图灵机实验报告

匡亚明学院
181240035
刘春旭

一、实验分析

实验中需要能够实现一个模拟图灵机的程序，即对于每一步都能给出当前的图灵机状态。一共有三个任务需要完成：

- 解析器：能够正确解析读入的图灵机程序；
- 模拟器：能够判断输入串的合法性以及模拟运行合法的输入串；
- 设计多带图灵机程序；

二、设计思路

前两个任务是coding的部分，下面尝试对它们进行任务的拆分；

1. 解析器：基本就是读入文件后利用各种数据结构把关键数据存起来，下面讲一些自己实现中比较有趣的思路：
    - 使用`set`来储存状态集Q、输入符号集S、纸带符号集G和终极状态集F，这样可以比较方便的对于某个特定元素快速查询它是否属于这个集合，比如`Q.count(q) != 1`则说明状态q不属于状态集Q；
    - 包括一个`trim()`函数，可以去掉读入文件中每一行开头和结尾的空格，并且可以去掉句尾的注释；
    - 每个转移函数都用一个结构体来储存，结构体有5个`string`，分别对应转移函数中的五个成分；最后用一个`vector`数组来存起这些转移函数。
2. 模拟器：这里的两个模式基本相同，只需要分析`--verbose`模式即可：
    1. 检查输入是否合法：

        按照上述描述依次判断每一个输入的符号即可，空输入在下面进行描述；

    2. 初始化操作：

        按照上一步任务中读入的纸带数量N，初始化对应数目的纸带。每一个纸带都由一个`Tape`结构体来表示，其中对于第0条纸带，由于需要在这上面储存输入字符串，所以单独拿出来处理；其余的纸带都初始化成空的纸带即可；这里我的实现保证了每条纸带的最后一定会存在一个空格字符`B`，所以对于空输入也有鲁棒性。并且初始化当前的状态为`q0`

        纸带结构如下：

```
struct Tape{
  int ptr;             // 当前指针所指
  int start_index;     // 第一个不为空格的字符角标
  int end_index;       // 最后一个不为空格的字符角标
  vector<char> w;      // 纸带上的字符串
  vector<int>  index; // 纸袋上的角标
};
```

3. 进入循环：

   循环终止的条件是当前的状态属于终止状态集，或者是循环执行过程中没有找到合适的转移函数：

   1. 展示当前纸带状态，利用几个循环完成纸带的展示；利用数字位数判断的函数`digit()`判断输出数字的位数，来实现输出的纸带内容和角标的左对齐；
   2. 根据目前的状态和指针找到合适的转移函数，如果没有合适的转移函数那就终止循环，并展示结果；
   3. 根据找到的转移函数对所有纸带进行修改：
      1. 首先更新`tape[k].ptr`，根据转移函数中的方向来确定第`k`条纸带的指针是应该$+1$、-1还是不动；
      2. 如果`tapes[k].ptr`已经指到了当前`tapes[k].w`的最后一个字符，那么向最后再插入一个空格字符`B`，保持循环不变式的正确性（上方荧光笔高亮的位置）；
      3. 接着统计B的位置，更新`tape[k].start_index`和`tape[k].end_index`；
      4. 如果`tapes[k].ptr`指到了`-1`的位置，那么应该向`tapes[k].w`最前面插入一个空格字符`B`，并且在`tapes[k].index`最前面也插入正确的角标；然后再将`tape[k]`的`start_index`、`end_index`、`ptr`都$+1$（向后移1）；
4. 退出循环，展示结果即可（这里我还展示了所有纸带的最终状态）；

三、多带图灵程序的设计

1. $L_1 = \{a^i b^j a^i b^j | i, j > 0\}$

   使用两条纸带实现：

   基本思路是把一个输入拆成两个$a^i b^j$，把前面的一个$a^i b^j$复制到第二条纸带，然后和后面的$a^i b^j$进行比较，如果完全相同，则接受，否则不接受；具体状态定义见`case1.tm`中的注释说明。

2. $L_2 = \{1^m \times 1^n = 1^{mn} | m, n \in N^+\}$

   使用两条纸带实现：

基本思路是以1^m为蓝本，每次读到1^n中的1时都在第二条纸带上复制一遍1^m，最后把等号后面的字符串与第二条纸带上复制好的字符串进行比较，如果完全相同则接受，否则不接受；具体状态定义见`case2.tm`中的注释说明。

四、实验完成度

上述实验已经全部完成

五、实验中遇到的问题及解决方案

Q：对于`vector<Tape>`这个结构，`Tape`中还有变长的`vector`数组，应该如何处理?

A：这里当想要插入更多元素的时候，一定要用`insert()`或者是`push_back()`，只有它们可以对`vector`的大小进行调整，否则直接赋值是UB，有时会成功（内存分配的连续性），有时就会失败。

Q：如何设计`Tape`结构体?

A：就像上面提到的那样设计，多使用一些变量可以简化思考量；现在的设计是最直观并且符合直觉的。（之前考虑过很多奇怪的想法，比如考虑用head记录下当前指针并且用remaining记录当前字符串剩余长度，打印时再判断该从哪里打印到哪里，但是这样太不直观了）

六、想说的话

1. 深度好课，感谢卜老师和助教老师；
2. 这个实验对理解图灵机有很大贡献，希望能一直保留；
3. 有点小遗憾就是没太利用好C++的特性写出这份代码，C的味道还是重了些；
4. 如果把这门课的学分再提高，或许可以考虑把FA和PDA做成编程小实验，肯定对理解更加有帮助；
5. 错误编号以及一部分使用说明均在`README.md`里。