实验报告

181240078|张思拓
Dec 15, 2020

实验完成度

完整完成了全部实验内容，具体地：

- 实现一个Makefile编译工具，方便记录git信息以及编译和测试；
- 实现了多带图灵机程序解析器，尽可能完备的进行了错误处理的实现，并且在指定`-v|--verbose`选项时在报错的同时会给出相应的不符合语法的图灵机程序片段（错误提示）；
- 完成了多带图灵机模拟器，可以判断输入串的合法性，并模拟输入串的图灵机执行过程，返回最后的执行结果且输出第一条纸带上的内容。当指定`-v|--verbose`选项时，会详细给出输入错误以及每个转移时刻图灵机的瞬时描述。
- 实现了两个多带图灵机程序`case1.tm`和`case2.tm`，分别判定$L_1 = \{a^ib^ja^ib^j|i,j>0\}$和$L_2 = \{1^m \times 1^n = 1^{mn}|m,n \in \mathbb{N}^+\}$，并在自己实现的模拟器上进行了详尽的测试。

分析与设计思路

命令行参数解析

首先，程序需要按照命令行的参数和传入的文件来执行后续操作，因此整体的设计从命令行`parse_args`开始。使用`getopt_long`函数可以获得命令行的`-h|--help,-v|–verbose`选项，之后剩下的参数分别为待解析的图灵机程序文件以及输入的字符串，在参数个数不够的时候报错并提示命令行工具的正确使用方法。

解析器

解析器的设计需要严格参照**图灵机程序语法**来进行，目标是从`.tm`程序中获得图灵机形式化描述$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$这七部分的定义，并存储下来。首先要打开`.tm`文件，然后逐行读入，并删去注释部分以及忽略空行以便后续处理。

- 接下来根据读入的行中第一个字符是否是“#”来判断该行是否为转移函数的描述。若不是，则使用**正则表达式**匹配相应的状态、集合以及参数的设定，上述匹配使用宏定义`Assert`来完成，若匹配不成功，则说明输入的图灵机程序不符合语法，转入错误处理。在错误处理中会输出`syntax error`，返回exit code以及给出该错误行的描述。若匹配成功则使用迭代器获取对应位置的字符串或参数值。
- 若是转移函数，则同样地，使用迭代器根据空格的分割获取对应位置的字符串，并判定旧状态和新状态是否在状态集`Q`中。

模拟器

对于模拟器的实现，我们需要维护`n`个纸带的全部状态信息。为此我创建了`Tape`类，维护的信息有纸带上的内容、读写头位置，以及纸带最左边的索引数字。因为运行过程中纸带上会写入和删除符号，长度会动态变化，因此我使用了`STL string`来模拟这一**无限长**的纸带。同时为模拟图灵机每运行一步行为，我创建了move函数，会根据输入的当前状态以及读写头所指符号，通过遍历所有可能的状态转移函数，找到与当前状态匹配的转移函数，相应改变纸带记录的信息。重点关注两种边界情况，即读写头到达了当前类记录纸带的有限位置的边界，需要相应的增加string的长度，并可能修改最左边的索引数字（因为Tape类只记录访问过的纸带上的有限位置）。

verbose模式的实现较为繁琐，主要是通过`Log`宏定义完成。`Log`可以代替printf的功能，仅在传入verbose参数的时候在执行printf输出任务。而对于纸带的内容打印，因为只需打印纸带上最左非空格符号到最右非空格符号之间的符号及对应索引，因此考虑使用string类的内置函数`find_first_not_of`和`find_last_not_of`来获取上述纸带的内容。而控制格式和对齐则是利用`printf`的格式化打印方法，记录应打印字符数（根据`Index`值来确定），控制左/右对齐来实现。

多带图灵机程序

第一个语言为：$L_1 = \{a^i b^j a^i b^j | i, j > 0\}$，观察可以发现满足该语言的字符串由前后相同的两部分组成，且都是$a^i b^j$的形式。因此设计思路是使用两条纸带，开始时输入在第一条纸带上，将能够按$a^i b^j$形式匹配到的最长的字符串剪切到第二条纸带上。然后两条纸带同步运行比较内容是否相等，来确定输入串是否在L_1中。

第二个语言为：$L_2 = \{1^m \times 1^n = 1^{mn} | m, n \in \mathbb{N}^+\}$，可以看出，符合该语言的字符串应该具有三个部分，每个部分都是由1组成的字符串序列，且他们的长度（x,y,z）满足一种乘积关系$x \times y = z$.因此思路是使用三条纸带，经过剪切和处理后分别记录上述三个序列。然后比较是否满足乘积关系的执行方式类似于**算盘**的工作方式，一个纸带代表低位，一个代表高位。低位的纸带每遍历一遍，高位的纸带就移动一位。最后判断是否能刚好消耗完第三个序列1^z，来确定输入串是否在L_1中。

具体的实现细节，（如：第一条纸带上打印`true/false`，且不出现其他内容）详见`programs/case1.tm`和`programs/case2.tm`

总结感想

完成这次大作业我收获很大，首先是加深了对于图灵机的理解。通过完整实现了图灵机，我清楚了一个图灵机在形式化设定之后，接受一个字符串之后它的内部到底发生了什么。同时我也真切感受到了要保证一个程序的鲁棒性和完备性的困难之处（因为不能保证用户的输入都是严格合法的），因此图灵机需要严格的形式化定义。

意见方面是我觉得以后可以增加一点点实验框架（例如Makefile，git记录等等）