Relation b/w FSA and regular grammar

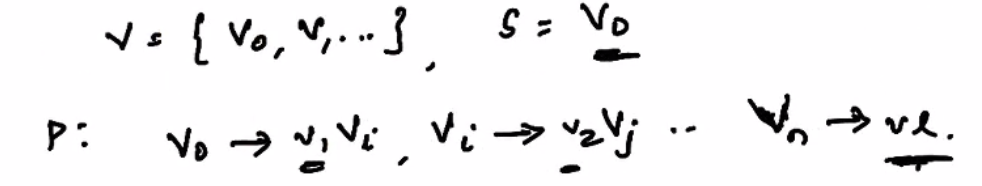
Eg

G = {{s}, {a,b}, S, P}

P = S -> aSb/lambda

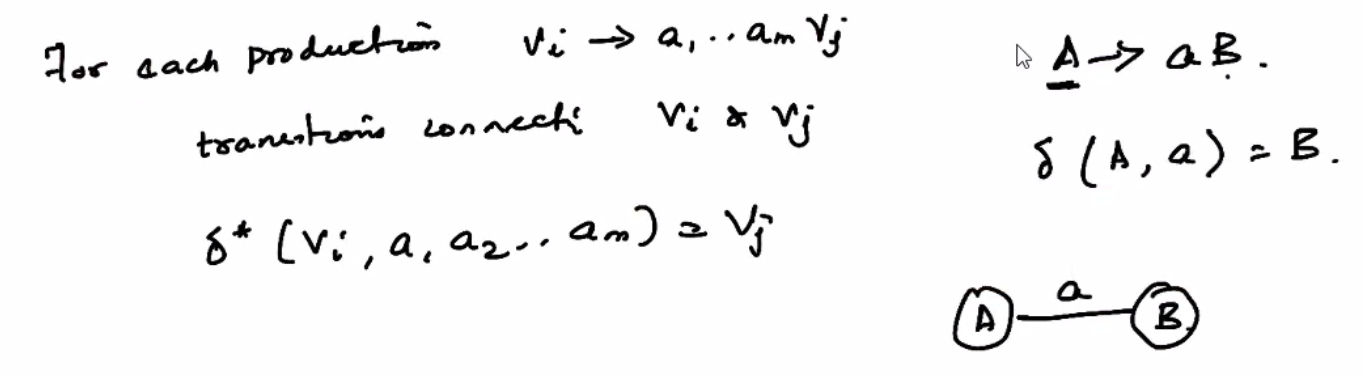
So L = {lambda, ab, aabb, aaabbb ...}

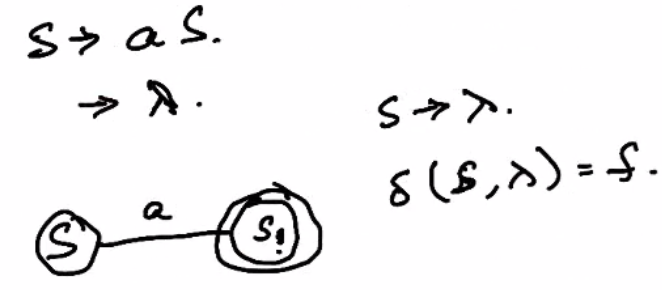
Let G = {V,T,S,P} be a right linear grammar, then L(G) is a regular language, so we can represent it using an FSA.



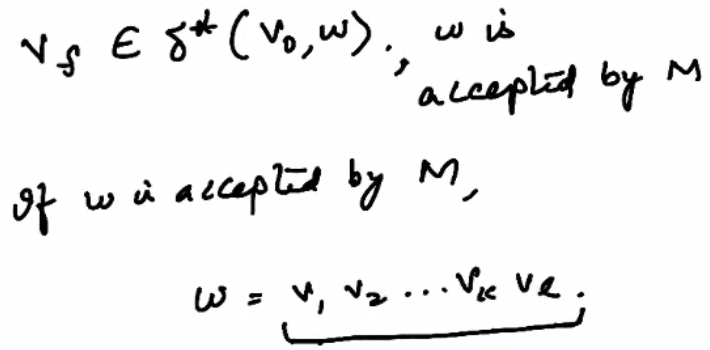
Steps for conversion

1. Start state = v0
2. For every non-terminal variable, have a non-final state vi





1. Given a production, construct the corresponding FSA as shown above.



Construction of NFA from a given grammar

Let

V = {v0,v1}

T = {a,b}

S = {v0}

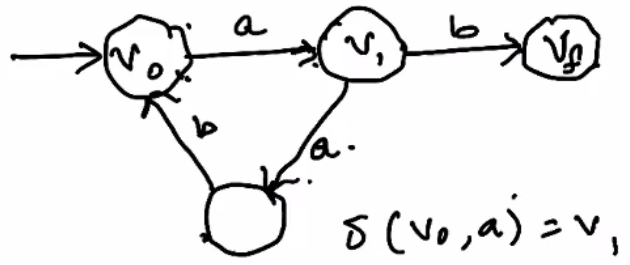
P = V0 -> av1 and v1 ->abv0/b

Delta(v0, a) = v1

Delta(v1, ab) = v0

Delta(Delta(v0,a), ab) = v1

Delta(v1, b) = vf



This is the corresponding NFA.

If L is a regular language on the alphabet Sigma, then there exists a right linear grammar

G = {V,Sigma,S,P}

Such that L = L(G)

Given a DFA that accepts a language L:

M = {Q, Sigma, Delta, q0, F}

Where

Q = {q0,q1...qn}

Sigma = {a1, a2... am}

To construct the RLG from M

V = {q0, q1 .. qn}

T = Sigma

For each transition

Delta(qi, aj) = qk

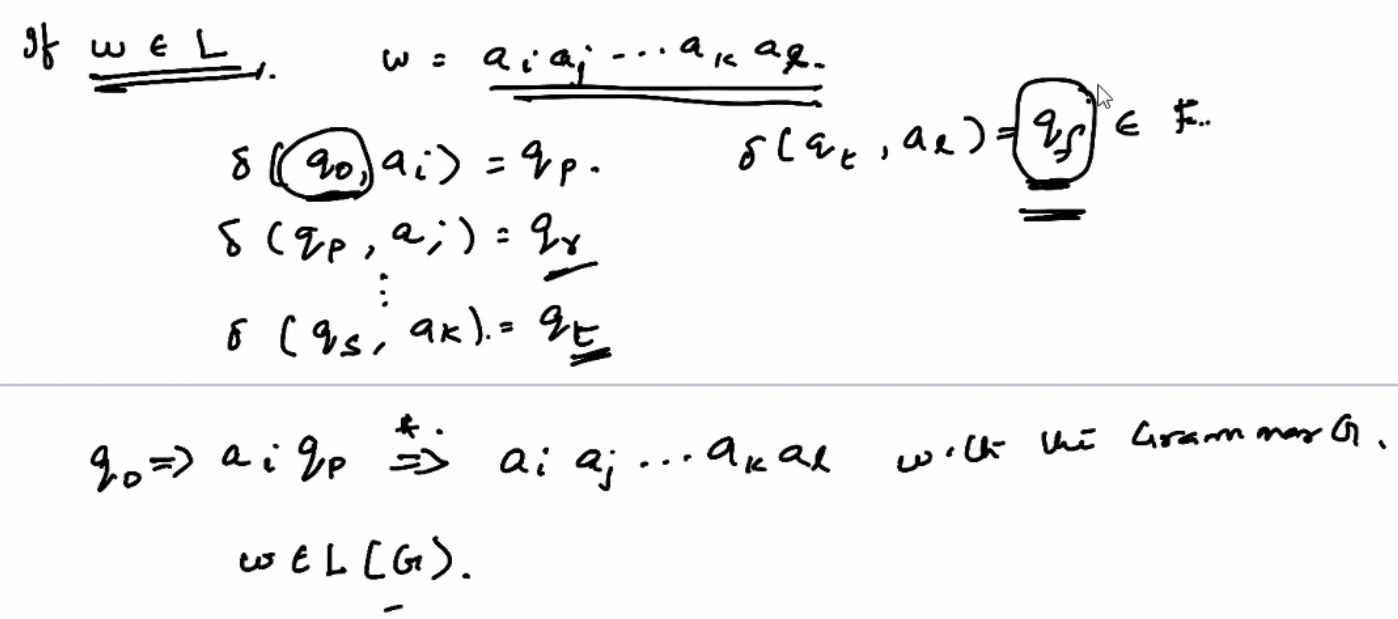
Replace this with the production qi -> ajqk

If qk is in F, then add

Qk -> lambda

To the set of productions

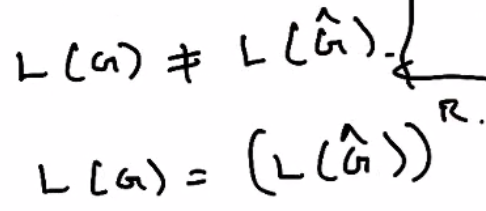
In this way, we have all four V, T, S, P from M. This proves that a right linear grammar G always exists corresponding to an NFA M.



This proves that the 2 representations of the language L are equivalent.

The above also holds true for left linear grammars. This can be observed by simply reversing the productions.

Consider an RLG G and a corresponding LLG G^ then



Closure properties

These properties are used to check whether a family of regular languages are closed under certain operations.

Eg.

For union operation. If L1 and L2 are regular, and if L1 U L2 is also regular, then the family of these 2 regular languages is said to be closed under union operation