FLA-2022 实验报告

201502009 孙智鑫

分析与设计思路

因为图灵机的构成主要有两个部分，一个是解析器，一个是模拟器。所以我写了两个类，一个是`turing_machine`类，用于把`.tm`文件解析成对应的图灵机规则，另一个是`simulator`类，用于把解析好的图灵机在给定的输入上运行，并且打印结果。自然的，`turing_machine`类是`simulator`类的成员。

关于解析器，我使用了STL里的`set`去存储图灵机对应的集合，比如Q，S，G等等。对于转移函数，我使用了一个`set` + 结构体去存储对应的转移函数。本来想用的是`unordered_set`遇到的麻烦是我的`unordered_set`不知道为什么一直报初始化函数的错，所以最后给我的结构体根据其所在的行号定义了一个比较函数，使用了`set`作为存储方式。

解析器的报错就是在解析时遇到意料外情况就报错。我实现的主要有检查`#`,`,`,`{`,`}`是否缺失，集合名字是否正确，集合内元素是否在语法限定的ASCII码范围内，`delta`转移函数是否有空格缺失。在`verbose`模式下，程序会报出哪一行出错，普通就只有`syntax error`。

关于模拟器，除了成员变量`turing_machine`外，还有纸带的模拟，读写头位置的模拟，以及图灵机当前状态。在实现上，纸带我是用`deque`模拟的。由于`deque`在前向插入时原有的元素下标都会加一位，所以还需要维护一个`vector`，用于记录纸带`0`位置的信息。在模拟之前，模拟器会对输入进行合法性检查，主要就是比对输入符号是否属于图灵机输入集合。若不符合就直接illegal input，并退出。转移函数就比较简单。只要收集每个读头的信息，加上当前状态，就可以在转移函数里搜索到相应的规则，从而进行转移。若搜不到就直接打印第一条带内容，退出程序。

在verbose模式下，`simulator`的输入正确性检查会在检测到不合法输入时打印不合法输入位置。若合法，则会在每次转移时打印图灵机瞬时描述。为了使index和tape内容对齐，我使用了一个vector来记录每个index打印时的起始位置，因为index两位数时位置要加3，而一位数时只需要加2。

对于命令行参数的问题，我的处理是参数不合法或者有`-h | --help`时直接打出`usage: turing [-v|--verbose] [-h|--help] <tm> <input>`这一行字。在其他情况正常运行。

关于任务三的第一个任务，我是通过把最后一个元素移动到开头，并把整体全部右移一位实现的，只使用了一条纸带。

第二个任务用了四条纸带。第一条存储输入，第二条为第一个乘数，第三条为第二个乘数，第四条为乘法结果，通过计算乘法，并且比较第一条纸带和第四条纸带的结果决定是否继续增大乘数，还是擦除第一条纸带打印 true 或 false。

实验完成度

如在上一部分中的内容所说，基本上完成了要求的所有内容。

遇到的问题及解决方案

问题可能主要集中在`simulator`部分，关于如何打印`tape`，我一开始的`start`和`end`在`tape`长度为0时老是找不好位置，后来加了一个关于`start`和`end`的`start < end`检测就好了

总结感想

这次实验完成的还算迅速，整体也不算太难，不过做完后感觉对图灵机的运行方式又有了更深的理解，看着自己设计的图灵机规则在自己写的模拟器上成功运行也很有成就感，总之感觉写的很快乐。

对课程设计的意见和建议

图灵机真好玩，嘿嘿嘿。