형식 언어 및 오토마타

본 프로젝트 2

20130273 박준희

1. 프로그램 설명

Python용 lex & yacc 패키지인 ply를 이용하여 정규식을 parsing하고 syntax에 따라서 eNFA를 생성하도록 하였습니다. 생성한 eNFA 는 예비 프로젝트 2-1을 이용해서 mDFA로 바꾸도록 하였습니다.

regular.py
- lexer를 위해서 총 7가지 토큰을 설정하였습니다.(SIM, EPSILON, LPAREN, RPAREN, UNION, CONCATE, CLOSURE)
- lexer를 통해 생성된 form을 제가 지정한 BNF(production)에 따라서 parser가(yacc)이 parsing하고, parsing하기 위한 함수를 호출함과 동시에 그에 대응하는 eNFA를 생성하도록 하였습니다.

2. 실행방법

실행 방법은 아주 간단합니다. regular.py파일을 실행하면, 콘솔에 'input : '이라는 문구가 뜨는데 여기에 정규 표현식을 입력하면 됩니다.

정규 표현식의 format 다음과 같습니다.
1) union은 '+' 로 표현한다.
2) concatenation 은 '.'으로 표현한다.
3) closure는 '*'로 표현한다.
4) parenthesis는 '(', ')'으로 표현한다.
5) symbol은 항상 알파벳으로 표현한다.(각 심볼이 한 글자가 아니더라도 정상작동하지만, 한 글자로 이루어진 형태가 가장 좋습니다.)
6) epsilon은 'eps'로 표현한다.

example)
input : (a*+b).a.(b+eps).a*

위와 같은 예제를 입력하면 자동으로 결과를 출력합니다.
출력되는 결과는 eNFA, DFA, mDFA 세가지 입니다.
출력되는 포맷은 예비 프로젝트 2-1에서 주어진 것과 같습니다.
regular expression을 eNFA로 변환하는 과정에서 state들의 이름은 각각 integer로 표현됩니다. 하지만 모든 state의 이름이 연속적이지 않을 수도 있습니다.

예비프로젝트 2-1에서 구현할시에 partial function을 허용하는 dfa를 출력하도록 만들었기 때문에 여기서도 마찬가지로 regular expression에서 만들어지는 dfa와 mdfa는 partial function을 허용합니다. 따라서 mDFA의 경우에는 dead state가 표현되지 않을 수 있습니다.(제 프로그램에서는 거의 모든 경우가 표현되지 않는 것 같습니다.)

(regular.py가 ply 패키지와 예비프로젝트2-1을 import하기 때문에 반드시 같은 디렉토리 내에 모듈이 존재해야 합니다!)

실행 결과 (샘플 테스트 1가지 — 프로젝트2-1에서 주어진 첫번째 샘플 : (a+b)*.a.b.b)