形如A→α•Bβ $B\in V_N$、接收项目:形如S→α•
- 有效项目、有效项目集
- 若A→α•Bβ对活前缀δα有效，则B →•η对活前缀δα也有效。
- 构造：闭包函数closure(I)、转换函数Go(I,X)
  - （1）求项目S′→●S为初态闭包（有A→α●Bβ，则也有B→●γ项目）
  - （2）应用转换函数GO(I，X)=CLOSURE(J)，求出新状态J的项目集。
  - （3）重复b)

LR(0)分析表的构造

1. 拓展文法
2. 确定所有项目
3. 构造LR(0)项目规范族

$A\rightarrow\alpha\bullet B\beta\in closure(I)$
且$B\rightarrow\eta$为文法G的一个产生式

则$B\rightarrow\bullet\eta\in closure(I)$

$I: A\rightarrow\alpha\bullet X\beta$

闭包函数closure(I)

转换函数Go(I,X)

$Go(I,X)=Closure(A\rightarrow\alpha X\bullet\beta)$

$C=\{I_0\}=\{Closure(S'\rightarrow\bullet S)\}$

对C中每一个项目集I和每一个文法符合X
go(I,X)不空 且 go(I,X)$\notin$C 把go(I,X)加入C中

自下而上的语法分析

SLR（1）分析表的构造

- 设一个LR(0)项目集规范族有一个项目集I：
  $I=\{A_1\rightarrow\alpha\bullet a_1\beta_1, A_2\rightarrow\alpha\bullet a_2\beta_2, \cdots, A_m\rightarrow\alpha\bullet a_m\beta_m, B_1\rightarrow\alpha\bullet, B_2\rightarrow\alpha\bullet, \cdots, B_n\rightarrow\alpha\bullet\}$
- 若满足
  ① $\{a_1, a_2, \ldots, a_m\}\cap FOLLOW(B_i)=\varnothing, i=1, 2, \ldots, n$
  ② $FOLLOW(B_i)\cap FOLLOW(B_j)=\varnothing, i, j=1, 2\ldots, n$, 且$i\neq j$,
- 则可以通过判断当前的输入符号a属于哪一个集合来解决冲突。

- 按如下策略构造SLR(1)分析表：
  ① 若$a=a_i (i=1, 2, \ldots, n)$，则移进$a_i$；
  ② $a\in FOLLOW(B_i) (i=1, 2\ldots, n)$，则用$B_i\rightarrow\alpha$归约；
  即：若$A\rightarrow\alpha\bullet\in Ii$，$A\rightarrow\alpha$为第j个产生式，则
  $\forall b\in FOLLOW(A)$，action[i,b]=rj;
  ① 此外，按“出错”处理；

语义分析

1. 语义分析

(1) 语义检查
- 一致性检查
- 越界检查

(2) 语义处理
- 说明语句 —— 信息登记
- 可执行语句 —— 四元式

2. 语法制导翻译

在语法分析的过程中，由各个产生式对应的语义子程序对源代码进行翻译（生成中间代码）的方法称为语法制导翻译。

3. 说明语句的翻译

不产生可执行指令，仅负责填表，将被说明对象的类型及相对存储位置记入各自的符号表中。

语义分析

D→D;D | i:T

T→real | integer | array[num] of T_1 | ↑T_1

1. 类型　　T.Type

2. 相对存储位置　　T.Width

全部变量Offset：相对位移量

1. 赋初值　S → MD { Offset=0 }
　　M → ε
2. 增值　Offset=Offset+T.Width

3. 记入符号表　enter(i.name,T.type,offset)

语义分析

简单赋值语句的翻译

- 语义变量及函数：
  - E.Place， X.a， ip
  - Newtemp， entry(i)， gen(....)
- 翻译方案-语义子程序
  - 对应产生式
  - 临时变量的使用
- 类型检查
  - 增加类型属性：E.type
  - 类型转换指令：(itr,x,_,t)

语义分析

布尔表达式的翻译

- 控制语句中，布尔表达式一定为早期句柄
- B.T 与 B.F
- 两个地址，两条中间代码

B→i

```
{
    B.T := ip;
    emit (if i goto 0);
    B.F := ip;
    emit (goto 0)          }
```

B→i_1 rop i_2

```
{
    B.T:=ip;
    B.F:=ip+1 ;
    emit (jrop,entry(i1), entry(i2),0);
    emit(j,-,-,0)  }
```

B.T的记录也可以省略

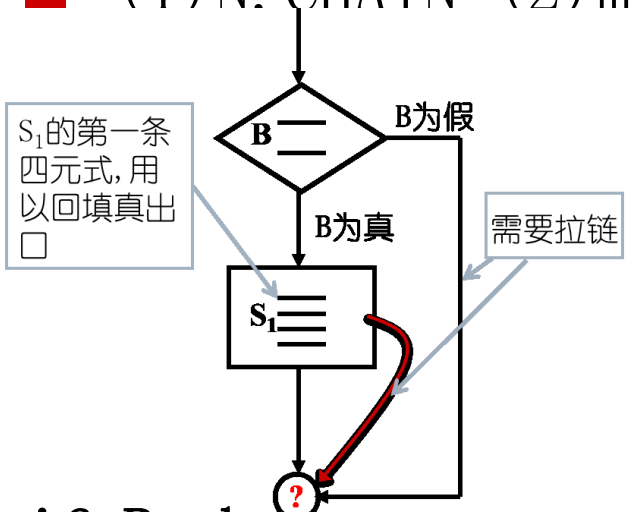
B→i

```
{    emit(jnz, entry(i), -, ip+2);
     B.F:=ip;
     emit(j,-,-,0)    }
```

语义分析

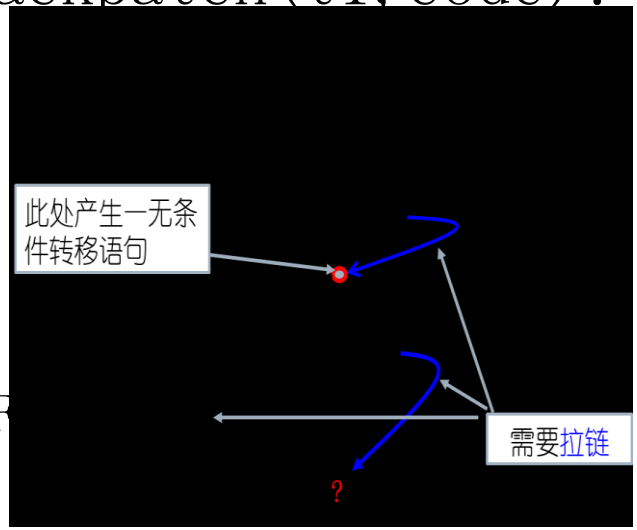
If语句的翻译

(1)N.CHAIN (2)merge(t1,t2)(3)backpatch(t1,code);



S_1的第一条四元式,用以回填真出口

M→if B then
{backpatch(B.T, ip);
M.CHAIN:=B.F }
S→M S_1
{S.CHAIN:=
merge(M.CHAIN, S_1.CHAIN) }



M→if B then
{backpatch(B.
M.CHAIN:=B.F
N→M S_1 else
{q:=ip;
gen(j,-,-,0);
backpatch(M.CHAIN,ip);
N.CHAIN:=merge(S_1.CHAIN,q) }
S→N S_2
{S.CHAIN:=
merge(N.CHAIN, S_1.CHAIN)}

代码优化和目标代码生成

- 基本块的划分
  - 出口语句
  - 入口语句
- 基本块内的优化，局部优化
  - 合并已知量
  - 删除公共子表达式
  - 删除无用赋值
  - 删除死代码
- 全局优化
  - 只讨论循环的优化
  - 循环的定义
- 如何查找循环
  - 必经节点d
  - 回边 n->d
  - 以回边寻找循环 LOOP={n, d}∪M
    - M是流图中有通路到达n而该通路不经过d的结点集合，
- 循环的优化
  - 1. 代码外提
  - 2. 强度削弱
  - 3. 删除归纳变量
- 目标代码生成
  - 循环中的寄存器的分配
  - 节省代价计算
  - $\sum_{B \in L}[USE(x,B) + 2 * LIVE(x,B)]$