Pushdown automata (PDA) are abstract automata that accept all context-free languages. PDAs are essentially NFAs with an additional infinite stack memory. (Or NFAs are PDAs with no additional memory!)
Pushdown Automata

- Input is read left-to-right
- Control has finite memory (NFA)
- State transition depends on input and top of stack
- Control can push and pop symbols to/from the infinite stack
Let’s look at $L = \{ a^n b^n \mid n \geq 0 \}$

How can we use a stack to recognize $w \in L$?

1. Push a special bottom of stack symbol $\$\$ to the stack
2. As long as you are seeing $a$’s in the input, push an $a$ onto the stack.
3. While there are $b$’s in the input AND there is a corresponding $a$ on the top of the stack, pop $a$ from the stack
4. If at any point there is no $a$ on the stack (hence you encounter $\$\$), you should reject the string – not enough $a$’s!
5. If at the end of $w$, the top of the stack is NOT $\$\$, reject the string – not enough $b$’s.
6. Otherwise accept the string.
How can we use a PDA to recognize

\[ L = \{ w \mid n_a(w) = n_b(w) \} \]

Remember how we argued that the grammar generates such strings

- Keep track of the difference of counts

We do something similar but using the stack.

- Push a special bottom of stack symbol $ to the stack
- An a in the input “cancels” a b on the top of the stack, otherwise pushes an a
- A b in the input “cancels” an a on the top of the stack, otherwise pushes a b
- At the end nothing should be left on the stack except for the $, if not reject.
We have two alphabets $\Sigma$ for symbols of the input string and $\Gamma$ for symbols for the stack. They need not be disjoint.

Define $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$

A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where $Q, \Sigma, \Gamma,$ and $F$ are finite sets, and

- $Q$ is the set of states
- $\Sigma$ is the input alphabet, $\Gamma$ is the stack alphabet
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$ is the state transition function. ($\mathcal{P}(S)$is the power set of $S$. Earlier we used $2^S$.)
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of final or accepting states.
A PDA computes as follows:

- Input $w$ can be written as $w = w_1 w_2 \cdots w_m$ where each $w_i \in \Sigma \epsilon$. So some $w_i$ can be $\epsilon$.
- There is a sequence of states $r_0, r_1, \ldots, r_m$, $r_i \in Q$.
- There is a sequence of strings $s_0, s_1, \ldots, s_m$, $s_i \in \Gamma^*$. These represent sequences of stack contents along an accepting branch of $M$’s computation.

- $r_0 = q_0$ and $s_0 = \epsilon$.
- $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, $i = 0, 1, \ldots, m - 1$
- $a$ is popped, $b$ is pushed, $t$ is the rest of the stack.

- $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma \epsilon$ and $t \in \Gamma^*$
- $r_m \in F$
Example PDA

- PDA for $L = \{a^n b^n \mid n \geq 0\}$
  - $\Sigma = \{a, b\}$, $\Gamma = \{0, \$\}$
  - $\$ \text{ keeps track of the “bottom” of the stack}$

![Diagram of PDA](Diagram.png)
**Example PDA**

- PDA for \( L = \{ww^R \mid w \in \{0, 1\}^*\} \)
- **Palindromes**: See [http://norvig.com/palindrome.html](http://norvig.com/palindrome.html) for interesting examples:
- A 17,826 word palindrome starts and ends as:
  - *A man, a plan, a cameo, Zena, Bird, Mocha, Prowel, a rave, Uganda, Wait, a lobola, Argo, Goto, Koser, Ihab, Udall, a revocation, Ebarta, Muscat, eyes, Rehm, a cession, Udella, E-boat, OAS, a mirage, IPBM, a caress, Etam, . . . , a lobo, Lati, a wadna, Guevara, Lew, Orpah, Comdr, Ibanez, OEM, a canal, Panama*
Example PDA

- PDA for $L = \{ww^R \mid w \in \{0, 1\}^*\}$
- $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \$\}$

\[
\begin{align*}
&\text{start} &\rightarrow& q_1 &\rightarrow& q_2 &\rightarrow& q_3 &\rightarrow& q_4 \\
&\varepsilon, \varepsilon &\rightarrow& \$ &\varepsilon, \varepsilon &\rightarrow& \$ &\varepsilon, \varepsilon &\rightarrow& \varepsilon \\
\end{align*}
\]

Transitions:
- $1, \varepsilon \rightarrow 1$
- $0, \varepsilon \rightarrow 0$
- $1, 0 \rightarrow \varepsilon$
- $0, 1 \rightarrow \varepsilon$
- $0, 0 \rightarrow \varepsilon$
- $1, 1 \rightarrow \varepsilon$
- $\varepsilon, \$ \rightarrow \varepsilon$
Let’s construct a PDA for
\[ L = \{ w \mid n_a(w) = n_b(w) \} \]
It is usually better and more succinct to represent a series of PDA transitions using a shorthand.
PDAs and CFGs are equivalent in power: they both describe context-free languages.

**Theorem**

A language is context free if and only if some pushdown automaton recognizes it.
If a language is context free, then some pushdown automaton recognizes it.

If $A$ is a CFL, then it has a CFG $G$ for generating it. Convert the CFG to an equivalent PDA.

- Each rule maps to a transition.
CFGs to PDAs

- We simulate the leftmost derivation of a string using a 3-state PDA with \( Q = \{ q_{\text{start}}, q_{\text{loop}}, q_{\text{accept}} \} \).
- One transition from \( q_{\text{start}} \) pushes the start symbol \( S \) onto the stack (along with $).
- Transitions from \( q_{\text{loop}} \) simulate either a rule expansion, or matching an input symbol.
  - \( \delta(q_{\text{loop}}, \epsilon, A) = \{(q_{\text{loop}}, w) \mid A \rightarrow w \text{ is a production in } G\} \)
  - If the top of the stack is \( A \), nondeterministically expand it in all possible ways.
  - \( \delta(q_{\text{loop}}, a, a) = \{(q_{\text{loop}}, \epsilon)\}, \text{ for all } a \in \Sigma \).
    - If the input symbol matches the top of the stack, consume the input and pop the stack.
- One transition takes the PDA from \( q_{\text{loop}} \) to \( q_{\text{accept}} \) when \( $ \) is seen on the stack.
CFGs TO PDAs

start → $q_{start}$

$\epsilon, \epsilon \rightarrow S\$

for each $a \in \Sigma$, $a, a \rightarrow \epsilon$

$q_{loop}$

$\epsilon, A \rightarrow w$ for rule $A \rightarrow w$

$\epsilon, S \rightarrow \epsilon$

$q_2$
Let’s convert the following grammar for $L = \{ w \mid n_a(w) = n_b(w) \}$.

- $S \rightarrow aSb$
- $S \rightarrow bSa$
- $S \rightarrow SS$
- $S \rightarrow \epsilon$