**Closure properties of regular languages**

Regular languages are closed under union operation, i.e. union of 2 regular languages is regular.

Reg lang are closed under intersection operation, complement and concatenation as well.

Eg

1. **Proof of closure under complement**

M = (Q, Sigma, delta, q0, F) represents language L

Then to represent language L’ (complement of L), we can simply take complement of the final states

M’ = (Q, Sigma, Delta, q0, Q - F)

Since this is also a DFA, the language L’ must be regular as well.

1. **Proof of closure under intersection**

Method 1:

Consider L1 = L(M1) and L2 = L(M2), where

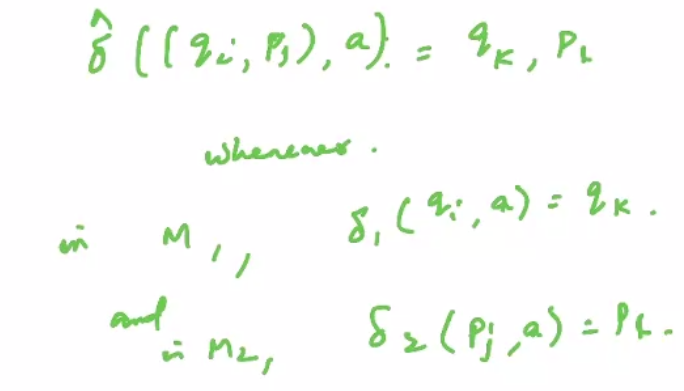
M1 = (Q, Sigma, delta1, q0, F1) and M2 = (P, Sigma, delta2, p0, F2)

Then L1 Intersection L2 can be represented using

M^ = (Q^, Sigma, delta^, q0, F^) where

Q^ = Q x P (Cartesian product)

Delta^ = (qi, pj) Whenever M1 is in state qi and M2 is in state pj



F^ = set of all (qi,pj) where qi belongs to F1 and pj belongs to F2.

Hence if an input symbol w belongs to L1 intersection L2, then it will be accepted by the DFA M^. Hence L1 intersection L2 is regular.

Method 2 Proof

We know by DeMorgan’s laws:



We already know that L1’ and L2’ are regular since reg lang are closed under complement. Similarly we know their union is regular, and so complement of the union is also regular.

1. **Proof of closure under difference**



We know L2’ is regular because of closure under complement. Similarly, we know their intersection must be regular because of proof 2. Hence reg lang are closed under difference operation.

1. **Proof of closure under reversal**

Consider a DFA

M1 = (Q, Sigma, delta, q0, F) that represents language L.

Then for obtaining the reverse of the language L^R, we can do the following:

Make the initial vertices of M1 the final vertices of M2.

Make the final vertices of M1 the initial vertices of M2.

Reverse the direction of all edges in the transition graph

**Homomorphism**

Suppose Sigma and Gamma are the alphabets, then the function

h: Sigma --> Gamma\* is a homomorphism if

For any w belonging to Sigma\*, and w = a1a2a3...an

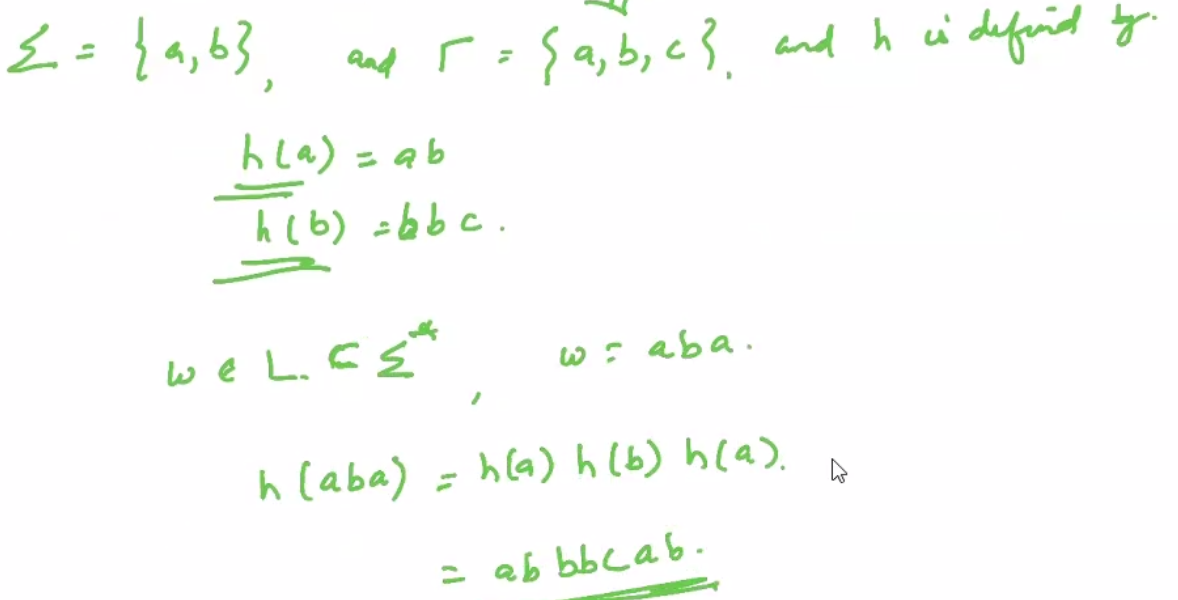
we have

h(w) = h(a1).h(a2)...h(an)

If a language L is given, we define its homomorphic image as

h(L) = {h(w): w belongs to L}

Eg:

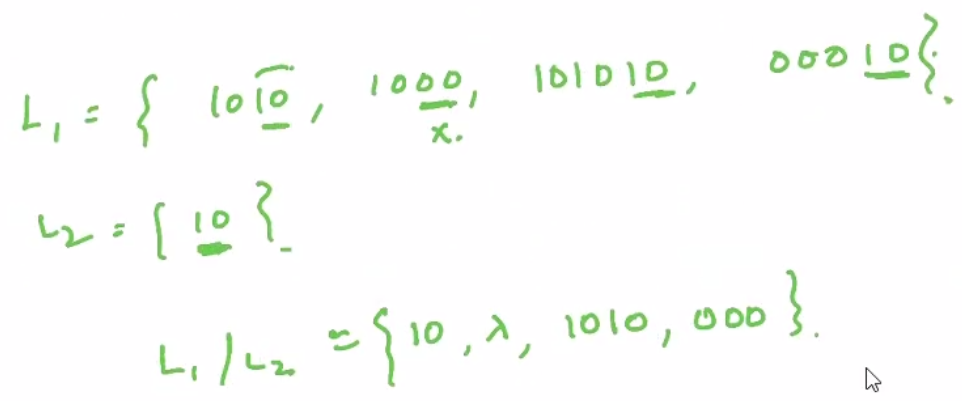


**A family of regular languages is closed under homomorphisms.**

Right and left quotients

Let L1 and L2 be 2 languages having the same alphabet. Then the right quotient of L1 with L2 is defined as:

L1 | L2 = {x: xy belongs to L1 for some y belongs to L2}



**Regular languages are closed under right and left quotient.**

**Identifying non-regular languages**

If there is an expression that requires memory capabilities to represents the strings of the language more than what is possible by a finite state automaton, then the language represented by the expression cannot be represented by an FSA, and hence the language is not regular.

Consider an expression

r = a^n.b^n for n >= 0

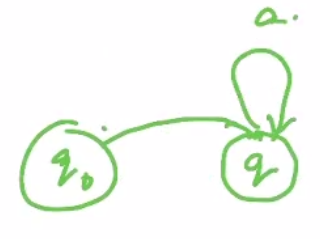
Then L(r) = {lambda, ab, aabb, aaabbb ...}

Let us examine whether the above language is regular or not.

Consider an automaton

M = (Q, {a,b}, delta, q0, F)

Let Delta\*(q0, a^i) = q



Since no of states is not unlimited, our transition will loop at some point.

We know that

Delta\*(q0, a^n) = Delta\*(q0, a^m) = q

Then

Delta\*(Delta\*(q0, a^m), b^m) = f

i.e. Delta(q, b^m) = f which is a final state

Similarly,

Delta\*(Delta\*(q0, a^n), b^n) = f

i.e. Delta(q, b^n) = f which is a final state

This would mean that

Delta(q0, a^n.b^n) = Delta(q0, a^n.b^m) = f, even if m != n

Hence it accepts both a^n.b^n as well as a^n.b^m, however it should have accepted only if m = n.

By design of an FSA,we have no way to represent how many times the symbol a occurred in order to determine how many times the transition to the state f needs to loop.

This proves that the language L is not regular.