CS322 예비 프로젝트 #2-1

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

프로젝트 설명

이 프로젝트는 ε-NFA를 입력받아 m-DFA로 변환하는 프로그램을 작성하는 프로젝트이다. 변환하는 과정은 다음과 같다.

1. 입력받은 ε-NFA에서 각 state의 epsilon closure를 구한다
2. Subset construction을 통해 DFA로 변환한다. 단, subset construction을 할 때 도달 가능한 state만 고려한다.
3. Minimization을 통해 m-DFA로 변환한다.

코드 설명

각 코드에 대한 자세한 설명은 주석으로 남겨놓았으니 여기서는 각 코드의 전체적인 기능을 설명한다.

- class FA

  다루고자 하는 finite automata를 class FA로 만들어 사용한다.

- make_nfa

  변환하고자 하는 ε-NFA를 입력받는다.

- nfa2dfa

  nfa의 initial state의 epsilon closure를 dfa의 initial state로 설정하는 것을 시작으로 subset construction을 통해 dfa로 변환하되, dfa가 현재 갖고 있는 state들이 각 input symbol에 대해 도달할 수 있는 state만을 dfa의 state에 추가하는 방식으로 새로운 state가 나오지 않을 때까지 반복하여 dfa의 state 및 state transition function을 확인한다.

  모든 state와 state transition function을 조사한 뒤, nfa의 final state를 원소로 가지는 모든

state를 dfa의 final state로 한다.

- dfa2mdfa

  table filling algorithm을 통해 모든 가능한 두 state의 pair에 대해 두 state가 distinguishable 한지 아닌지를 판별한다. 그 후 indistinguishable states끼리는 같은 state로 취급하여 mdfa의 state와 state transition function을 작성한다.

- check_dis

  주어진 pair의 두 state가 distinguishable 한지 아닌지를 판별하기 위해 dfa2mdfa에서 사용된다. 단, 한번에 판별되지 않는 경우가 발생할 수 있는데 이 경우는 dfa2mdfa에서 while-loop를 반복하며 dis 또는 indis에 pair가 채워지면 판별이 가능해질 것이다.

- epsilon_one / epsilon_closure

  epsilon_one에서는 nfa의 각 state에 대해 epsilon이 1개 이하로 input 되었을 때 전이될 수 있는 모든 state를 구하여 저장한다. epsilon_closure에서는 epsilon_one에서 얻은 정보를 바탕으로 각 state의 epsilon closure를 구하여 저장한다.

- states2string / string2states

  state가 list 혹은 set의 형태로 되어있을 때 string으로 바꾸어 써야한다면 states2string 함수를, state가 string으로 되어있는데 set의 형태로 바꾸어 써야한다면 string2states 함수를 쓴다. 이 때, states2string 함수에서 .sort()를 통해 state의 원소 순서를 항상 같게 유지시켜주기 때문에 같은 state에 대해서 states2string의 return 값은 항상 같은 string이 된다.

입출력 예시

- ε-NFA 입력

```
State
q0,q1,q2,q3
Input symbol
a,b
State transition function
q0,E,q2
q0,a,q3
q1,a,q0
q1,a,q2
q1,b,q1
q2,E,q1
q2,b,q3
q3,b,q2
q3,a,q3
Initial state
q0
Final state
q3
```

m-DFA 출력

```
State
q0,q1,q2,q3
Input symbol
a,b
State transition function
q3,b,q2
q3,a,q0
q0,b,q0
q0,a,q0
q1,b,q3
q1,a,q0
q2,b,q3
q2,a,q1
Initial state
q1
Final state
q0,q3
```

프로그램을 모두 작성한 후, 예비프로젝트 1.1에서 사용한 DFA 시뮬레이터에서 output 텍스트 파일을 작성하는 함수를 복사해 온 뒤 예비프로젝트 1.1에서의 DFA 예시를 input해 ε-NFA -> m-DFA 변환 과정을 거친 뒤 input string 예시를 input 하여 그 결과를 확인해 보았다. 확인 결과 다음과 같이 m-DFA가 생성되었으며 (input한 DFA에서 state number만 바뀌었음), 역시 output도 동일하게 나오는 것을 확인할 수 있었다. 다만 '예비프로젝트 1.1의 DFA 시뮬레이터 결과를 이용하여' 변환기가 잘 작동함을 보이는 방법을 제대로 이해하지 못하여 이 이상의 검증은 실시하지 못하였다.