NJU-FLA-2023FALL-PROJ

Standard Turing Machine Description Language
2023秋FLA大作业
姓名:闻嘉迅
学号:211220091
日期:2024.1.2(最后修改)

编译与运行

1. 在含有`CMakeLists.txt`的文件夹下，使用指令`cmake -B build`;
2. 在含有`CMakeLists.txt`的文件夹下，使用指令`cd ./build; make`。

测试框架将使用上述方法对工程进行编译。执行完毕后，在工程根目录下应当出现`/bin`文件夹，其中含有可执行文件`turing`。

实验完成度

(自认为)完成了所有实验要求.

文件结构

- turing-project: 源代码文件夹
  - main.cpp: 主函数
  - ReadTMfile: 图灵机解析器文件夹
    - ReadTMfile.cpp: 图灵机解析器类的实现
    - ReadTMfile.h: 图灵机解析器类的定义
  - TuringMachine: 图灵机模拟器文件夹
    - TuringMachine.cpp: 图灵机模拟器类的实现
    - TuringMachine.cpp: 图灵机模拟器类的定义

实现思路

主函数部分

首先根据命令行输入决定运行模式,然后通过解析器读取图灵机文件,再根据读取的数据对模拟器进行初始化,最后得到运行结果.

图灵机解析器

实现了一个图灵机文件解析器类 `ReadTMfile`，用于解析图灵机描述文件。下面是对该类的主要分析：

成员函数解析：

1. **`readrule` 函数**：
    - 参数：`string str` 表示要分割的字符串，`string split` 表示分隔符。
    - 返回值：`vector<string>`，表示分割后的子字符串数组。
    - 功能：根据给定的分隔符，将输入字符串分割成子字符串，并存储在一个 vector 中。
2. **`readline` 函数**：
    - 参数：`char *buf`，表示用于存储读取的一行内容的字符数组。
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取一行内容，排除空行和注释行。
3. **`readStateSet` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取图灵机的状态集合。
4. **`readCharSet` 函数**：
    - 参数：`char mode`，表示读取字符集合的类型（输入符号集合或图灵机字符集合）。
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取字符集合。
5. **`readStartState` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取图灵机的初始状态。
6. **`readBlankChar` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取图灵机的空白符。
7. **`readFinalSet` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取图灵机的最终状态集合。
8. **`readTapeNum` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取图灵机的磁带数目。
9. **`pushTransRule` 函数**：
    - 返回值：`bool`，表示读取是否成功。
    - 功能：从文件中读取并压入转移规则。
10. **`checkInput` 函数**：
    - 返回值：`bool`，表示输入是否有效。
    - 功能：检查输入字符串是否符合图灵机的输入符号集合。
11. **`checkError` 函数**：
    - 返回值：`int`，表示错误代码。
    - 功能：检查文件解析过程中是否出现错误，并输出错误信息。

12. **`printInvInput` 函数**：
    - 功能：输出无效输入的错误信息。
13. **构造函数 `ReadTMfile`**：
    - 参数：`const char * filename`，表示图灵机描述文件的文件名；`const char * input`，表示输入字符串。
    - 功能：初始化图灵机文件解析器，打开文件，并进行图灵机描述文件的解析。

成员变量解析：

1. `FILE *file`：用于存储打开的文件指针。
2. `int errorno`：用于存储错误代码。
3. `int line`：用于记录当前读取的行号。
4. `set<char> Gset`：用于存储图灵机字符集合。
5. `set<char> Sset`：用于存储输入符号集合。
6. `set<string> STset`：用于存储图灵机状态集合。
7. `vector<string> stateset`：用于存储图灵机状态集合的字符串形式。
8. `string start`：用于存储图灵机的初始状态。
9. `char blankchar`：用于存储图灵机的空白符。
10. `vector<string> finalset`：用于存储图灵机最终状态集合的字符串形式。
11. `vector<queue<string>> rulelst`：用于存储转移规则。
12. `int tapenum`：用于存储图灵机的磁带数目。
13. `string input`：用于存储输入字符串。
14. `int step`：用于记录输入字符串的无效位置。

图灵机模拟器

实现了一个简单的图灵机模拟器类 `TuringMachine`，用于模拟运行图灵机。

主要功能和结构分析：

1. **构造函数 `TuringMachine`**：
    - 参数：接受图灵机的初始状态集合、最终状态集合、初始状态、空白符、磁带数目、是否打印模拟过程。
    - 功能：根据传入的参数初始化图灵机的状态、规则等信息。
2. **`loadTransRule` 函数**：
    - 参数：接受转移规则的字符串数组。
    - 功能：解析转移规则并加载到相应的状态中。
3. **`getcharlst` 函数**：
    - 返回值：当前磁带上的字符序列。
    - 功能：获取当前磁带上的字符序列。
4. **`updatecharlst` 函数**：
    - 参数：新字符序列、磁带移动方向。
    - 功能：根据转移规则更新磁带上的字符序列。

5. **`statetrans` 函数**：
    - 返回值：表示是否成功进行状态转移。
    - 功能：根据当前状态、磁带上的字符序列，执行状态转移操作。
6. **`reset` 函数**：
    - 功能：重置图灵机的状态、计数器、是否接受等信息。
7. **`runTuringMachine` 函数**：
    - 参数：输入字符串。
    - 返回值：表示是否接受输入字符串。
    - 功能：运行图灵机，模拟图灵机的工作过程。
8. **`printstep` 函数**：
    - 功能：在每一步模拟之后打印当前的状态、磁带内容、磁头位置等信息。
9. **`printfinal` 函数**：
    - 功能：在模拟结束后打印最终的结果，包括接受状态、磁带内容等信息。
10. **`printinput` 函数**：
    - 功能：在开始模拟前打印输入信息。

成员变量：

1. `nowstate`：当前图灵机的状态。
2. `start`：图灵机的初始状态。
3. `blankchar`：图灵机的空白符。
4. `tapenum`：磁带的数目。
5. `accept`：表示是否接受输入。
6. `print`：表示是否打印模拟过程。
7. `step`：模拟的步数计数器。
8. `statelst`：存储图灵机状态的 map。
9. `tapelst`：存储磁带上的字符序列的 vector。
10. `tapeidx`：存储每个磁带的磁头位置的 vector。
11. `offset`：存储每个磁带的偏移量的 vector。

模拟过程：

- 模拟过程通过 `statetrans` 函数实现，该函数会根据当前状态和磁带内容执行状态转移，并更新磁带上的字符序列和磁头位置。
- 在每一步模拟后，通过 `printstep` 函数打印当前的状态、磁带内容和磁头位置信息。
- 模拟结束后，通过 `printfinal` 函数打印最终的结果，包括接受状态、磁带内容等信息。

注意事项：

- 代码中使用 `printf` 函数进行输出，适用于 C++ 的标准输出流。
- 对于磁带内容的表示，空白符用 `_` 表示。
- 磁头位置用 `^` 表示。

遇到的问题和解决方法

进行实验时,在判断输入是否合法和查找相应状态的转移规则时遇到了困难:直接采用暴力搜索的方式效率低下且实现较为复杂,因此采用了STL中的set类和map类,简化了实现并提高了效率.