# PTC Lab

# Lab Report

**Name：王辛有**

**STUDENT ID：171860691**

# 一、实验分析及设计思路

本实验要求设计一个图灵机模拟器。根据给定的图灵机文件输入，构建出图灵机，并可以根据输入进行正确的图灵机运行的模拟。

程序的架构：

```
|— turing
    |— 头文件     //包含图灵机的数据结构，及所有需要用到的函数
        |— Machine.h
    |— 源文件
        |— MachineInit.cpp    //图灵机的构建
        |— MachineRun.cpp         //图灵机的模拟运行
        |— main.cpp
```

先介绍图灵机的基本结构：

```
class Turing_Machine {
private:
    vector<string> states;//状态集
    vector<char> alphabets;//输入字母表
    vector<char> tape;//纸带符号集
    string start_state;//初始状态
    string blank;//空格符号
    vector<string> finalstates;//终结符号集
    int tape_num;//纸袋数
    map<pair<string, string>, TransResults> trans;//转移函数
public:
```

用一个类来表示图灵机。其中每个成员的含义已通过注释说明。

对转移函数进行解释：

因为一个 ID（即当前状态，和当前读写头的指向字符），可以唯一决定一个转移函数，所以决定采用 map 数据结构来表示转移函数。

其中 Key 是一个 pair 二元组，代表的含义是<当前状态,当前读写头所指向的字符>.比如有两个纸带，第 0 个纸带的读写头指向的字符是'0'，第 1 个纸带的读写头指向的字符是'1'，当前状态处

于 start，则这个二元组则表示为<”start”,”01”>。

而 Value 则是一个结构体，结构体的含义表示转移的结果。结构体的定义如下：

```
struct TransResults {
    string rewrite;//对读写头指向字符的改写
    string l_or_r;//读写头向左移动还是向右移动
    string nextstate;//下一个状态
};
```

每个域的含义已通过注释说明。

所以，在图灵机的运行过程中，只需要构造一个 pair 二元组，即可得到相应的转移函数。

而对于带*通配符的转移函数，则会在下面进行详细说明。

下面针对逐个任务进行分析。

## 1. 任务一：多带图灵机程序解析器

由于是通过读文件的方式来获得图灵机的信息，所以采取逐行读取的方式。将读到的每一行传到 create 函数中进行处理。

1） `if (line[0] != '#'&&line[0] != ';')`

该行第一个字符不是#或者;，说明只可能是转移函数。所以会直接跳转到转移函数的处理函数

2） `if (line[i] == '#')`

接下来只可能是状态集、输入符号集、纸带符号集、初始状态、空格符号、终结状态集、纸袋数的信息。根据不同的情况，作不同的处理。（主要是字符串的截取）

## 2. 任务二：多带图灵机程序模拟器

我设置了用于打印当前 ID 信息的结构体，其定义如下：

```
//用于打印ID快照
struct print {
    int num;//读写头的数量
    int step;//当前的步数
    int** index;//纸带的索引
    char** tape;//表示纸带上的符号
    int* head;//head有tape_num个元素
    string state;//当前的状态
};
```

其中 index，tape 是动态二维数组，根据读写头的数量决定申请的空间。

根据要求，先初始化 ID 的信息，把 0 纸带的内容拷贝为输入，其余纸带内容为空，然后通过 find_trans 函数，找到当前 ID 下对应的转移函数。

下面就来分析一下 find_trans 的机制。

先根据 map 结构自带的 find 函数，查找有没有匹配的转移函数。如果找到了，就直接返回该转移函数。如果没有找到，则有两种情况：

1. 可能存在带有通配符*的转移函数与之匹配

2. 没有转移函数与之匹配

对于 1 情况，只需要遍历整个 map 结构的 Key，即遍历 map 中的每一个二元组<state,current>，对于 state 与当前状态相同的二元组，取出它的 current，然后逐个字符与当前读写头所指向的字符比较，如果匹配成功，统计其中*的数量，选择*数量最小的那个二元组。这个二元组对应的 TransResult 就是我们想要

的转移函数的结果。

而对于 2 情况，如果没有转移函数与之匹配，返回 error，即到达停机状态。

得到了转移函数的结果，调用 modify_id()函数对当前 ID 进行修改。

如果得到的结果是 error，则打印出 0 纸带的内容，并结束程序。

3. 任务三：多带图灵机程序

$$1.\{0^k \mid k\text{是一个斐波那契数}\}$$
$$2.\{ww|w \in \{0,1\}^*\}$$

对于 这两道题的整体思路和转移函数运行我已经详细地写在了它们对应的 tm 文件中，此处便不再赘述。

# 二、实验完成度

完成了本实验要求的全部内容。

# 三、实验中遇到的问题及解决方案

在设计转移函数的数据结构时考虑了一段时间。理想当中的数据结构应该具有查找方便，时间复杂度相对较低等优点。于是选择了 map 数据结构。但是对于带有通配符的转移函数 map 又没办法处理，于是无奈之下，只好对于带有通配符的转移函数采取遍历搜索法。考虑到带通配符的转移函数毕竟只是部分，所以所增加的时间开销也并不显著，可以接受。

# 四、编译运行说明

进入 MyProject 文件夹，执行下列命令

**g++ main.cpp MachineInit.cpp MachineRun.cpp -o turing**

即可完成编译。

# 五、实验感想及建议

我对于本次实验的任务三比较欣赏。这两道题难度适中，既考察了课上所讲过的知识，同时也相当于给了我们两个测试用例，帮助我们自己检测一下前面所写的图灵机模拟器是否存在问题（我在调试的时候出现的问题基本都是通过自己编写图灵机文件测试运行才找到的）。

我对本实验的建议：

明年的实验可以继续沿用这个形式，同时我觉得可以加一个 bonus：在实现多纸带的同时，实现多磁道，从而达到更真实的模拟。