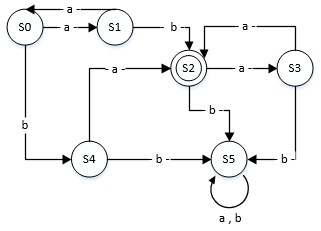
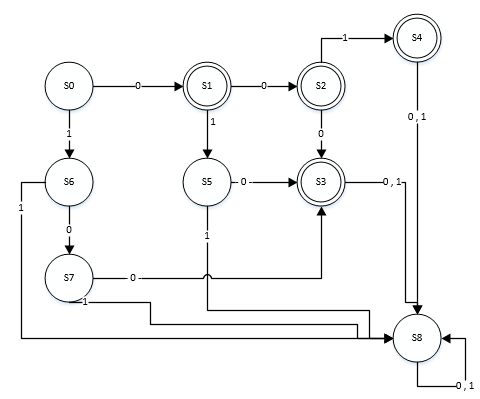
Nathan Brooks - I used Microsoft Visio to make diagrams

L1: For alphabet {a,b}, all strings that contain an odd number of a's and exactly one b.



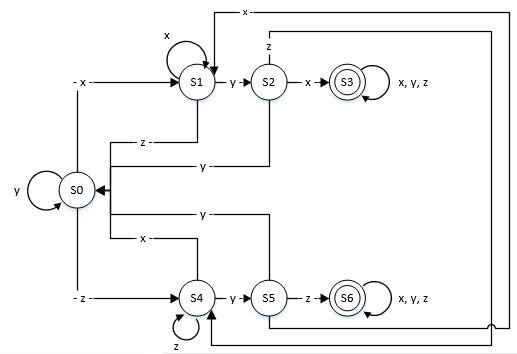
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A = I = {a, b}  S = {S0, S1, S2, S3, S4, S5}  S0 = S0  T = {S2} | |  |  |  | | --- | --- | --- | |  | a | b | | S0 | S1 | S4 | | S1 | S0 | S2 | | S2 | S3 | S5 | | S3 | S2 | S5 | | S4 | S2 | S5 | | S5 | S5 | S5 | |
| Regular Expression  (aa)\*ab(aa)\* v (aa)\*ba(aa)\*  Reduced  = (aa)\*(ab v ba)(aa)\* | |

L2: For alphabet {0,1}, all strings of length 3 or less that have more 0's than 1's.



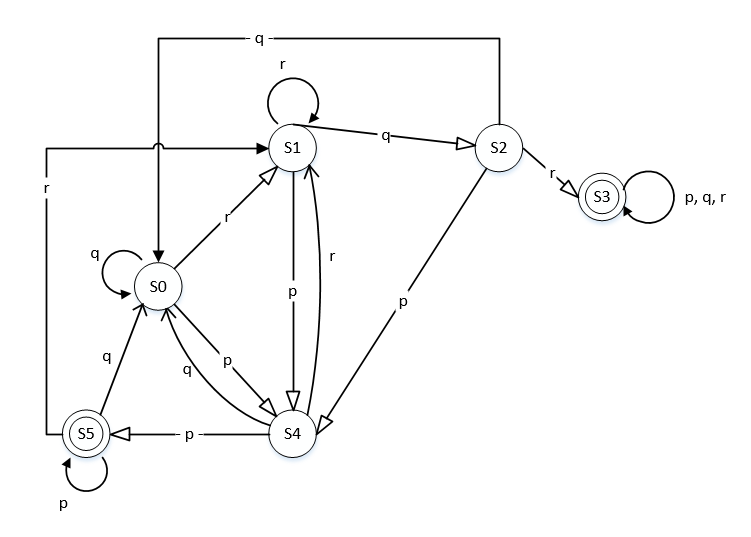
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A = I = {0, 1}  S = {S0, S1, S2, S3, S4, S5, S6, S7, S8}  S0 = S0  T = {S1, S2, S3, S4} | |  |  |  | | --- | --- | --- | |  | 0 | 1 | | S0 | S1 | S6 | | S1 | S2 | S5 | | S2 | S3 | S4 | | S3 | S8 | S8 | | S4 | S8 | S8 | | S5 | S3 | S8 | | S6 | S7 | S8 | | S7 | S3 | S8 | | S8 | S8 | S8 | |
| Regular Expression  0 v 00 v 000 v 001 v 010 v 100  Reduced  = 0 v 0(0 v 00 v 01 v 10) v 100  = 0 v 0(0 v 0(0 v 1) v 10) v 100 | |

L3: For alphabet {x,y,z}, all strings that contain xyx or zyz.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A = I = {x,y,z}  S = {S0, S1, S2, S3, S4, S5, S6, S7, S8}  S0 = S0  T = {S1, S2, S3, S4} | |  |  |  |  | | --- | --- | --- | --- | |  | x | y | z | | S0 | S1 | S0 | S4 | | S1 | S1 | S2 | S0 | | S2 | S3 | S0 | S4 | | S3 | S3 | S3 | S3 | | S4 | S0 | S5 | S4 | | S5 | S1 | S0 | S6 | | S6 | S6 | S6 | S6 | |
| Regular Expression  (x v y v z)\*xyx(x v y v z)\* v (x v y v z)\*zyz(x v y v z)\*  Reduced  = (x v y v z)\*(xyx v zyz)(x v y v z)\* | |

L4: For alphabet {p,q,r}, all strings that contain the substring rqr or end with pp.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A = I = {p, q, r}  S = {S0, S1, S2, S3, S4, S5}  S0 = S0  T = {S3, S5} | |  |  |  |  | | --- | --- | --- | --- | |  | p | q | r | | S0 | S4 | S0 | S1 | | S1 | S4 | S2 | S1 | | S2 | S4 | S0 | S3 | | S3 | S3 | S3 | S3 | | S4 | S5 | S0 | S1 | | S5 | S5 | S0 | S1 | |  |  |  |  | |
| Regular Expression  (p v q v r)\*rqr(p v q v r)\* v (p v q v r)\*pp  Reduced  = (p v q v r)\*(rqr(p v q v r)\* v pp) | |

INSTRUCTIONS!!!!

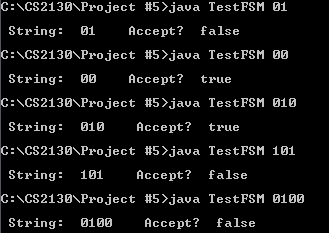
All 4 sets of information are here, just uncomment one set at a time if you want to run it. You will see what I did at the start of the code.

// Test Finite State Machine Class  
public class TestFSM  
{  
 public static void main(String[] args)  
 {  
   
 /\*  
 // L1  
 String A = "ab";  
 int[][] ST = {{1,4},  
 {0,2},  
 {3,5},  
 {2,5},  
 {2,5},  
 {5,5}};  
 int[] AS = {0,0,1,0,0,0};  
 \*/  
   
 /\*  
 // L2  
 String A = "01";  
 int[][] ST = {{1,6},  
 {2,5},  
 {3,4},  
 {8,8},  
 {8,8},  
 {3,8},  
 {7,8},  
 {3,8},  
 {8,8}};  
 int[] AS = {0,1,1,1,1,0,0,0,0};  
 \*/  
   
 /\*  
 // L3  
 String A = "xyz";  
 int[][] ST = {{1,0,4},  
 {1,2,0},  
 {3,0,4},  
 {3,3,3},  
 {0,5,4},  
 {1,0,6},  
 {6,6,6}};  
 int[] AS = {0,0,0,1,0,0,1};  
 \*/  
   
 // L4  
 String A = "pqr";  
 int[][] ST = {{4,0,4},  
 {4,2,1},  
 {4,0,3},  
 {3,3,3},  
 {5,0,1},  
 {5,0,1}};  
 int[] AS = {0,0,0,1,0,1};  
  
  
 String inString;  
 boolean accept1 = false;  
 FSM FSM1 = new FSM(A, ST, AS);  
 if(args.length >= 1)  
 {  
 // Input string is command line parameter  
 inString = args[0];  
 accept1 = FSM1.validString(inString);  
 System.out.println("\n String: " + inString   
 + " Accept? " + accept1);  
 }  
 } // end main  
} // end class

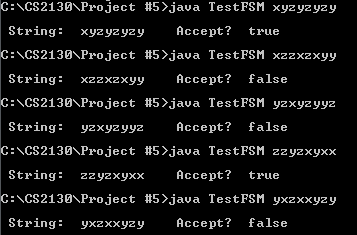
L1



L2



L3



L4

