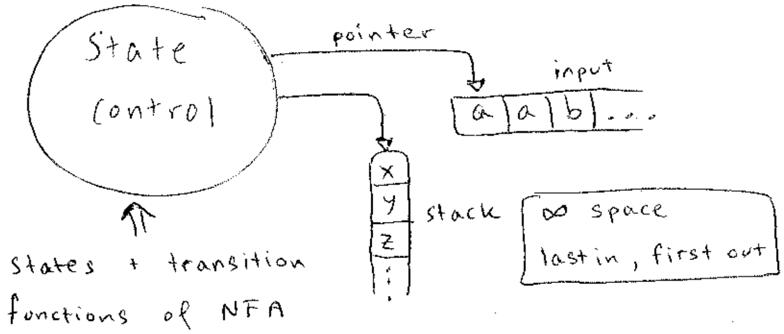
Chapter 2.2 : Pushdown Automata (PDA)



- Basically, a PDA is an NFA with a stack
- A context-free language (CFL) is
  Generated by a CFG
  Recognized by a PDA

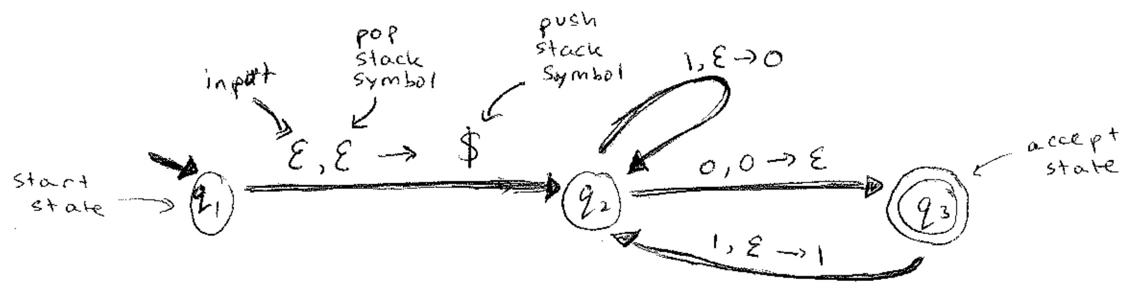


Figure 1 : PDA P

CFG: Context-free grammar
NFA: Nondeterministic Finite Automata

The PDA P, drawn above, can be described by the set $(Q, \Sigma, \Gamma, \delta, q_0, F)$

Q : the set of states $= \{q_1, q_2, q_3\}$

Σ : the input alphabet $= \{0, 1\}$

Γ : the stack alphabet $= \{0, 1, \varepsilon, \$\}$

δ : the transition function of the form $f: Q \times \Sigma_\varepsilon \times \Gamma \to \mathcal{P}(Q, \Gamma)$ (usually represented as a state digram)

q_0 : start state $= q_1$

F : the accept states $= \{q_3\}$

Rules to accept a string ω :

- if ω is made of symbols in Σ
- if the PDA can get from the start state to at least one accept state following δ

} then $\omega \in L(PDA)$ and the PDA accepts ω.

In Figure 1:

Input = 110

State	input	stack
q_1	110	Γ
q_2	110	\$
q_2	110	0, \$
q_2	110	0, 0, \$
q_3 (accept state)	110 _	0, \$

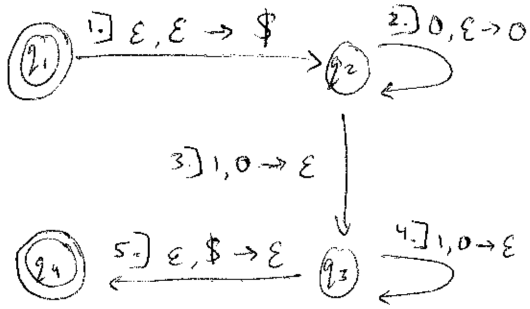
- ω is made of symbols in Σ ✓
- follows δ ✓

so $110 \in L(P)$ and P accepts 110

Example: PDA M where $L(M) = \{0^n 1^n \mid n \geq 0\}$

$$M = (\underbrace{\{q_1, q_2, q_3, q_4\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, \underbrace{\{0, \$\}}_{\Gamma}, \underbrace{\delta}_{\delta}, \underbrace{q_1}_{q_0}, \underbrace{\{q_1, q_4\}}_{F})$$

δ is defined as:



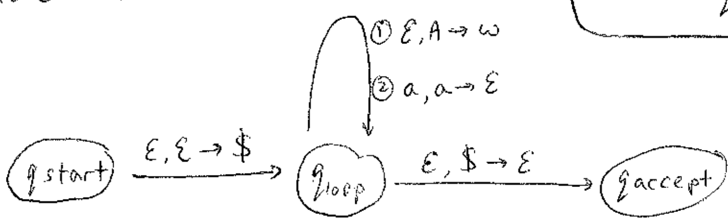
Note:
The \$ helps to mark an empty stack.

Steps:

1. Mark empty stack, $\Rightarrow q_2$
2. If input is 0, push 0. Ignore current stack, stay at q_2
3. If input is 1 and 0 ontop of stack, pop 0 and $\Rightarrow q_3$
4. Repeat step (3.)
5. If no more input, empty stack (\$) $\Rightarrow q_4$ and ACCEPT (else REJECT)

A CFG is equivalent to PDA: We can switch between the two of them!

From CFG to PDA:



① If top of stack is variable (A), ignore input and replace A with rule ($A \to w$) from CFG

② If top of stack is terminal (a), compare it to the input.
- if they match, go on. Else, reject the nondeterministic branch.

③ When stack is empty, ACCEPT!!

From PDA to CFG:

G will have rules to move from every state to every other state in the PDA

Start variable = $A_{q_{start}\, q_{accept}}$, the rule to move from the start to accept state

Each variable can be either a:

Combo of 2 states ($A_{13} \to A_{12} A_{23}$)

or

Input sequence where a pushed symbol is popped ($A_{23} \to 0 A_{22} 1$)

For M, G will have the following variables:

$A_{14} \to A_{12} A_{24} \mid A_{13} A_{34} \mid A_{14} A_{44}$

$A_{11} \to A_{12} A_{21} \mid A_{13} A_{31} \mid A_{14} A_{41}$

$\vdots$

$A_{44} \to \varepsilon$

Switching between a CFG and PDA relies on the following theorem:

THM: $A_{pq} \Rightarrow x$ iff x can take a PDA from p with an empty stack to q with an empty stack.

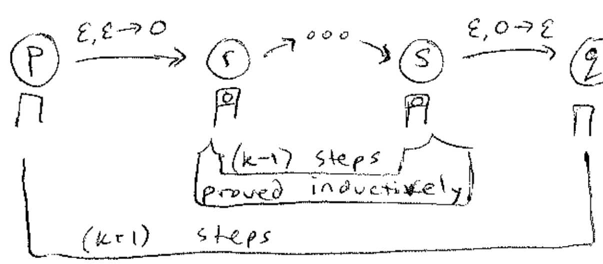
Here, we summarize the two-part proof.

PART 1

To show x taking PDA from p to q means $A_{pq} \Rightarrow x$, we use induction

Base case: q_1 a state to itself $(A_{q_1 q_1})$ takes 0 steps, generates ε.

Induction: If A_{pq} takes $k+1$ steps, then getting from the State after p to the state before q will take $k-1$ steps, and can generate the string based on our inductive hypothesis

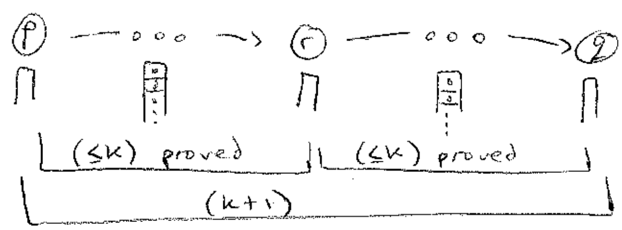


$A_{rs} \Rightarrow y$ and

$x = 0y0$ so

$A_{pq} \Rightarrow x$

Or if the stack is EVER empty at a middle state, then divide the string that way, and you're done!



$A_{pr} \Rightarrow y$ and $A_{rq} \Rightarrow z$

$x = yz$ so

$A_{pq} \Rightarrow x$

PART 2

To show $A_{pq} \Rightarrow x$ means x takes PDA from p to q, we just do the proof in part 1 in reverse. Base case: $A_{pp} \to \varepsilon$ means a state goes to itself with an empty string, we divide the variables and prove that this theorem is true for more than 0 steps

MAIN POINTS

- → PDA's recognize CFL's AND
- → regular languages are recognized by finite automaton AND
- → finite automaton are PDA's without a stack so...
- → regular languages are CFL's!!