Modeling Computation

Introduction to Formal Languages and Automata
Pushdown Automata
Pushdown Automata (PDAs)

PDAs recognize CFGs.
Pushdown Automata (PDAs)

\[ A = (Q, \Sigma, \Gamma, \Delta, s_0, Z_0, F) \]

- **\( Q \)** = States
- **\( \Sigma \)** = Input alphabet
- **\( \Gamma \)** = Stack symbols
- **\( \Delta \)** = Transition function
- **\( s_0 \)** = Initial state
- **\( Z_0 \)** = Empty/bottom of stack symbol
- **\( F \)** = Set of favorable
If PDA
- is in state $s$
- reads $@$ from top
- pop $\beta$ from stack

Then
- move to state $q$
- push $\gamma$ to the stack

$((s, \alpha, \beta), (q, \gamma)) \in \Delta$
PDA Acceptance

PDA $A$ accepts a string $w$ if there exists a sequence of transitions such that

- All symbols of input have been read
- AND (the stack is empty OR $A$ is in a favorable state)
Graphical Representation of PDA

- tape symbol
- top of stack
- new top of stack

Start state: $S_0$

Transition: $a, \beta/\lambda \rightarrow \gamma$
push only

Favorable state: $T$

Transition: $a, \delta/\lambda \rightarrow \beta\gamma$
pop only
Edge label $a, \beta / \gamma$

\[ a \in (\Sigma \cup \{\epsilon\}) \text{ is read from tape} \]
\[ \beta \in \Gamma^* \text{ is popped from stack} \]
\[ \gamma \in \Gamma^* \text{ pushed to stack} \]
Example: PDA that accepts

\[ L = \{a^n b^n \mid n > 0\} \]

Idea: push "a"s to stack then, for each "b", pop "a" from stack.
Nondeterministic PDAs

Or

Or
Some languages can be accepted by a nondeterministic PDA, but not by a deterministic PDA.

Ex: $L = \{ w w^R \mid w \in \Sigma^* \}$

Some languages are not context-free, so no PDA can recognize them.

Ex: $L = \{ a^n b^n c^n \mid n > 0 \}$
Construct PDA to recognize $L = \{ww^R \mid w \in \Sigma^*\}$
Construct a PDA that recognizes strings that have an equal number of 0s and 1s.
Construct a CFG that generates strings that have an equal number of 0s and 1s.