第一组（本组）听课笔记

——注释语法分析树

1. 定义

注释语法分析树(注释分析树)：各个结点标记了相应的属性值的语法分析树。

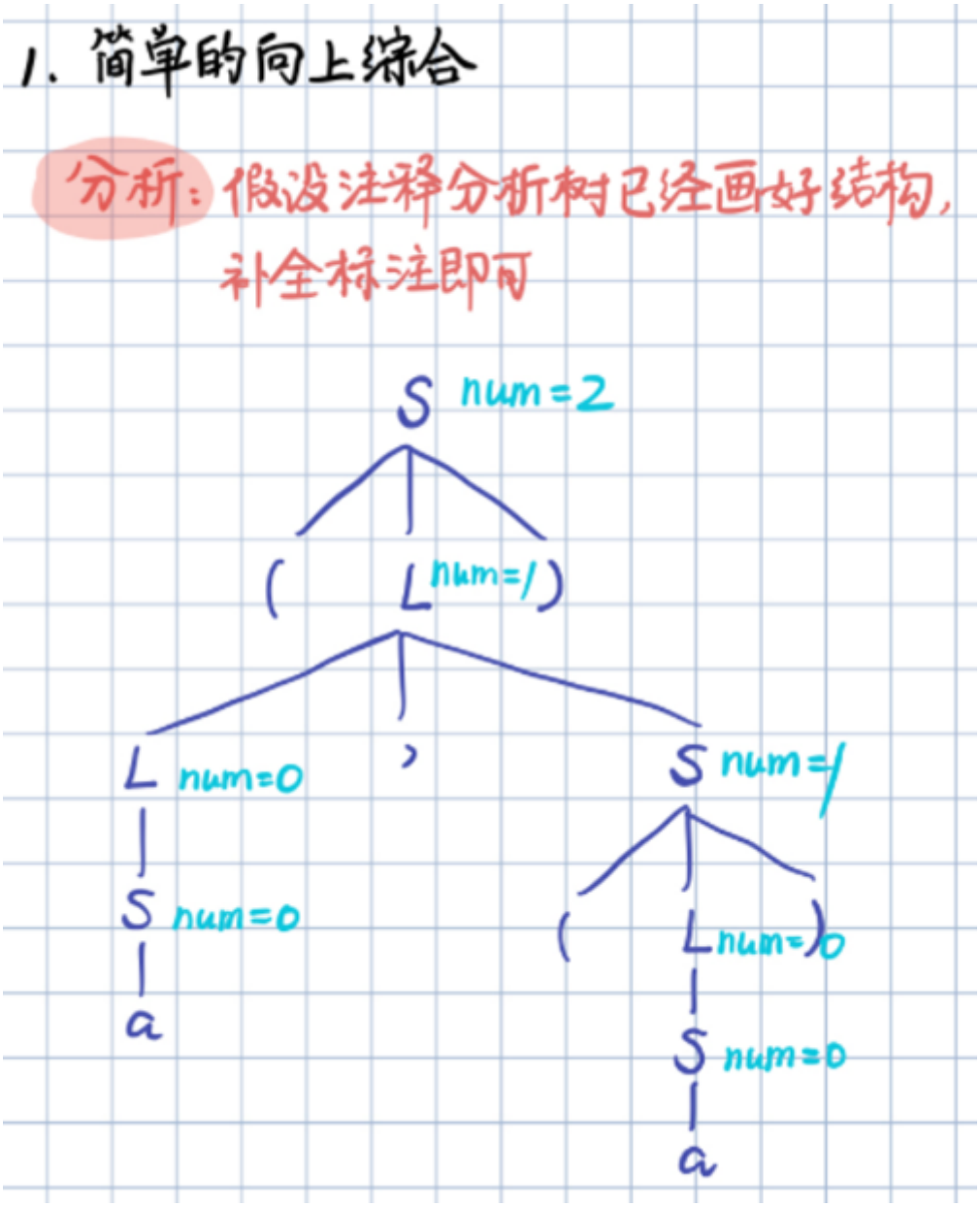
综合属性

- 自下而上传递信息
- 语法规则：根据右部候选式中的符号的属性计算左部被定义符号的综合属性
- 语法树：根据子结点的属性和父结点自身的属性计算父节点的综合属性

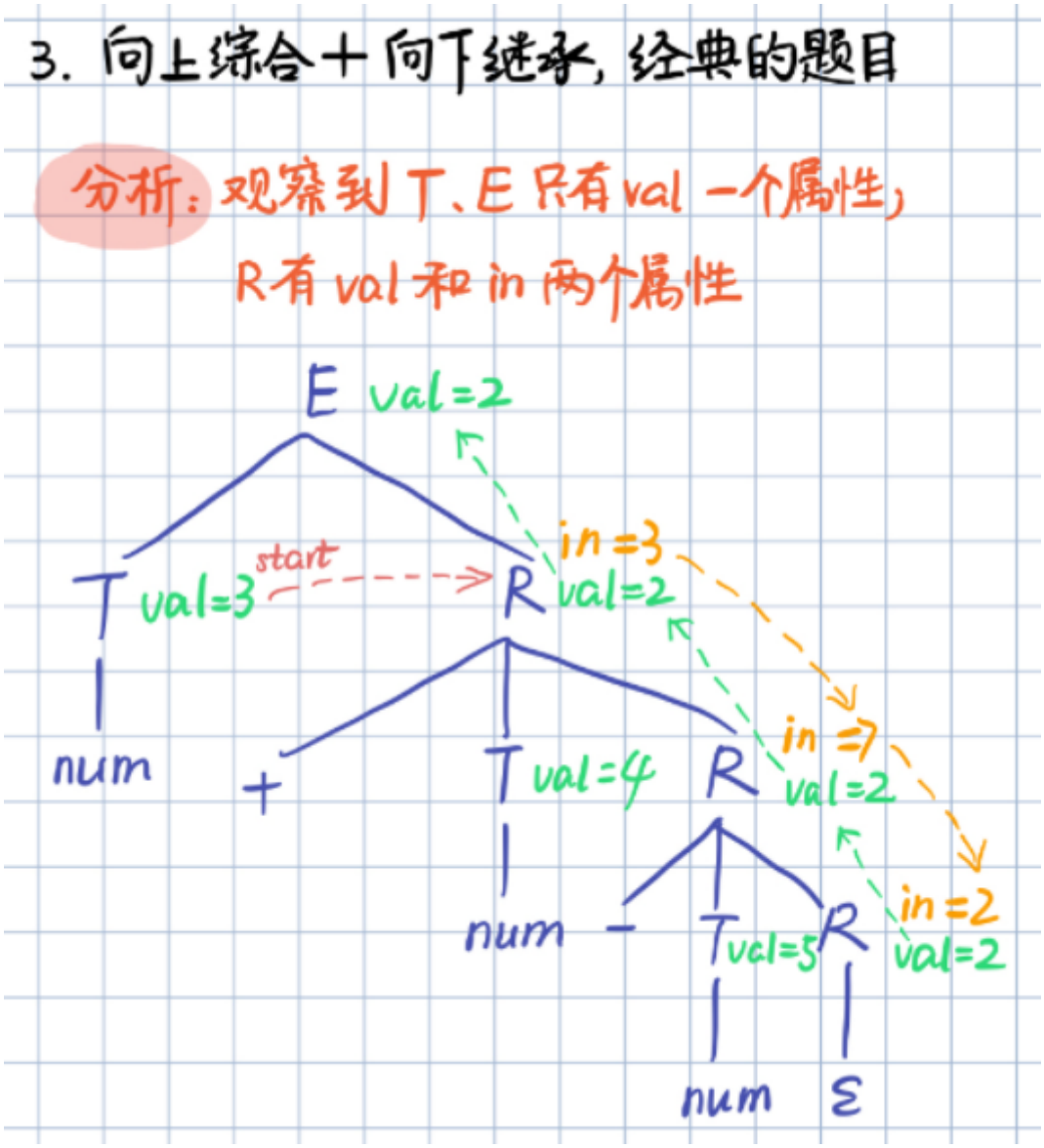
继承属性

- 自上而下传递信息
- 语法规则：根据右部侯选式中的符号的属性和左部被定义符号的属性计算右部候选式中的符号的继承属性
- 语法树：根据父结点和兄弟节点的属性计算子结点的继承属性

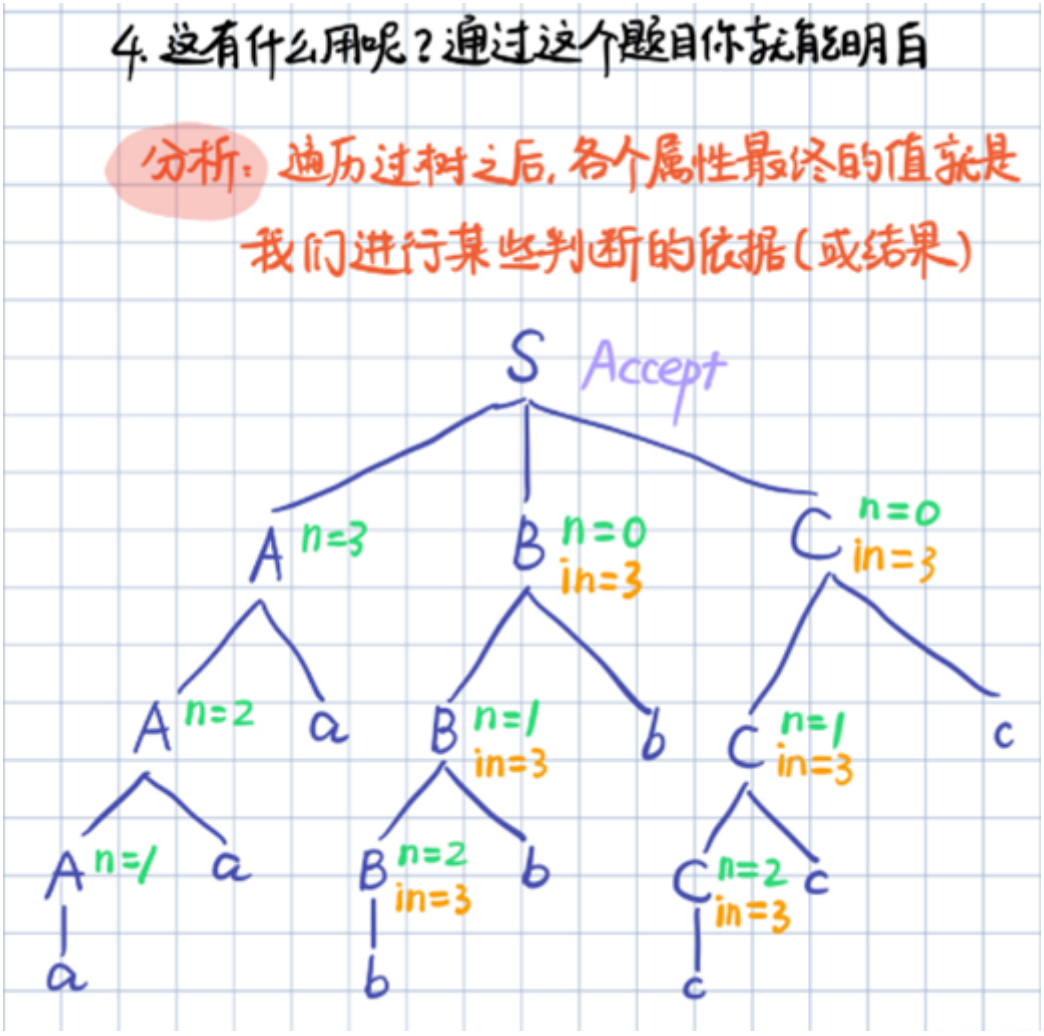
2. 简单的向上综合


1. 简单的向上综合
分析：假设注释分析树已经画好结构，
补全标注即可
S num=2
(L num=/)
L num=0
,
S num=/
S num=0
(L num=)0
a
S num=0
a

3. 向上综合+向下继承


3. 向上综合+向下继承，经典的题目
分析：观察到T、E只有val一个属性，
R有val和in两个属性
E val=2
T val=3
start
R
in=3
val=2
num
+
T val=4
R
in=7
val=2
num
-
T val=5
R
in=2
val=2
num
ε

4. 应用



第二组听课笔记

——属性文法

1. 基础概念

语法制导定义SDD：

- 将每个文法符号和一个语义属性集合相关联；
- 将每个产生式和一组语义规则相关联，用来计算该产生式中各文法符号的属性值

文法符号属性：综合属性、继承属性。

综合属性：在分析树结点 N上的非终结符A的综合属性只能通过 N的子结点或 N本身的属性值来定义；

继承属性：在分析树结点 N上的非终结符A的继承属性只能通过 N的父结点、N的兄弟结点或 N本身的属性值来定义。

2. S属性文法

定义：仅仅使用综合属性的SDD称为S属性的SDD，或S-属性定义、S-SDD。

如果一个SDD是S属性的，可以按照语法分析树节点的任何自底向上顺序来计算它的各个属性值。

3. L属性文法

定义：一个SDD是L-属性定义，当且仅当它的每个属性要么是一个综合属性，要么是满足如下条件的继承属性：假设存在一个产生式A→X1X2...Xn，其右部符号Xi(1≤i≤n)的继承属性仅依赖于下列属性。

- A的继承属性
- 产生式中Xi左边的符号 X1, X2, ... , Xi-1 的属性

语义规则：给出了属性计算的定义，没有属性计算的次序等细节；

翻译模式：给出使用语义规则进行计算的次序，把实现细节表现出来，在翻译模式中，和文法符号相关的属性和语义规则(也称语义动作)用花括号括起来，插入到产生式右部合适的位置。

建立翻译模式：

1、**只需要综合属性**

(1) 为每一个语义规则建立一个包含赋值的动作；

(2) 把这个动作放在相应的产生式右边的末尾。

2、**既有综合属性又有继承属性**

(1) 产生式右边的符号的继承属性必须在这个符号以前的动作中计算出来；

(2) 一个动作不能引用这个动作右边的符号的综合属性；

(3) 产生式左边非终结符的综合属性只有在它所引用的所有属性都计算出来以后才能计算。计算这种属性的动作通常可放在产生式右端的末尾。

第三组听课笔记

——依赖图

1. 依赖图的构造

由于语义动作可以出现在产生式右部的任何位置，因此会带来依赖关系。如何保证正常计算?

可以通过附注语法树，找出属性间的依赖关系，从而确定属性操作的先后顺序（拓扑排序）

依赖图中为每一个属性设置一个结点，如果属性b依赖于属性c，则从属性c的结点有一条有向边连到属性b的结点。

2. 根据依赖图求出文法属性的拓扑排序

(1) 从 DAG 图中选择一个 没有前驱（即入度为0）的顶点并输出；

(2) 从图中删除该顶点和所有以它为起点的有向边。

(3) 重复 1 和 2 直到当前的 DAG 图为空或当前图中不存在无前驱的顶点为止。后一种情况说明有向图中必然存在环。

第四组听课笔记

——语法制导翻译方案的应用

1. 语法制导翻译方案定义

语法制导翻译的根本上是在一个上下文无关文法中通过向结果中添加动作（action）来工作的，从而形成语法制导定义（Syntax-Directed Definition）。动作是指，一个结果在推导过程中被使用的时候，将要被执行的步骤或过程。 一个嵌入了将要执行的动作的语法规则，称为一个语法制导翻译计划（有时简称为“翻译计划”）。

2. 应用：抽象语法树

详细介绍了S属性和L属性的SDD，从构造SDD、构造SDT到生成a - 4 + c的抽象语法树的过程，并给出了使用clang生成AST的代码。

3. 应用：中间代码生成

采用中间语言的好处：
（1）便于进行与机器无关的代码优化工作；
（2）使编译程序改变目标机更容易；
（3）使编译程序的结构在逻辑上更为简单明确。

一遍扫描的中间代码生成：

- 通过makelist创建四元式链表——通过merge合并链表——通过backpatch对链表中的四元式回填；
- 通过回填的方式实现了一遍扫描生成中间代码；
- 通过拉链与回填操作，在一遍扫描实现语法分析的同时完成控制流翻译生成了中间代码；
- 有C语言的中间代码和编程实现的举例。

4. 应用：类型检查

静态检查和动态检查

二者的区别：

- 静态检查倾向于类型错误，即与特定的值无关的错误。
- 动态类型检查倾向于特定值才会触发的错误。

静态检查

1.语法错误

2.错误的名字

3.参数个数不对

4.参数类型不对

5.错误的返回类型

动态检查

1.非法变量名

2.无法表示的返回值

3.越界访问

4.使用一个null对象解引用

第五组听课笔记

——语法制导定义转换为语法制导翻译方案

1. 概述

语法制导翻译包含语法分析、语义分析、中间代码生成

语法制导基本思想：为CFG中的文法符号设置语义属性，用来表示语法成分对应的语义信息.

语法制导定义SDD是对CFG的推广：将每个文法符号和一个语义属性集合相关联>将每个产生式和一组语义规则相关联，用来计算该产生式中各文法符号的属性值

2. SDD的求值顺序、S-属性定义与L-属性定义

3. 语法制导翻译方案SDT

SDT是在产生式右部嵌入了程序片段的CFG，这些程序片段称为语义动作。按照惯例，语义动作放在花括号内

- 可以看作是对SDD的一种补充，是SDD的具体实施方案
- 显式地指明了语义规则的计算顺序，以便说明某些实现细节

将L-SDD转换为SDT的规则

- 将计算某个非终结符号A的继承属性的动作插入到产生式右部中紧靠在A的本次出现之前的位置上
- 将计算一个产生式左部符号的综合属性的动作放置在这个产生式右部的最右端

4. 在非递归的预测分析&在递归的预测分析过程中进行翻译&L-属性定义的自底向上翻译

列出详细图表进行介绍。

第六组听课笔记

——中间代码形式

1. 中间语言的特点和作用

特点

- 独立于机器
- 复杂性界于源语言和目标语言之间

引入中间语言的优点

- 使编译程序的结构在逻辑上更为简单明确
- 便于进行与机器无关的代码优化工作
- 易于移植

2. 常用的中间语言

- 后缀式，逆波兰表示
- 图表示：抽象语法树(AST)、有向无环图(DAG)
- 三地址代码
  - 三元式
  - 间接三元式
  - 四元式

第七组听课笔记

——综合属性和继承属性

介绍了综合属性和继承属性，S - 属性文法，L-属性文法。

1. 综合属性

（1）在语法树中，一个结点的综合属性的值由其子结点和它本身的属性值确定。
（2）使用自底向上的方法在每一个结点处使用语义规则计算综合属性的值
（3）仅仅使用综合属性的属性文法称S - 属性文法

2. 继承属性

（1）在语法树中，一个结点的继承属性由此结点的父结点、兄弟结点和它本身的某些属性确定
（2）用继承属性来表示程序设计语言结构中的上下文依赖关系很方便

3. L-属性文法

如果对于每个产生式A→X1X2...Xn的每个语义规则中，每个属性或者是综合属性，或者是Xi(1≤i≤n)的一个继承属性且这个继承属性仅依赖于:
（1）产生式的左边符号Xi的属性;
（2）A的继承属性。
L-属性文法也就是"左属性"文法，计算每一个继承属性时不能引用右边符号的属性。也就是说，在这类属性文法的依赖图中，不能有从右到左的边。