编原往年题总结

rls LindaLydia

梳理了一下三套样题里的考点，大概分析了一下思路，具体算法详见PPT和讲义。靠前多动动手做做题会有帮助的

样题涉及知识点

(2014 2016 2018) 开作用域，作用域栈，符号表

(2016 2018) 嵌套过程声明，动态链 DL，静态链 SL，静态作用域规则，动态作用域规则

(2014 2016 2018) 流图回边，自然循环，活跃变量信息迭代求解（未给方程）

(2016 2018) 到达定值 OUT 求解

(2016) 寄存器相干图着色

(2014 2016 2018) DU链，UD链

(2018) DAG 图，最小寄存器数

(2014 2016 2018) LL(1) 文法，First、Follow、PS 集合，预测分析表

(2014 2016 2018) 翻译模式，继承/综合属性，自下而上翻译、自上而下递归下降翻译

(2014 2018) 句形、短语、直接短语、右句型、句柄

(2014) 悬垂二义性非 LL(1) 文法，强制优先匹配解决二义性

(2018) 解决二义性，指出强制措施

(2014 2016 2018) LR(0) 、SLR(0)、LR(1)文法及自动机，冲突类型，最近匹配解决悬垂二义性，LR(1)和SLR(1) 错误捕获

(2014 2016 2018) 类型检查、S / L翻译模式产生 TAC 语句

(2018) 拉链和代码回填

第二讲 词法分析

PL/0会出现在题干里。PL/0语言是语法分析调用词法分析的。

应该没有考题

1. 自顶向下语法分析
2. First、Follow、PS集合的求解，预测分析表填表
   1. First(各种串) 集合：不断往后看直到能看到终结符或者看到尽头还是ε
   2. Follow(非终结符)集合：就是看非终结符后跟的第一个终结符是什么，不可能有ε，会有#作为字符串的终结标志
   3. PS(产生式)集合：预测集合
   4. 【考前多背公式，考场上无情迭代】
3. 表驱动的LL(1)分析表【注意压栈顺序是从右到左，为配合最左推导】
4. 左递归的消除【背！】
5. 文法二义性解决【结合方式、优先级】
6. 自底向上语法分析
7. 短语（分析树子树叶子序列）、直接短语（一步归约）、句柄（右句型最左直接短语）
   1. LR分析中的分析栈中符号串是活前缀。活前缀在右句型中不超过句柄。
8. LR(0)，SLR(1)，LR(1)，LALR(1)的自动机、分析表
   1. LR(0)的归约项目占分析表的一整行
   2. SLR(1)的归约项目仅占该行Follow集合对应的列
   3. LR(1)的归约项目仅占该行向前搜索符对应的列
   4. LALR(1)的状态和LR(0)一样多，可以由LR(1)合并同芯状态得到
9. 两类冲突的发现
   1. 移进-归约冲突：一个归约项目所在的项目集存在一条终结符转移边时可能出现
      1. 如果转移边上的终结符a不在归约项目左侧非终结符E的Follow集合里，用SLR(1)就能解决
      2. 如果a不在LR(1)该归约项目的的向前搜索符集合里，LR(1)能解决
   2. 归约-归约冲突：两个归约项目出现在同一个项目集里（LR(1)也解决不了）
10. 文法冲突的解决【结合方式、优先级】
11. SLR(1)和LR(1)的错误捕获【一般来说LR(1)错误捕获快，要举例子】
12. 最不可能的报错【题目描述+尝试LR分析推导】
13. 符号表
14. 开作用域、作用域栈、符号表
    1. 注意区分静态or动态
    2. 区分“编译分析”和“运行时”
15. 语法制导的语义计算基础
16. （知识点）S-属性文法和L-属性文法
    1. 【所有的语义动作都放在产生式的最后】
    2. 单遍分析的方法——依赖图，确定值传递的方向
    3. S-属性文法适合自底向上分析，即LR分析
    4. L-属性文法适合自顶向下分析（深度优先从左至右遍历），即LL分析
17. （知识点）S-翻译模式和L-翻译模式
    1. 【L-翻译模式中产生式右侧非终结符的有关的语义动作在相关非终结符左侧】
    2. L-翻译模式可以看做“规定好了属性处理顺序的L-属性文法”
18. 翻译模式变换
    1. 引入新的非终结符M，仅有产生式M→ε 配合M的综合属性的生成
    2. 产生式中间的语义动作集仅含复写规则【之后就可以写v[top]代码片段了】
19. 写代码片段用语义栈实现语义计算（S-翻译模式）
    1. top，v[ ]
    2. 【不要忘记标明属性（v[top].x）】
    3. 代码片段中出现的所有属性都必须是综合属性【根据产生式寻找根本来源】
20. 递归下降翻译程序的构造（L-翻译模式）
    1. 手写代码
    2. 要加上lookahead的判断【先计算PS，First，Follow集合】
    3. 形参是继承属性，返回值时综合属性
21. 消除左递归并调整语义动作（往年题没有，我也不会）
22. 静态语义分析与中间代码生成
23. 很有可能是压轴大题【照猫画虎，照葫芦画瓢，个人感觉比较难】
24. 类型检查【会给产生式含义，按含义来】
25. S-翻译模式产生TAC
    1. 只能用综合属性
    2. 可以借助翻译模式变换+拉链和代码回填将L-翻译模式改成S-翻译模式【明确M和N的用途和含义】
26. L-翻译模式产生TAC
27. 拉链和代码回填
28. 运行时存储组织
29. Display表【D[i]对应第i层的函数，有保存旧值的操作】
30. 静态链SL，动态链DL
    1. 可能导致运行表现的不同
    2. 静态链指向外层，动态链指向调用者
31. 目标代码生成和代码优化基础
32. 支配节点、回边、自然循环
    1. 支配节点包含自己
    2. 回边要回到支配节点
33. 到达定值数据流、活跃变量数据流
    1. 【考前多背公式，考场上无情迭代】
    2. Write I是使用变量I，Read I是给变量I定值
    3. 到达定值是前向数据流分析
       1. out(B)=gen(B)∪(in(B)-kill(B)), in(B)=∪out(P(B))
       2. 初始out(B)=gen(B)，in(B)=φ
       3. 集合里是语句编号
    4. 活跃变量是后向数据流分析
       1. LiveIn(B)=LiveUse(B)∪(LiveOut(B)-Gen(B)), LiveOut(B)=∪LiveIn(S(B))
       2. 初始LiveIn(B)=LiveUse(B)，LiveOut(B)=φ
       3. 集合里是各个变量
       4. LiveUse(B)∪Gen(B)=B中出现的所有变量
34. DU链、UD链
    1. Define-Use：找这个定值点所有可能的使用点
    2. Use-Define：找这个使用点所有可能的定值点
    3. 【不丢就好】
35. 寄存器相干图着色
    1. 寄存器相干图：每条定值语句中定值变量和该语句定值后活跃变量不可共用寄存器
    2. 启发式算法=>图着色问题
36. DAG图和Ershov数计算最少寄存器使用数【会就行】