CS322 프로젝트 #2

20150112 김동성

언어는 Python 3.4.4를 사용하였고 코드 작성에 사용한 IDE의 정보는 다음과 같다.

PyCharm Community Edition 2016.2.3

Build #PC-162.1967.10, built on September 7, 2016

JRE: 1.8.0_112-release-b343 x86

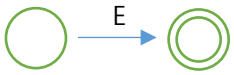
JVM: OpenJDK Server VM by JetBrains s.r.o

사용한 PLY의 버전은 3.9이다.

프로젝트 설명

이 프로젝트는 입력받은 regular expression을 e-NFA로 바꾼 뒤 최종적으로 m-DFA로 변환시키는 프로그램을 작성하는 프로젝트이다. Regular expression을 e-NFA 로 바꾸는 데는 lex와 yacc (PLY)으로 regular expression을 파싱하며 만들어지는 Abstract Syntax Tree를 이용할 것이고, e-NFA를 m-DFA로 변환하는 데에는 예비프로젝트 #2-1의 결과물을 이용할 것이다. 이때, 최종 결과물인 m-DFA의 state transition function은 dead state를 배제한 partial function으로 한다.

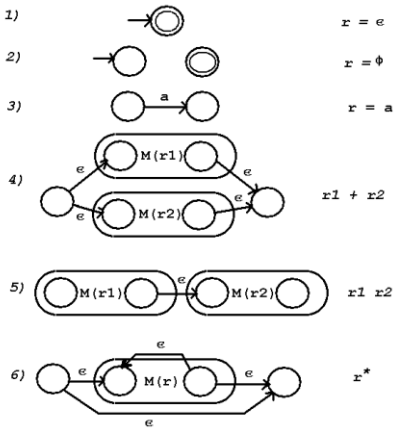
주어진 regular expression에 empty string을 나타내는 ()이 들어가 있으면 그 부분은 epsilon move를 하게 처리해주었다. 또한 입력 문자는 0~9까지의 숫자와 a~z까지의 영어 소문자로 제한하였다.

Ex) regular expression이 ()일 경우 e-nfa는 다음과 같다. 

코드 설명

e-NFA에서 m-DFA로 변환하는데 필요한 코드는 예비프로젝트 #2-1에서 소개를 하였으므로 여기서는 RE에서 e-NFA를 만드는 과정에 필요한 코드만 소개하겠다.

- class ExpressionUnit



이 프로그램에서 Regular expression를 e-NFA로 바꿀 때는 왼쪽 그림 중 3, 4, 5, 6번과 같은 방법을 따른다. 이때, ExpressionUnit은 왼쪽 그림에서 M(r)을 나타내는 단위체 역할을 한다. 이 class에서 init은 이 단위체 내에서의 initial state, final은 이 단위체 내에서의 final state를 저장한다.

- function re2enfa

  (위 RE->e-NFA 그림 참고)

  tree[0] == 'CHAR' 일 때는 그림에서 3번과 같은 상황이다. 단 tree[1] == 'E', 즉 empty string 의 경우에는 'E'는 epsilon을 나타내는 것이므로 e-nfa의 input symbol에 포함시키지 않는다.

  tree[0] == '*' 일 때는 그림에서 6번과 같은 상황이다.

  tree[0] == '+' 일 때는 그림에서 4번과 같은 상황이다.

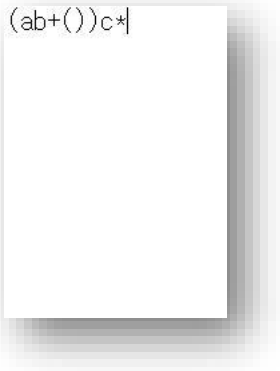
  tree[0] == 'CONCATENATION' 일 때는 그림에서 5번과 같은 상황이다.

  위 과정을 AST의 root에서부터 recursive하게 반복하며 e-NFA를 완성해 나간다.

이후 m-DFA로 변환하는 과정은 예비프로젝트 #2-1과 같다.

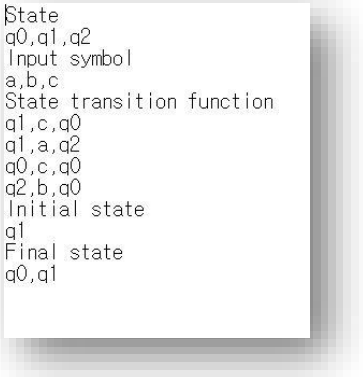
입출력 예시

Regular expression 입력



<re.txt>

m-DFA 출력



<re2mdfa.txt>

run() 함수의 check_valid(result_mdfa) 함수의 주석처리를 풀고 같은 폴더에 input.txt라는 이름으로 input string이 담긴 텍스트 파일을 만들어 실행하면 output.txt라는 이름으로 각 string의 accept 여부가 출력된다. 오토마타가 올바르게 작동하는지는 이 기능을 이용해 확인하면 된다.