重拾此项目时可参考本文档

# 概述

截至2023年4月17日09:18:51，我要暂停GrammarMentor的编写。这是因为还有其他事要做，不得不暂停。为此，也为了一般的说明的目的，我在此记录GrammarMentor目前的情况。

这个GrammarMentor，作用是替代Lex+Yacc，即：实现一个编译器的编译器。当然，通过观察、试用GrammarMentor，也能起到快速深入地学习编译原理的作用。这两个作用是统一的：容易上手，就容易搞明白原理；容易学，就容易上手。

更精确地说，这个GrammarMentor，要实现的是解析器的生成器。任何一个实用的软件系统，都需要先解析某种格式的输入文件，而后才能开展业务活动。GrammarMentor就是为此而作。

各种解析器的工作流程，都有相似之处。换句话说，各种解析器，在抽象的层面上，是相同的。根据面向对象的程序设计思想，这个抽象的共同点，就应当化为一个发挥基础设施作用的类库。这就是bitzhuwei.Compiler。不用bitzhuwei.Format这个名字，是为了突出它运用编译原理的特点。

有了基础设施，就可以做几个简单的例子了。我搜集了ABB、SES、Calc、Step、WRL这几个文法。什么是文法？要解析的格式，来个自我介绍，就是文法。这个自我介绍里，包含了这个格式的全部规则。编写这个格式的解析器，就是用代码实现这个文法。

首先，我根据搜集的资料，手动编写上述文法的解析器。这些解析器，运用了正则表达式、LL(1)、LR(0)、SLR(1)、LALR(1)、LR(1)的原理，这也是今后可以一键生成的部分。对于每种格式，都可以输出它的格式化的输入文件；对于Calc（四则运算），还可以得到计算结果；对于Step和WRL，还可以用OpenGL渲染出三维图形。这是语义部分，与各自的业务功能相关，无法完全一键生成，只能分别手动实现。

然后，整理出Grammar（描述文法的文法），手动实现它，就得到了解析器的生成器。我已经实现了分别根据LL(1)、LR(0)、SLR(1)、LALR(1)、LR(1)的原理一键生成语法分析器的功能，但还没有做完一键生成词法分析器的功能，只好留待将来。

最后，用Grammar解析器（bitzhuwei.GrammarFormat）一键生成我搜集到的所有文法对应的解析器，并与手动编写的代码对比，以验证其正确性。

这样，就连bitzhuwei.GrammarFormat（的词法分析器和语法分析器）都是一键生成的了。手工业进化为重工业。

# 基础设施

解析器的输入数据是`string`，中间经历`Token`、`Node`，最终抽取出业务内容。

|  |
| --- |
|  |

词法分析器`LexicalAnalyzer`根据具体解析器的单词规则，用状态机的原理分析`string`，输出`Token`数组。

语法分析器`LLSyntaxParser`或`LRSyntaxParser`根据具体解析器的文法规则，用LL(1)、LR(0)、SLR(1)、LALR(1)、LR(1)的原理分析`Token`数组，输出`Node`这个树结构。

语义分析器`TExtracter`根据具体解析器的语义规则，按后序优先遍历的方式分析`Node`，输出业务需要的内容。当然，这个语义分析器`TExtracter`是泛型的，因为具体业务的数据结构各不相同，只能手动定义。

# 一键生成

所谓一键生成，就是将`bitzhuwei.GrammarFormat`输出的业务内容`GrammarDraft`，进行一系列算法操作，输出对应的词法分析表、语法分析表和其他杂项信息，这些东西合起来就是`Grammar`的解析器了。因而，我要做的就是实现很多算法。

## 词法分析

一个字符串，如果符合某个正则表达式，它就是某种类型的`Token`。换句话说，我必须解析正则表达式，才能得到词法分析表，才能一键生成词法分析器代码。实际上，我一直在通过肉眼分析正则表达式，手动编写词法分析器。

目前，我整理出了正则表达式的文法（在文件`Regex.st`中），已经能够解析正则表达式了。词法分析、语法分析，语义分析之后，我已得到了ε-NFA（含ε的非确定有限自动机），但还未实现ε-NFA转NFA再转DFA的算法。且`Regex.st`还未支持<Aaa>'Xxx'（在某单词之后有某单词）这样的功能，因此也需要改进。

这是当下最急需完成的功能。否则，就仍然要手写词法分析器。

## 语法分析

我已经实现了一键生成语法分析器的功能，且对LL(1)、LR(0)、SLR(1)、LALR(1)、LR(1)全部支持。

## 语义分析

我已经实现了一键生成文法中非叶结点对应的`class`的功能。这个太简单了，不必在意。

# 其他

## Mermaid

为了画出词法分析表和语法分析表，我找到了这个工具。现在只需很简单的实现`ItoMermaid`接口，就可以得到壮丽的词法分析自动机和语法分析状态机的状态图了。这对于学习和编码，都很有益处。

我在项目中掺入了对C#正则表达式的研究和对OpenGL的研究。

研究正则表达式，是为了一键生成词法分析器。

研究OpenGL，是为了做一个三维图形引擎。这是后话的后话了。