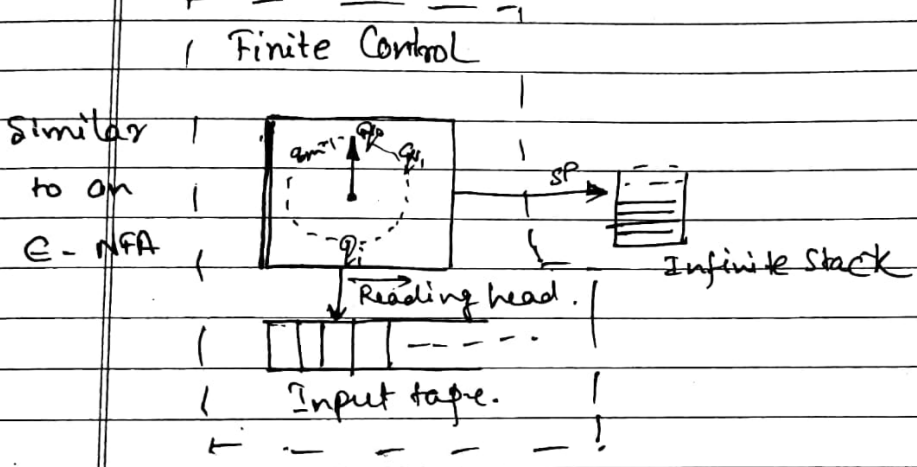
Push-down Automata (PDA)

- an automata equivalent to CFGs in language defining power

* Only the non-deterministic PDA defines all the CFL's

- But the deterministic version models parsers
  - Most programming languages have deterministic PDA's.



* block diagram of a PDA

- A PDA is defined to have 7 components such as
  1. A finite set of states (Q, typically)
  2. An input alphabet (Σ, typically)
  3. A stack alphabet (Γ, typically)
  4. A transition function (δ, typically)

$$\delta(q_i, a, Z) = \text{A set of zero or more actions of the form } (p, \alpha)$$

q_i: A state in Q
a: An input, either a symbol in Σ or ε
Z: A stack symbol in Γ
p: A state
α: A string of stack symbols

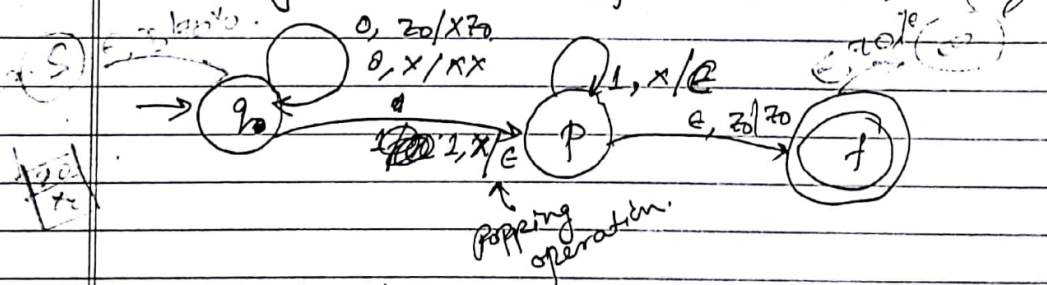
  5. A start state (q_0 in Q typically)
  6. A start symbol (Z_0 in Γ typically)
  7. A set of final states ($F \subseteq Q$ typically)

* Conventions:
  - a, b, c - --- are input symbols
    - Sometimes we allow ϵ as input symbol.
  - ---X, Y, Z are stack symbols
  - ... w, x, y, z are strings of input symbols
  - α, β, ... are strings of stack symbols

- Actions of the PDA
  - If $\delta(q, a, Z)$ contains (p, α) among its actions, then one thing the PDA can do in state q, with a at the front of the input and Z on top of stack is:
    1. Change the state to p
    2. Remove a from the front of the input (but a may be ϵ)
    3. Replace Z on the top of the stack by α.

Example:

Design a PDA to accept $\{0^n 1^n \mid n \geq 1\}$



q = Start state
p = an internal state
f = final state.

The transitions:

$\delta(q, 0, Z_0) = \{(q, XZ_0)\}$
$\delta(q, 0, X) = \{(q, XX)\}$
$\delta(q, 1, X) = \{(p, \epsilon)\}$
$\delta(p, 1, X) = \{(p, \epsilon)\}$
$\delta(p, \epsilon, Z_0) = \{(f, Z_0)\}$

- is a triple (q, w, α)
  - q: Current state
  - w: the remaining input
  - α: the stack contents,

* The "Goes-to" Relation
  - To say the ID I can become ID J in one move of the PDA, we write $I \vdash J$
  - Formally,
    $(q, aw, X\alpha) \vdash (p, w, \beta\alpha)$ for any w and α if (q, a, X) contains (p, β)
  - Extend $\vdash$ (goes to) to $\vdash^*$ meaning "zero or more moves" by.

Basis: $I \vdash^* I$.

Induction: If $I \vdash^* J$ and $J \vdash K$ then $I \vdash^* K$

Example

$(q, 011, Z_0) \vdash (q, 11, XZ_0)$
$\vdash (q, 1, XXZ_0)$
$\vdash (p, 1, XZ_0)$
$\vdash (p, \epsilon, Z_0)$
$\vdash (f, \epsilon Z_0)$ Accept

$(q, 011, Z_0) \vdash (q, 11, XZ_0)$
$\vdash (p, 1, Z_0)$ (Reject)

* Language of a PDA

- The common way to define the language of a PDA is by final state.
- If P is a PDA then $L(P)$ is the set of strings w such that $(q_0, w, Z_0) \vdash^* (f, \epsilon, \alpha)$ for final state f and any α

* Language of a PDA -(2)

- Another language defined by the same PDA is by empty stack
- If P is a PDA then $N(P)$ is the set of strings w such that $(q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)$ for any state q.

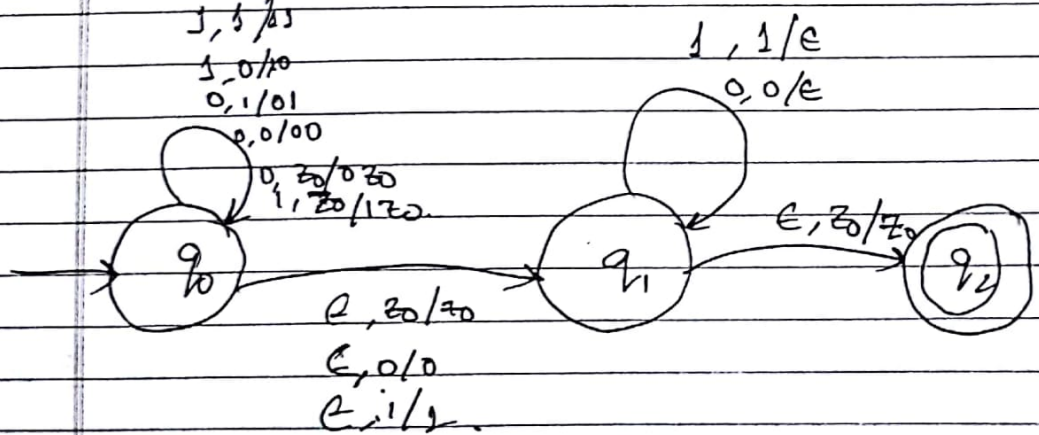
Deterministic PDA's

To be deterministic, there must be at most one choice of move for any state q, input symbol a and stack symbol X

- In addition, there must not be a choice between using input ϵ or real input.
- Formally $\delta(q, a, X)$ and $\delta(q, \epsilon, X)$ cannot both be non-empty.

Design a deterministic PDA to accept $\{ww^R \mid w \in \{0,1\}^*\}$

Can you design a deterministic PDA to accept $\{ww^R w \mid w \in \{0,1\}^*\}$



- CFG's and PDA's are both useful to deal with properties of CFL's
- PDA's being 'algorithmic' are often easier to use when arguing that a language is CFL.
- But all depends on knowing that CFG's and PDA's both define the CFL's

Converting a CFG to a PDA

- Let $L = L(G)$
  - Construct PDA P such that $N(P) = L$
    - ✓ P has:
      - ✓ one state q
      - ✓ Input symbols = terminals of G
      - ✓ Stack symbols = all symbols of G
      - ✓ Start symbol = start symbol of G.
- Given input w, P will step through a left most derivation of w from the start symbol S.
- Since P can't know what this derivation is, or even what the end of w is, it uses non-determinism to guess the production to use at each step

~~* Given input w, P will step.~~

* At each step, P represents some left-sentential form (step of a left most derivation)

* If the stack of P is α, and P has so far consumed w' from its input then P represents left-sentential form $w\alpha$.

* At empty the input consumed is a string in L(G)

* Transition function of P

1. $\delta(q, a, a) = \{(q, \epsilon)\}$ [Type 1 rule]

2. If $A \to \alpha$ is a production of G then $\delta(q, \epsilon, A)$ contains (q, α) [Type 2 rule]

Proof : That $L(P) = L(G)$

- We need to show that
  $(q, wx, S) \xrightarrow{*} (q, x, \alpha)$ for any α
  if and only if $S \overset{*}{\Longrightarrow}_{lm} w\alpha$
  [In pic mobile]

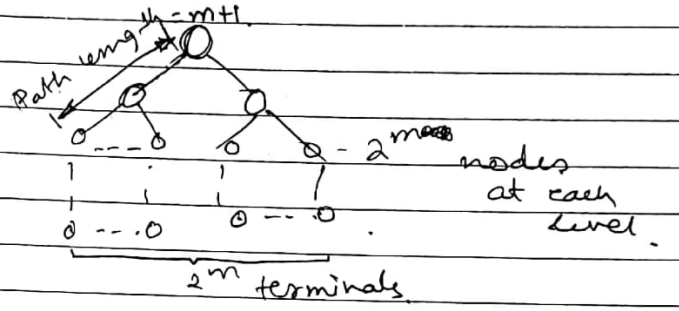
Pumping Lemma for CFL's

* For every context free language L
  There is an integer n such that
  For every string z in L of length $\geq n$
  There exists $z = uvwxy$ such that

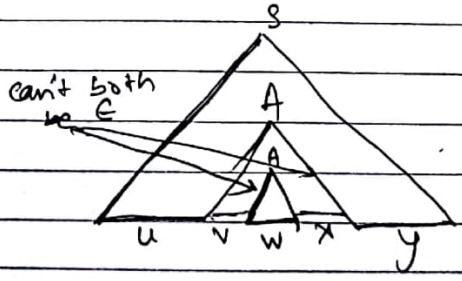
1. $|vwx| \leq n$.
2. $vx \neq \epsilon$
3. For all $i \geq 0$ $uv^i wx^i y$ is in L.

Let us start with a CNF grammar for $L - \{\epsilon\}$
Let the grammar have m variables
Pick $n = 2^m$
Let $|z| \geq n$

We claim ("lemma 1") that a parse tree with yield z must have a path of length $(m+2)$ or more.



- Now we know that the parse tree for z has a path with at least m+1 variables
- Consider some longest path.
- There are only m different variables, so among the lowest m+1 we can find two nodes with the same label, say A.
- The parse tree thus looks like.



can't both be ϵ

v & x both can't be ϵ

Non-decision Properties

Many questions that can be decided for regular sets cannot be decided for CFL's

- Example : Are two CFL's same?
- Example : Are two CFL's disjoint?

* We need theory of Turing Machines (TM's) and decidability to prove no algos exists for the above mentioned problems
  - ~~Eliminate variables that generate no.~~

★ Testing Emptiness.

- ~~Eg~~ Eliminate variables that generate no terminate string
- If the start symbol is of these then the CFL is empty; ~~then~~ otherwise not.

★ Testing Infiniteness

- Suppose the CFG has a repeating variable A, which is neither nullable nor a useless symbol. Say $A \Rightarrow^{+} xAy$, the grammar has no ϵ-production nor a unit production and x and y cannot simultaneously be empty.

So $S \overset{*}{\Rightarrow} uAv \overset{*}{\Rightarrow} w$, u, v, w are all strings of terminals
And $A \Rightarrow z$
Then $S \overset{*}{\Rightarrow} uAv \overset{*}{\Rightarrow} ux^m A y^n v \Rightarrow ux^m z y^n v$ is possible for all n, So $L(G_1)$ is infinite.

Closure of CFL's

Under union

- Let L and M be CFL's with grammar G_1 and H respectively
  - Assume G_1 and H have no variables in common.
  - Names of variables do not affect the language
  - Form a new grammar for $L \cup M$ by combining all the symbols and productions of G_1 & H
  - Then add a new start symbol S
  - Add productions $S \to S_1 | S_2$

- In the new grammar, all derivations start with S
- In the first case, the result must be a string in $L(G_1) = L$ and in the second case a string in $L(H) = M$.

The first step replaces S by either S_1 or S_2

Closure under Star

- Let L have grammar G_1 with start symbol S_1
- Form a new generation for L^* by introducing to a new start symbol S and the production $S \to S_1 S | \epsilon$

• The right most derivation from S generates a sequence of zero or more S_i's, each of which generates some string in L.

Closure of CFL's under Reversal

If L is a language with grammar G, form a grammar for L^R by reversing the right side of every production.

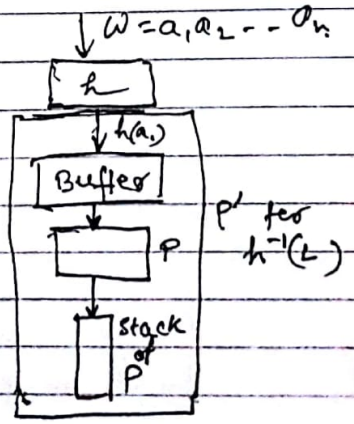
• Example : Let G have $S \to 0S1 \mid 01$
The reversal of L(G) has grammar
$S \to 1S0 \mid 10$

* Closure under Homomorphism
• Let L be a CFL with grammar G.
• Let h be a homomorphism on the terminal symbols of G.
• Consider a grammar for h(L) by replacing each terminal symbol a by h(a)

* Inverse Homomorphism

• If h is a homomorphism and L is any language then $h^{-1}(L)$ is the set of strings w such that h(w) is in L

• Closure of CFL's under Inverse Homomorphism.



Read first remaining symbol in buffer as if it were input to P.

Date 23/11/17

Formal construction of P'

- States are pairs [q, b] where:
  1. q is a state of P
  2. b is a suffix of h(a) for some symbol a,
- Stack symbols of P' are those of P
- Start state of P' is [q0, ε]
- Input symbols of P' are the symbols to which h applies
- Final states of P' are the states [q, ε] such that q is a final state of P.

* Transitions of P'

1. $\delta([q, \epsilon], a, X) = \{[q, h(a)], X\}$ for any input symbol a of P' and any stack symbol X.
   ✓ when the buffer is empty P' can rebuild it
2. $\delta([q, bw], \epsilon, X)$ contains $([p, w], \alpha)$ if $\delta(q, b, X)$ contains (p, α) where b is either an input symbol of P or ε
   ✓ simulate P from the buffer.

Example: Using the Pumping Lemma, prove that the following languages are not context-free

(i) $\{0^i 1 0^i 1 0^i \mid i \geq 1\}$

(ii) $\{0^i 1^i 2^i \mid i \geq 1\}$.

Self study:

1. Non-closure of CFL's under intersection
2. Non-closure of CFL's under difference

Prove that intersection of a CFL with a RL is always a CFL