# 《语法分析》学案

1. **学习目的**

能力获取：掌握构造语法分析器所需的基本理论、具体算法和开发工具。

1. 描述源语言语法结构的能力——对上下文无关文法更深入的理解，包括与有限自动机的关系、为适应语法分析算法进行改写等。
2. 构造非递归预测分析器的能力——理解如何用一种表驱动的机制实现预测分析，掌握计算FIRST集、FOLLOW集以及用它们构造预测分析表的方法。
3. 构造自底向上语法分析器的能力——理解“移进-归约”机制及活前缀等基本概念，理解算符优先分析算法和LR分析算法确定句柄、实现正确移进-归约的思想，掌握构造算法优先关系表、LR分析表（包括SLR、规范LR及LALR）的方法以及用表驱动移进-归约分析过程的方法。
4. 理解一些相关问题，如消解分析表冲突以实现对二义性文法的分析等。

**最重要的算法：LR分析表构造算法，其中的关键词是“活前缀性质”，重点理解活前缀的定义，特别是从“移进-归约”过程角度给出的定义——这就是活前缀性质，基于此再理解如何利用此性质构造正确的自底向上语法分析算法。**

**知识点的关联：预测分析法表驱动实现与第二章介绍的递归下降实现的异同；LR分析表构造中项目集规范族的计算本质上是子集构造法——根本原因是这个过程其实是在构造识别活前缀的DFA。**

1. **学习内容**

**第一部分：基本概念和正则表达式**

1. 基本概念和正则表达式：
2. 了解程序错误的特点，理解编译器错误处理的基本方式。
3. 更深入理解正则表达式的基本概念——相对第二章的进一步内容，符号约定：“占位符”，在一些演算中表示一类符号、符号串；推导：一步推到、多步推导符号，最左（右）推导概念；二义性：基于推导的定义。
4. 理解正则表达式（有限自动机）与CFG的关系，掌握NFA到CFG的转换。
5. 理解CFG的验证——用数学归纳法证明CFG生成的语言与要描述的符号串集合相等（两者相互包含）。
6. 掌握CFG的改写方法，理解改写的目的，消除二义性——所有语法分析算法所必须的，消除左递归、提取左公因子——预测分析法所要求的，消除产生式、消除回路——消除左递归算法所要求的。

**难点：理解NFA到CFG转换算法的思想，CFG的非终结符Ai对应NFA中状态i🡪终态所有可能路径上的符号串的集合。如果令其对应初态🡪NFA中状态i所有可能路径上的符号串集合，则可得到另一种转换算法。**

**第二部分：自顶向下语法分析**

1. 非递归预测分析法
2. 回顾预测分析法的思想，选定的产生式应该推导出与输入单词串一致的结果——简化为首单词一致，每个产生式预先算出首单词集合，语法分析时选择与下一输入单词匹配的产生式。
3. 理解如何用状态转换图表示产生式、进行语法分析。
4. 理解非递归预测分析法算法思想，核心是表驱动，理解预测分析表结构、表项含义，进而理解如何根据语法符号栈当前栈顶和输入缓冲区下一单词查表、表项内容如何指导操纵语法符号栈和输入缓冲区——即为预测分析。
5. 掌握如何构造预测分析表，如何计算FIRST函数，理解根据预测分析思想用FIRST函数指导将普通产生式填入分析表；理解FOLLOW函数的含义、掌握其计算方法，理解如何用FOLLOW函数指导将产生式填入分析表。
6. 理解结合预测分析法设计错误处理方法，表项为空即为语法错误，设计Panic模式恢复错误——一种策略是认为栈顶NT A已无法与输入单词匹配、另一种是认为仍可匹配——产生两种单词丢弃策略，前者丢弃符号直至遇到FOLLOW(A)、后者直至FIRST(A)。

**重点&难点：熟悉表驱动语法分析的模式，后面算符优先分析和LR分析也都是表驱动，与递归下降纯手写递归函数方式完全不同，将CFG无关的分析算法框架和CFG相关的分析表解耦。**

**理解FOLLOW函数在预测分析中的作用，这里涉及一个基本能力——复现语法分析实时状态，即栈和输入缓的冲状态，并对其进行相关分析，这一能力会反复用到。对本问题，复现栈和输入状态后，理解两者是语法等价关系，前者推导出后者，从而推断出，如果是用产生式进行下一步推导，则栈顶NT与输入T是FOLLOW关系，所以用FOLLOW指导产生式的使用。**

**理解如何给出错误信息，复现栈和输入的状态后，根据其体现的语法含义，并结合采用的哪种Panic策略，来给出错误信息。**

**第三部分：自底向上语法分析**

1. 自底向上语法分析
2. 理解移进-归约过程，“移进”——T从输入缓冲到栈，“归约”——**栈顶**符号串（产生式右部）替换为产生式左部，理解这个过程与最右推导过程恰好是相反的，归约是一步最右推导的逆操作。
3. 理解句柄的概念，归约动作替换掉的符号串，注意，不是“与产生式右部匹配”就是句柄——应是最右推导的拟操作！理解活前缀的概念，最右**句型**的**前缀**，“活”——其末端不超过最右**句柄**的末端——自底向上分析过程可出现在栈中的符号串——在设计LR分析算法时会利用这一特性。
4. 理解自底向上语法分析算法设计的核心也是难点——定位句柄、确定用哪个产生式进行归约，一个自底向上语法分析算法的基本动作包括移进、归约、接受、错误。
5. 算符优先分析算法
6. 理解算符文法定义，无产生式、无连续NT。理解运算符优先关系的定义——与我们通常理解的优先级不尽相同、指的就是运算顺序，理解与数值大小关系的不同——可能三种关系都不成立、高于和低于不是逆关系、等于也不是自反关系。
7. 理解算符优先分析的思想，通过运算符优先级——即运算顺序——来确定句柄，优先级高于前后的运算符即为句柄——因为先归约就意味着先运算，理解将这一思想在移进-归约框架下实现。
8. 理解算符优先分析的核心——构造运算符优先关系表，一种方法是基于对CFG的理解，直接给出运算符优先关系（运算顺序）；另一种方法是在不了解CFG的情况，采用算法计算出优先关系——核心思想：语法树中后代节点先运算（优先级高、先归约）。
9. 理解错误处理，本质上与预测分析法的错误处理类似，也是复现栈和输入缓冲的状态，理解其语法含义，决定给出什么错误信息（认为发生了什么语法错误），进而给出错误恢复策略。
10. LR分析方法
11. 理解LR分析方法的核心思想，活前缀特性——如果有一个正确的自底向上语法算法，则其分析过程中可出现中栈中的符号串都是活前缀，倒过来利用此特性设计自底向上的语法分析算法——它能识别活前缀（构造了一个DFA）、保证栈中始终是活前缀，即为正确的自底向上语法分析。
12. 理解LR分析算法框架。理解栈中“状态”的作用——要保证栈中符号串是活前缀，读取栈中所有符号太低效，用状态表示其下所有符号构成的符号串，读取状态即可知符号串。理解LR分析表的结构和表项含义，进而理解分析表驱动语法分析的机制，查action表知道进行移进、归约、接受、错误等基本操作，归约时查GOTO表得到栈顶新状态。
13. 理解SLR分析表的构造方法，理解LR(0)项目的含义——不可能穷举所有活前缀（无穷）来构造识别它的活前缀，LR(0)项目就是或前缀的替代物，“.”就是栈顶。理解项目及规范族的计算，不要死记硬背——就是子集构造法。理解根据项目集规范族（DFA）填SLR分析表的方法。
14. 理解规范LR分析表的构造，为什么需要这种方法，SLR的归约动作只根据FOLLOW确定，不精确，因而扩充LR(0)项目，加上搜索符，直接指出活前缀与单词的跟随关系，从而精确确定归约动作。与SLR分析表的构造相比，闭包运算和归约动作的确定有所变化，其他相似。
15. 理解LALR分析表的构造，SLR和规范LR的折中，掌握同心集合并方法即可。
16. 理解LR分析算法的错误恢复和处理二义性文法，关键技术还是恢复栈和输入缓冲的状态，分析其语法含义（语法错误类型和二义性），然后给出合理的错误信息和恢复方法，或是消除二义性的方法。

**重点&难点：理解为什么项目集规范族的计算就是子集构造法，不需要任何计算，直接就可以构造出一个识别活前缀的NFA，LR(0)项目即为其状态，边的构造也很直接（参见讲义中例子）。而项目集规范族的计算是构造DFA，本质上就是从这个NFA转换为DFA，因此实际上就是子集构造法过程，将其中状态集的-closure和运算对应为项目集的closure和goto运算，不用显示构造出NFA，用项目集即可进行子集构造法过程。**