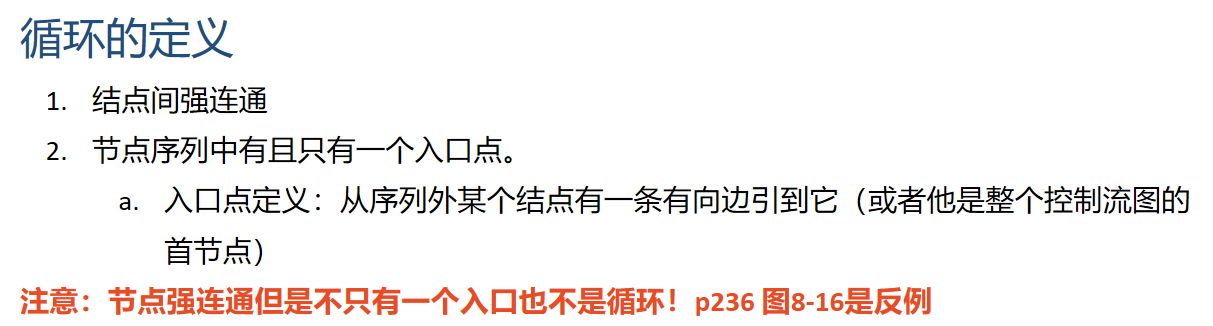
编译原理复习

# 易错点



1. LL(1)分析中，画分析表的时候，没有空串作为终结符，但是如果first集合出现了空串，要看follow集合，如果follow集合里面有#，需要加上#。
2. 计算follow集合的时候，首先：**如果取出first集合计算，不能有空串**，其次：计算左边的follow时候，**还有一种情况是ab，b中含有空串**。
3. 如果把非LL(1)转变成LL(1)后，还需要验证，这一步不能少。
4. 在LR(0)分析表中，归约式中，#也要加上，但是如果是文法开始符号的归约，则是acc
5. 对文法进行LR分析时，要仔细登记状态栈的数量，状态栈中的每一个状态都对应与历史的移进或者规约动作。当规约时查goto表栈顶弹出规约串时，弹出的候选串的数量和项目和状态栈是一一对应的！(即：可能弹出不止一个状态进行规约)
6. 注意LR(1)项目求FIRST确认搜索符的范围

# 填空题

1. DAG图可以完成的优化为\_\_\_\_\_, \_\_\_\_\_\_, \_\_\_\_\_。
2. 通常代码优化阶段由\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_组成。
3. 从优化所涉及的源程序的范围而言，可以把优化分为\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_。

## 答案

1. 常量合并与传播，无用赋值删除，公共子表达式删除
2. 控制流分析，数据流分析，代码变换
3. 局部优化，循环优化，全局优化

# 问题

* DAG优化可以完成包括
  + 常量合并与传播
  + 公共子表达式的删除
  + 无用赋值的删除
  + 死代码删除

的这些所有优化？（如果在DAG重建的时候适当删除一些变量的话）

如何从生成式判断这个文法是否有二义性？

判断文法是否是LL(1)的？

充要条件：LL(1)分析表M不含多重定义入口(满足唯一候选：FIRST∩别的FIRST和FIRST∩FOLLOW为空？)

一眼就能扫描的判定：

1. 无左递归
2. 产生式的候选式无左公因子(无回溯)

# 做题小技巧

1. 流图查找循环的时候，得到回边<a,b>，找到到达a的通路但不经过b，这个过程可以把b节点删除后，再查找。（这个好像只是充分条件？并不是所有循环都能找到
2. 入口唯一性：或许需要找到循环序列以后，再判断外部结点指向这个序列的边是否唯一
3. 找循环步骤
   1. 从n0开始，广度优先搜索，搜每个结点的D(?)
   2. 看看有没有像回边的边
   3. 对着这个回边的边看<n,d>中的d是否在D(n)中
   4. 如果在，可以试着求循环序列：图中所有(是所有！！不要漏掉下面的点）不经过d可达n的点
   5. 得到循环序列，进行入口唯一性的判断
4. 如何在文法中体现优先级：低优先级可以包括高优先级的左部（非终结符）。如E->E+T|T;T->T\*F;如果一刚开始使用T->T\*F，那么就不会有因子是加法表达式，也就满足了高优先级的特点。

# 可能的题型/考点

## 来自课本

* 有限自动机的确定化
* 有限自动机的最小化
* 对应正则语言正规式的书写
* 二义性文法的判断
* 判断是否是LL(1)文法/LR(0)文法/SLR(1)文法/LR(1)文法
* 构造LL(1)分析表
* 构造LR(0)分析表/SLR(1)分析表/LR(1)分析表
* 左递归文法的消除
* 左公因子的消除
* 改写成LL(1) [然后对改写后的文法进行判断
* 自上而下分析，并画出语法树
* 自下而上分析，给出分析表，写对文法分析的步骤
* 二义性文法的LR分析表构建
* 四元式的翻译
* 拉链反填顺序的书写
* 画出程序的控制流图
* 对DAG进行优化，并写出优化后的代码
* 程序中的活跃变量OUT
* 程序中的到达-定值集IN
* 找到控制流图中的循环
* 循环内代码是否可以外提
* 线性扫描法寄存器分配
* 图着色法分配步骤