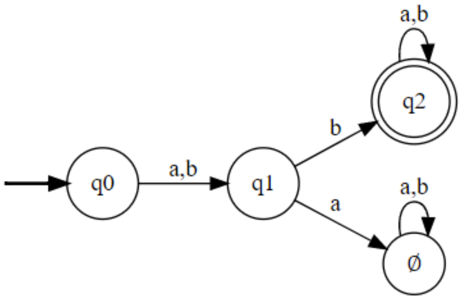
Assignment 1

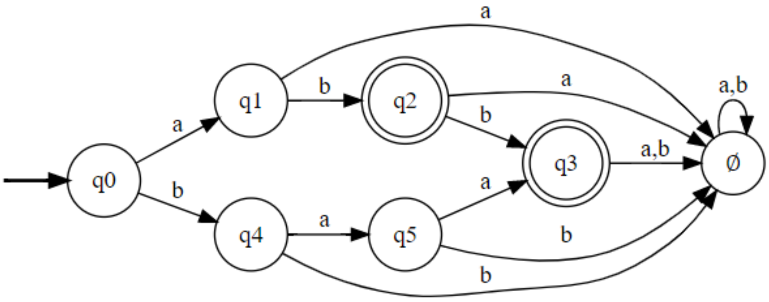
Consider alphabet set Σ = {a, b} unless specified.

Define the language, regular expressions, and finite automata for the following:

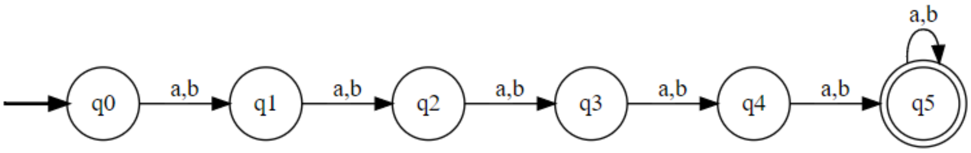
1. Strings with 'b' as the second letter.
   Language = {ab, bb, aba, abb, bba, bbb, …}
   RegEx = (a + b) b (a + b)*



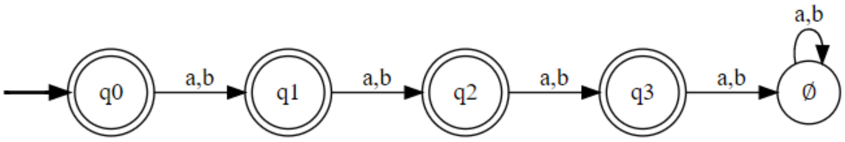
2. Strings that accept only baa, ab and abb.
   Language = {baa, ab, abb}
   RegEx = baa + ab + abb



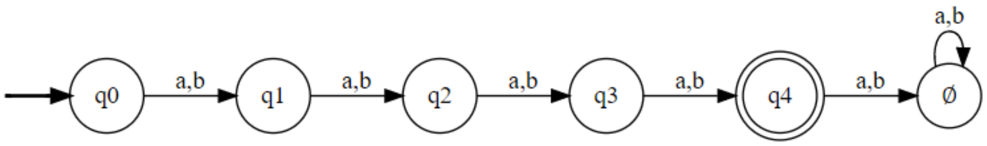
3. Words that have more than 4 letters.
   Language = {aaaaa, aaaab, aaaba, aaabb, aabaa, …}
   RegEx = (a + b) (a + b) (a + b) (a + b) (a + b) (a + b)*



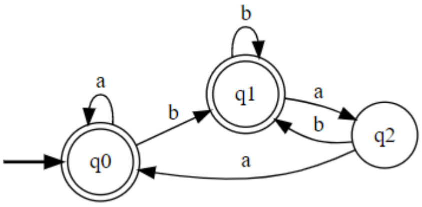
4. Words that have less than 4 letters.
   Language = {Λ, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, …}
   RegEx = Λ + (a + b) + (a + b) (a + b) + (a + b) (a + b) (a + b)



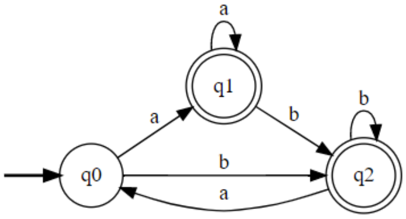
5. Words that have exactly 4 letters.
   Language = {aaaa, aaab, aaba, aabb, abaa, …}
   RegEx = (a + b) (a + b) (a + b) (a + b)



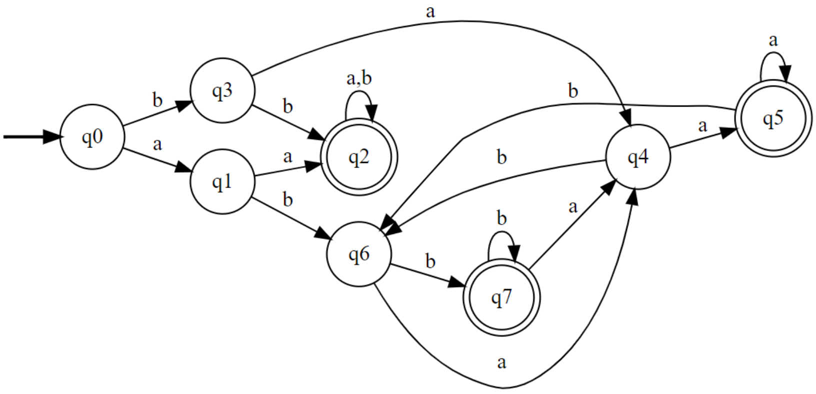
6. Strings that do not end with 'ba'. (include Λ, exclude Λ)
   - Including Λ
     Language = {Λ, a, b, aa, ab, bb, aaa, aab, abb, baa, bab, bbb, …}
     RegEx = (a + b)* (aa + ab + bb) + a + b + Λ



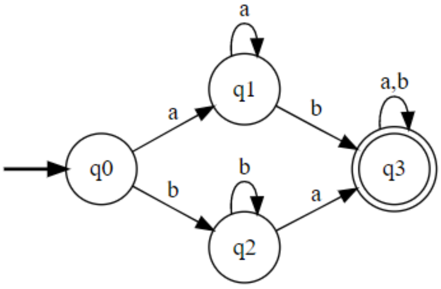
   - Excluding Λ
     Language = {a, b, aa, ab, bb, aaa, aab, abb, baa, bab, bbb, …}
     RegEx = (a + b)* (aa + ab + bb) + a + b



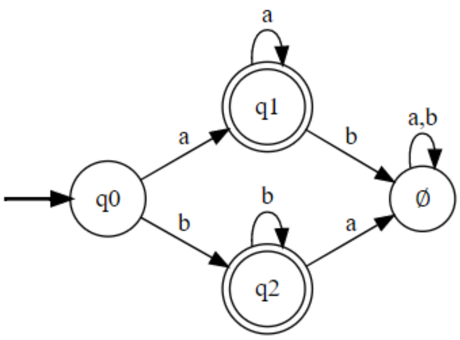
7. Strings that begin or end with a double letter.
   Language = {aaaa, aaab, aaba, aabb, abaa, …}
   RegEx = (aa + bb) (a + b)* + (a + b)* (aa + bb)



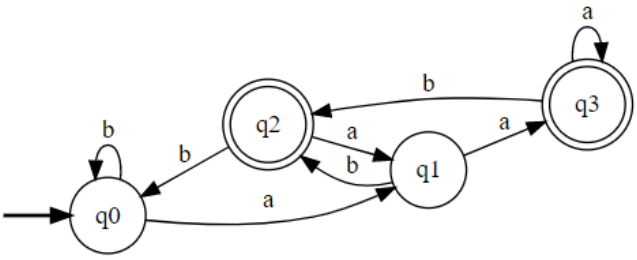
8. Words that have both the letters 'a' and 'b' in them in any order.
   Language = {ab, ba, aab, aba, abb, baa, bab, bba, …}
   RegEx = (a + b)* a (a + b)* b (a + b)* + (a + b)* b (a + b)* a (a + b)*



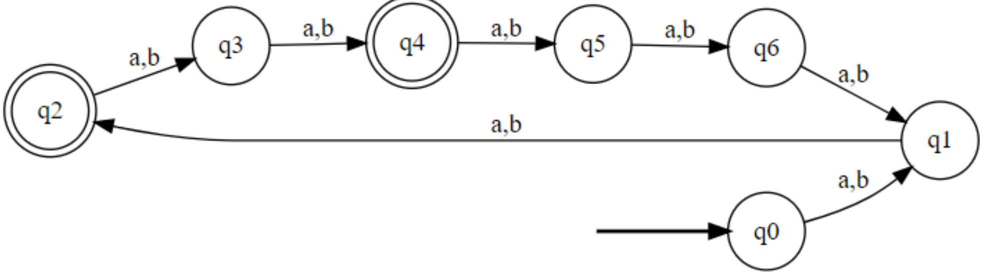
9. Words with only a’s or only b’s in them.
   Language = {a, b, aa, bb, aaa, bbb, …}
   RegEx = aa* + bb*



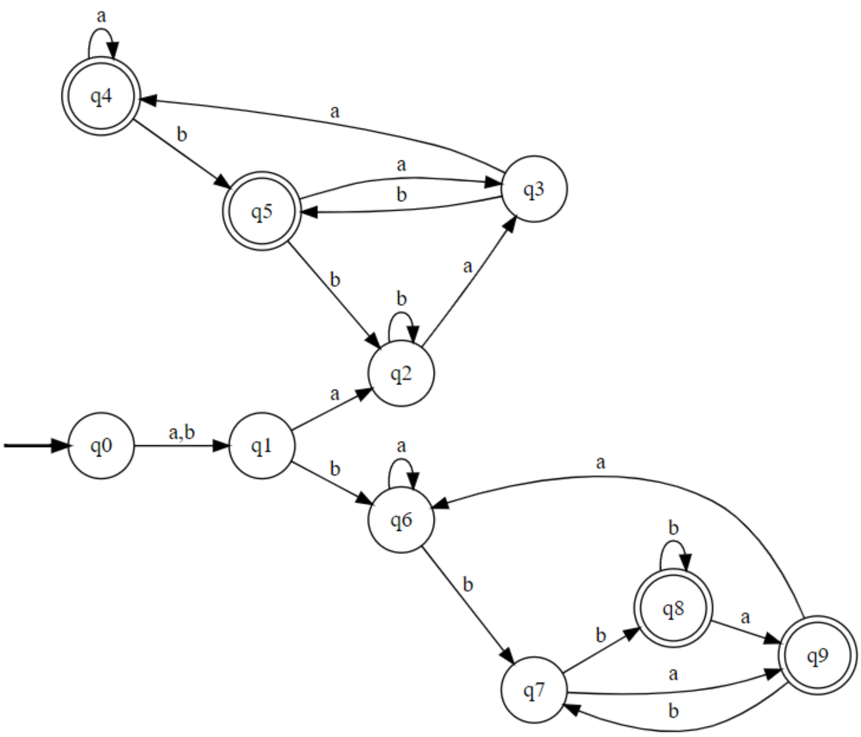
10. Strings of a's and b's such that previous to last letter is ‘a’.
    Language = {aa, ab, aaa, aab, baa, bab, aaaa, aaab, …}
    RegEx = (a + b)* a (a + b)



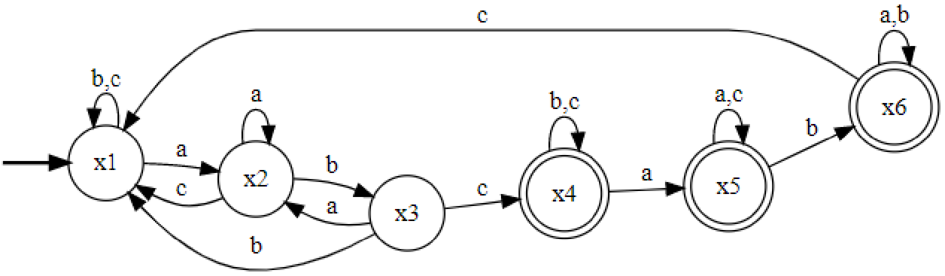
11. Strings that have even length that is not divisible by 6.
    Language = {aa, ab, ba, bb, …, aaaa, …, abababab, …}
    RegEx = (a + b) ((a + b) (a + b) (a + b) (a + b) (a + b) (a + b))*((a + b) (a + b) (a + b) + a + b)



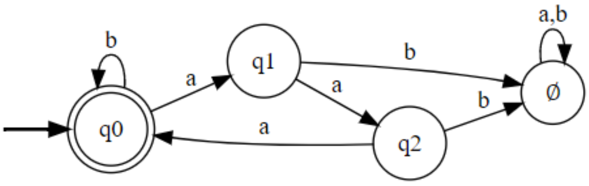
12. Strings of length 4 or more such that previous to last letter is equal to the second letter of the input string.
    Language = {aaaa, aaab, abba, abbb, baaa, baab, bbba, bbbb, …}
    RegEx = (a + b) a (a + b)* a (a + b) + (a + b) b (a + b)* b (a + b)



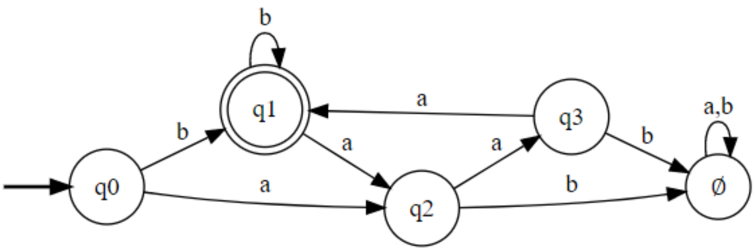
13. Words that have an odd number of occurrences of the substring ‘abc’. (consider Σ = {a, b, c})
    Language = {abc, aabc, abca, babc, abcb, cabc, abcc, aaabc, abca, abcaa, …}
    RegEx = (b + c + (a a* c) + (a a* b) (a a* b)* (b + a a* c) + (a a* b) (a a* b)* c (b + c)* a (a + c)* b (a + b)* c)* ((a a* b) (a a* b)* c (b + c)* + (a a* b) (a a* b)* c (b + c)* a (a + c)* + (a a* b) (a a* b)* c (b + c)* a (a + c)* b (a + b)*)



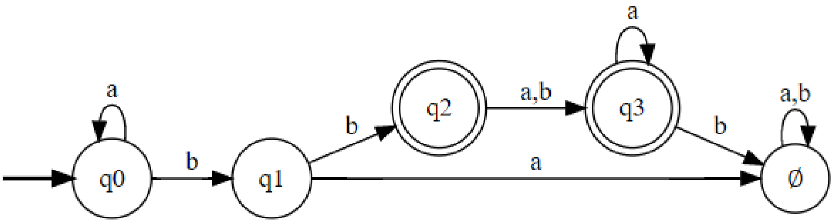
14. Strings in which ‘a’ appears in triplets.
    - Including Λ
      Language = {Λ, b, bb, aaa, bbb, aaab, baaa, …}
      RegEx = (aaa + b)*



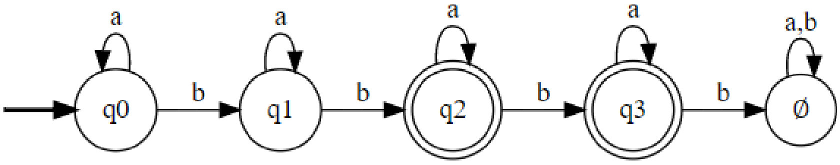
    - Excluding Λ
      Language = {b, bb, aaa, bbb, aaab, baaa, …}
      RegEx = (aaa + b) (aaa + b)*



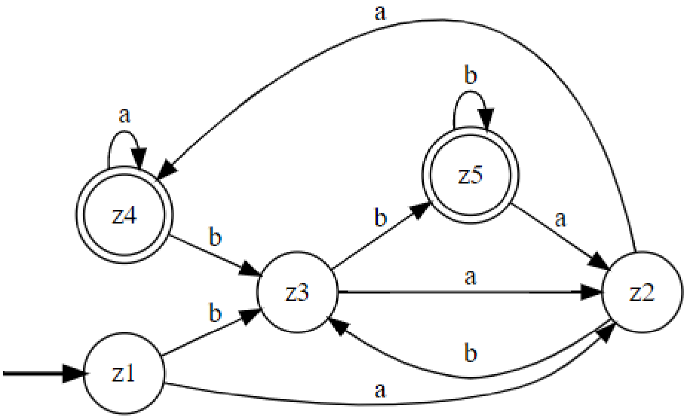
15. Strings that contain exactly two ‘b’ or three ‘b’, not more than that. (in clumpses).
    Language = {bb, abb abb, bba, bbb, abba, abbb, bbaa, bbba, …}
    RegEx = a* (bb + bbb) a*



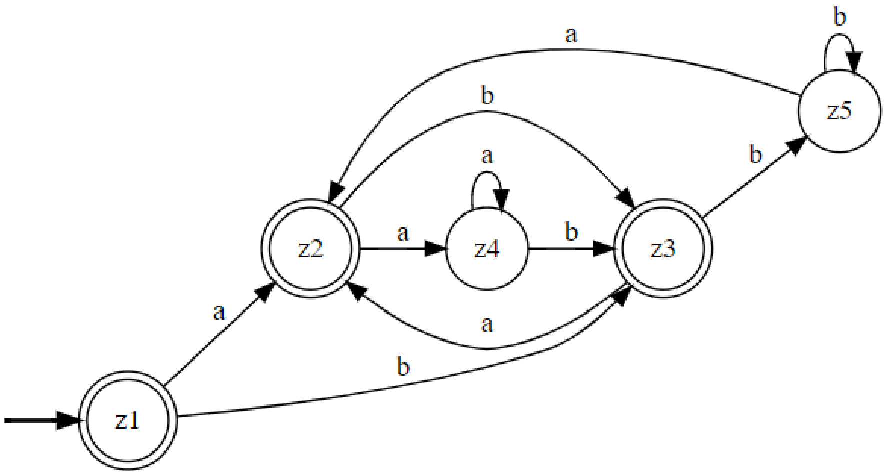
16. Strings that contain exactly two ‘b’ or three ‘b’, not more than that. (distributed anywhere)
    Language = {bb, abb, bab, bba, abab, abba, baba, …}
    RegEx = a* b a* b a* + a* b a* b a* b a*



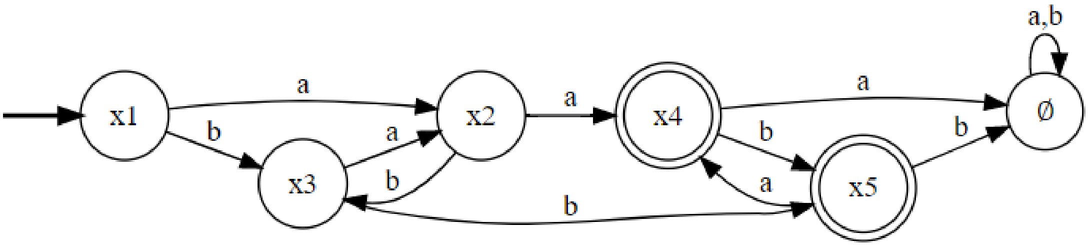
17. Strings that end in a double letter.
    Language = {aa, bb, aaa, abb, baa, bbb, aaaa, aabb, abaa, abbb, …}
    RegEx = (a + b)* (aa + bb)



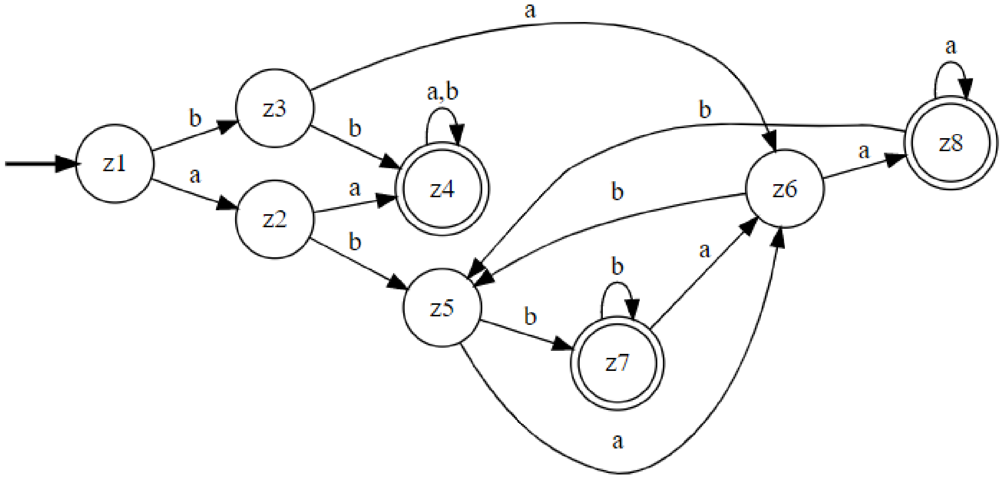
18. Strings that do not end in a double letter.
    Language = {Λ, a, b, ab, ba, aab, bab, aba, bba, …}
    RegEx = (a + b)* (ab + ba) + a + b + Λ



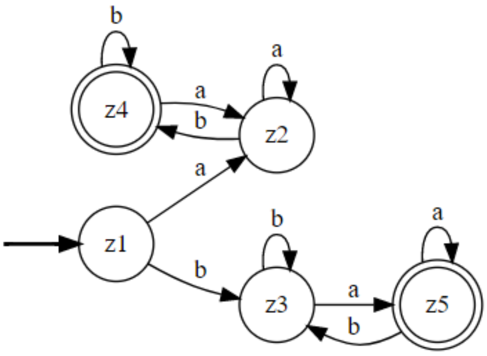
19. Strings that have exactly one double letter in them.
    Language = {aa, bb, abb, baa, bba, aaba, abaa, babb, bbab, …}
    RegEx = (b + Λ) (ab)* aa (ba)* (b + Λ) + (a + Λ) (ba)* bb (ab)* (a + Λ)



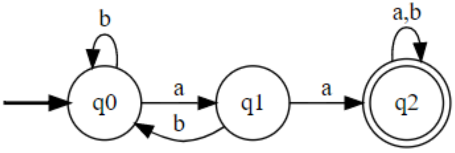
20. Strings that begin or end with a double letter.
    Language = {aa, bb, aaa, aab, abb, baa, bba, bbb, …}
    RegEx = (aa + bb) (a + b)* + (a + b)* (aa + bb)



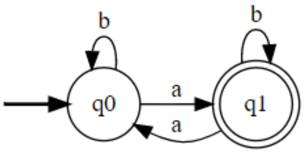
21. Strings that have different first and last letters.
    Language = {ab, ba, aab, abb, baa, bba, aaab, aabb, …}
    RegEx = a (a + b)* b + b (a + b)* a



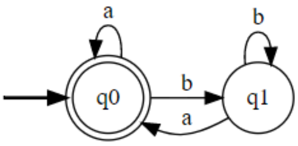
22. Strings having double a in them somewhere.
    Language = {aa, aaa, aab, baa, aaaa, aaab, baaa, baab, …}
    RegEx = (a + b)* aa (a + b)*



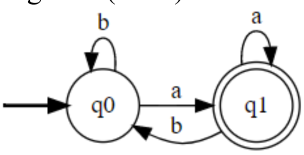
23. Strings that accept an odd number of a’s.
    Language = {a, ab, ba, aaa, aaab, aaba, abaa, baaa, …}
    RegEx = b* (a b* a b*)* a b*



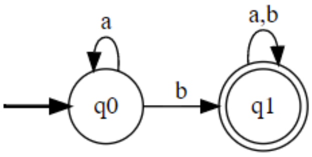
24. Strings that end with ‘a’. (include Λ, exclude Λ)
    - Including Λ
      Language = {Λ, a, aa, ba, aaa, aba, baa, bba, …}
      RegEx = (a + b)* a + Λ



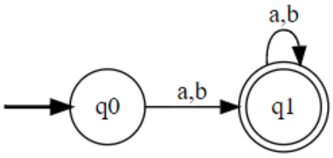
    - Excluding Λ
      Language = {a, aa, ba, aaa, aba, baa, bba, …}
      RegEx = (a + b)* a



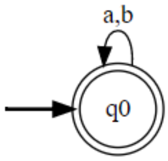
25. Words that accept at least one b.
    Language = {b, ab, ba, aab, aba, abb, baa, bab, bab, bbb, …}
    RegEx = (a + b)* b (a + b)*



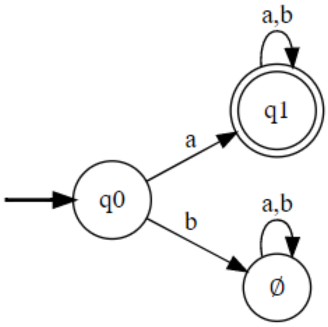
26. All strings of a's and b's except Λ.
    Language = {a, b, aa, ab, ba, bb, aaa, aab, aba, abb, …}
    RegEx = (a + b) (a + b)*



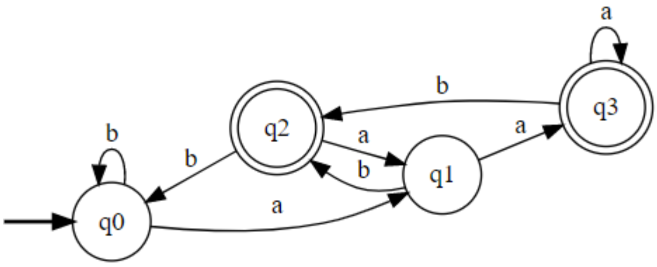
27. All strings of a's and b's including Λ.
    Language = {Λ, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, …}
    RegEx = (a + b)*



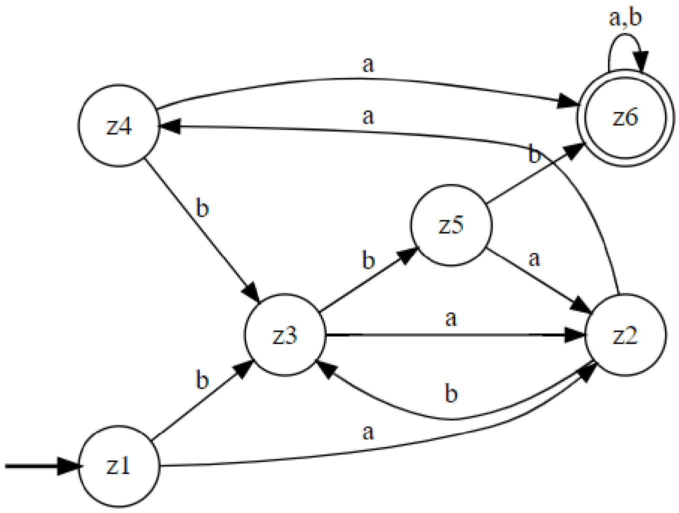
28. Strings that have the first letter as ‘a’.
    Language = {a, aa, ab, aaa, aab, aba, abb, …}
    RegEx = a (a + b)*



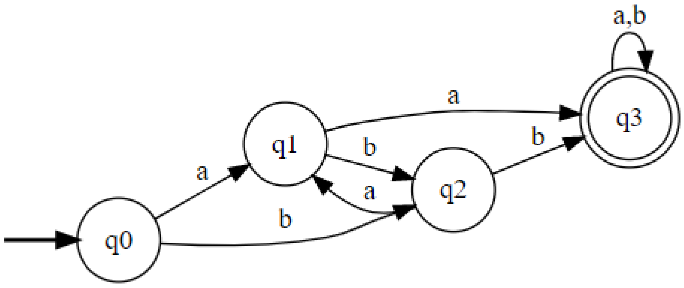
29. Strings that have the second last letter as ‘a’.
    Language = {Λ, aa, ab, aaa, aab, baa, bab, …}
    RegEx = (a + b)* a (a + b)



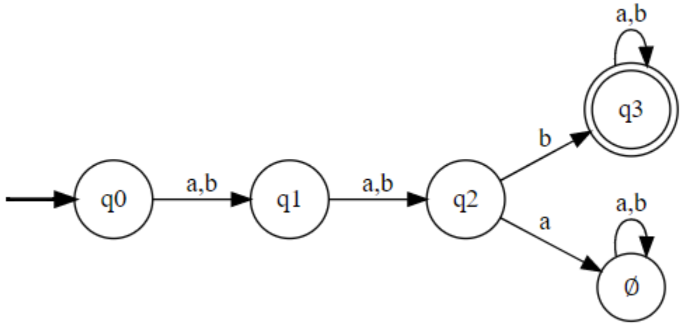
30. Strings that accept at least one ‘aaa’ or ‘bbb’. (at least one triplet of aaa or bbb).
    Language = {aaa, bbb, aaaa, aaab, baaa, bbba, bbbb, …}
    RegEx = (a + b)* (aaa+bbb) (a + b)*



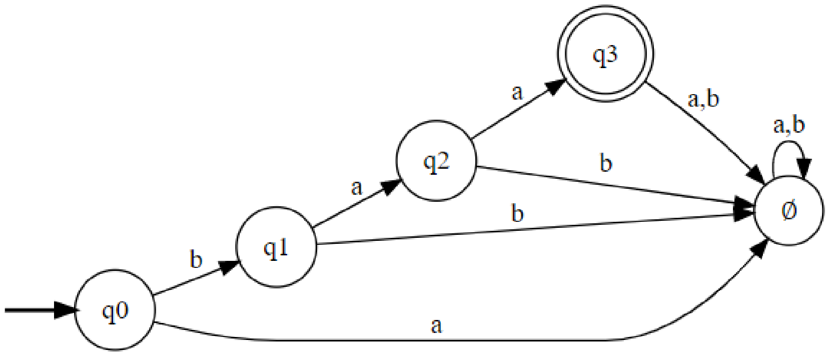
31. Strings that have double letters in them.
    Language = {aa, bb, aaa, aab, abb, baa, bbb, …}
    RegEx = (a + b)* (aa + bb) (a + b)*



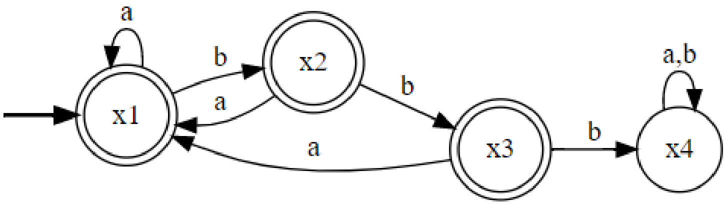
32. Strings that have ‘b’ as third letter.
    Language = {aab, abb, bab, bbb, aaba, abba, baba, bbba, …}
    RegEx = (a + b) (a + b) b (a + b)*



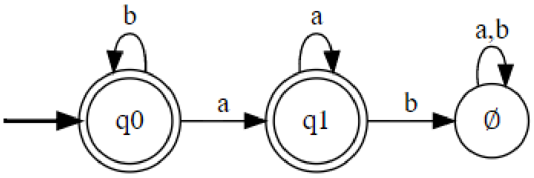
33. Language that accept only the word ‘baa’
    Language = {baa}
    RegEx = baa



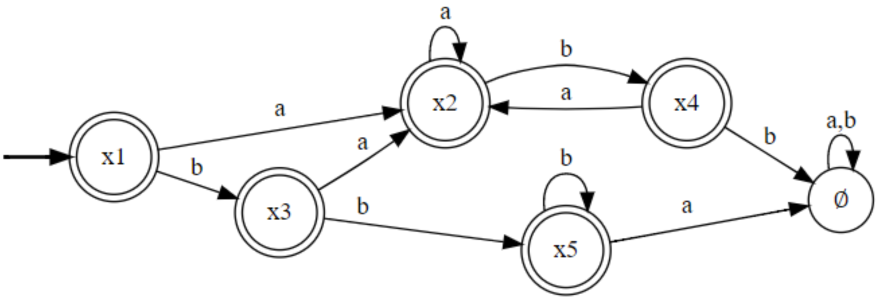
34. Strings in which letter ‘b’ is never tripled (no word contains the substring ‘bbb’)
    Language = {Λ, a, b, aa, ab, ba, bb, aaa, aab, …}
    RegEx = (Λ + b + bb) (a + ab + abb)* or (a + ba + bba)* (b + bb + Λ)



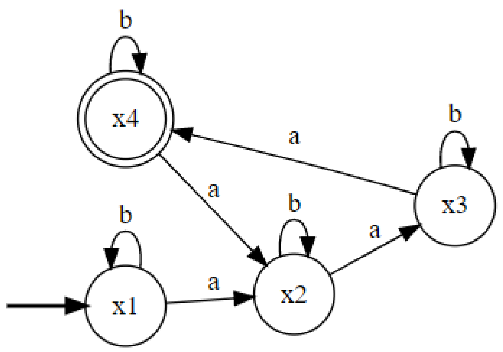
35. Words that do not have the substring ‘ab’?
    Language = {Λ, a, b, aa, ba, bb, aaa, baa, bba, …}
    RegEx = b* a*



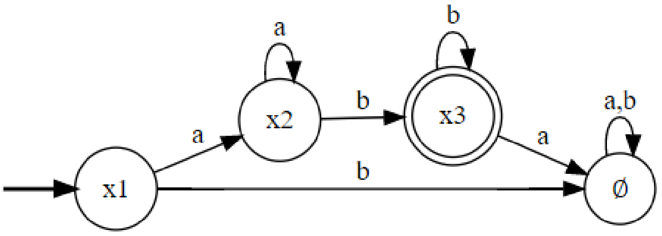
36. Words that do not have both the substrings ‘bba’ and ‘abb’.
    Language = {ab, ba, aab, aba, baa, bab, …}
    RegEx = Λ + b + ba + bbb* + (a + ba) (a + ba)* (Λ + a + b + ba)



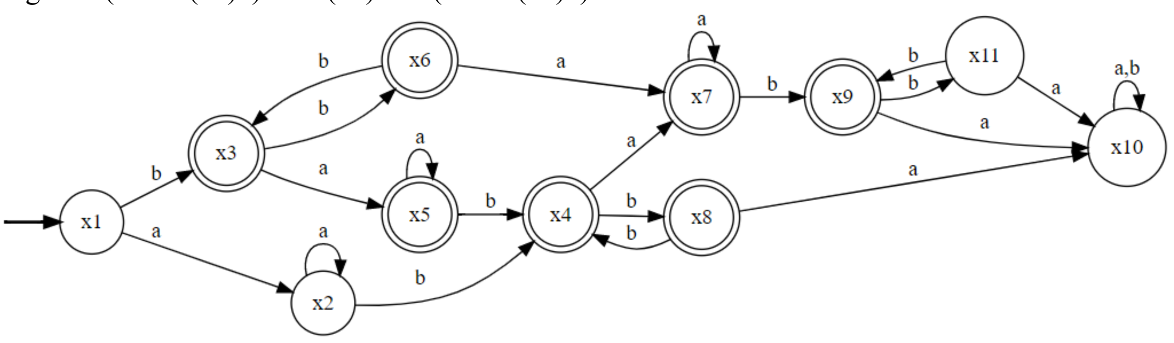
37. Total number of a's divisible by 3 no matter how they are distributed.
    Language = {aaa, aaab, baaa, aaabb, ababa, aabba, bbaaa, abbaa, …, aaaaaa, …}
    RegEx = (b* ab* ab* ab*) (b* ab* ab* ab*)*



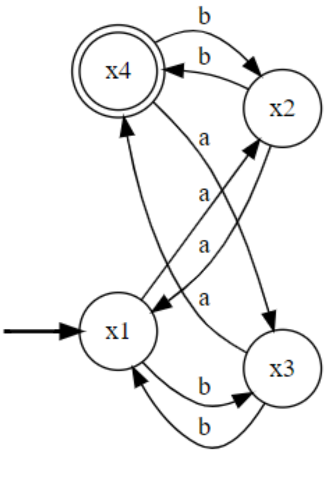
38. Strings in which all a’s precede all b’s.
Language = {ab, aab, abb, aaab, aabb, abbb, …}
RegEx = aa*bb*



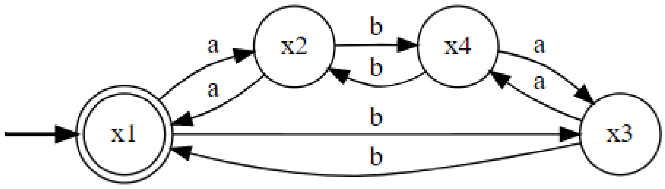
39. Strings in which any ‘b’ that occurs is found in clumps of an odd number at a time.
Language = {b, ab, ba, aab, aba, baa, bbb, …}
RegEx = (Λ + b (bb)*) a* b (bb)* a* (Λ + b (bb)*)



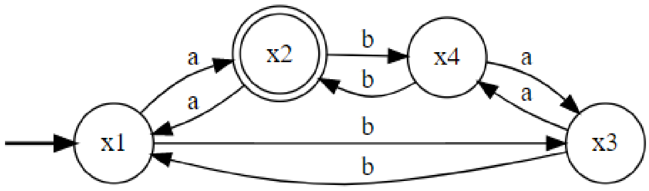
40. Strings that have an odd number of a's and odd number of b’s.
Language = {ab, ba, aaab, aaba, abaa, baaa, abbb, bbab, bbba, babb, …}
RgEx = (aa + bb + (ab + ba) (aa + bb)* (ab + ba))* (ab + ba) (aa + bb + (ab + ba) (aa + bb)* (ab + ba))*
Consider FA1 to be the DFA for a language that accepts odd number of a’s and FA2 to be the DFA for a language that accepts of odd number of b’s. The required DFA can be found by finding FA1 ∩ FA2.



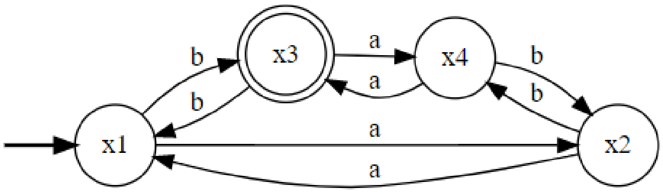
41. Strings that have an even number of a's and even number of b’s.
Language = {Λ, aa, bb, aabb, abab, abba, baba, bbaa, baab, …}
RegEx = (aa + bb + (ab + ba) (aa + bb)* (ab + ba))*
Consider FA1 to be the DFA for a language that accepts even number of a’s and FA2 to be the DFA for a language that accepts of even number of b’s. The required DFA can be found by finding FA1 ∩ FA2.



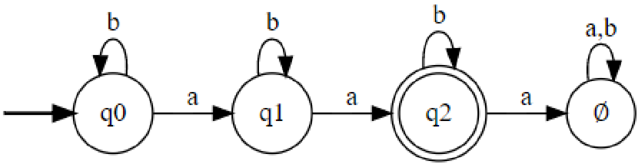
42. Strings that have an odd number of a's and even number of b’s.
Language = {a, abb, bab, bba, aaab, aaba, abaa, baaa, …}
RegEx = a (aa + bb + (ab + ba) (aa + bb)* (ab + ba))* + b (aa + bb + (ab + ba) (aa + bb)* (ab + ba))* (ab + ba) (aa + bb + (ab + ba) (aa + bb)* (ab + ba))*
Consider FA1 to be the DFA for a language that accepts odd number of a’s and FA2 to be the DFA for a language that accepts of even number of b’s. The required DFA can be found by finding FA1 ∩ FA2.



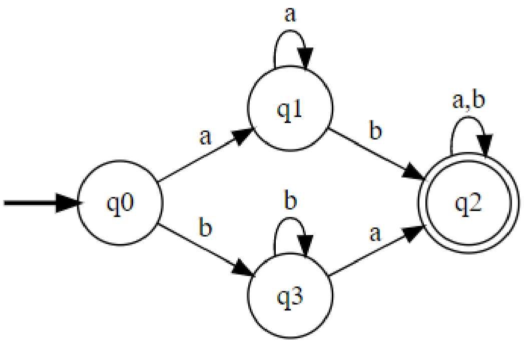
43. Strings that have an even number of a's and odd number of b’s.
Language = {aa, aab, aba, baa, aabbb, ababb, abbab, bbbaa, …}
RegEx = b (aa + bb + (ab + ba) (aa + bb)* (ab + ba))* + a (aa + bb + (ab + ba) (aa + bb)* (ab + ba))* (ab + ba) (aa + bb + (ab + ba) (aa + bb)* (ab + ba))*
Consider FA1 to be the DFA for a language that accepts even number of a’s and FA2 to be the DFA for a language that accepts of odd number of b’s. The required DFA can be found by finding FA1 ∩ FA2.



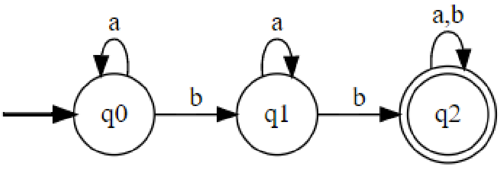
44. Strings that have exactly 2 a’s. (distributed anywhere)
Language = {aa, aab, aba, baa, aabb, abab, baba, bbaa, …}
RegEx = b* a b* a b*



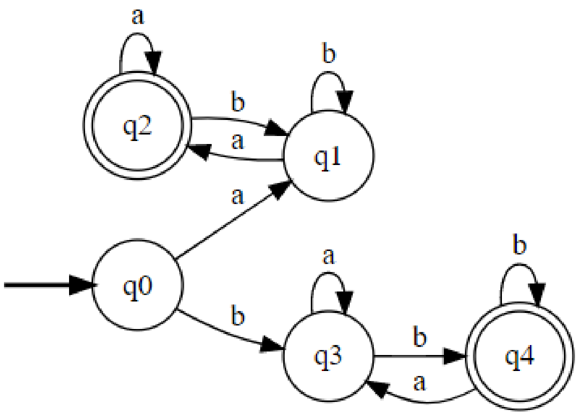
45. Strings that have at least one ‘a’ and at least one ‘b’.
Language = {ab, ba, aab, aba, abb, baa, bab, …}
RegEx = (a + b)* a (a + b)* b (a + b)* + (a + b)* b (a + b)* a (a + b)*



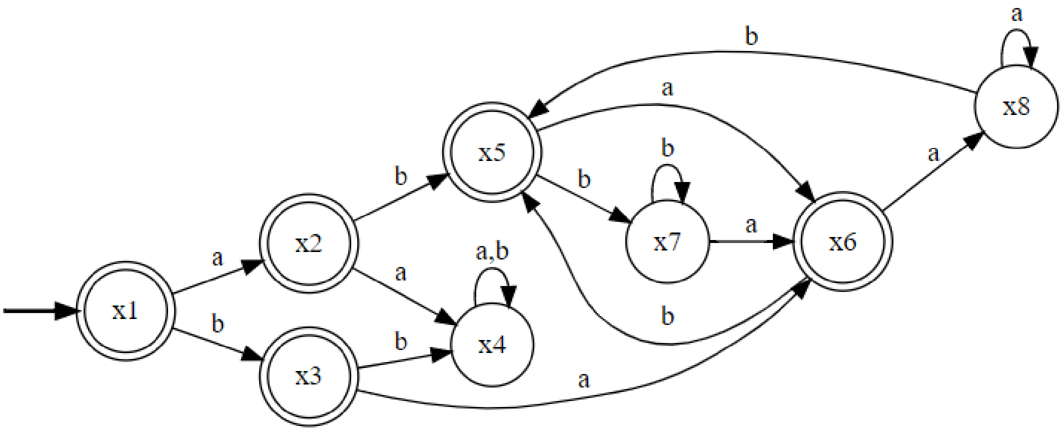
46. Strings that have at least two b's. (distributed anywhere)
    Language = {bb, abb, bab, bba, aabb, abab, baba, …}
    RegEx = (a + b)* b (a + b)* b (a + b)*



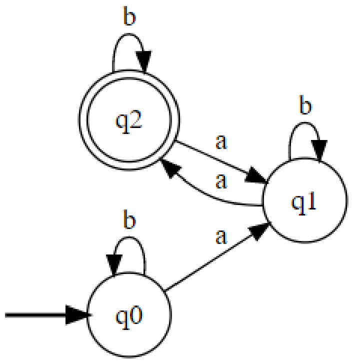
47. Strings that begin and end with the same letter.
    Language = {aa, bb, aaa, aba, bab, bbb, …}
    RegEx = a (a + b)* a + b (a + b)* b



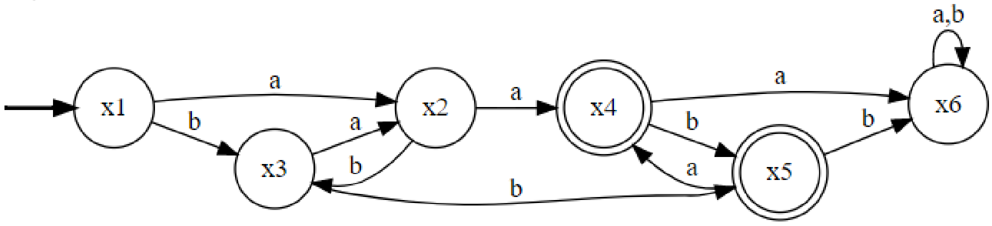
48. Strings that do not have double letters in the beginning and at the end.
    Language = {aa, bb, aaa, aab, abb, baa, bba, bbb, …}
    RegEx = Λ + a + b + (ab + ba) (a + b)* (ab + ba)
    Consider FA to be the DFA of a language that accepts strings that have double letters in the beginning or at the end. The required DFA can then be found by finding FA′.



49. Strings that have an even number of a's no matter where they are distributed.
    Language = {aa, aab, aba, aaaa, aabb, abab, abba, baba, …}
    RegEx = (b* a b* a b*) (b* a b* a b*)*



50. Strings that have exactly one double letter in them.
    Language = {aa, bb, abb, baa, bba, aaba, abaa, babb, bbab, …}
    RegEx = (b + Λ) (ab)* aa (ba)* (b + Λ) + (a + Λ) (ba)* bb (ab)* (a + Λ)


a,b
x1
a
b
x3
a
b
x2
a
x4
a
b
a
b
x5
b
x6