**对LR语法分析错误恢复的一些思考——规约动作的填入**

以龙书P169的SLR1分析表为例（其实与教材上相同），试图探讨以下几个问题：

1. **在状态3/7/8/9中，规约动作被扩展至一行，这是为什么？有什么好处？**
2. **用另一种方法，比如填入e类信息（已有的或新设计的报错指针e1,e2,e3…）而不进行规约，是否可以？**
3. **什么条件下可以进行这样的扩展？**
4. 首先观察这些状态的特征：**有且只有一个规约动作**（可能含有移进动作，若含有，则这个状态本质上是**没有产生**移进-规约冲突的状态，但是不能含有其他的规约动作）。这暗示了一个含义：无论当前输入字符是否是可进行规约的字符，如果想规约的话，只能用**唯一的**式子来规约。这种状态中的空白格子的含义是：当前输入字符不可能出现在 规约式（产生式）左边 的右侧，这是因为SLR最浅层次的规约要求就是利用 规约式左边 符号的FOLLOW集，而FOLLOW集代表着可能出现在这个非终结符之后的所有终结符。当遇到错误时，说明一个**不可能出现在其之后**的输入符号出现了，例如，符号栈顶为E+E，状态栈栈顶为7，此时遇到了输入字符a，由于分析表的空白，即使进行规约也是无效的（FOLLOW集告诉我们，a永远不可能出现在规约之后符号E的右边）。但是若强行进行规约，未尝不可：首先，当前输入的字符**并不干扰**符号栈内的状态；其次，就算进行了（唯一）规约，a（错误的输入）也绝不可能移入栈中。这是因为规约之后的 **符号和状态** 面对这个a的时候，一定只有两个选择：其一，**再次**进行其他的（唯一）规约，其二，在其他状态中接触到e类报错指针（e1,e2,e3…）。会不会规约之后这个错误就消失了？这是不可能的，因为FOLLOW集是规约条件的本质，这个错误会传递，而不是消失。这样做的好处是显而易见的：**以延迟报错为代价，换取报错种类的减少和集中**。可以说的更本质一些，这些新填入的规约动作，**将所有错误都引到了便于设计的e类错误信息中**（也就是某些状态，这些状态中只出现移进动作，没有规约动作）。
5. 如果想给（上述）这样的状态添加e类报错信息是**困难的**，首先，此类状态（3/7/8/9）的空白格子里并不好用统一的某个e来表征，这是因为此类状态中的错误是广义的“非法”，即出现了FOLLOW集（假设为SLR1）中不存在的字符，这好像是在告诉你：我这边看到了个非法的字符，你再怎么使劲儿推导，你也推不出这个字符来，至于怎么错的，这太宽泛了，我无法具体告诉你。所以量身定做一个e是麻烦的，不如上述的用规约动作延迟报错、将错误引入到那些设计好的e来处理这个错误更精巧。
6. 可以从以上两个问题中明显看出，这个条件是**规约动作的唯一性**。如果一个状态中出现两个或以上不同的规约动作（本质上是**没有产生**规约-规约冲突的状态），这样的扩展没办法进行，因为试图做的规约动作已经不再唯一，而此题并不涉及这样的问题。如果有这样的状态出现，则还需设计相应的e（或者有其他的方法？）。

**一句话小结：利用状态中规约动作的唯一性，将SLR（或更高层的LR分析）退化至LR0状态。以延迟报错的代价，换取程序设计上的简单和精巧。**

PS：以上所有观点仅为个人分析与猜想，如有谬误，还请张老师谅解！