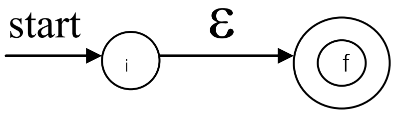
Construction of an NFA from a Regular Expression

Algorithm. (Thompson's construction)

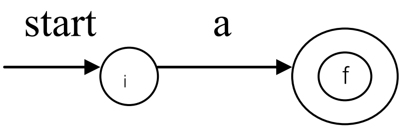
Input. A regular expression r over an alphabet Σ.

Output. An NFA N accepting L_r.

1. For ε, construct the NFA

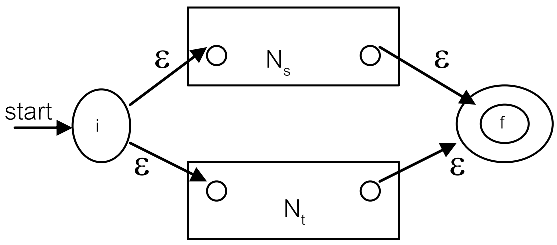


2. For a in Σ, construct the NFA

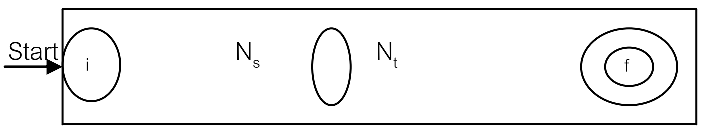


3. Suppose N_s and N_t are NFA's for regular expression s and t.

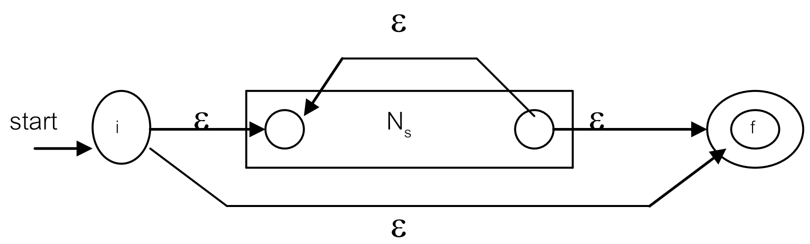
   a) For the regular expression s | t, construct the following composite NFA $N_{s|t}$:



b) For the regular expression st, construct the following composite NFA N_{st}:



c) For the regular expression s*, construct the following composite NFA N_{s*}:



d) For the parenthesized regular expression (s), use N_s itself as the NFA.

Example

Let us construct NFA from the regular expression (a | b)*abb

Conversion of an NFA into a DFA

Algorithm. Constructing a DFA from an NFA.

Input. An NFA N.

Output. A DFA D accepting the same language.

Initially, ε-closure({s0}) is the only state in Dstates and it is unmarked;

while there is an unmarked state T in Dstates **do begin**

  mark T;

  for each input symbol a **do begin**

    U := ε-closure(move(T, a));

    if U is **not in** Dstates **then**

      add U as an unmarked state **to** Dstates

    Dtran[T, a]:=U

  end

end

A state of DFA (Dstates) is a final state if it contains at least one final state of NFA.

Note! move(T, a) is a set of NFA states to which there is a transition on input symbol a from some NFA state s in T. Dtran[T, a]:=U is a transition of DFA on input symbol a from state T to U.

Computation of ε-closure(T)

---

push all states in T **onto** stack

Initialize ε-closure(T) to T

while stack is **not** empty **do begin**

  pop t, the top element, off of stack

  for each state u with an edge from t to u labeled ε **do**

    if u is **not in** ε-closure(T) **do begin**

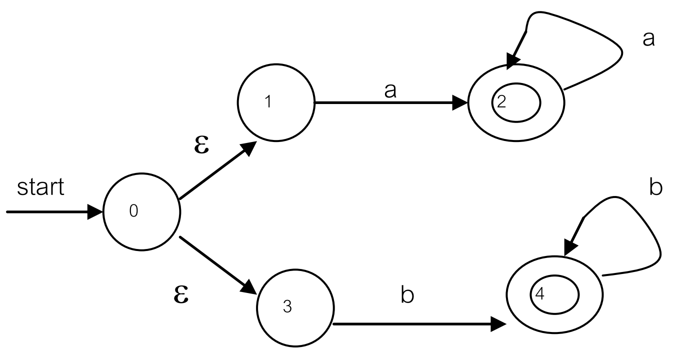
      add u to ε-closure(T)

      push u onto stack

    end

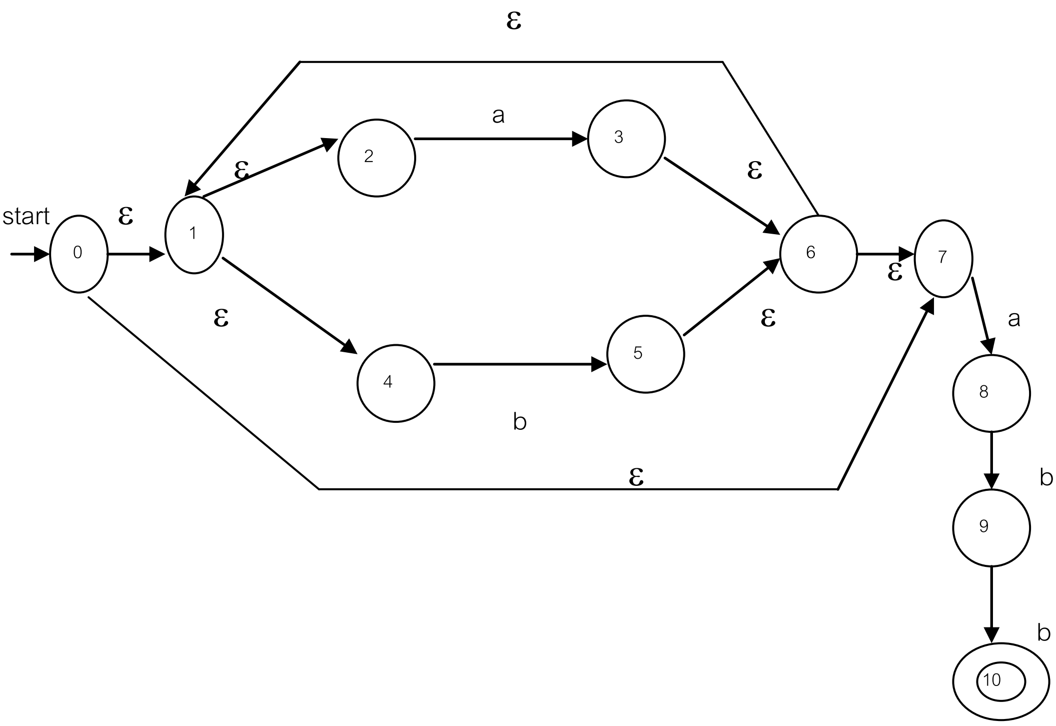
end

Example , Figure below shows NFA accepting the language aa* | bb*.



Example

Figure below shows NFA accepting the language (a|b)*abb.



The start state of the equivalent DFA is ε-closure(0), which is A = {0,1,2,4,7}, since these are exactly the states reachable from state 0 via a path in which every edge is labeled ε. Note that a path can have no edges, so 0 is reached from itself by such a path.

The input symbol alphabet here is {a, b}. The algorithm tells us to mark A and then to compute ε-closure(move(A, a)). We first compute move(A, a), the set of states of N having transitions on a from members of A. Among the states 0, 1, 2, 4 and 7, only 2 and 7 have such transitions, to 3 and 8, so ε-closure(move({0,1,2,4,7}, a)) = ε-closure({3,8}) = {1, 2, 3, 4, 6, 7, 8}. Let us call this set B. Thus, Dtran[A, a]=B.

Among the states in A, only 4 has a transition on b to 5, so the DFA has a transition on b from A to C = ε-closure({5}) = {1, 2, 4, 5, 6, 7}. Thus, Dtran[A, b]=C.

If we continue this process with the now unmarked sets B and C, we eventually reach the point where all sets that are states of the DFA are marked. This is certain since there are "only" 2^{11} different subsets of a set of eleven states, and a set, once marked, is marked forever. The five different sets of states we actually construct are:

A = {0, 1, 2, 4, 7}   D = {1, 2, 4, 5, 6, 7, 9}

B = {1, 2, 3, 4, 6, 7, 8}   E = {1, 2, 4, 5, 6, 7, 10}

C = {1, 2, 4, 5, 6, 7}

State A is the start state, and state E is the only accepting state. The complete transition table Dtran is shown below:

	Input Symbol	
state	a	b
>A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C

Finite Automata with output

One limitation of the finite automaton as we have defined it is that output is limited to a binary signal: "accept" | "don't accept". Models in which the output is chosen from some other alphabet have been considered. There are two distinct approaches; the output may be associated with the state (called a Moore machine) or with the transition (called a Mealy machine). We notice that the two machine types produce the same input-output mappings.

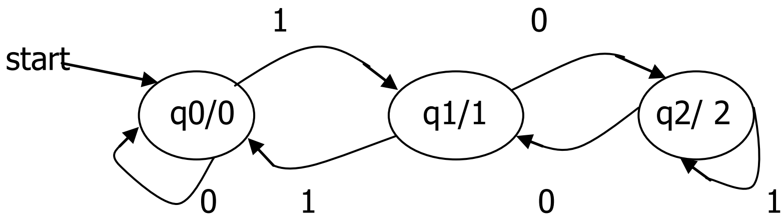
Moore machines

A Moore machine is a six-tuple (K, $\sum$, Γ, δ, χ, s), where K, $\sum$, δ, and s are as in the DFA. Γ is the output alphabet and χ is a mapping from K to Γ giving the output associated with each state. The output of M in response to input $a_1 a_2 \ldots a_n$, $n \geq 0$, is $\chi(q_0)\ \chi(q_1) \ldots \chi(q_n)$, where $q_0, q_2, \ldots, q_n$ is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$. Note that any Moore machine gives output $\delta(q_0)$ in response to input ε. The DFA may be viewed as a special case of a Moore machine where the output alphabet is {0, 1} and state q is "accepting" if and only if $\chi(q) = 1$.

Example

Suppose we wish to determine the residue mod 3 for each binary string treated as a binary integer. To begin, observe that if i written in binary is followed by a 0, the resulting string has value 2*i, and if i in binary is followed by a 1, the resulting string has value 2*i + 1. If the remainder of i/3 is p, then the remainder of 2*i/3 is 2*p mod 3. If p = 0, 1, or 2, then 2*p mod 3 is 0, 2, or 1, respectively. Similarly, the remainder of (2*i + 1)/3 is 1, 0, or 2, respectively.

It suffices therefore to design a Moore machine with three states, q_0, q_1, and q_2, where q_j is entered if and only if the input seen so far has residue j. We define χ (q_j) = j for j = 0, 1, and 2. The following figure shows the transition diagram, where outputs label the states.



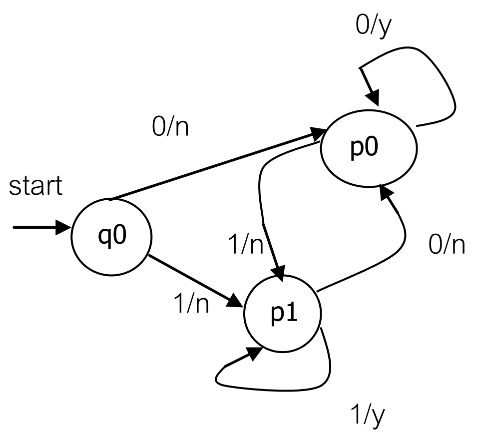
Note! We use q/a as a state indicate that χ(q)=a.

On input 1010 the sequence of states entered is q0, q1, q2, q2, q1, giving output sequence 01221. That is , ε (which has "value" 0) has residue 0, 1 has residue 1, 2 (in decimal) has residue 2, 5 has residue 2, and 10 (in decimal) has residue 1.

Mealy machines

A Mealy machine is a six-tuple (K, $\sum$, Γ, δ, χ, s), where all is as in the Moore machine, except that χ maps K$\times\sum$ to Γ. That is, Γ(q, a) gives the output associated with the transition from state q on input a. The output of M in response to input $a_1a_2 \ldots a_n$, n$\geq$0, is $\chi(q_0, a_1)\ \chi(q_1, a_2) \ldots \chi(q_{n-1}, a_n)$, where $q_0, q_{2, \ldots,}\ q_n$ is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$. Note that this sequence has length n rather than length n + 1 as for Moore machine, and on input ε a Mealy machine gives output ε.

Example,

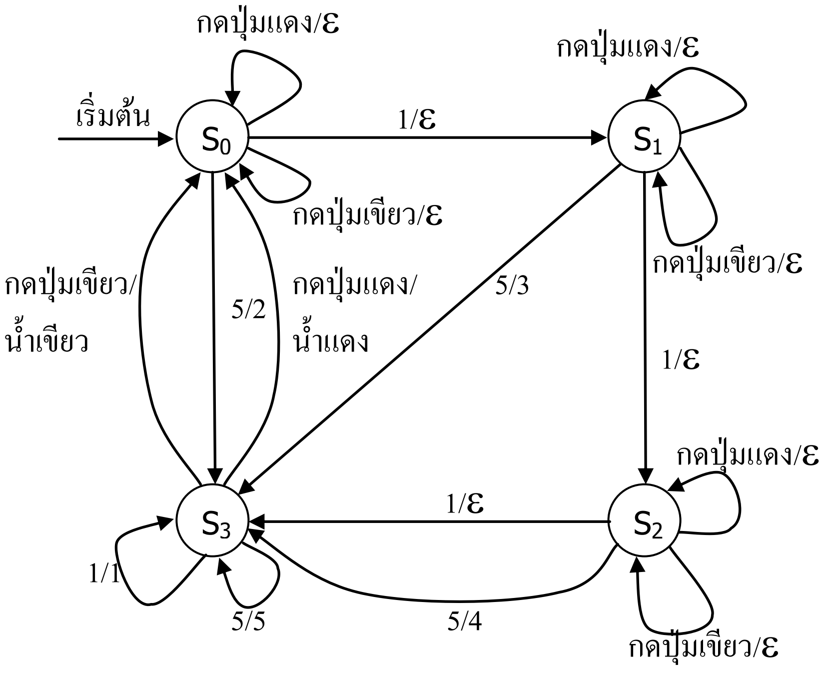


We use the label a/b on an arc from state p to state q to indicate that δ(p, a) = q and χ(p, a) = b. The response of M to input 01100 is nnyny, with the sequence of states entered being $q_0p_0p_1p_1p_0p_0$.

Theory: If M_1 is a Moore machine, then there is a Mealy machine M_2 equivalent to M1.

Theory: If M_1 is a Mealy machine, then there is a Moore machine M_2 equivalent to M1.

ตัวอย่าง การออกแบบเครื่องขายน้ำหวานอัตโนมัติ สมมุติว่าเครื่องๆ นี้รับเฉพาะเหรียญ 1 บาท และ 5 บาทเท่านั้น และน้ำหวานที่ขายราคาถ้วยละ 3 บาท โดยมีน้ำหวานสองประเภท คือ น้ำเขียวและน้ำแดง เมื่อผู้ซื้อหยอดเหรียญมูลค่าครบ 3 บาทแล้ว สามารถเลือกกดปุ่มสีเขียวหรือสีแดงก็ได้ เพื่อรับถ้วยน้ำเขียวหรือน้ำแดงตามลำดับ ในกรณีที่หยอดเหรียญมูลค่าเกิน 3 บาท เครื่องจะทอนเงินให้ด้วย



หรือ แสดงด้วย Transition table ได้ ดังนี้

สถานะ	δ				χ			
	1	5	กดปุ่มเขียว	กดปุ่มแดง	1	5	กดปุ่มเขียว	กดปุ่มแดง
s0	s1	s3	s0	s0	-	2	-	-
s1	s2	s3	s1	s1	-	3	-	-
s2	s3	s3	s2	s2	-	4	-	-
s3	s3	s3	s0	s0	1	5	น้ำเขียว	น้ำแดง