Kushagra Agarwal
2018113012

Automata Theory Assignment 1

Theory:

We read the json file input and convert it into a dictionary using the json.loads() function.
Next we define the initial definitions for our dfa and nfa from the NFA provided. The DFA has 2**(states) as compared to the number of states in the parent NFA. The input alphabet for both the automata is the same. The starting state for both are same just that in the DFA we express it as a set of state(singleton set). The final state for the NFA defines the final state for the DFA with the following definition: The final state for the DFA comprises of all the sets where at least one member of the set is an element in the NFA’s final state. The transition function is defined as follows:
Lets denote the delta function by the symbol F(x,a) so the F(x,a) for DFA is:
F({1,2,3},a)= F(1,a) U F(2,a) U F(3,a)
WIth this we complete the definition of our automata.
Now we construct the DFA by defining a transition table using the definition of the transition function previously defined. We use the set

operation union present in the python library to compute the union of all the states.

Code snippets:

```
f=open("input.json","r")
r=f.read()
nfa=json.loads(r)
f.close()
```

Reading the input json as a dictionary

```
nfa["states"]=nfa["states"]
nfa["letters"]=nfa["letters"]
nfa_start=nfa["start"]
nfa_final=nfa["final"]
nfa_t_func=nfa["t_func"]
```

Initialising the NFA

```
dfa_states = 2**nfa["states"]
dfa["letters"] = nfa["letters"]
dfa_start = nfa["start"]
dfa_final = []
```

Initialising the DFA

```
for finalstate in nfa_final:
   for i in superset:
       if(i!=[]):
           for elements in i:
               if(finalstate == elements):
                   dfa_final.append(i)
```

Constructing the DFA's final states using the definition above, if a set in the powerset is such that an element of it appears in the final state then the set is part of the final state of the DFA

```
for transition in nfa_t_func:

nfa_updatedt_func[transition[0]+1][nfa["letters"].index(transition[1])+1]
= transition[2]
```

The nfa's transition table is being constructed

```
for ss in superset:
   if(ss != []):
       for letters in dfa["letters"]:
           answer = set([])
           for state in ss:
               answer =
answer.union(set(nfa_updatedt_func[state+1][dfa["letters"].index(letters)+
1]))
           if(len(answer) == 0):
               answer = []
           dfa_t_func.append([ss,letters, list(answer)])
       counter = counter + 1
```

Creating the transition table for the DFA using the idea that the transition is basically a union of all the elements getting transitioned individually. In case a set has no element then we say that the transition function is a null set or an empty set with no elements.

```
finalanswer={'states':dfa_states, "letters":dfa["letters"],
"t_func":dfa_t_func, "start": dfa_start, "final":dfa_final}
jso=json.dumps(finalanswer)
f=open("output.json","w")
h=f.write(jso)
f.close()
```

We define the finalanswer using the formal definition of an automaton. We then convert it into a json using the dumps() function and write to a file.
