

형식 언어 및 오토마타

본 프로젝트 2

20130273 박준희

1. 프로그램 설명

- 두개의 Class DFA, eNFA를 정의

1) DFA class

- accept(a) : a라는 input string을 입력하여 dfa가 accept하는지를 '네', '아니오'로 출력한다.
 - project 1-1에서 사용된 함수이다.
- rename() : minimize하면서 이름이 바뀐 state들의 이름을 깔끔하게 다시 재정리한다.
 - state리스트에 들어있는 state들을 0, 1, 2, 3, ... 과 같은 순서로 rename하고 이에 맞게 transition function, initial, final states 들의 이름도 맞게 바꾼다.
- print_content(message) : dfa의 내용을 output format에 맞게 출력
 - guideline에서 주어진 output format에 따라서 그대로 출력한다.
- minimize() : 현재의 dfa를 minimize한다.
 - table filling algorithm을 코드로 구현

2) eNFA class

- e_closure(state, trans) : state의 epsilon closure를 구하여 반환한다.
 - 현재 state에서 eps를 input symbol로 하는 모든 transition을 찾아서 하나의 리스트로 묶어 반환한다.
- enfa_to_dfa() : e-NFA를 DFA로 변환하여 반환한다.

2. 실행방법

이번 eNFA to DFA 프로그램은 주어진 guideline을 따라서 최대한 편리하게 프로그램을 실행할 수 있도록 하였습니다. 프로그램을 실행한뒤에, console에 guideline 에서 주어진 input format 을 그대로 넣고 실행하면 console에 ouput format에 따라서 DFA와 mDFA가 순서대로 출력됩니다.

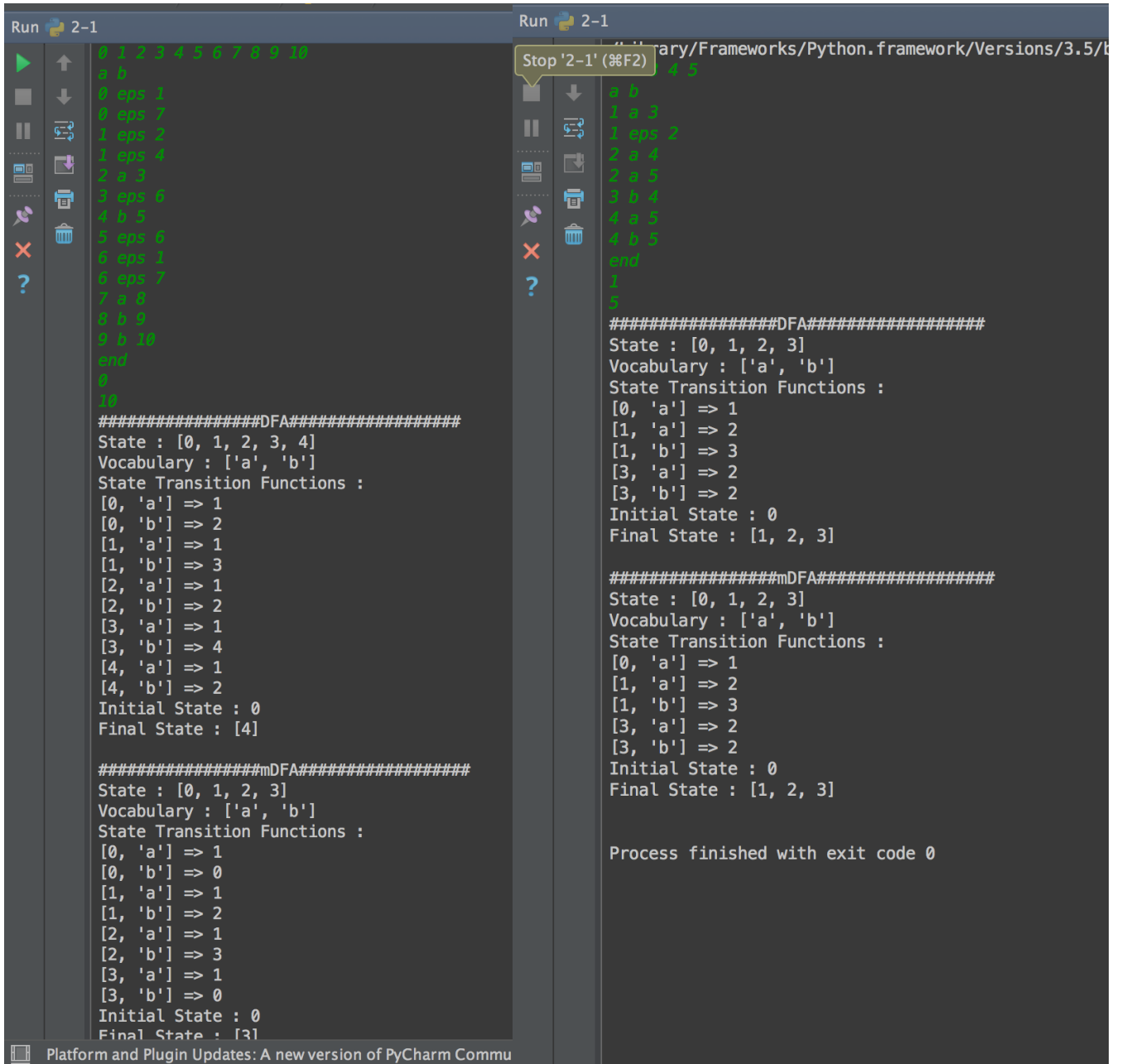
input의 경우 symbol과 state는 콘솔에서 받는 어떠한 심볼, 숫자도 가능하게 구현했습니다. 띄어쓰기가 있을 때 마다 구분하므로 string도 가능합니다. 하지만 guideline에서 주어주신 것 처럼, state는 0, 1, 2, 3... 과 같은 정수로, symbol은 a, b, c, ... 과 같은 alphabet으로 표현하는 것이 제일 좋습니다.(다른 경우는 확실하게 테스트해보지 않았습니다.) epsilon symbol의 경우는 guideline과 같이 'eps'로 입력합니다.

output의 경우도 guideline에서 제시해주신 것과 정확하게 일치합니다. input을 넣고 실행하면, 제일 먼저 dfa로 변환한 결과를 출력하고 그 다음부터 mdfa로 변환한 결과를 출력합니다. transition funcntion의 경우에는 현재 state가 값이 작은 순서부터(0, 1, 2, 3... 의 순서로) 출력되므로 guideline과 순서가 조금 다를 수 있습니다. 또는 state의 번호를 매기는 순서가 달라서 결과가 다를 수도 있으니 이는 확인해주시면 감사하겠습니다.

input에서 사용되는 enfa와 ouput에서 출력되는 dfa는 모두 partial function을 허용합니다. 결과로 출력되는 dfa는 dead state를 표시하지 않습니다!

ouput이 다 출력되면 콘솔에 'check accepting : ' 이라는 문구가 뜨고, 테스트할 string을 입력하면 예비프로젝트 1-1의 결과와 같이 accept될 경우 '네'를 출력하고, 그렇지 않을 경우 '아니오'를 출력합니다. (dead state로 갈 경우 '아니오'를 출력합니다.)

실행 결과 (샘플 테스트 2가지)



```
Run 2-1
0 1 2 3 4 5 6 7 8 9 10
a b
0 eps 1
0 eps 7
1 eps 2
1 eps 4
2 a 3
3 eps 5
4 b 5
5 eps 6
6 eps 1
6 eps 7
7 a 8
8 b 9
9 b 10
end
0
10
#####DFA#####
State : [0, 1, 2, 3, 4]
Vocabulary : ['a', 'b']
State Transition Functions :
[0, 'a'] => 1
[0, 'b'] => 2
[1, 'a'] => 1
[1, 'b'] => 3
[2, 'a'] => 1
[2, 'b'] => 2
[3, 'a'] => 1
[3, 'b'] => 4
[4, 'a'] => 1
[4, 'b'] => 2
Initial State : 0
Final State : [4]

#####mDFA#####
State : [0, 1, 2, 3]
Vocabulary : ['a', 'b']
State Transition Functions :
[0, 'a'] => 1
[0, 'b'] => 0
[1, 'a'] => 1
[1, 'b'] => 2
[2, 'a'] => 1
[2, 'b'] => 3
[3, 'a'] => 1
[3, 'b'] => 0
Initial State : 0
Final State : [3]

Run 2-1
a b
1 a 3
1 eps 2
2 a 4
2 a 5
3 b 4
4 a 5
4 b 5
end
1
5
#####DFA#####
State : [0, 1, 2, 3]
Vocabulary : ['a', 'b']
State Transition Functions :
[0, 'a'] => 1
[1, 'a'] => 2
[1, 'b'] => 3
[3, 'a'] => 2
[3, 'b'] => 2
Initial State : 0
Final State : [1, 2, 3]

#####mDFA#####
State : [0, 1, 2, 3]
Vocabulary : ['a', 'b']
State Transition Functions :
[0, 'a'] => 1
[1, 'a'] => 2
[1, 'b'] => 3
[3, 'a'] => 2
[3, 'b'] => 2
Initial State : 0
Final State : [1, 2, 3]

Process finished with exit code 0
```

Platform and Plugin Updates: A new version of PyCharm Commu