周心瑜 181870290

181870290@smail.nju.edu.cn

Project 2020 形式语言与自动机

一、分析与设计思路

主要思路：生成关于输入程序的解析器；再根据解析器、输入字符串和模拟模式生成模拟器。运行模拟器，并在退出后展示第一条纸带上的结果。

1、文件功能

my_exception.*	错误类声明
TM_parser.*	解析器实现
parser_help.*	解析器辅助宏
TM_simulator.*	模拟器实现
main.cpp	主程序

2、主要类的设计

解析器（**Parser**）类

成员变量：状态集、输入符号集、纸带符号集、初始状态、空格符、终止状态集、纸带数目、转移函数存储文件的文件描述符和这一临时文件的文件路径。

```
31      vector<string> state_group;
32      vector<char> input_symbol_group;
33      vector<char> tape_symbol_group;
34      string initial_state;
35      char blank_symbol;
36      vector<string> final_state_group;
37      int tape_number;
38      int delta_function_fd;
39      char delta_function_file[];
```

成员函数：

Part 1 图灵机集合与符号的信息记录

parser_certain_line（对程序指令行进行分类解析）、func_about_variable（对指定成员变量函数声明的宏，其中clean表示提取行内有用信息，parse表示对类内成员变量赋值）

```
41      Parser();
42      Parser(string prog_filename);
43      void parse_certain_line(string line);
44
45      func_about_variable(clean_)
46      func_about_variable(parse_)
```

宏的定义（Q, S, G, q0, B, F, N各代表对应的成员变量）：

```
20  #define func_about_variable(name) \
21      make_func(concat2(name,Q));  \
22      make_func(concat2(name,S));  \
23      make_func(concat2(name,G));  \
24      make_func(concat2(name,q0));  \
25      make_func(concat2(name,B));  \
26      make_func(concat2(name,F));  \
27      make_func(concat2(name,N));
```

周心瑜 181870290

181870290@smail.nju.edu.cn

在TM_Parser.cpp中进行的函数具体实现：

```
242 make_parse_group_func(Q)
243 make_clean_group_func(Q)
```

在clean_<state>函数中具体实现了清洗行信息并调用parse_<state> 函数；parse_<state> 函数中则通过对于不同的数据类型（group赋值或是单个值复制）定义宏来进行对应成员变量的选择：

```
68          concat2(variable_, name) = concat2(proccess_, name)(input); \
```

```
81          while (istr >> s){                                              \
82              concat2(variable_, name).push_back(concat2(proccess_, name)(s));\
83          }                                                               \
```

以下，是对于图灵机程序信息规范化的判定：

```
11  #define variable_Q state_group
12  #define variable_S input_symbol_group
13  #define variable_G tape_symbol_group
14  #define variable_q0 initial_state
15  #define variable_B blank_symbol
16  #define variable_F final_state_group
17  #define variable_N tape_number
18
19  #define proccess_Q(a)\
20      (regex_match(a, regex("[a-zA-Z0-9_]+")) == true)? a:throw(MyException())
21
22  #define proccess_S(a)\
23      ((a.length() == 1) && regex_match(a.substr(0,1), regex("[^,;{}*_]")))? a[0]:throw(MyException())
24
25  #define proccess_G(a)\
26      ((a.length() == 1) && regex_match(a.substr(0,1), regex("[^,;{}*]")))?
            a[0]:(throw(MyException()))
27
28  #define proccess_q0(a)\
29      (regex_match(a, regex("[a-zA-Z0-9_]+")))? a:(throw(MyException()))
30
31  #define proccess_B(a)\
32      ((a.length() == 1) && (a[0] == '_'))? a[0]:(throw(MyException()))
33
34  #define proccess_F(a)\
35      (regex_match(a, regex("[a-zA-Z0-9_]+")))? a:(throw(MyException()))
36
37  #define proccess_N(a) atoi(&a[0])
```

Part 2 图灵机程序转移函数的信息记录

transition_function（输入存在定义则返回新的纸带符号、指针移动方向和新状态）、write_temp_file()/read_rule_line()/initial_transition_file()（将转移函数的五元组存储到临时文件中（*记录形式见问题一））：

```
49      vector<string> transition_function(string old_state, string old_symbols);
50      // new symbols ## direction ## state
51      void write_temp_file(char* buffer, int length);
52      char* read_rule_line(int* length, int index);
53      int transition_number();
54      bool is_in_input_symbols(char ch);
55      void initial_transition_file();
```

周心瑜 181870290

181870290@smail.nju.edu.cn

Part 3 图灵机模拟步骤相关

对于图灵机内容各部分合法性相关判定的函数：

```
57      bool is_in_input_symbols(char ch);
58      bool is_in_tape_symbols(string str);
59      bool is_in_state_group(string cur_state);
60      bool is_in_final_state_group(string cur_state);
61      bool is_in_form_of_five_tuple(string line);
```

模拟器（Simulator）类

成员变量：解析器、纸带数量、当前状态、各纸带指针位置、各纸带端点位置（偶数为左端点，奇数为右端点）、各纸带零点位置。

```
24      Parser* my_tm_parser;
25      int tape_num;
26      string cur_state;
27      vector<Node*> tape_head;
28      vector<Node*> tape_most;
29      //vector<Node*> tape_rightmost;
30      vector<Node*> tape_zero;
31      bool is_halt;
32      bool is_verbose;
```

成员函数：

build_tape（建立含初始内容的纸带）、input_verification（判断输入是否含有非法字符）、tm_next_step（根据模拟器的当前状态决定下一步迁移）、tm_start（模拟进程）、get_symbols（获取每个纸带当前的字符）、tape_result（展示指定纸带的信息）、show_step_info(展示指定步的信息)、show_final_result（打印第一条纸带最终结果）：

```
34      Simulator();
35      Simulator(Parser* tm_parser, string input, bool flag);
36      void build_tape(int index, string content);
37      bool input_verification(string input);
38      void tm_next_step();
39      void tm_start();
40      string get_symbols();
41      void tape_result(int index);
42      void show_step_info(int steps);
43      void show_final_result();
```

3、程序实现过程

对设计思路，在main.cpp中实现：

```
30      string filename = argv[index];
31      Parser my_parser(filename);
32      //cout<<"finished parsing"<<endl;
33      //my_parser.show();
34      string input = argv[index + 1];
35      Simulator my_simulator(&my_parser, input, flag);
36
37      my_simulator.tm_start();
38      my_simulator.show_final_result();
```

周心瑜 181870290

181870290@smail.nju.edu.cn

二、实验完成度

完成了解析器的普通错误处理，模拟器的普通模式与verbose模式与接收实验要求中两个语言的多带图灵机程序。

三、实验中遇到的问题以及解决方法

问题一：

关于Transition function数据的存储。Transition function(δ)为一个五元组，存储在类内所占的空间复杂度 上限为 $O(|Q|\times|G|^n)$。随着纸带数量的增加，消耗空间是指数增长的，需要临时文件来进行存储。

解决方法：temp文件中，将以如下形式记录transition function：

记录的总数	记录1		……	记录n	
	记录长度	记录内容	……	记录长度	记录内容

具体实现：Parser:: initial_transition_file()、Parser::write_temp_file(char*, int)、Parser::read_rule_line(int*, int).

注：注意初始化记录总数，否则将在随意位置开始记录δ，查询时无法找到。

问题二：关于Makefile的使用。

解决方法：查阅资料后进行简单的使用。

问题三：将可执行文件turing加入/bin后，直接使用turing工具解析程序时出现“free(): invalid pointer” 的错误。

解决方法：代码行中运用了strcpy函数，需要注意的是strcpy会给字符串结尾自动添加’\0’。所以再给赋值式左端指针分配空间时，空间大小应该为右端指针所指向字符串的长度加一。

四、总结感想

做好对版本库的整理可以有效的在功能实现出现bug时做好版本退回，同时对于文件的意外删除事件有较好的预防。

五、对课程和实验的建议与意见

非常满意！谢谢。