语法制导翻译预习

讲义5.1~5.6

继承属性

- 综合属性：父节点依赖孩子节点

  S-属性定义

  计算时遍历语法树的顺序?
- 继承属性：孩子节点依赖父节点和兄弟节点，更强

  L-属性定义：只依赖左兄弟

  计算时遍历语法树的顺序?
- 本章难点：翻译（属性计算）工作融入到语法分析过程
- 难在哪里?
- 预测分析+L-属性定义：自顶向下
- LR分析+S-属性定义：自底向上
- 但我们显然希望强强联合

  LR分析+L-属性定义：顺序冲突

LR分析+S-属性定义

- 都是自底向上，顺序自然
- 技术手段
  - 语法符号栈之外再引入属性栈
  - 前者中每个语法符号的属性值位于后者中相同为止
- 语法符号压栈——对应属性值压入属性栈
- 语法符号弹出栈——对应属性值从属性栈弹出

预测分析+L-属性定义

- 都是自顶向下，顺序自然
- 难点：文法消除左递归，翻译模式如何相应修改？
- 具体的，原本两个运算对象在同一个产生式中（兄弟节点）

  $E \to E_1 + T$——E_1 和 T

  消除左递后分散到两个产生式中

  $E \to T\ R$ ， $R \to +\ T\ R$ ——两个T
- 解决方法？

  必须在有一个中间桥梁将两个T联系起来——R!

  R把第一个T的计算结果传递到第二个产生式——继承属性!

LR分析+L-属性定义

- 自底向上 vs. 自顶向下，天然矛盾
- 困难1：嵌入语义动作无法归属产生式

  解决方法：用marker NT（只有产生式，因此不会影响接受的符号串集合）代替嵌入语义动作的位置

  所有语义动作都在产生式末尾，只会在归约时执行
- C→XYZ归约时用到C的继承属性

  但C还不在栈中，它的属性也不在!

  解决方法：“子/弟债父/兄还”，C的继承属性来自于父/左兄弟节点

  A→BCD：来自于B——B在栈中!

  来自于A：再找A的左兄弟/父节点

  最坏情况一直到根节点

类型检查预习

讲义6.1~6.4

类型系统

- 类型表达式：类型的形式化描述

  像算术表达式、正则表达式一样去理解它!

  基本类型：int、char、…

  类型构造符：基于其他类型(基本类型/复杂类型)构造出复杂类型
- 类型系统:

  检查每个操作/运算的对象的类型是否符合要求

  将结果类型赋予操作/运算结果

  可用语法制导翻译方法实现

类型表达式等价

- 结构等价：透过表面看本质

  两个类型的类型表达式树（已完全展开，没有别名、变量等）等价

  算法：递归判定两棵树等价

  优化：树链，类型构造符忽略次要因素，一个二进制串表示

  二进制串不等类型不等价
- 名字等价：只看表面

  披着羊皮的狼与狼不等价

  Pascal语言采用的类型等价判定方式

  对C/C++而言，类/结构体的等价判定也必须采用名字等价