Formal Languages and Automata(FLA-2022-Autumn) Project Report

> 实验人员 : 201220096 钟亚晨
> 实验时间 : 2022/12/30
> 实验环境 : Win10 + VSC + WSL2(Ubuntu20.04 LTS、g++9.3.0, gcc9.3.0)

实验成果/实验完成度

- [x] 完成了所有要求的必做功能、选做功能，并自行编写了一系列的示例验证正确性(均可通过);
- [x] 追加了一些错误处理，以提升鲁棒性;
- [x] 完成了本篇实验报告;

分析与设计

Task-01:图灵机程序解析器

该模块的目的就在于:

1. 读取并检查.tm文件中的内容，如果出现不合规范的则打印错误信息并返回exit code退出程序;
2. 如果所有内容都符合规范，则将其转化为内存中的"图灵机"程序实体;

该模块的实现在: `./turing-project/parser.cpp`

实现思路为:

1. 使用C++的regex正则库对.tm程序的语法规范进行检查、匹配，以及抽取出可用片段用于构建程序实体;
2. 使用C++的OOP、STL来定义"图灵机"类，其数据成员包括了多带图灵机定义的八个部分:
    - 一个string向量，保存所有的**状态名**
    - 一个string向量，作为**输入符号表**
    - 一个string向量，作为**纸袋符号表**
    - 一个string变量，存储**起始状态名**
    - 一个string变量，存储**空白符号**
    - 一个string向量，存储**终止状态名**
    - 一个int变量，存储**纸带条数**
    - 一个以string为键值、二维string矩阵为值的哈希表，存储**迁移函数**
3. 迁移函数使用表格的方式进行存储，这也是图灵机的一种典型表现形式，后续对多带图灵机的模拟就通过不断查表进行实现;

Task-02:图灵机程序模拟器

该模块的目的在于:

1. 从图灵机解析器中获取到"多带图灵机"程序实体，从输入中获得待检测串，而后将待检测串输入到"多带图灵机"中模拟其运行;
2. 如果输入串为非法串，而输出对应的错误信息;
3. 在具备"-v"参数的条件下，每一个单步运行后输出当前图灵机各个读写头存在的位置及纸带情况;

该模块的实现在:`./turing-project/simulator.cpp`

实现思路为:

1. 模拟部分的一个难点在于对"双向无穷纸带"的模拟，这里使用了STL里的deque作为栈来进行"双栈模拟"，一方面易于遍历获取各个元素，另一方面可作为栈进行使用，其结构及运行边界如下:

```
Structure:

|<-head1/bottom1 deque1/stack1 tail1/top1->| |<-tail2/top2 deque2/stack2 head2/bottom2->|
<-Left                                        ↑                                    Right->
                                        read-write head

--------------------------------------------------------------------------------------------

How to work:
由于需要对下标也一同进行处理，因此每个双栈模拟的纸带又对应了另一对双栈模拟对应的下标序列（也可视为一条纸带），即四个栈模拟一个纸带

Basis:(初始状态)
1. 将输入串从尾到首压入第一个纸带对应的stack2中，此时deque2的尾部为输入串首个字符，即读写头指向输入串首个字符，且已将输入串写入到了第一个纸带上;
2. 将其它纸带对应的stack2中压入空符号_，如果输入串为空，则对第一个纸带也这样做;
3. 记录下标的纸带也需要对应进行操作;

Induction:(状态迁移)
每次图灵机的状态迁移包含三部分:
1. 将各个纸带读写头所指内容依次组合起来，获得当前符号串，结合当前状态进行查表，找到对应的迁移函数;
2. 根据查找到的迁移函数(未查找到则停机)，更改当前图灵机所处状态以及复写各个纸带当前读写头下的内容;
3. 根据查找到的迁移函数移动各个读写头:
    - 向左移动:即将stack1尾部pop出，并将其压入到stack2尾部;如果stack1为空(到达非空最左端),则直接向stack2中压入空白符;
    - 向右移动:即将stack2尾部pop出，并将其压入到stack1尾部;如果操作后stack2为空(到达非空最右端，不能使指针悬空)，则向stack2中补充一个空白符;
    - 不移动:什么也不做即可;
另外，为了方便不打印出多余空格，在完成上述3步后，还需删除掉deque1、deque2头部的连续空白符;
```

Task-03:图灵机程序

Task-03.01:循环右移的.tm程序

1. 使用2条纸带的多带图灵机;
2. 读写头1移动到纸带1最右端非空字符，并将该字符拷贝到纸带2上，纸带2读写头右移一格;
3. 读写头1移回到纸带1最左端，之后读写头1、2一同向右移动，将读写头1指向的内容复制到读写头2所指单元格，直至读写头1指向空白符;
4. 将读写头1、2均向左移动到最左端非空字符;
5. 同步向右移动，在读写头1不指向空字符的前提下，将读写头2所指内容拷贝到纸带1上，直至读写头1指向空字符时结束;
6. 如果一开始就遇到空字符，则直接到终止态停机，作为输出;

Task-03.02:判断是否为完全平方个1的.tm程序

1. 主要利用到等差数列的性质，任何一个非零完全平方数都可表达为"以1为首项，以2为公差"的等差数列之和，而对于零(也即输入为空串)，进行处理，直接进入终止态即可;
2. 使用2条纸带的多带图灵机;
    - 纸带1用于存储输入串，及输出结果
    - 纸带2用于进行计数
3. 初始时，先在纸带2上写一个1，且不移动读写头2;
4. 此后，不断重复循环:读写头1和读写头2在同时指向1的前提下同步右移，如果:
    - 读写头1指向1，而读写头2指向空白符，则再向纸带2的1序列后追加2个1，并将读写头2指向最左端非空字符，再进入下一轮循环;
    - 读写头2指向1，而读写头1指向空白符，则进入拒绝状态，清空纸带1内容，并打上false
    - 读写头1、2均指向空白符，则进入接受状态，清空纸带1内容，并打上true

Task-Other:其它事宜

- 在图灵机解析模块中追加了下述检查，以保证程序鲁棒性:
    1. 检查每个迁移函数中的新旧状态是否处于状态集中;
    2. 检查每个迁移函数中的新旧符号序列长度是否与纸带数目相等;
- 使用了下述C++内容进行编程:
    1. fprintf函数，指定输出内容到stdout/stderr;
    2. STL中的unordered_map、unordered_set、vector、string等类型，提升效率并保证稳定性;
    3. regex库中的regex_match等内容，进行正则匹配、搜索以检查.tm内容和提取有效片段;

实验中遇到的问题及解决方案

Probelm01: 在Win10环境下使用VSC编写的.tm程序，放入到WSL2(Ubuntu20.04)中编译运行后，无论如何都无法被所编写的图灵机解析器正确解析。

Solution01:

1. 最先是比对默认框架下的.tm文件内容和自己的.tm文件内容，发现每行末尾总会出现多余的不可见符号，从而导致正则匹配失败报错。而后遂使用WSL2(Ubuntu20.04)下的vim直接编写.tm文件，顺利通过解析与模拟运

行;
2. 后由于仍未弄清背后原因，在群中提问，得到同学们的解答，得知在Win10下未对VSC进行合理设置，导致每次按下回车键时会使用`\r\n`进行换行，与Linux下的`\n`换行有所出入，从而获知多出的不可见符号就是`\r`，后经验证也确实如此。并且也注意到了VSC中对编码、回车LF的设置。

总结感想

1. 对于图灵机长什么样、如何工作、如何定义、如何设计图灵机程序、多带图灵机这一系列的tricks对于图灵机程序设计的便利性有了更为深刻、真切的认识;
2. 在开发过程中遇到了使用跨平台的工具的编码问题，增加了跨平台、跨环境开发中的注意点;
3. 首次使用Win10配合WSL2进行开发，很喜欢WSL2的兼容性和便利性;

对课程和实验的意见与建议

1. 实验量相较于隔壁编译原理课程要小很多很多，事实上有许多可以拓展之处，并且该部分的代码可以复用到编译原理课程当中，比如:
    - 在学习FA时，可以考虑设计一套实验，开发一个类似Bison的组件，输入需要的正则表达式，自动生成对应的FA程序，并提供相关对外接口;