编译原理

实验三：典型语句的语义分析及中间代码生成

规格严格，功夫到家

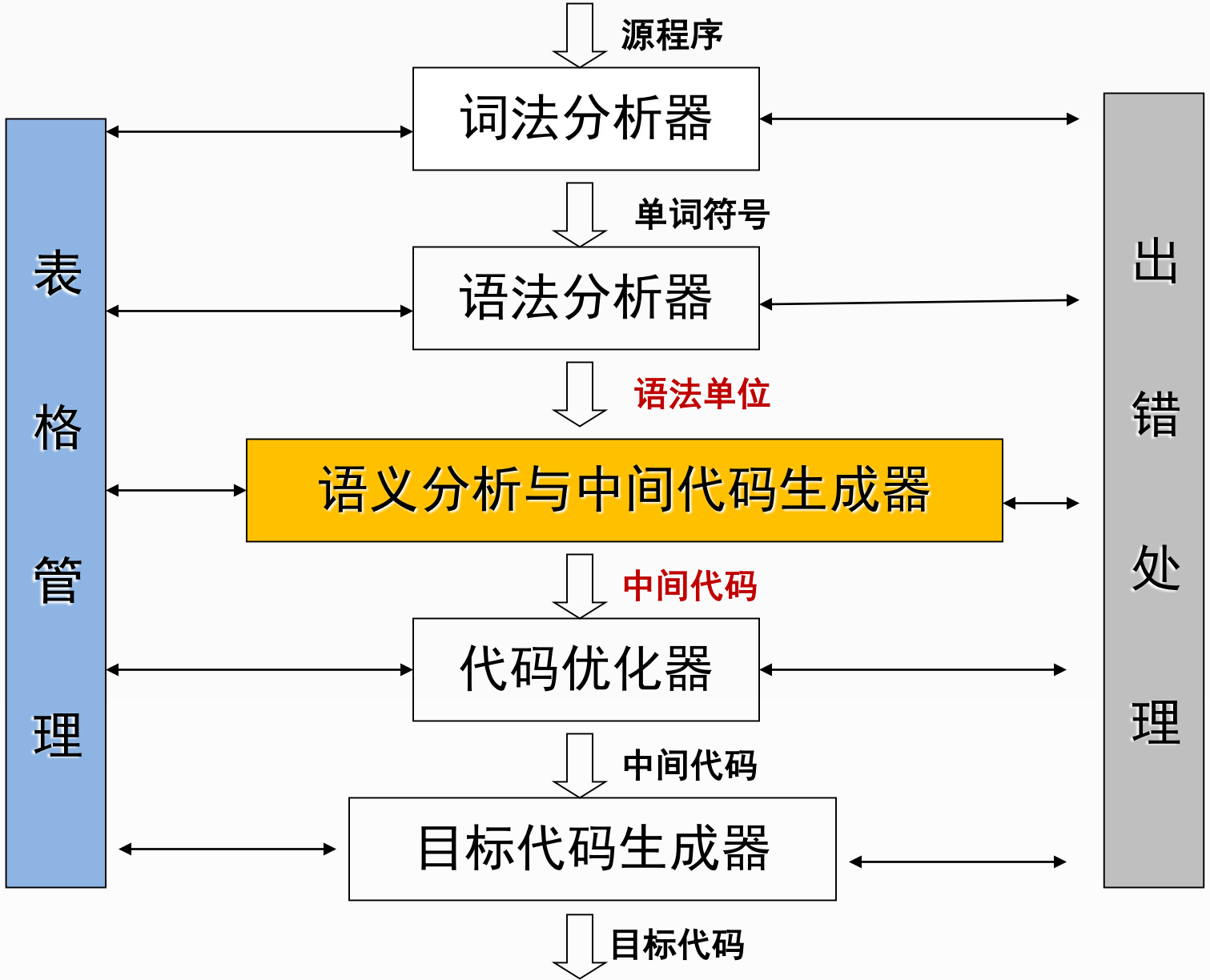
编译程序的总体结构

✔实验一：

✔实验二：

✔实验三：

✔实验四：



实验目的

1. 巩固对语义分析的基本功能和原理的认识；

2. 加深对自底下上语法制导翻译技术的理解，掌握声明语句、赋值语句和算术运算语句的翻译方法。

3. 理解中间代码的表示形式，掌握三地址码的实现方式。

实验学时数：4学时

实验内容

1. 采用实验二中的文法，为语法正确的单词串设计翻译方案，完成语法制导翻译。

2. 利用该翻译方案，对所给程序段进行分析，输出生成的中间代码序列和更新后的符号表，并保存在相应文件中，中间代码使用三地址码的四元式表示。

注：更新后的符号表只需要保存id和type两个属性。

3. 实现声明语句、简单赋值语句和算术表达式的语义分析与中间代码生成（参考教材P260声明语句的翻译, P267赋值语句的翻译）。

4. 使用框架中的模拟器IREmulator验证生成的中间代码的正确性。

实验原理

1、语义分析器的主要任务是检查各个语法结构的静态语义，亦即验证语法正确的程序结构是否真正有意义，例如类型检查，唯一性检查等。

2、语法制导定义：将一个属性集合和文法符号相关联；将产生式和一组语义动作相关联。

3、语法制导翻译：在语法制导定义的基础上，将语义规则插入到产生式右部相应位置。使用语法制导翻译将语义检查和中间代码生成结合到语法分析中进行。

S-属性定义的自底向上翻译

S-属性定义是只含有综合属性的语法制导定义，可以在自底向上的语法分析过程中被高效的实现。

注：综合属性指的是只能通过该节点自己或者子节点的属性来计算的属性

S-属性定义的翻译方案，只需要将每个语义规则放到相应产生式的末尾即可得到翻译模式，在对产生式规约时执行相应的语义动作。

S-属性定义的自底向上翻译

赋值语句和声明语句的翻译方案：

输入：

int a;

a = 3;

S -> D id	{p = lookup(id.name); if p != nil then enter(id.name, D.type) else error}
D -> int	{D.type = int;}
S -> id = E	{gencode(id.val = E.val);}
E -> A	{E.val = val;}
A -> B	{A.val = B.val;}
B -> IntConst	{B.val = IntConst.lexval;}

Lookup(name)：查询符号表，返回name对应的记录

gencode(code)：生成三地址指令code

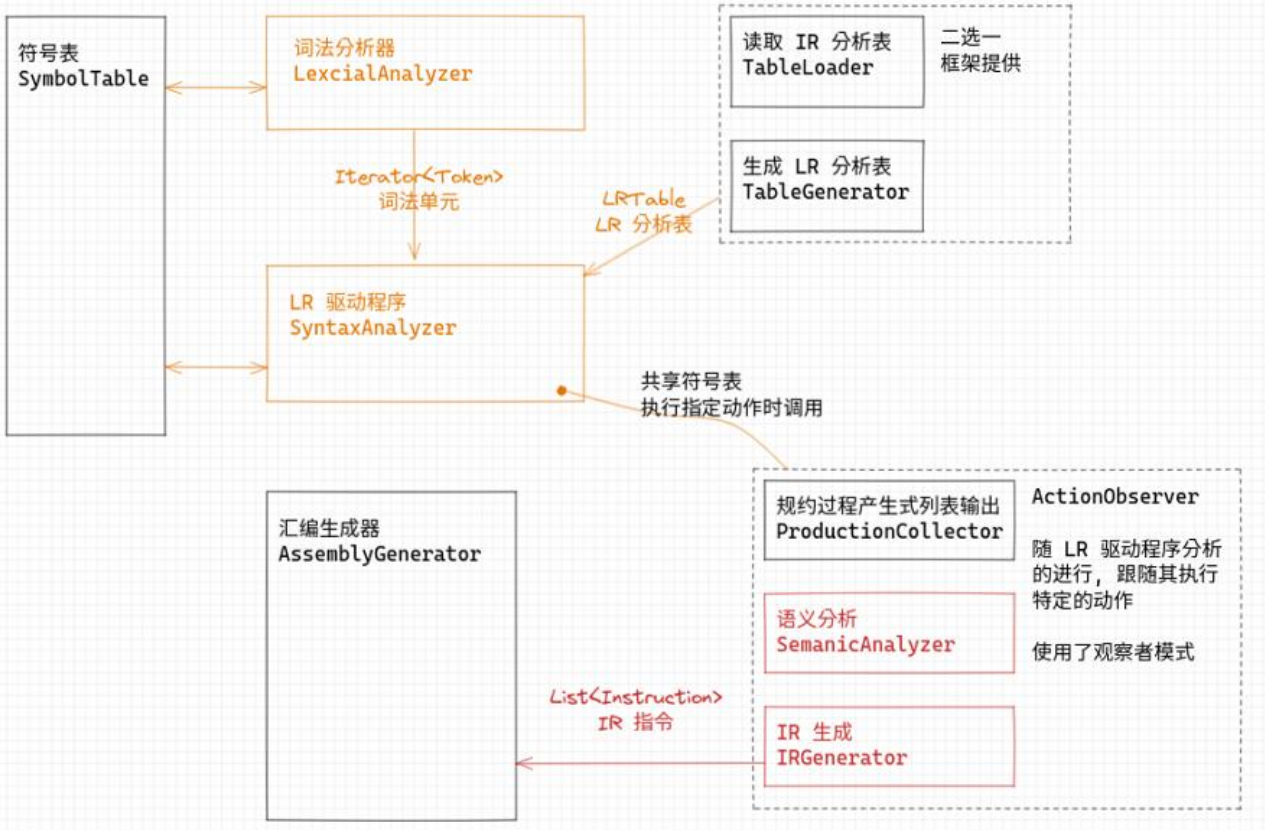
enter：将变量的类型填入符号表

语义分析栈

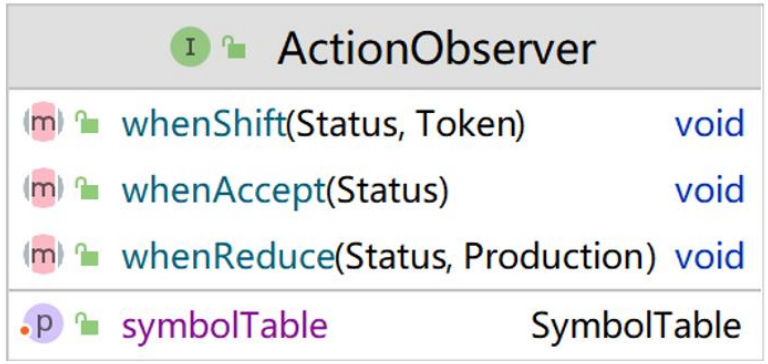
- 语义分析的结果是更新符号表，根据翻译方案，语义分析栈需要保存type属性；
- 与语义分析相关的产生式有S -> D id和 D -> int；
- 自底向上的分析过程中Shift时将符号的type属性（从Token中获得）入栈；
- 自底向上的分析过程中Reduce时，先出栈产生式右侧符号的属性。如果规约时使用的产生式是D -> int，需要将D符号的type属性入栈，如果规约时使用的产生式是S -> D id，更新符号表中相应变量的type信息，压入空记录占位，如果使用其他产生式规约，直接压入空记录占位。

注：中间代码生成过程与语义分析过程类似，可以使用栈来处理，栈中需要保存翻译方案中的value属性。

实验步骤

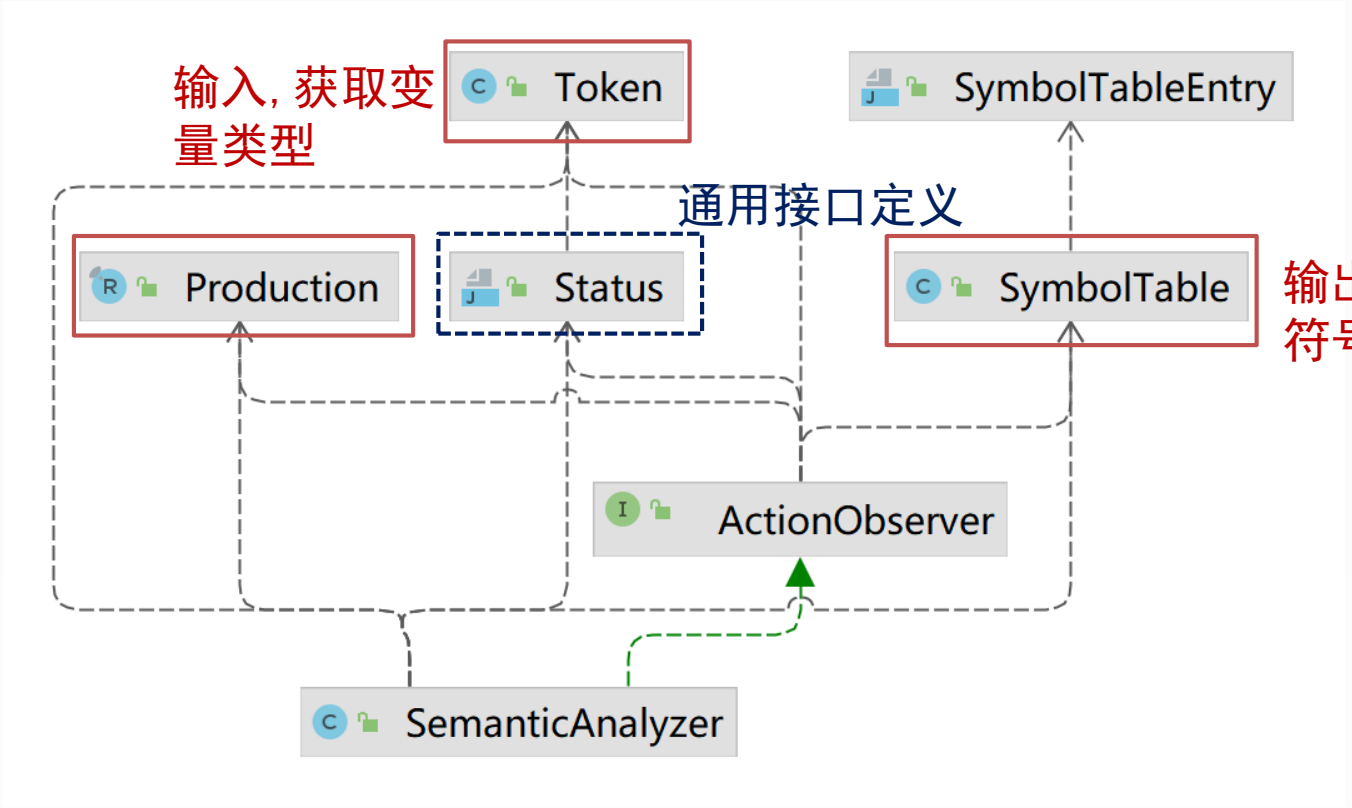



```java
// 加载 LR 分析驱动程序
final var parser = new SyntaxAnalyzer(symbolTable);
parser.loadTokens(tokens);
parser.loadLRTable(lrTable);

// 加入生成规约列表的 listener
final var productionCollector = new ProductionCollector(GrammarInfo.getBeginProduction());
parser.registerObserver(productionCollector);

// 加入用作语义检查的 listener
final var semanticAnalyzer = new SemanticAnalyzer();
parser.registerObserver(semanticAnalyzer);

// 加入用作 IR 生成的 listener
final var irGenerator = new IRGenerator();
parser.registerObserver(irGenerator);
```



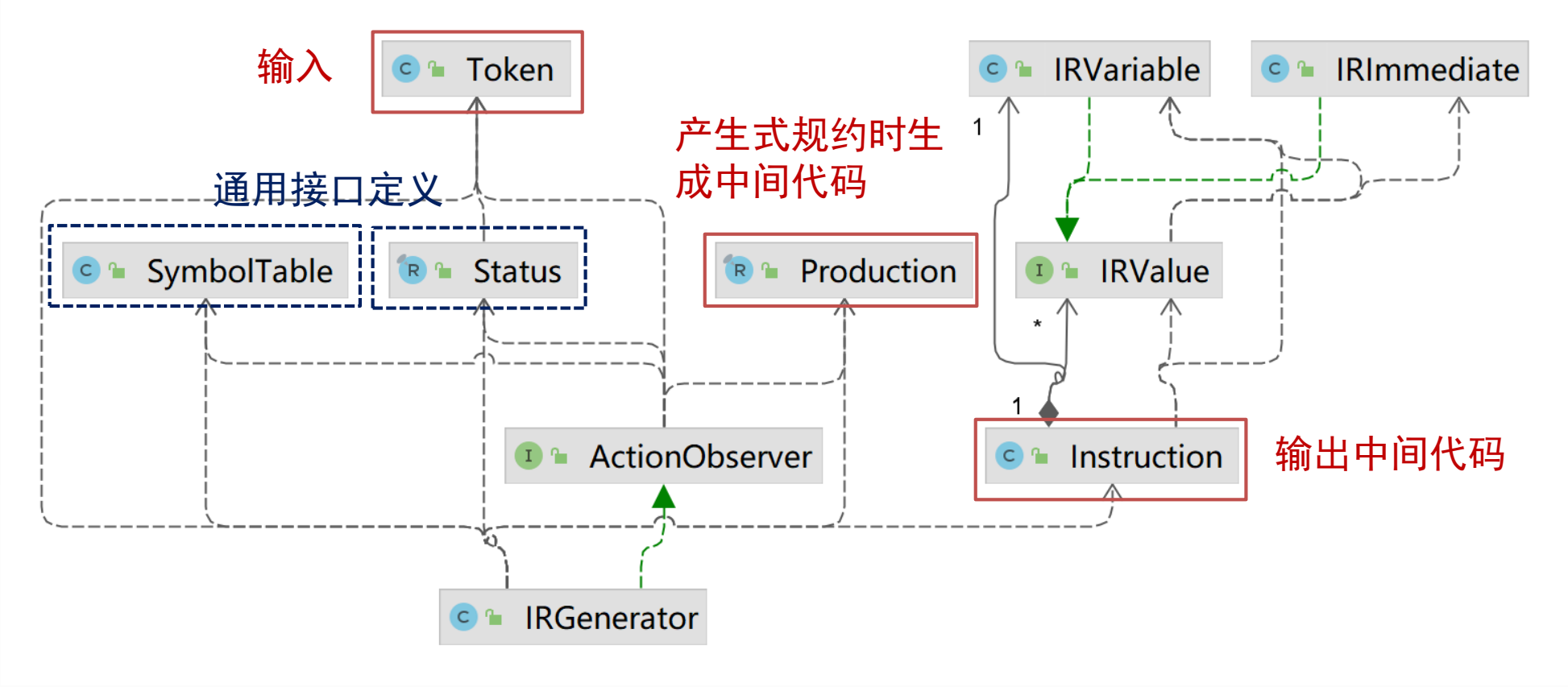
实验步骤

语义分析SemanticAnalyzer ： 在自底向上的分析过程中更新符号表。



实验步骤

中间代码生成IRGenerator： 在自底向上的分析过程中生成中间代码（使用四元式表示）。



实验步骤

1、设计文法的翻译方案；

2、设计各个观察者需要的用于语义分析和中间代码生成的数据结构（栈）；

3、实现各观察者在分析过程中的动作处理；

4、使用IREmulator对生成的中间代码进行验证。

同学们，
独立开始实验