Andy, Yu-Guang Chen

Final Project: Finite State Machine Minimization

(Due:2021/01/19 23:59 Late submission is NOT accepted)

//Last update: 20210108 00:30 update Fig 2 and Fig 4

Introduction

You are asked to implement a **Finite State Machine (FSM) Reduction Algorithm**. The program reads given Mealy models which contain several states and transitions in a file. Then, it applies an arbitrary optimization algorithm to reduce the state number of the given FSM. Finally, an equivalent FSM with less states will be recorded in a file as the output.

Problem Specification:

Input: We will provide a file contains a couple of completely specified mealy machines.

An example of a given file is shown as follows:

```
.t 2

.i 1
.o 1
.s 7
0 a a 0
1 a b 0
0 b c 0
0 c a 0
0 e a 0
1 e f 1
1 b d 0
0 f g 0
1 f f 1
0 g a 0
1 g f 1
1 c d 0
0 d e 0
1 d f 1
.e

.i 2
```

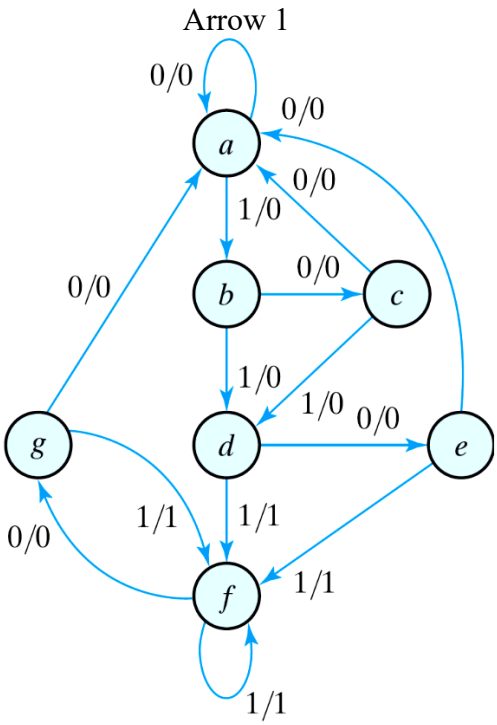

Fig 1

```
.o 1
.s 3
00 S0 S0 0
01 S0 S1 1
10 S0 S2 0
11 S0 S0 1
00 S1 S0 0
01 S1 S1 1
10 S1 S2 0
11 S1 S0 1
00 S2 S2 0
01 S2 S2 1
10 S0 S0 0
11 S2 S1 1
.e
```

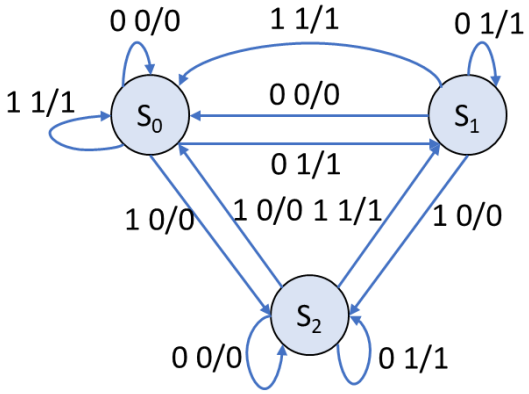

Fig 2

The first line in the file gives the number of FSMs contain in the file with the tag ".t". Then an empty line is encountered. After that, the first FSM is specified as follows: the bit number of inputs is shown with ".i", and the bit number of outputs is shown with ".o". Then, the following line gives the number of the states contain in the FSM. After that, each line gives a transition with "input current_state next_state output" description. For example, "0 a a 0" represents the arrow 1 in Fig.1. Note that the order for transition description maybe random. The line with ".e" shows the end of the FSM description. Then, an empty line is encountered again, following by the specification of the next FSM.

Note that since we provide the completely specified mealy machines, all input combinations will be specified through each state. The state name can be any combinations with [a-z], [A-Z], [0-9] and _. When you need remove a state, you have to compare the state name of the two sequence and remove the larger one. For example, if you find state "e" and "g" are identical, remove "g" instead of "e".

Output: Your program will generate an output file which specifies reduced FSMs.

The output format should be as follows:

```
.r 2
.s 5
0 a a 0
1 a b 0
0 b c 0
1 b d 0
0 c a 0
1 c d 0
0 d e 0
1 d d 1
0 e a 0
1 e d 1
.e

.r 1
.s 2
00 S0 S0 0
01 S0 S0 1
10 S0 S2 0
11 S0 S0 1
00 S2 S2 0
01 S2 S2 1
10 S2 S0 0
11 S2 S0 1
.e
```

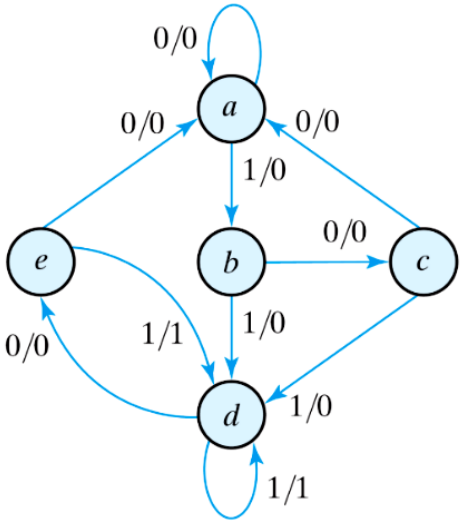

Fig 3

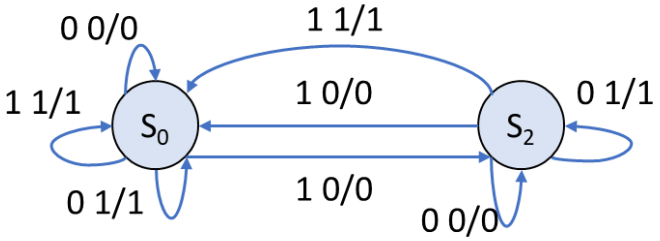

Fig 4

The first line in the file gives the number of reduced states for the first FSM with ".r". Then the second line gives the number of states of the reduced FSM with ".s". After that, the following lines give the transitions with "input current_state next_state output" descriptions. The line with ".e" shows the end of the FSM description. Then, an empty line is encountered again, following by the specification of the next reduced FSM.

Bonus: (at most 20% of the final project)

You are asked to implement a **Reduction algorithm for Incompletely Specified Mealy models.** An incompletely specified finite state machine (ISFSM) is the one where either the next state or the output is not specified for at least one input vector. Minimization of ISFSMs is an important task in the optimal design of sequential circuits. The program reads given Mealy models with the same format as described above but some edges may miss. You can treat the missing edges as don't cares. You can apply an arbitrary optimization algorithm to reduce the state number of the given FSM. Finally, an equivalent FSM with less states will be recorded in a file as the output.

Requirements:

You have to implement your code with C++/C/Python. You can use any of the algorithms you want. Your program has to take the *input_file_name* and *output_file_name* from the command line, as show as follows:

```
C:\Users\andyr>1095875871_Final.exe testcase1.txt testcase1_out.txt
```

[Hint: You may have to use *argc* and *argv[]* in your program]

You have to submit a source code named as StudID_Final.cpp. You also need to submit a report named StudID_Final_report.pdf (in pdf format, .docx is NOT acceptable). In your report, you have to describe 1) how to compile and execute your program; 2) The structure of your program (i.e., the functions you create and their relationships); 3) The data structure and the algorithm you used; 4) the hardness of this project and how you overcome it(them); and 5) the bonus part if you work on it. The bonus should be named as StudID_Final_bonus.cpp.

Your program will be judged with Code::Blocks 20.03 and GNU GCC Compiler.

The grading is as follows:

(1) Correctness of your code 50%
(2) Readability of your code 10%
(3) The report 40%
(4) The bonus (at most 20%)

If you do NOT follow the naming rule above, you will get 50% penalty. No late submission is accepted.

If you have questions, please Email me or TA.

Here are some references for you:

[1] Minimizing the number of states – implication tables
[2] State reduction in incompletely specified finite state machines