German University in Cairo
Media Engineering and Technology
Assoc. Prof. Haythem Ismail

Compilers Lab, Spring term 2019

Task 3

Fallback DFA

Please read the following instructions carefully:

- Read Rules & regulations first
- It is **YOUR responsibility** to ensure that you have:
  - Submitted before the deadline.
  - Submitted the correct file(s).
  - Submitted the correct file(s) names.
  - Submitted correct logic of the task as it will be tested both publicly & privately.
  - Submitted your code in the format XX_XXXX_lab_3.zip where XX_XXXX is your ID for example 34_8000_lab_3.zip if your ID is 3 digits, append a zero to the left to be 34_0800_lab_3.zip to the correct google form link https://goo.gl/forms/bkpNhUOyGgzzWDZv2.

  .
- Good luck! =D

1 DFA to Fallback DFA

In this part, you are required to implement the algorithm explained in the lecture about the operation of the Fallback DFA. **Following the exact output file name for each one** & with the following format, you must follow the sample file on MET as well. **Note that the format of the sample file has been changed so make sure to understand it beforehand, failure to adhere to the sample file will result in deductions**

Follow the exact file name "task_3_1.py" & output file name "task_3_1_result.txt".

Your code must be generic for any Fallback DFA given.

Assume the input will not have symbols outside the alphabet & the start state will never be an accepted state.

Listing 1: Format

```
Line #1 '('string matched',' '"'action'"' ')', ...
e.g.: (xy, "Bye World"), (xy, "Bye World"), (yxyy, "Bye!")
```

For example, the input Fallback DFA will be:

Listing 2: Format

```
Line #1 state(s) separated by commas, the dead state is
represented by "DEAD"
e.g.: A , B, C, ..., DEAD
Line #2 alphabet separated by commas
e.g.: a ,b, c, etc.
Line #3 start state
e.g.: A
Line #4 final state(s) separated by commas
e.g.: A, B, C, ...
Line #5 transition(s) in a tuple form separated by commas
(state, alphabet, result state)
e.g.: (A, a, B), (A, b, C), (B, a, DEAD), ...
Line #6 State(s) label in a tuple form separated by commas
(state, expression)
e.g.: (A, "x|y"), (B, "y*"), (C, "DEFAULT"), ...
Line #7 Expression(s) action in a tuple form separated by commas
(expression, action)
e.g.: ("x|y", "Hi"), ("y*", "Bye"), ("DEFAULT", "Fail"), ...
```

```
A, B, C, D, DEAD
x, y
A
B, D
(A, x, B), (A, y, C), (B, x, DEAD), (B, y, D), (C, x, DEAD),  (C, y, DEAD), (D, x,
↪  DEAD), (D, y, D), (DEAD, x, DEAD), (DEAD, y, DEAD)
(A, "DEFAULT"), (B, "x|y"), (C, "x|y"), (D, "xy*"), (DEAD, "DEFAULT")
("x|y", "Hello world"), (''xy*``, "Bye World"), ("DEFAULT", "Fail!")
```

The output for following inputs, would be:

1. x

```
x, "Hello World"
```

2. xyxy

```
xy, "Bye World"
xy, "Bye World"
```

3. yxxxyy

```
yxxxyy, "Fail!"
```