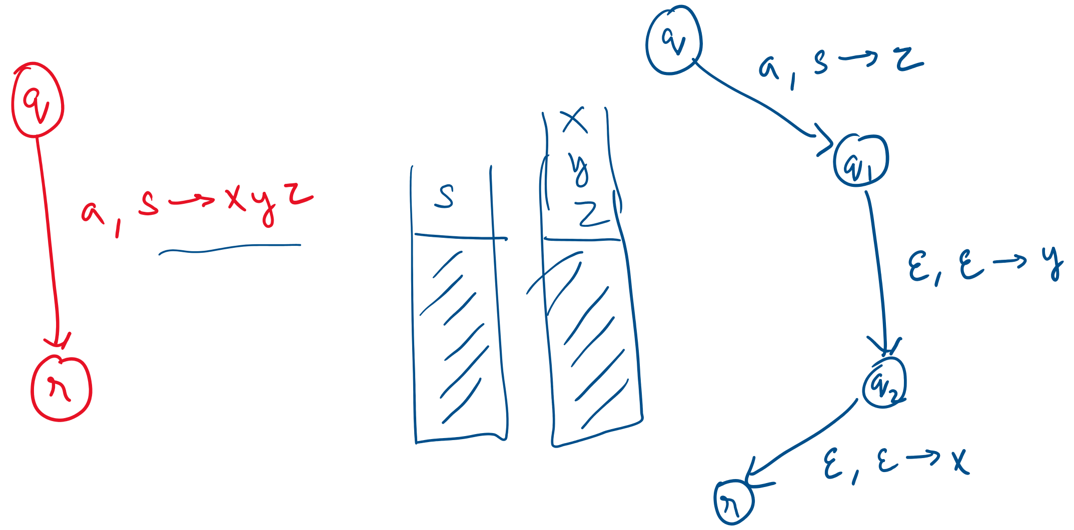
Equivalence of CFGs and PDAs

14 February 2021 23:14

Shorthand Notation : $a, s \to \underbrace{xyz}_{\in \Gamma^*}$



Theorem 2.20 : A language is context-free if and only if some PDA recognizes it.

Lemma 2.21 : If a language is context-free, then some PDA recognizes it.

Proof : We have A which is generated by a CFG. We will construct a PDA P for A.

P accepts $w \iff w$ is generated by G.

The PDA can access only the top of the stack.

So it cannot apply production rules to the

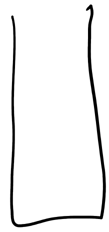
$\phi\,\vartheta\,\phi\,\Gamma\,\lambda\,\Upsilon$

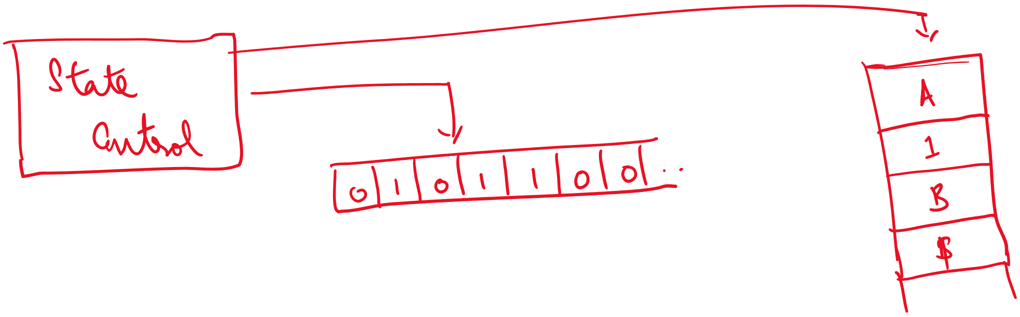
$A \to 0A1 \mid \varepsilon$

So it cannot apply ...

intermediate symbols.

$A \to 0A1 \mid \varepsilon$





1. Keep \$ in the stack at the beginning, followed by the start variable.

2. Repeat
   (a) If top = variable, choose a substitution rule non. det. and replace.

   (b) If top = terminal, pop off and verify that the next symbol in the input is the same. If yes, advance. If not, reject.

   (c) If top = \$, move to accept state.

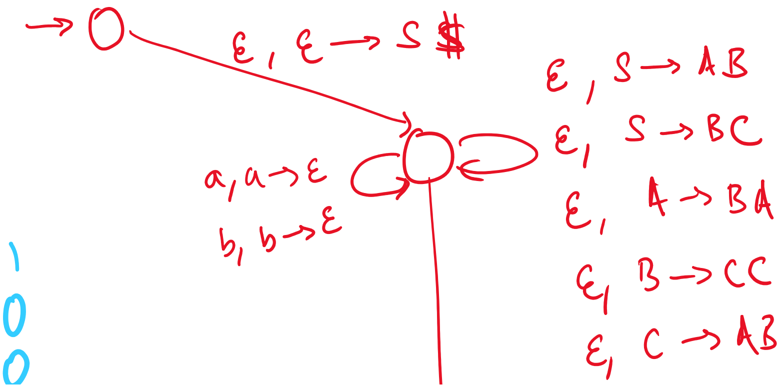
$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

1
0
0



$C \rightarrow AB \mid a$

0
0
1
B
0
\$

$\varepsilon, \$ \rightarrow \varepsilon$

$\varepsilon, C \rightarrow AB$
$\varepsilon, A \rightarrow a$
$\varepsilon, B \rightarrow b$
$\varepsilon, C \rightarrow a$

<u>Proof</u>: $P = (Q, \Sigma, \Gamma, \delta, q, F)$.

$Q = \{ q_{start}, q_{loop}, q_{accept} \} \cup E$

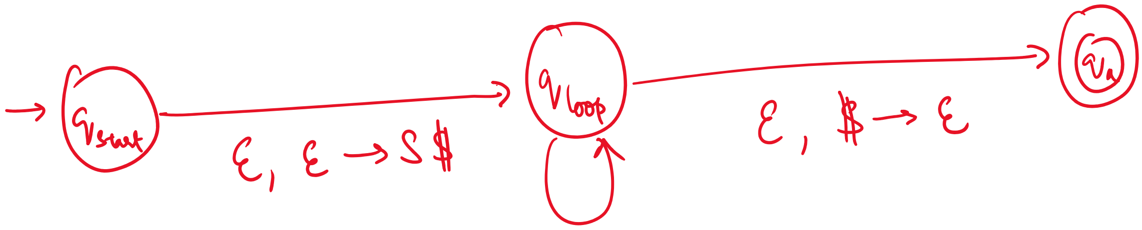
additional states required to implement shorthand notation

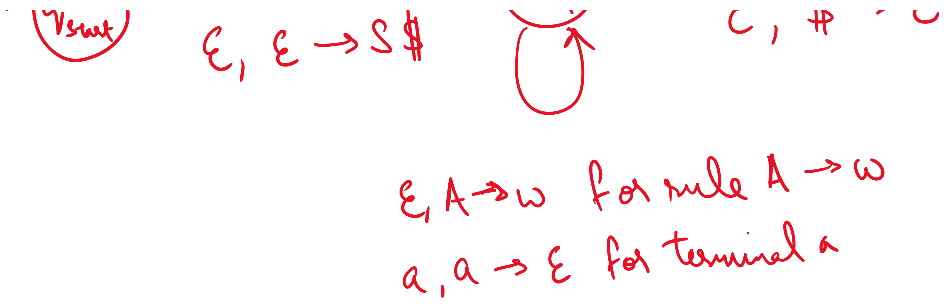
$\delta(q_{start}, \varepsilon, \varepsilon) = \{ (q_{loop}, S\$) \}$

$\delta(q_{loop}, \varepsilon, A) = \{ (q_{loop}, w) \mid \text{where } A \rightarrow w \text{ is a rule in } R \}$

$\delta(q_{loop}, a, a) = \{ (q_{loop}, \varepsilon) \}$

$\delta(q_{loop}, \varepsilon, \$) = \{ (q_{accept}, \varepsilon) \}$





Exercise: Read Example 2.25.

Lemma 2.27: If a pushdown automaton recognizes a language, then it is context-free.

Assume WLOG,
1. PDA P has a single accepting state, q_{accept}.
2. P empties stack before accepting.
3. Each transition either pushes, or pops, but not both.

GOAL: Obtain CFG G, that generates all the strings that can take P from q_{start} to q_{accept}.

Variable A_{pq} will generate all the strings that can take P from state p to state q.

$A_{p,q}$ = { all strings that move P from

$A_{p,q} = \{$ all strings that move P from

$(p, \text{empty stack}) \longrightarrow (q, \text{empty stack})\}$

$A_{q_{start}, q_{accept}}$ generates $L(P)$.

For any string x, P's first move must be a push. The last move must be a pop. There are two possibilities:

* last move pops the same symbol that was pushed in first move. ($\Rightarrow$ Stack never gets empty till the end)

$$A_{pq} \longrightarrow a \, A_{rs} \, b$$

where a is input read in first move

b is input read in last move

and state r follows p and q follows s.

* Stack becomes empty in between, at state r.

$$A_{pq} \longrightarrow A_{pr} A_{rq}.$$

… }) and

Proof: let $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$ and

let us construct G.

G has variables $\{A_{pq} \mid p, q \in Q\}$.

The start variable is $A_{q_0, q_{accept}}$.

Rules of G are:

* For each $p, q, r, s \in Q$, $t \in \Gamma$, $a, b \in \Sigma_\varepsilon$

  If $(r, t) \in \delta(p, a, \varepsilon)$ and $(q, \varepsilon) \in \delta(s, b, t)$,

  add $A_{pq} \to a A_{rs} b$

* For each $p, q, r \in Q$,

  add $A_{pq} \to A_{pr} A_{rq}$

* For all $p \in Q$, add $A_{pp} \to \varepsilon$.

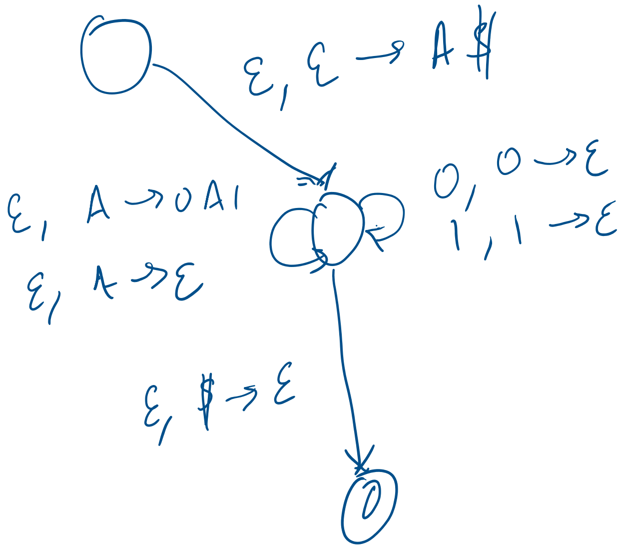
Now we have to show that $A_{p,q}$ generates the string x if and only if x can take P from p with empty stack to q with empty stack.

Both directions of the proof use induction.

Claim 2.30 and Claim 2.31.

$A \to 0A1$

$A \to \varepsilon$



$\cancel{0}\,\cancel{0}\,\cancel{1}$

\$