Andreas Slovacek

Fall 2018, CS5700

**Project 01 Report**

**Technical Approach**

**Documentation**: please see ***PJ01/html/index.html*** for the complete documentation, generated with [Doxygen](https://www.stack.nl/~dimitri/doxygen/).

**Source Repository:**<https://github.com/Andreas237/NFA_DFA_Maker>

My simulator implements an NFA. Since DFAs are a subset of NFAs some checks are done to limit functionality on DFAs, such as ceasing processing if epsilon is in the input string. There were three aspects implemented:

* the finite automaton (***finite\_automaton.py***)
* a logger as a subclass of the finite automaton (***fa\_logger.py***)
* master package (***fa\_master.py***) to read in the definitions, read in the input strings, and request action from finite automatons.

The project is configured to run as configured on a Unix environment, directory and file paths are relative to ***PJ01/***.

***>> python3 fa\_master.py*** from within ***code/*** and the project should on any Unix machine.

Pseudo code for building the FA is in ***finite\_automaton.FA(infile)***

|  |
| --- |
| **def process\_def(self,from\_file)**    Save the name of the file which this finite automaton is defined in  **Read in the file**:  Check that the file can be opened, raise error if not  Check if accept states are formatted correctly, cease processing if not    Read the rest of the lines into the transition table |

I approached classifying the machine in the following way (***finite\_automaton.*** ***fa\_type()****)*:

1. If the input file has a faulty accept state line classify as INVALID, cease processing
2. Are “duplicate” transitions, (current state, symbol) are the same but next state is different? NFA if so
3. Are there any epsilon transitions defined in the machine? NFA if so
4. Check the range of states, if outside of [0,255] INVALID
5. Check the range of accept states, if outside of [0,254] INVALID

Pseudo code for processing strings in ***finite\_automaton.process\_string(in\_string)***

|  |
| --- |
| **def process\_string(self,in\_string)**  Increment the number of strings processed  Copy the input string  Reset the current state    Cease processing if the FA is classified as INVALID  **Check empty accept string:**  If the accept states and input string are empty, accept empty string  **Check if this is a DFA:**  If this is a DFA reject the string if it contains epsilon transitions  **Check if the final symbol leads to an accept state**  Reject if the final symbol doesn’t lead to an accept state  **Check if the input string verse the alphabet**  Reject if the string contains characters not in the alphabet  **Recursively process the string**  No reason to reject the string so process it, check the final state |

Finally the FA\_Master, which orchestrates file I/O, building the FAs from definitions, passing strings to the FAs, then asking the FAs to log themselves once all strings have been processed.

***fa\_master.run()***

|  |
| --- |
| **def run(self)**  Log the start time of run    **Build FAs**  Self.file\_dir specifies the prefix where to look for .fa files  **Get input string**  Self.test\_str\_file has the name of the input string file  Process the file into a list stored in self.in\_strings  Setup variables for the progress bar  **Process each FA**  Feed every FA with an input string  Update the load bar  After an FA has processed all strings call **fa\_finalize()** for logging  Print the execution time |

**Implementation Details**

**Aspect Detail Comment**

**Language** Python 3

**Development Tool** Atom.io A+ Git integration

**OS** OSX 10.13.4

Kali Linux/GNU Rolling

**SCM** Git/Github

**Documentation** Doxygen Used Java as default lang

**Notes**

Python 3 on Unix works easily with the filesystem. There are a few checks for ‘linux’ system throughout the code.