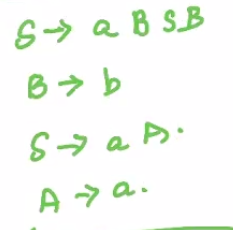
Every production in GNF begins with terminal symbol

Eg



Both productions violate GNF. After normalizing



Now it is in GNF

For every context free grammar G with Lambda not belonging to L(G), there exists a CFG G’ where G’ is in Greibach normal form. In other words, all context free grammars without lambda in their language can be normalized to GNF.

Pushdown Automata (PDA)

A context-free language can be represented using a pushdown automaton. Since context-free languages are a superset of regular languages, pushdown automata can also be used to represent regular languages.

While FSA are memory-less in nature, Pushdown automata have a memory stack of infinite size.

A PDA is represented using a 7-tuple

P = {Q, Sigma, Gamma, Delta, q0, z0, F}

Q - Finite set of start symbols

Sigma - Finite set of alphabet

Gamma - Finite set of stack alphabets

Delta - transition function

q0 - start state

z0 - stack start symbol

F - set of final states

Here the transition function Delta takes 3 arguments:

Delta(q, a, x) ---> P, gamma

q - state in Q

a - either an input symbol in Sigma or Lambda

x - stack symbol at the top of the stack, belonging to Gamma

Returns

P - P is the new state to transition to

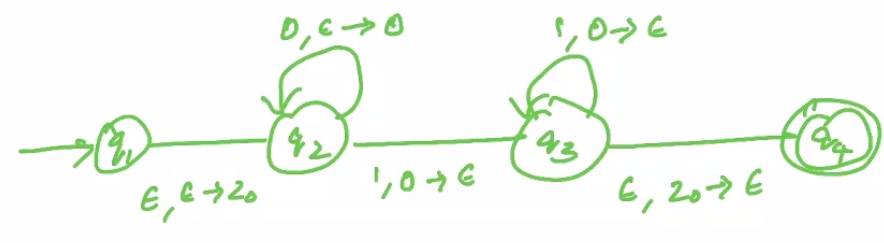
gamma - string of the stack alphabet Gamma that replaces x at the top of the stack

If gamma = Epsilon (empty string), the stack is popped

If gamma = x, the stack remains unchanged (x is popped and then pushed again)

If gamma = yz, the stack is popped, and **z followed by y** are pushed (i.e. x is replaced by z,y)

Eg



This can accept strings of the form 0^n.1^n where n >= 1. This is not a regular language, but it is context-free. Hence it cannot be represented using a FSA, but can be represented using a PDA. Using the stack, we were able to count the number of 0s and 1s, ensure they’re equal and hence fulfil the constraint of the language.