(예비프로젝트 2-1) ε-NFA to m-DFA 변환기

제 7주 수요일 10월 12일 출제 제 11주 수요일 11월 09일 까지 마감 (27일)

임의의 ε-move NFA를 minimal state DFA로 바꾸어주는 프로그램을 작성하고, 예비프로젝트 1.1의 DFA 시뮬레이터 결과를 이용하여[1], 당신의 ε-NFA to m-DFA 변환기가 잘 동작함을 보여주시오.

이 프로젝트는 주 프로젝트 2(정규식 to m-DFA 변환기)과 연결될 부분 프로젝트이다.

(1) 임의의 ε-NFA를 입력 데이터로 읽어서[2]
(2) ε^*[3]를 구하고,
(3) ε^*와 subset construction을 이용하여 동등한 DFA로 바꾸고,
(4) DFA의 상태 수를 최소화(minimization)하는 과정을 거친다.

앞의 예비프로젝트 1.1이나 1.2보다는 좀 어려울 것이다.

본 프로젝트 2는 예비프로젝트 2.1 앞에 정규식 to ε-NFA 변환기를 붙여서 완성 될 것이다.

제출 시 유의사항

작성할 프로그래밍 언어는 다음으로 제한한다. **C, C++, Python, Java**. 코드와 함께, 작성한프로그램을 실행시키기 위한 **환경, 컴파일 및 실행 방법**을 명시하여 **README**를 첨부해야 한다. 프로그램의 자세한 세부사항은 아래 제출파일 및 입출력 예시를 참고하라.

---

1) 재사용이 불가능할 경우에는 이를 보고서에 명시하고, 고치시오.
2) ε-NFA의 일종인 (1) DFA나 (2) 부분함수를 허용하는 DFA나, (3) NFA도 입력하는 것까지도 허용하면, 당신의 ε-NFA to m-DFA 프로그램이 좀 더 강력해 질 것이다.
3) ε^*는 깊이우선탐색(Depth First Search; DFS) 나 넓이우선탐색(Breadth First Search; BFS)을 이용하면 될 것이다.

입력 방식 및 입출력 테스트 케이스 설명

0. Readme

아래와 같이 ε-NFA 입력 방식과 Input 문자열 입력 방식을 제한합니다. 다음 포맷을 지키지 않을 경우에 불이익이 있을 수 있습니다. 첨부한 'e-nfa.txt', 'm-dfa.txt' 파일은 프로젝트 수행에 있어서 자신의 코드를 테스트하기 위한 간단한 test case로 사용할 수 있습니다.

1. ε-NFA 입력 방식

ε-NFA를 입력하는 방식은 'e-nfa.txt' 파일을 읽어 결정합니다.

e-nfa.txt	
State	// ε-NFA가 $M_E = (Q_E, \Sigma, \delta_E, q_{0E}, F_E)$로 표현될때
q0,q1,q2,q3	// Q는 **상태**(State)를 의미
Input symbol	// 각 상태 q0~q3는 ","로 구분됨
a,b	// Σ는 **입력문자**(input symbol)를 의미
State transition function	// 각 입력문자 a,b는 ","로 구분됨.
q0,E,q2	// δ는 **상태변화함수**를 의미
q0,a,q3	// 현 상태 (q0)에서 입력문자(b)를 보고 다음 상태(q1)을
q1,a,q0	의미하며 ,로 구분됨
q1,a,q2	// ε-transition에 해당하는 입력문자는 대문자 E로 표시
q1,b,q1	
q2,E,q1	// $q_0 \in Q$는 **처음상태**를 의미
q2,b,q3	
q3,E,q2	// $F \subseteq Q$는 **끝나는 상태**를 의미
q3,a,q3	// 다수의 끝나는 상태가 존재하면 ","로 구분
Initial state	
q0	
Final state	
q3	

2. m-DFA 출력 방식

m-DFA로의 변환 결과는 다음과 같이 'm-dfa.txt' 텍스트 파일 형식으로 출력하여야 합니다. 예비프로젝트 1-1에서의 DFA 입력 방식과 유사합니다.

m-dfa.txt	
	// DFA가 $M_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$로 표현될때
	// Q는 **상태**(State)를 의미
State	// 각 상태 q0,q3 ... 는 ","로 구분됨.
q0,q1	// Σ는 **입력문자**(input symbol)를 의미
Input symbol	// 각 입력문자 a,b는 ","로 구분됨.
a,b	// δ는 **상태변화함수**를 의미
State transition function	// 현 상태 (q0)에서 입력문자(b)를 보고 다음 상태(q1)을
q0,a,q1	의미하며 ,로 구분됨
q0,b,q1	
q1,a,q1	
q1,b,q1	
Initial state	
q0	// $q_0 \in Q$는 **처음상태**를 의미
Final state	
q1	// $F \subseteq Q$는 **끝나는 상태**를 의미
	// 다수의 끝나는 상태가 존재하면 ","로 구분