编译技术复习提纲

**第一章 引论**

1. **编译器和解释器之间的区别：编译器将高级语言程序翻译成汇编或机器语言程序，完整翻译后执行、执行速度快。解释程序逐句处理源程序，交互性好、错误诊断效果好。**
2. **编译器的模块结构：编译器有8个功能模块，包括4个分析模块、2个综合模块和2个辅助模块。分析模块有词法分析、语法分析、语义分析、中间代码生成；综合模块有代码优化、目标代码生成；辅助模块有符号表管理、出错处理。分析和综合模块构成了编译的6个有序步骤，2个辅助模块贯穿始终。**
3. **编译器各阶段/步骤的主要功能：（1）词法分析器从左到右读入源程序的字符并组成词素，生成词法单元序列并给语法分析使用、过滤注释和空白符、定位词法错误、查填符号表——进行标识符登记。（2）语法分析器按语法规则将词组成各类语法成分（如表达式、因子、项、语句、子程）、指出语法错误、指导翻译。（3）语义分析器分析语法单位的语义、获取标识符的属性（类型、作用域等）、进行语义检查（运算的合法性、取值范围等）、子程序和变量的静态地址绑定。（4）中间代码生成：生成与机器无关的简单规范的中间代码，如逆波兰表达式、三地址码、语法分析树等，中间代码易于优化和转换成目标机器代码。（5）代码优化分为与机器无关的优化和与机器有关的优化。其中与机器无关的优化包括局部优化和循环优化等，局部优化包括常量合并（常数运算在编译期间完成，如8+9\*4）、公共子表达式的提取；循环优化包括强度削减（用较快的操作代替较慢的操作）、代码外提（将循环不变计算移出循环）。与机器有关的优化包括基于寄存器的优化（将常用量放入寄存器，以减少访问内存的次数）、基于体系结构的优化（MIMD、SIMD、SPMD、向量机、流水机）、基于存储策略的优化（根据算法访存的要求安排：Cache、并行存储体系——减少访问冲突）、基于任务划分的优化（按运行的算法及体系结构划分子任务，即MPMD）。（6）目标代码生成将中间代码转换成目标机上的机器指令代码或汇编代码，包括确定源语言的各种语法成分的目标代码结构（机器指令组/汇编语句组）、制定从中间代码到目标代码的翻译策略或算法、确定目标代码的形式（具有绝对地址的机器指令、汇编语言形式的目标程序、模块结构的机器指令--需要链接程序）。**

**第二章 语言与文法**

1. **理解字母/字符、字母表∑、符号串/字/单词、符号串的连接和方幂、字母表的乘积和n次幂、字母表的正闭包和克林闭包、字母表上的语言和句子、推导和规约、句型和句子、最左推导（最右规约）和最右推导（最左规约）、文法的分类**
2. **深刻理解文法G的定义，能够根据要求规法描述文法，具体要求参考课堂例题和作业：文法G为一个四元组:Ｇ = (ＶT，ＶN，Ｐ，Ｓ)，ＶT：终结符集；ＶN：非终结符集（ＶT∩ＶN=Φ），语法成分—代表某个语言的各种子结构；Ｓ：开始符号，S∈ＶN代表文法所定义的语言，至少在产生式左侧出现一次；Ｐ：产生式集合，产生式定义各个语法成分的结构，即描述各种规则。**
3. **深刻理解最左推导（最有规约）和最右推导（最左规约）的定义：最左推导：每次推导都施加在句型的最左边的语法变量上；最右推导：每次推导都施加在句型的最右边的语法变量上（最左规约称规范规约）**
4. **上下文无关文法的条件：（1）满足文法G的定义；（2）产生式集合中所有｜α｜≤｜β｜， 除 Ｓ→ε外；（3）α∈ＶN，β∈(ＶN∪ＶT)\*。**

**第三章 词法分析**

1. **理解词法单元、模式和词素的概念：词法单元是表示某种词法单位的抽象符号（如常量、关键字、标识符），词法单元由一个词法单元名和一个可选的属性值组成；模式是描述一个词法单元的词素可能具有的形式（即构成规则）；词素是源程序中的一个字符序列，它和某个词法单元的模式匹配。每个词法单元对应于一个词素。**
2. **正则表达式RE是描述词素模式的重要方法。深刻理解正则表达式的定义，能够熟练写出∑有2-3个字母、规则较简单（包含、不包含或以特定子串结尾）的词素的正则表达式，具体参见课堂例题和作业。正则表达式RE的定义：设∑是一个字母表，⑴ Φ是∑上的RE，L(Φ)=Φ；⑵ ε是∑上的RE，L(ε)={ε}；⑶ 对于a∈∑，a是RE，L(a)={a}；⑷ 如果r和s是RE，L(r)=R,L(s)=S，则： r与s的“和”(r|s)是RE，L(r|s)=R∪S；r与s的“乘积”(rs)是RE，L(rs)=RS；r的克林闭包(r\*)是RE，L(r\*)=R\*；⑸ 只有满足⑴、⑵、⑶、⑷的才是RE。**
3. **状态转换图通过描绘系统的状态及引起系统状态转换的事件，来表示系统的行为。掌握状态转换图的组成要素、NFA、DFA的组成和应用，理解不确定的有穷自动机NFA和确定的有穷自动机DFA的主要区别，能够熟练画出∑有2-3个字母、规则较简单（包含、不包含或以特定子串结尾）词素的NFA和DFA，具体参见课堂例题和作业。**
4. **状态转换图的组成要素：（1）状态表示在识别词素中可能出现的情况（状态看作部分处理的总结）。开始状态（初始状态）用start边表示；某些状态为接受状态或最终状态，表明已经找到词素，加上\*的接受状态表示最后读入的符号不在词素中。（2）边从一个状态指向另一个状态；边的标号是一个或者多个符号。如果当前符号为s，下一个输入符号为a，就沿着从s离开标号为a的边到达下一个状态。**
5. **应用子集构造法完成NFA到DFA的等价转换：**

|  |  |
| --- | --- |
| **操作** | **描述** |
| **ε-Closure(s)** | **s∈S，从状态s只经过ε-转换就可以到达的NFA状态集** |
| **ε-Closure(T)** | **T为S子集，从T中的状态s只经过ε-转换就可以到达的NFA状态集** |
| **move(T,a)** | **从状态集T中的状态s只经过输入符号a-转换就可以到达的NFA状态集** |

1. **NFA和DFA的组成：NFA由以下几部分组成：（1）一个有穷的状态集合S；（2）一个输入符号集合Σ；（3）转换函数：对于每个状态和ΣU{ε}中的符号，给出相应的后继状态集合；（4）一个状态s0被指定为开始状态/初始状态（有些定义中可以有多个开始状态）；（5）S的一个子集被指定为接受状态。一个NFA被称为DFA，如果没有ε之上的转换动作；对于每个状态s和每个输入符号a，有且仅有一条标号为a的离开s的边。**

**第四章 语法分析**

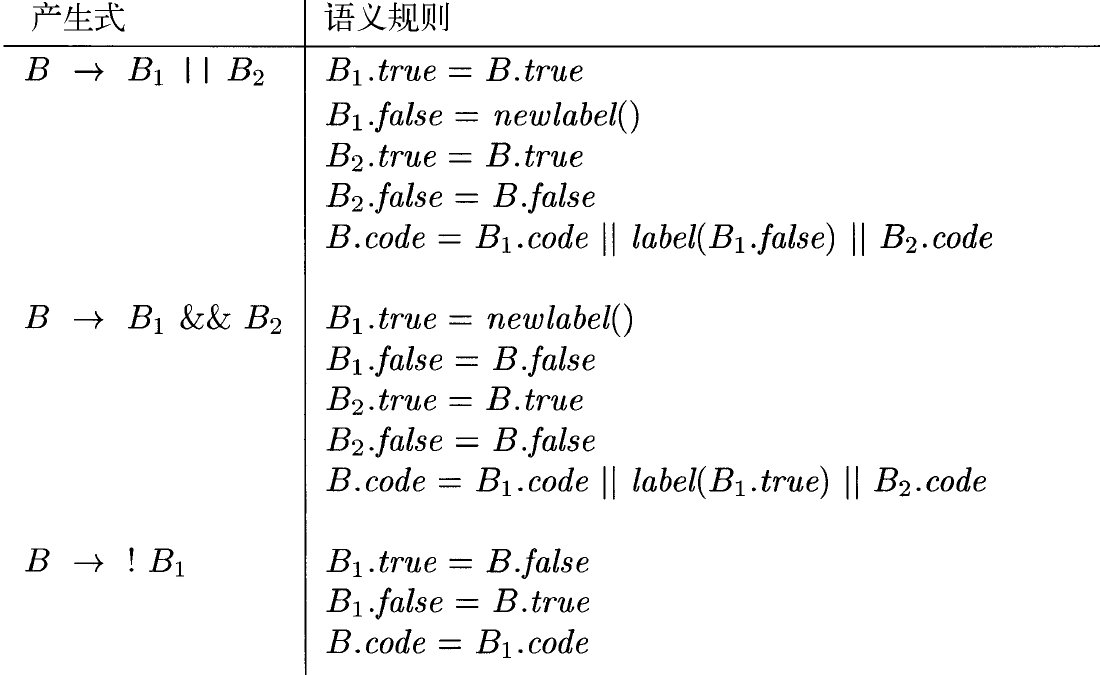
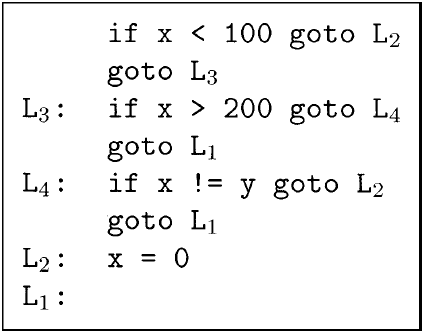
1. **熟知LL文法和 LR 文法的常用分析方法：LL文法常用递归下降分析法（递归子程序法）或预测分析法；LR文法常用算符优先分析法或LR分析法。另：LL文法多用于编译的手工实现；LR文法多用于编译的自动生成。**
2. **深刻理解回溯、左递归和二义性，并能举例说明；掌握它们的消除方法：（1）消除回溯：提取左因子，引入语法变量（2）消除左递归基本方法：将A→Aα|β替换为A→βA′和 A′→αA′｜ε。（3）消除二义性：引入语法变量，使文法含有更多的信息（如IF-THEN-ELSE引入语法变量并提取公因子）。**
3. **深刻理解FIRST集、FOLLOW集。FIRST集：对于α∈(ＶT∪ＶN)\*, 定义:α的首符号集FIRST(α)={a|α=>\*a…，a∈ＶT \*}。α的FIRST集的意义:可从α推导得到的首符号(是终结符)集合。FOLLOW集：对于A∈ＶN，FOLLOW(A)={a|S=>\*…Aa…，a∈ＶT }。**
4. **能熟练计算FIRST集、FOLLOW集，熟练构造预测分析表。**
5. **FIRST集求法：（1）终结符：FIRST集由它本身一个元素构成；（2）非终结符：对X∈ＶN，则FIRST(X)初值={a|X→a…∈P}，X→ε不属于P或={a|X→a…∈P}∪{ε}，X→ε∈P；对X∈ＶN，还要重复如下过程直到所有FIRST集不变：若 X→Y1…Yn∈P，且Y1...Yi-1=>\*ε，则对k=1到i-1:FIRST(X)= FIRST(X)∪(FIRST(Yk)-{ε}) ；若 Y1...Yn=>\*ε，则 FIRST(X)=FIRST(X)∪{ε}。（3）句型α的FIRST集：令α= X1…Xm，初始化FIRST(α)={}；然后将X1…Xm当成某产生式右部借鉴（2）方法求出FIRST(α)。**
6. **FOLLOW(A)的求法：（1）若B→αAaβ,a是终结符,则把a放在FOLLOW(A)中；（2）若B→αAXβ,X是非终结符,则把FIRST(X)放入FOLLOW(A)中；（3）若B→αA或B→αAβ,但β=>ε,则把FOLLOW(B)放到FOLLOW(A)中；（4）若A是开始符，则#/$放入FOLLOW(A)。**
7. **预测分析表的构造方法：（1）对于每一产生式 A→α，执行①和②：①对于FIRST(α)中的每一终结符a，将A→α填入 M[A,a]；②如果ε属于FIRST(α)，则对FOLLOW(A)中的每个终结符b，将A→α填入M[A,b]；（2）若ε属于 FIRST(α)，且#/$在FOLLOW(A)中，则将A→α填入M[A,#/$]；（3）将所有无定义的 M[A,a] 标上错误标志（手工可留空白）。**
8. **简单描述非递归预测分析器模型结构（包括输入输出）：输入缓冲区(符号序列)、栈、预测分析控制程序、预测分析表、输出（产生式序列）。**
9. **理解句柄的概念：最右句型（规范规约）中某个产生式体匹配的子串，对它的归约代表了该最右句型的最右推导的最后一步；正式定义：如果S=>\*αAw =>\*αβw；那么紧跟α之后的A→β是最右句型αβw的一个句柄；在一个最右句型中，句柄右边只有终结符号；如果文法是无二义性的，那么每个句型都有且只有一个句柄。**

**第五章 语法制导的翻译**

1. **理解综合属性和继承属性的概念：（1）综合属性：在分析树结点N上的非终结符号A的属性值由N的产生式所关联的语义规则来定义，又称为S-属性定义。综合属性必然通过N的子结点或N本身的属性值来定义。（2）继承属性：在分析树结点N上的非终结符号B的属性值由N的父结点所关联的语义规则来定义，又称为L-属性定义。继承属性依赖于N的父结点、N本身和N的兄弟结点上的属性值。（3）不允许N的继承属性通过N的子结点上的属性来定义，但是允许N的综合属性依赖于N本身的继承属性。（4）终结符号有综合属性（由词法分析获得），但是没有继承属性。**
2. **深刻理解注释语法分析树的概念：注释语法分析树是显示了各个结点的各属性值的语法分析树。对于算术运算相关SDD（可能含继承属性），能够熟练画出给定简单表达式的注释语法分析树。**

**第六章 中间代码生成**

1. **三地址代码及四元式表示，熟练应用（P222 例6.5、P223例6.6）。三地址代码：每条指令右侧最多有一个运算符，一般情况可以写成x=y op z。四元式的结构及顺序 (op, arg1, arg2, result)，一般画成表格形式。**
2. **三地址码中三地址指的哪三个地址？第一操作数、第二操作数和目的操作数的地址。这些地址可能是：源程序中的名字或常量，编译器生成的常量或临时变量。**
3. **在处理一个过程/函数时，它声明的变量应该放到单独的符号表中去；可以为每个记录创建单独的符号表。**
4. **布尔表达式的代码的SDD，理解语义规则并翻译简单逻辑与和逻辑或的短路（P243例6.15）：if (x<100 || x > 200 && x!= y ) x = 0;相应的代码：**



1. **理解回填的基本概念：回填技术解决某些跳转指令应该跳转到哪里的关键问题；生成跳转指令时暂时不指定目标标号，使用列表分别记录这些不完整的指令，等直到正确的目标时再填写目标标号，每个列表中的指令都指向同一个目标。**
2. **布尔表达式的回填翻译：引入两个综合属性truelist和falselist。truelist包含跳转指令的列表，为true时执行这些指令；falselist包含跳转指令的列表，为false时执行这些指令。**