If a grammar **G** is context-free, we can build an equivalent nondeterministic PDA which accepts the language that is produced by the context-free grammar **G**. A parser can be built for the grammar **G**.

Also, if **P** is a pushdown automaton, an equivalent context-free grammar G can be constructed where

**L(G) = L(P)**

In the next two topics, we will discuss how to convert from PDA to CFG and vice versa.

Algorithm to find PDA corresponding to a given CFG

**Input** − A CFG, G = (V, T, P, S)

**Output** − Equivalent PDA, P = (Q, ∑, S, δ, q0, I, F)

**Step 1** − Convert the productions of the CFG into GNF.

**Step 2** − The PDA will have only one state {q}.

**Step 3** − The start symbol of CFG will be the start symbol in the PDA.

**Step 4** − All non-terminals of the CFG will be the stack symbols of the PDA and all the terminals of the CFG will be the input symbols of the PDA.

**Step 5** − For each production in the form **A → aX** where a is terminal and **A, X** are combination of terminal and non-terminals, make a transition **δ (q, a, A)**.

Problem

Construct a PDA from the following CFG.

**G = ({S, X}, {a, b}, P, S)**

where the productions are −

**S → XS | ε , A → aXb | Ab | ab**

Solution

Let the equivalent PDA,

P = ({q}, {a, b}, {a, b, X, S}, δ, q, S)

where δ −

δ(q, ε , S) = {(q, XS), (q, ε )}

δ(q, ε , X) = {(q, aXb), (q, Xb), (q, ab)}

δ(q, a, a) = {(q, ε )}

δ(q, 1, 1) = {(q, ε )}

Algorithm to find CFG corresponding to a given PDA

**Input** − A CFG, G = (V, T, P, S)

**Output** − Equivalent PDA, P = (Q, ∑, S, δ, q0, I, F) such that the non- terminals of the grammar G will be {Xwx | w,x ∈ Q} and the start state will be Aq0,F.

**Step 1** − For every w, x, y, z ∈ Q, m ∈ S and a, b ∈ ∑, if δ (w, a, ε) contains (y, m) and (z, b, m) contains (x, ε), add the production rule Xwx → a Xyzb in grammar G.

**Step 2** − For every w, x, y, z ∈ Q, add the production rule Xwx → XwyXyx in grammar G.

**Step 3** − For w ∈ Q, add the production rule Xww → ε in grammar G.