CS322 프로젝트 1-2. Mealy machine

개요

한글 오토마타를 구현하기 위한 보조 프로젝트의 일환으로 Mealy machine을 구현하였다.

이론 개요

Mealy machine $M = (Q, \Sigma, \Pi, \delta, \lambda, q_0)$은 아래와 같은 6가지 원소로 이루어져 있다.

> Q: 가능한 상태의 identifier들의 유한집합이다. 서로 다른 상태는 같은 이름을 가질 수 없다.
> Σ: Mealy machine에 입력할 수 있는 문자들의 집합이다.
> Π: Mealy machine이 출력할 수 있는 문자들의 집합이다.
> δ: 상태 변환 함수 $Q \times \Sigma \to Q$로, 한 상태에서 입력을 받았을 때 이동하는 다음 상태를 지시한다.
> λ: 출력 함수 $Q \times \Sigma \to \Pi$로, 한 상태에서 입력을 받았을 때 출력할 문자를 지시한다.
> q_0: 오토마타의 초기 상태로, Q에 속해 있어야 한다.

Mealy machine M은 입력 $s \in \Sigma^*$을 받아 $\bar{s} \in \Pi^*, |s| = |\bar{s}|$을 출력한다. 만약 s를 따라 진행할 경로가 δ에 정의되어 있지 않다면, (1) 아무것도 출력하지 않거나(입력이 1글자 이상일 때에는 애매하지 않은 리턴 값이 된다), (2) 출력으로 나올 수 없는 문자를 포함한 문자열을 출력한다.
(2)의 예로, 경로가 없을 때 출력되는 "No path exists!"의 "!" 기호는 출력 문자열로 사용하지 않는다.

프로그램 설명

project 1-2:

> testcase/mealy.txt
> testcase/input.txt
> testcase/output.txt
> Mealy.py
> main.py

main.py를 실행하면, Mealy.py에 정의된 Mealy 클래스를 불러온다. testcase/ 폴더의 mealy.txt를 읽어들여 Mealy machine을 생성한다. 같은 폴더의 input.txt를 읽어들여 Mealy machine에 입력하고, 실행한 결과를 output.txt에 저장한다.

> 개발 언어: python 3.5 (IDLE 사용)
> 개발 운영체제: Windows 10

Mealy.py

Mealy 클래스는 아래 세 가지 method를 제공한다.

1. __init__ (생성자): 6가지 input(Mealy machine의 6요소)를 받아 새 Mealy machine을 생성한다.
2. step_transit: 상태와 입력(alphabet)을 받아 그 다음 상태를 리턴한다. Transition function에 직접 접근하지 않도록 메소드를 생성하였다. 두 입력에 대응되는 다음 상태가 없다면 None을 리턴한다.
3. step_output: 상태와 입력(alphabet)을 받아 출력 문자를 리턴한다. 이 역시 output function에 직접 접근을 막기 위해 생성되었으며, 두 입력에 대한 대응 상태가 없다면 None을 리턴한다.
4. run: (alphabet 위의) 스트링을 받아 가 정의하는 언어에 포함되는지를 판별한다.
   Debug 옵션을 추가하면(debug=True) Mealy machine의 중간 상태들을 점검할 수 있다.
   만약 입력이 알파벳의 정의역을 벗어났거나, 혹은 입력과 중간 상태에 대응하는 Transition function(혹은 Output function)이 없을 경우에는 False로 간주한다.

main.py

testcase의 mealy, input을 파싱한다.

mealy.txt를 파싱하는 방식은 아래와 같다.

1. 6개의 요소를 설명하는 줄을 기준으로 잘라낸다("State", "Input symbol", …)
2. 각각의 요소(한 줄, 혹은 여러 줄)를 형식에 알맞게 파싱한다
   A. 상태, 입력 문자, 출력 문자: 반점(comma)을 기준으로 잘라낸다
   B. 상태 전환 함수, 출력 함수: 줄 나눔 문자(₩n), 반점을 기준으로 두 번 잘라내어 사전(dict)에 저장한다.
   C. 시작점: string이므로 별도의 해석이 필요하지 않다.

input.txt는 줄 나눔 문자로만 잘라낸다. 내용이 없는 줄도 입력으로 간주한다(empty string).

testcase/

프로젝트 안내 문서의 '입력 방식 및 입출력 테스트 케이스 설명'과 일치하므로 별도의 기재 안함.

실행 예시

선행 조건

testcase 폴더 안의 mealy.txt, input.txt를 형식에 맞게 수정해주어야 한다.

윈도우 예시(cmd)

..〉 python main.py

..〉 notepad testcase/output.py
...

Linux 예시(bash)

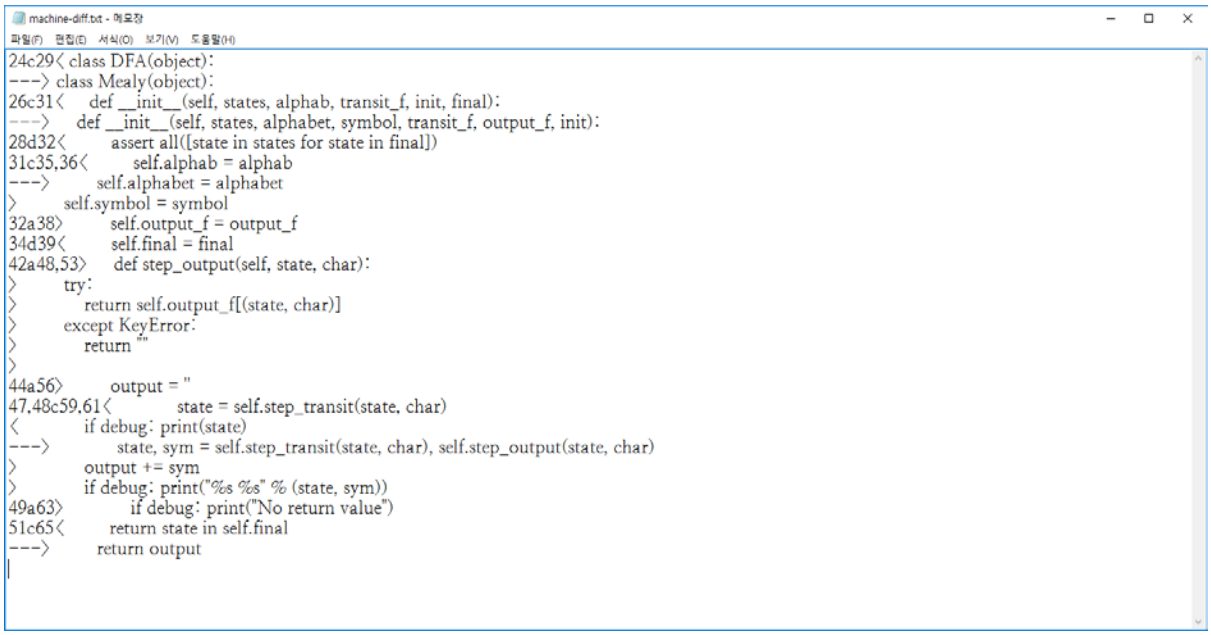
$ python main.py

$ cat testcase/output.py

변경 요소

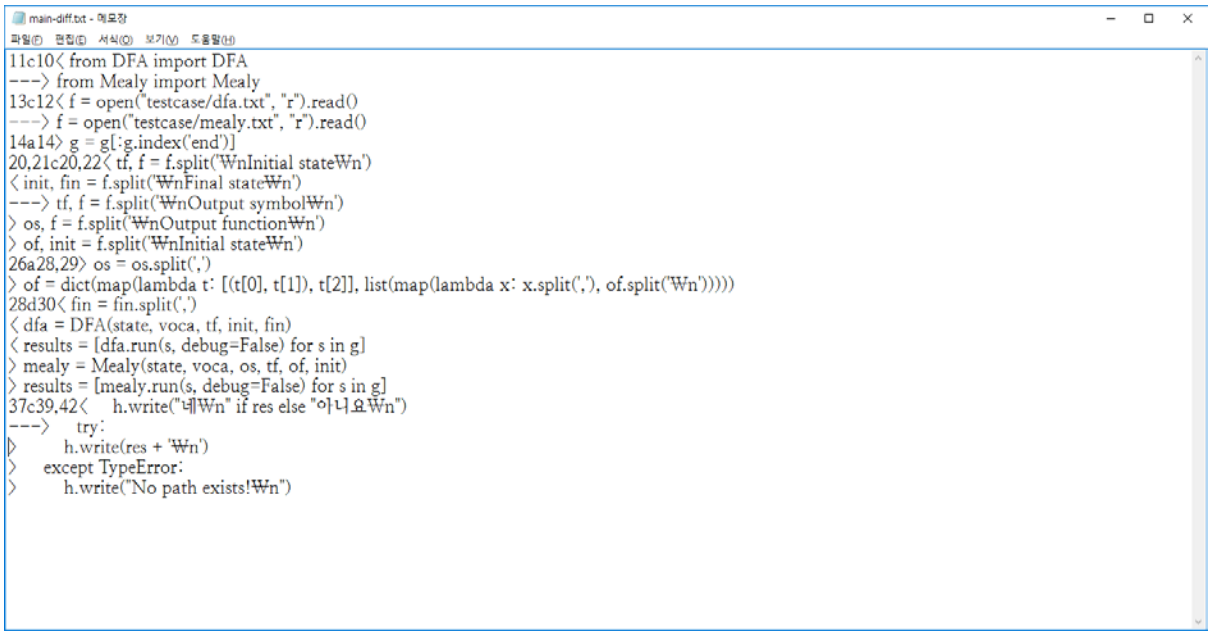
주석을 제외하면 60줄 미만 수정하였다.

$ diff PA1-1/DFA.py PA1-2/Mealy.py > machine-diff.txt



```
24c29< class DFA(object):
---> class Mealy(object):
26c31<     def __init__(self, states, alphab, transit_f, init, final):
--->     def __init__(self, states, alphabet, symbol, transit_f, output_f, init):
28d32<         assert all([state in states for state in final])
31c35,36<         self.alphab = alphab
--->         self.alphabet = alphabet
>         self.symbol = symbol
32a38>         self.output_f = output_f
34d39<         self.final = final
42a48,53>     def step_output(self, state, char):
>         try:
>             return self.output_f[(state, char)]
>         except KeyError:
>             return ""
>
44a56>         output = ''
47,48c59,61<             state = self.step_transit(state, char)
<             if debug: print(state)
--->             state, sym = self.step_transit(state, char), self.step_output(state, char)
>             output += sym
>             if debug: print("%s %s" % (state, sym))
49a63>             if debug: print("No return value")
51c65<         return state in self.final
--->         return output
```

$ diff PA1-1/main.py PA1-2/main.py > main-diff.txt



```
11c10< from DFA import DFA
--- > from Mealy import Mealy
13c12< f = open("testcase/dfa.txt", "r").read()
--- > f = open("testcase/mealy.txt", "r").read()
14a14> g = g[:g.index('end')]
20,21c20,22< tf, f = f.split('₩nInitial state₩n')
< init, fin = f.split('₩nFinal state₩n')
--- > tf, f = f.split('₩nOutput symbol₩n')
> os, f = f.split('₩nOutput function₩n')
> of, init = f.split('₩nInitial state₩n')
26a28,29> os = os.split(',')
> of = dict(map(lambda t: [(t[0], t[1]), t[2]], list(map(lambda x: x.split(','), of.split('₩n')))))
28d30< fin = fin.split(',')
< dfa = DFA(state, voca, tf, init, fin)
< results = [dfa.run(s, debug=False) for s in g]
> mealy = Mealy(state, voca, os, tf, of, init)
> results = [mealy.run(s, debug=False) for s in g]
37c39,42<     h.write("네₩n" if res else "아니요₩n")
--- >     try:
>         h.write(res + '₩n')
>     except TypeError:
>         h.write("No path exists!₩n")
```