CS310 : Automata Theory 2019

IITB, India

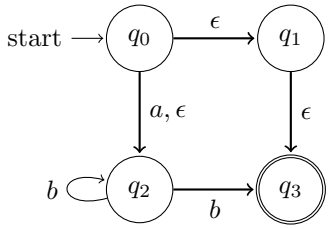
Tutorial sheet 2- ϵ-NFA, regular expression(RE), RE == DFA

Ashutosh Gupta and S. Akshay

Compile date: 2019-01-22

1. In the lecture, we added an extra initial state while going from ϵ-NFA to NFA. Can we get away without adding an extra state?

2. Convert the following ϵ-NFA into an NFA.



3. Consider the following algorithm that converts ϵ-NFA to NFA without adding any new states.

Algorithm 2.1: ENFA2NFA(ENFA $A = (Q, \Sigma, \delta, q_0, F)$)

Output: NFA $A' = (Q', \Sigma, \delta', Q_0, F')$

$Q' := Q_0$; $\delta' := \emptyset$; $F' := F \cap Q_0$;
$\delta'' = \emptyset$; $worklist := \{(q, \alpha, q') \in \delta | q \in Q_0\}$;
while $worklist \neq \emptyset$ **do**
  choose $(q, \alpha, q') \in worklist$;
  $worklist := worklist \setminus \{(q, \alpha, q')\}$;
  if $\alpha \neq \epsilon$ **then**
    $Q' := Q' \cup \{q'\}$; $\delta' := \delta' \cup \{(q, \alpha, q')\}$;
    if $q' \in F$ **then** $F' := F' \cup \{q'\}$;
    foreach $(q', \epsilon, q'') \in \delta$ **do**
      if $(q, \alpha, q'') \notin \delta'$ **then** $worklist := worklist \cup \{(q, \alpha, q'')\}$;
    foreach $(q', a, q'') \in \delta$ **do**
      if $(q', a, q'') \notin \delta'$ **then** $worklist := worklist \cup \{(q', a, q'')\}$;
  else
    $\delta'' := \delta'' \cup \{(q, \epsilon, q')\}$;
    if $q' \in F$ **then** $F' := F' \cup \{q\}$;
    foreach $(q', \alpha', q'') \in \delta$ **do**
      if $(q, \alpha', q'') \notin \delta' \cup \delta''$ **then** $worklist := worklist \cup \{(q, \alpha', q'')\}$;
return $(Q', \Sigma, \delta', Q_0, F')$

The presentation of the above algorithm is different from the class in the following two ways

- the algorithm allows multiple initial states.
- $q' \in \delta(q, a)$ is denoted by $(q, a, q') \in \delta$.

Please find the partial implementation of the above algorithm in python nfa-eps2nfa.py on the webpage. Complete the implementation.

4. Let us suppose there are n transitions in an ϵ-NFA, what is the maximum number of transitions that may be added during translation from ϵ-NFA to NFA? Give an ϵ-NFA that exhibits the worst case behavior.

5. Give regular expressions for

   (a) the set of binary strings whose number of 0's divisible by five

   (b) the set of binary strings such that each 00 appears before some 11.

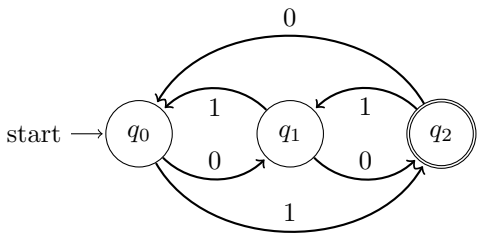
   (c) the set of binary strings such that no prefix of the stings has the difference between the number of 0s and 1s more than 2

6. Give a regex to accept only valid email addresses

   A valid email address:

   - Must contain only characters from a b 0 1 . @
   - No two symbols (. or @) are consecutive
   - Must start with a letter (a or b)
   - Has exactly one @
   - No digits (0 or 1) after @
   - At least one . after @

7. Give equivalent regular expression for the following DFA. (Please also try using the method presented in Hopcroft Section 3.2.2)



8. Give an ϵ-NFA for the following regular expressions. (You may simplify the expression as much as possible.)

   (a) $(aa^* + bb^*)^*$

   (b) $(ab + ba)(ab + ba)(ab + ba)$

9. Default Python3 uses backtracking NFA algorithm. The algorithm records a backtracking point each time it sees a nondeterministic choice and backtracks each time the match fails. There is potential of exponential blowup in the algorithm. Give a regular expression and a string, each less than 80 chars, where you can make Python3 run for at least 10mins.

   Here is a code sample for writing regular expression and matching against a string.

```
#!/usr/bin/python3
import re
str = '11010101'
# flag option is not necessary in the following call
p = re.compile(r'((01)+)', flags=re.DEBUG)
out = p.findall(str)
print( "\n\n Matched strings:")
for o in out:
    print( o )
```