语法分析讨论

技术的简单性

个人认为在调试的时候，自顶向下的程序更好调试。不过这里是指采用递归下降的方式比较好调试。首先，在这次实验当中，自顶向下的代码完全由自己实现，所以出错的话自己心中也会有一点概念大概知道哪里可能出问题。更重要的是，程序如果出错通常可以从调用栈中看出是哪里出错了。而自底向上的程序是由CUP工具生成的，所以在 查看调用栈的时候不太容易看出错误的原因。在实现语法制导翻译的时候，自顶向下通过lookahead可以立刻执行代码，而自下而上需要在规约的时候才执行，当然可以通过占位符来解决问题，但这本身就是增加代码调试难度了。

是否易于表达语义动作

我觉得自顶向下更易于表达语义动作。就像先前说的，自顶向下可以很方便的在符号之间插入代码，而且一定是代码执行完成之后才用执行后面符号相关的程序。而自下而上除非使用占位符，否则不能随意的在符号之间插入代码。

出错恢复

自下而上的出错恢复可以通过向分析表填入相应的error程序，这样在遇到错误的时候，自下而上可以更加精确的实现出错恢复。

递归下降要实现一个较好的出错恢复感觉不是很容易。因为栈不是显式实现的，那么我们对栈的操作就不是很方便了。如果自顶向下也是通过表格实现的话，那么它也可以通过向空白表项填入相应的处理程序来实现一个较好的出错恢复。

递归 VS 表格驱动

自顶向下的分析表大小为$non \times (term + 1)$，其中non为非终结符个数，term为终结符个数，而不同的自下而上有不同的表格大小，但通常要比自顶向下的大。

速度对比

如果考虑上穿插在文法间的代码的话，那么自下而上的效率应该更高一些。因为自顶向下在lookahead看到token之后立刻执行代码，但是自下而上是在遇到了一系列token之后才开始执行。这时如果遇到代码出错的情况，那么自顶向下的程序很肯能已经做了很多无用功。