Kush Shah, kshah2@wpi.edu Acknowledgement: Keith DeSantis, Sam Parks

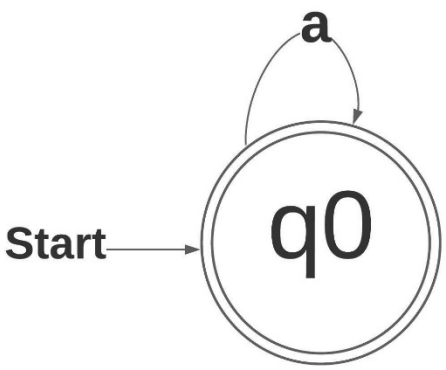
107. MakeNFAs

$\Sigma = a,b$

a)

Let K = a

Let *K* consist of the single string $x = a_1 \ldots a_i \ldots a_n$ where each a_i is an element of Σ. We can build an NFA *M* whose language is {x} as follows:

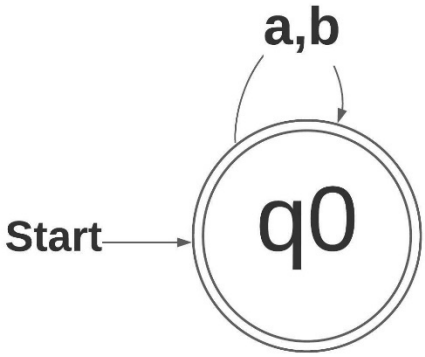


The above NFA is representative of the NFA that accepts K = {x}. An NFA could also be constructed that accepts any one element in x, individually, in this case 2, one for "a" and one for "b". Using subset construction on the NFA we get a DFA that has the same states and transitions. According to the definition of regular languages, a language is regular "if there is a DFA that accepts it".

b)

K = {a,b}

By the previous part, there are *k* NFAs, accepting precisely the individual strings x_i. We can prove that K is regular by doing the same thing.



The above NFA accepts {a,b}. Using subset construction again we get a DFA with the same states. And with the same definition of regular language, we can conclude that K is regular.

c)

If K is a coffinite language, K is aa language whose complement contains only finitely many strings then according to Theorem 8.3 "If A is regular then $\bar{A}$ is regular".

118. NFAUnionBigO

Algorithm A is charged $O(2^q)$ for subset construction and then $O(q_1q_2)$ for product construction. Since $O(2^q)$ is larger than $O(q_1q_2)$ the time complexity of algorithm A is $O(2^q)$.

Algorithm B is simply charged $O(q_1 + q_2)$ for the whole algorithm its time complexity is $O(q_1 + q_2)$.

Comparing the two algorithms in this way algorithm A will take much longer in the worst-case scenario. We know this by testing a few values.

States in M	States in N	Algorithm A	Algorithm B
10	10	O(1024)	O(20)
100	100	$O(1.256 \times 10^{30})$	O(200)
1000	1000	$O(1.071 \times 10^{301})$	O(2000)

126. RegExpCompare

a) Expression *i* is true. **O*** denotes {λ} and λ* is the same. This is because the star denotes a set containing zero or more and zero of a string is the same as empty string (λ).

b) Expression *iii* is true. **(a + b)*** is all the strings over {a,b} where as **a***+ **b*** is denoting the set of all strings either all a or all b.

c) Expression *ii* is true. **(a*b)*** is {λ,ab,aab,abababababab,aaaaaab}, in other words any number of a's followed by a single b zero or more times. (a*b*)* is any number of a followed by any number of b zero of more times. {λ,aaaaabbbbb,bbbbb, abababababab, aaaaaab }**.** The string bb is found in the second language but not the first.

d) Expression *iv* is true. **(ab** + **a)*** is the set containing any number of "ab" or any number of a, {λ,ab,ababab,a,aaa,aaaa…}. **(ba** + **a)*** is the set containing any number of "ba" or any number of a, {λ,ba,bababa,a,aaa,aaaa…}.

e) Expression *i* is true. **a*ba*b(a+b)*** and **(a+b)*b(a+b)*b(a+b)*** are the same language. Both start with any number of a followed by b the any number of a again followed by b then any number of a or b.

f) Expression *ii* is true. **a*ba*b(a+b)*** and **a*(a*ba*ba*)*** the first is a subset of the second. The second contain λ where as the first cannot.

g) **(ab)*a** and **a(ba)*** denote the same language. Both must start and end in a. Neither can have 2 of the same symbols consecutively. Also, concatenation is associative.

h) **(bba)*bb** and **bb(abb)*** are the same language. Neither contain λ, both must start with bb and end with bb.

i) **a(bca)*bc** and **ab(cab)*c** are the same language. Because concatenation is associative, and the symbols are the same the parenthesis can be moved around and * can be shifted. Both expressions require at least a singe a *a* single *b* and a single *c*.

129. MatchRegExpFA 1

1) M_1

2) M_5

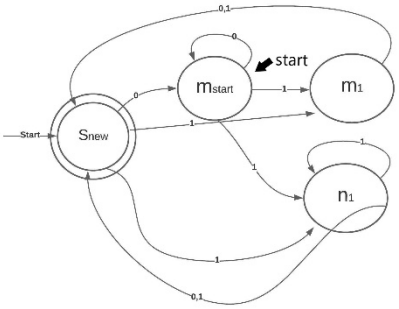
3) M_2

4) M_4

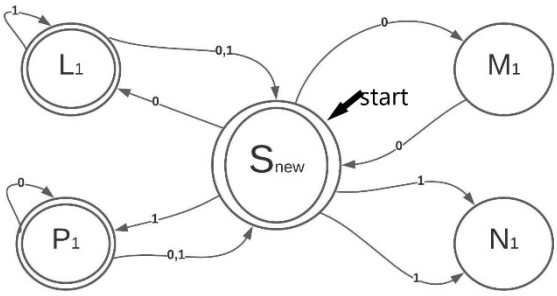
5) M_3

131. RegExpToNFA

a) (01+011+0111)*:

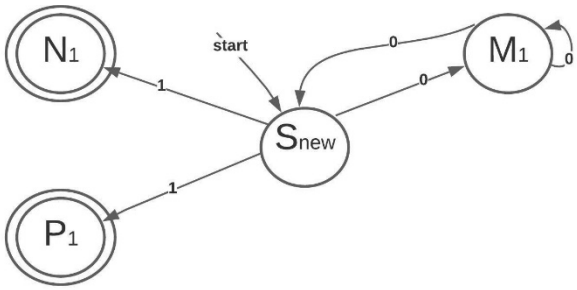


b) (00 + 11)*(01+10)(00+11)*:



Kush Shah, kshah2@wpi.edu Acknowledgement: Keith DeSantis, Sam Parks

c)(000)*1+(00)*1:



d) (0(01)*(1+00)+1(10)*(0+11))*: