**编译原理**

班级：21计算机科学与技术（4）班

姓名：陈昊天 学号：2021329600006

实验二 自顶向下语法分析器的构建

1. 实验要求

运用LL(1)分析法，针对给定的上下文无关文法，给出实验方案。预估实验中可能出现的问题。

1. 实验方案

（评价依据实验方案设计是否合理，包括输入输出的设计）

使用递归下降分析技术。

基本思想：

识别程序由一组子程序构成，其中每个子程序对应于文法的一个非终结符；这些子程序通常是递归子程序。

文法中每个非终结符的对应的子程序，根据选定的产生式右部出现的符号来设计，具体构造方法如下：

1. 每遇到一个终结符，则判断当前读入token是否与该终结符匹配。若匹配，则继续读下一个token，若不匹配，则进行错误处理。
2. 每遇到一个非终结符，则调用相应的子程序。

输入设计：

输入的序列分解为token序列，一个字符识别为一个token，简化了输入部分的处理。

输出设计：

在每个子程序的处理中，若成功则继续处理，若发生错误则进行错误处理。在错误处理时如果无法回退，则输出 LL1 parse failed 表示解析错误。

1. 预估问题

预估问题（是否有预估的问题，预估的问题是否合理）

1. 需要将给定文法转换为LL1文法
2. 输入的句子不符合给定文法，进行错误处理

理论基础（评价依据 理论知识非常清楚）

一般情况下，假定关于A的产生式是：

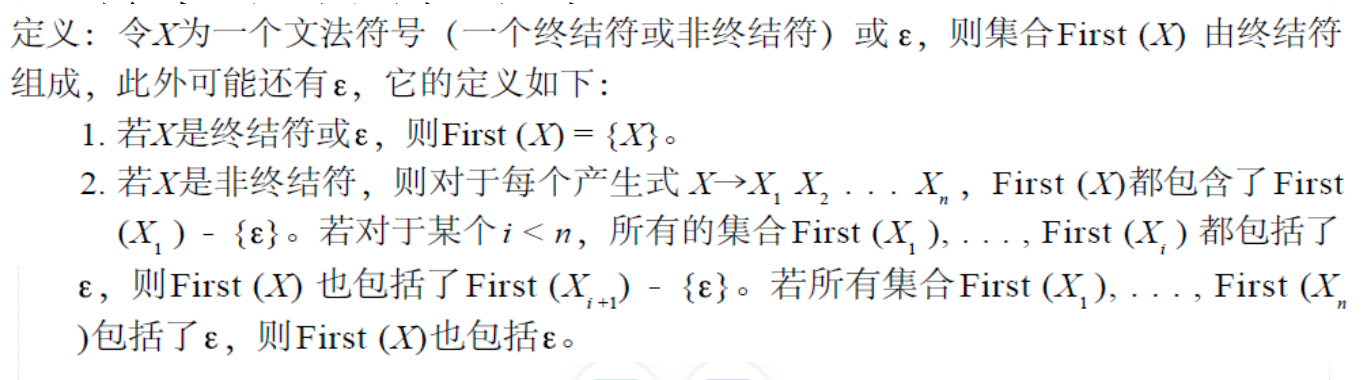
A→Aα1| Aα2 |… |Aαm|β1|β2 |…|βn

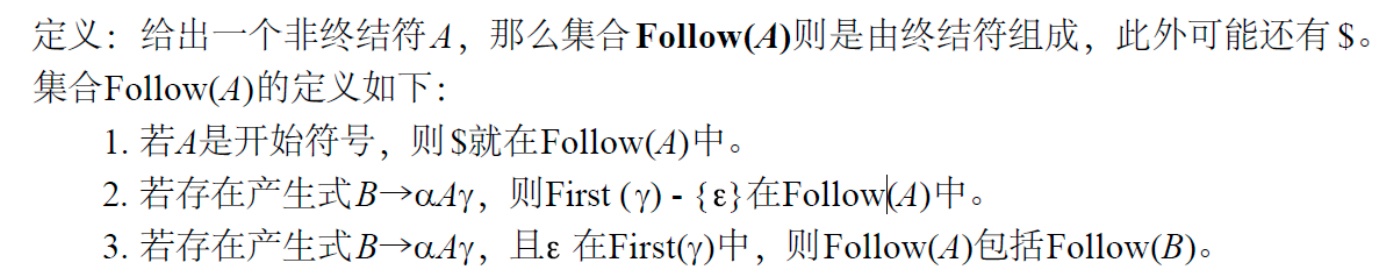
其中，αi(1≤i≤m)均不为空，βj(1≤j≤n)均不以A打头。

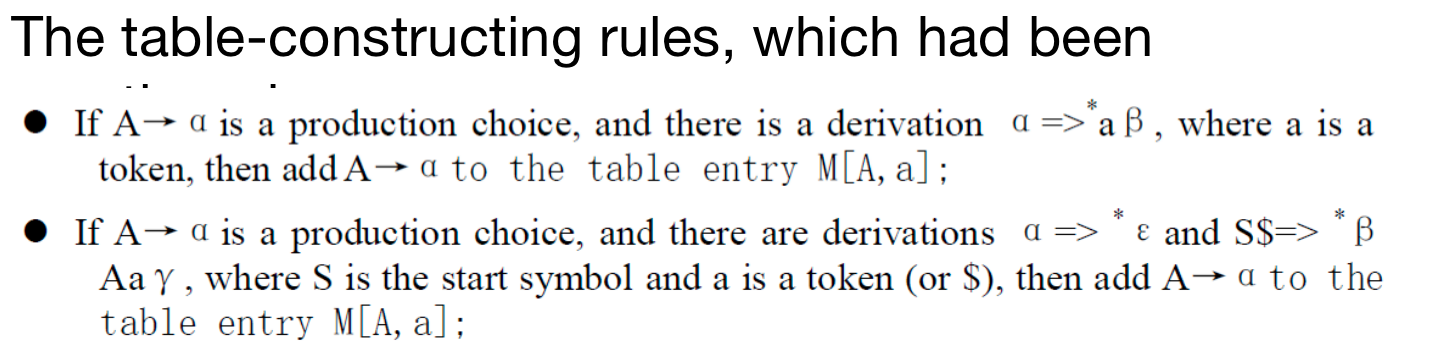
则消除直接左递归后改写为：

A→ β1A'| β2 A' |…| βnA'

A'→ α1A' | α2A' |…| αmA' |ε

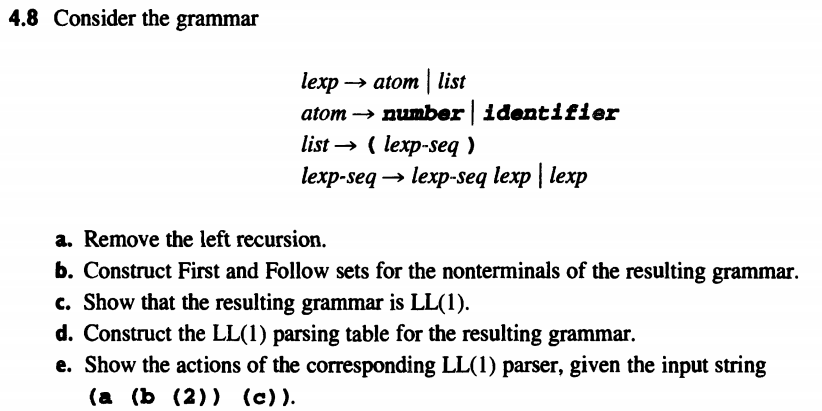






1. 内容和步骤

1、针对4.8习题输入和输出的设计及代码



**原文法**

lexp -> atom | list

atom -> **number** | **identifier**

list -> **(** lexp-seq **)**

lexp-seq -> lexp-seq lexp | lexp

1. 消除左递归

lexp -> atom | list

atom -> **number** | **identifier**

list -> **(** lexp-seq **)**

lexp-seq -> lexp lexp-seq’

lexp-seq’ -> lexp lexp-seq’ | ε

1. 构造First和Follow集
2. First集

First( lexp ) = { number, identifier, ( }

First( atom ) = { number, identifier }

First( list ) = { ( }

First( lexp-seq ) = { number, identifier, ( }

First( lexp-seq’ ) = { ε, number, identifier, ( }

1. Follow集

Follow( lexp ) = { $, number, identifier, (, ) }

Follow( atom ) = { $, number, identifier, (, ) }

Follow( list ) = { $, number, identifier, (, ) }

Follow( lexp-seq ) = { ) }

Follow( lexp-seq’ ) = { ) }

1. 判定是LL1文法

文法G是LL(1)的，当且仅当G的任意两个具有相同左部的

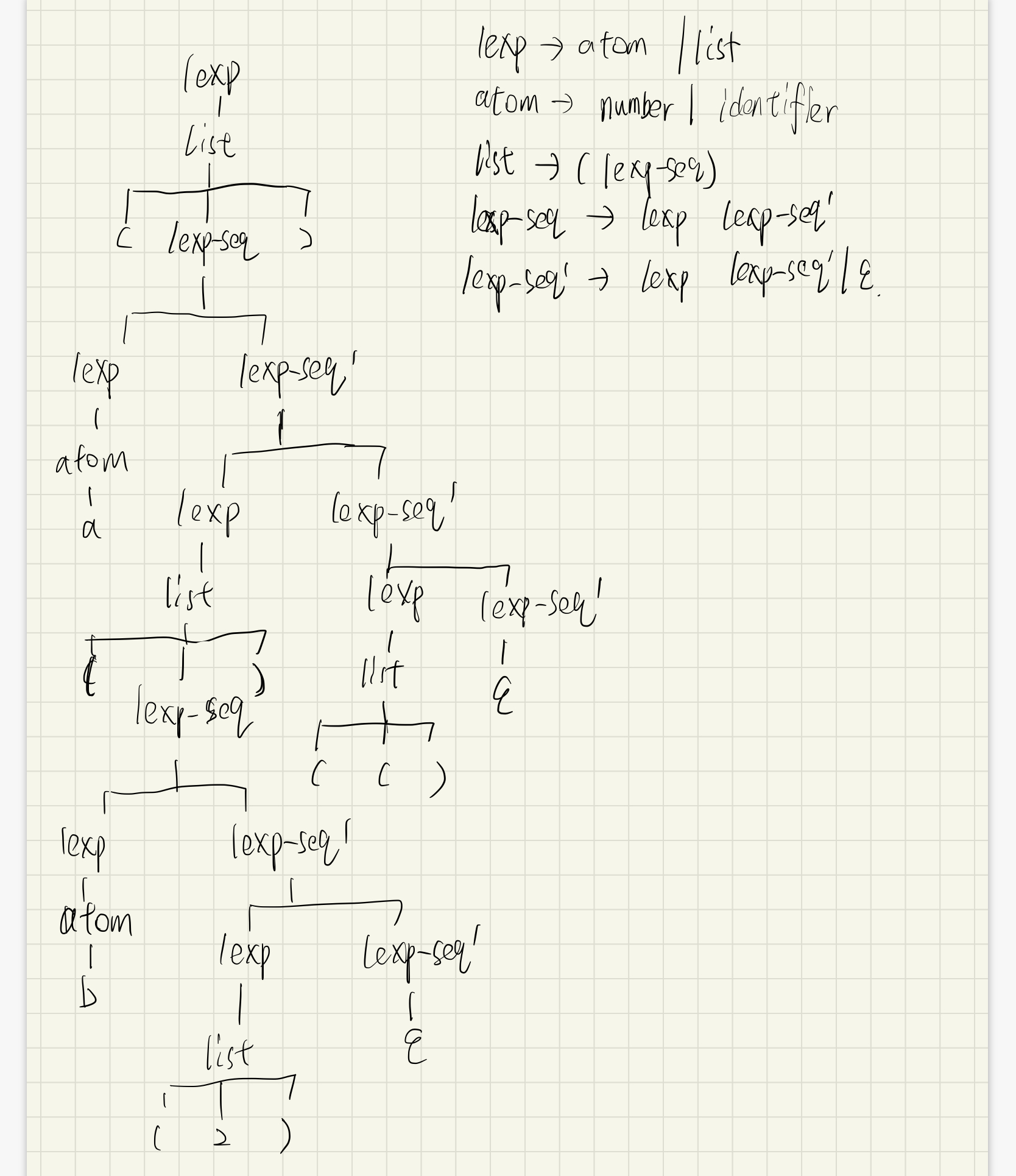
产生式A -> α|β满足下面的条件：

1. 不存在终结符a使得α和β都能够推导出以a开头的串
2. α和β至多有一个能推导出ε
3. 如果β可以广义推导出ε，则First (α)∩Follow( A ) = 空集
4. 如果α可以广义推导出ε，则First (β)∩Follow( A ) = 空集
5. 构造分析表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | number | identifier | ( | ) | $ |
| lexp | lexp -> atom | lexp -> atom | lexp -> list |  |  |
| atom | Atom -> number | Atom -> identifier |  |  |  |
| list |  |  | list -> **(** lexp-seq **)** |  |  |
| lexp-seq | lexp-seq -> lexp lexp-seq’ | lexp-seq -> lexp lexp-seq’ | lexp-seq -> lexp lexp-seq’ |  |  |
| lexp-seq’ | lexp-seq’ -> lexp lexp-seq’ | lexp-seq’ -> lexp lexp-seq’ | lexp-seq’ -> lexp lexp-seq’ | lexp-seq’ -> ε |  |

D-2: 分析树

其中list子树有一定简化



1. 用语法分析器处理(a (b (2)) (c))

代码和结果见《三、实验结果》部分

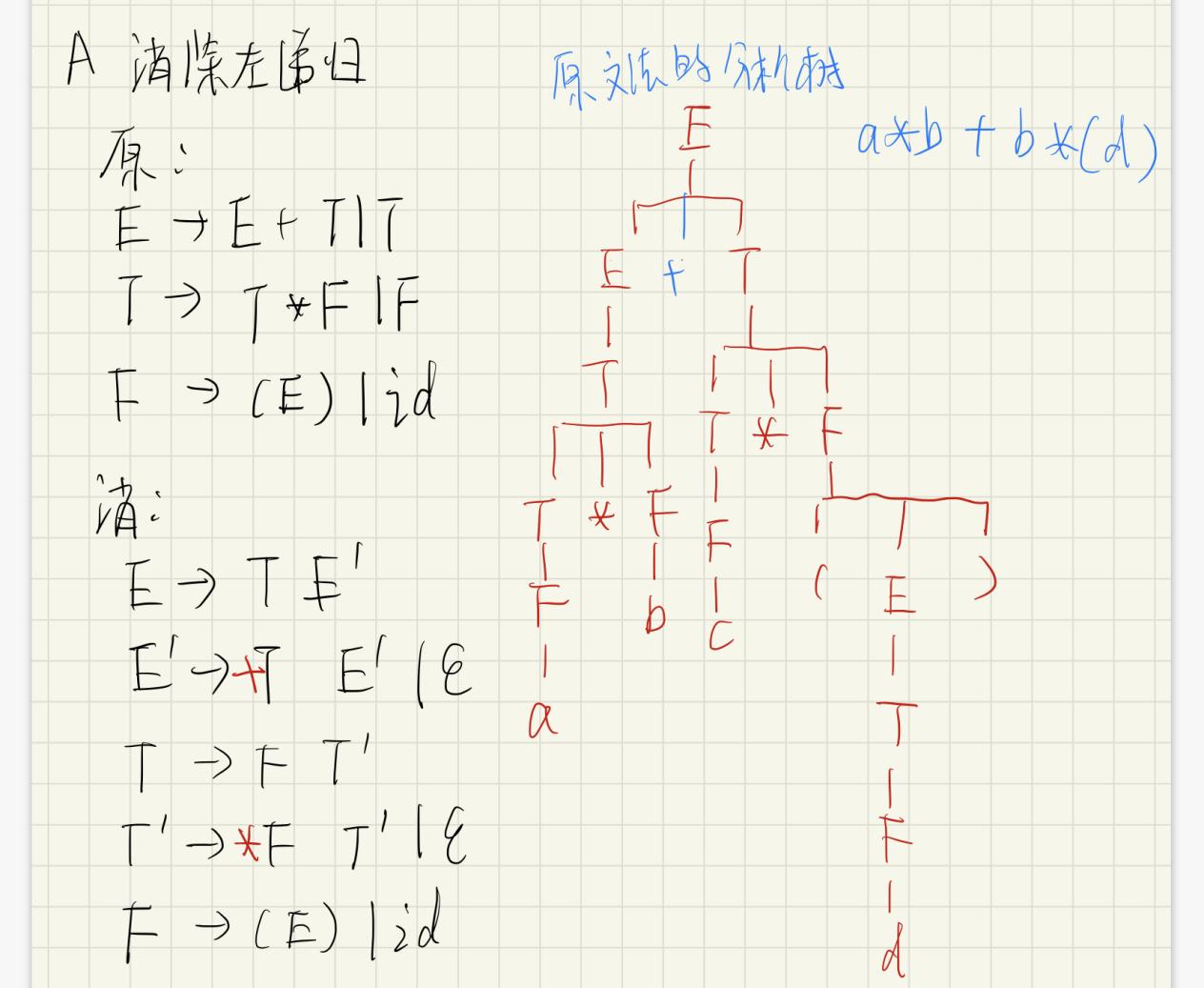
2、考虑简单算术表达式文法G:

E→E + T | T

T→T \* F | F

F→(E) | id

试设计LL(1)分析程序，以对任意输入的符号串进行语法分析。



3、实验具体步骤

已在1、2步中写出

1. 实验结果:
2. 代码
3. 截图
4. 实验结论:

1 、实验结论

（是否能够准确描述实验的结论）

2、分析和总结

1）对输入设计的结论

2）对输出设计的结论

3）对LL(1)分析法的结论

3、对预估问题的结论