1. PDAs are FSAs extended with an infinite pushdown store at each possible state

A Pushdown Automaton (PDA) is a tuple \( \langle Q, X, S, F, M \rangle \), where:

- \( Q \) is a finite set of states
- \( X \) is a finite set of observation symbols
- \( S \subseteq Q \) is a set of start states
- \( F \subseteq Q \) is a set of final states
- \( M \subseteq Q \times (Q \cup \{\epsilon\}) \times (X \cup \{\epsilon\}) \times (Q \cup \{\epsilon\}) \) is a set of store, state transitions, of form:
  - \( \langle q, \epsilon, \epsilon, q', q \rangle \) — called an ‘expansion’ (or ‘stack push’):
    transition from state \( q \) to \( q' \), replacing empty string \( \epsilon \) at front of store with \( q \)
  - \( \langle q, q''_0, x, q', q''_1 \rangle \) — called a ‘state transition’:
    transition from state \( q \) to \( q' \) on observation \( x \), replacing \( q''_0 \) at front of store with \( q''_1 \)
  - \( \langle q, \epsilon, q', \epsilon \rangle \) — called a ‘reduction’ (or ‘stack pop/pull’):
    transition from state \( q \in F \) to \( q' \), replacing \( q''_0 \) at front of store with empty string \( \epsilon \)

From ‘Candyland’ to ‘Trivial Pursuit’: pieces carry stacked-on sub-pieces corresp. to spaces

2. Language accepted by PDA \( A = \langle Q_A, X_A, S_A, F_A, M_A \rangle \)

During recognition, the automaton uses the above store, state transitions to nondeterministically explore all possible states combined with all possible infinite stores.

- The automaton starts with an empty store.
- The automaton pushes and pulls only to/from the front of the store.
- The automaton accepts the input on an empty store at any final state.

\( L(A) = \{ x_{1..T} \mid q \in S_A, \langle q, \epsilon \rangle \in V_A(x_{1..T}) \} \)

where \( V_A(x_{1..T}) \) returns a set of states from which observations \( x_{1..T} \) are acceptable:

\[
V_A(x_{1..T}) = \{ \langle q, \epsilon \rangle \mid q \in F_A, x_{1..T} = \epsilon \} \\
\quad \cup \{ \langle q, q''_0 \alpha \rangle \mid \langle q, q''_0, x_1, q', q''_1 \rangle \in M_A, \langle q', q''_0 \alpha \rangle \in V_A(x_{1..T+1}) \} \\
\quad \cup \{ \langle q, \alpha \rangle \mid \langle q, \epsilon, \epsilon, q', q \rangle \in M_A, \langle q', q \alpha \rangle \in V_A(x_{1..T}) \} \\
\quad \cup \{ \langle q, q''_0 \alpha \rangle \mid \langle q, q''_0, \epsilon, q' \epsilon, q \rangle \in M_A, \langle q', \alpha \rangle \in V_A(x_{1..T}) \}
\]

3. Graphical representation of PDAs

PDAs can be represented graphically:

- start states: \( q \in S \)
• final states: \( q \in F \)

\[
\text{Diagram 1: Final States}
\]

\( q \)

• state transitions: \( \langle q, \epsilon, x, q', \epsilon \rangle \in M \)

\[
\text{Diagram 2: State Transitions}
\]

\( q \xrightarrow{x} q' \)

• expansions: \( \langle q, \epsilon, \epsilon, q', q \rangle \in M \) (like transitions, but push previous state onto store)

\[
\text{Diagram 3: Expansions}
\]

\( q \xrightarrow{} q' \)

• reductions: \( \langle q, q'', \epsilon, q', \epsilon \rangle \in M, \ q_2 \in F \) (like trans, but remove last state from store)

\[
\text{Diagram 4: Reductions}
\]

\( q'' \xleftarrow{} q' \)

• conditional transitions: \( \langle q, q'', \epsilon, q', q'' \rangle \in M \)

\[
\text{Diagram 5: Conditional Transitions}
\]

\( q'' \xleftarrow{} q' \)

4. Example PDA for: \( a^n b^n; n > 0 \) (\( S \rightarrow a S b, S \rightarrow a b \))

\[
\begin{align*}
Q: \{q_0, q_1, q_2, q_3, q_4\}, \ X: \{a,b\}, \ S: \{q_0\}, \ F: \{q_3\}, \\
M: \{\langle q_0, \epsilon, a, q_1, \epsilon \rangle, \langle q_2, \epsilon, b, q_3, \epsilon \rangle, \langle q_0, \epsilon, a, q_1, \epsilon \rangle, \langle q_4, \epsilon, b, q_3, \epsilon \rangle, \langle q_1, \epsilon, \epsilon, q_0, q_1 \rangle, \langle q_3, q_1, \epsilon, q_2, \epsilon \rangle\}
\end{align*}
\]
5. PDAs can recognize CFGs; \( \mathcal{L} (\text{CFG}) \subseteq \mathcal{L} (\text{PDA}) \)

Given any CFG \( \langle C_G, X_G, S_G, R_G \rangle \), we can define a PDA \( \langle Q, X, S, F, M \rangle \)
(assume binary-branching CFG, since all CFGs can be translated into one):

\[
Q = \{ q_{c/e}, q_{c/e}, q_c \mid c \rightarrow d e \in R_G \} \cup \{ q_{x/x}, q_x \mid x \in X_G \}
\]

\( X = X_G \)

\( S = \{ q_{e/c} \mid e \in S_G \} \)

\( F = \{ q_c \mid c \rightarrow d e \in R_G \text{ or } e \in X_G \} \)

\( M = \{ \langle q_{c/e}, \epsilon, \epsilon, q_{d/d}, q_{c/e} \rangle, \langle q_{d/d}, q_{c/e}, \epsilon, q_{e/e}, \epsilon \rangle, \langle q_{c/e}, \epsilon, \epsilon, q_{e/e}, q_{e/e} \rangle, \langle q_{e}, q_{e/e}, \epsilon, q_{e}, \epsilon \rangle \mid c \rightarrow d e \in R_G \} \cup \{ \langle q_{x/x}, \epsilon, x, q_{x}, \epsilon \rangle \mid x \in X_G \} \)

for example, a grammar with the following rules:

\[ V \rightarrow N \ V-aN \]
\[ V-aN \rightarrow \text{hit} \ N \]
\[ N \rightarrow \text{the toy} \]

would yield the following PDA:
6. Practice

How would the store look after each expansion, reduction, or transition in the above PDA?

$q_{V/N}$ (start state)
$q_{N/N}$ $q_{V/N}$ (expand, pushing $q_{V/N}$ onto store)
...

7. Right-expansion elimination and left-expansion elimination

Easy to implement recognizer if only one expansion, reduction per transition

(a) Right-expansion elimination (‘awaited transition’) – right child ‘c’ disappears:

$$M^{(0)} = M$$
$$M^{(k)} = M^{(k-1)} \cup \{ \langle q_{a/c}, \epsilon, \epsilon, q_{d/d}, q_{a/c} \rangle, \langle q_{d}, q_{a/e}, \epsilon, q_{a/e}, \epsilon \rangle, \langle q_{a/e}, \epsilon, \epsilon, q_{c/e}, q_{a/e} \rangle, \langle q_{c}, q_{a/e}, \epsilon, q_{a} \rangle \}$$
$$M' = M^{[R]}$$

Recognition is equivalent because child sub-models are preserved in order:
for example:

Repeat by mapping $q_{a/e}, q_a$ in result to $q_{a/e}, q_a$ in subsequent iteration.
Now all sequences of reductions are replaced with a single reduction!

Result (hiding unnecessary structure):

(b) Left-expansion elimination (‘active transition’) – left child ‘c’ disappears:

$$M^{(0)} = M$$
$$M^{(k)} = M^{(k-1)} \cup \{ \langle q_{a/a'}, \epsilon, \epsilon, q_{d/d'}, q_{a/a'} \rangle, \langle q_d, q_{a/a'}, \epsilon, q_{c/e}, q_{a/a'} \rangle \mid \langle q_{a/a'}, \epsilon, \epsilon, q_{c/e}, q_{a/a'} \rangle \in M^{(k-1)}, \langle q_{c/e}, \epsilon, \epsilon, q_{d/d}, q_{c/e} \rangle \in M^{(k-1)},$$
\[\langle q_d, q_{c/e}, \epsilon, q_{c/e}, \epsilon \rangle \in M^{(k-1)} \} \}

$$M' = M^{[R]}$$

Add ‘conditional’ $\epsilon$-transition: $\langle q_d, q_{a/a'}, \epsilon, q_{c/e}, q_{a/a'} \rangle$ (drawn with dotted line)
Recognition is equivalent because child sub-models preserved in order:
for example:

Repeat by mapping $q_d$, $q_{a/a'}$ in result to $q_c$, $q_{a/a'}$ in subsequent iteration.

Now all sequences of expansions are replaced with a single expansion!

Result (hiding unnecessary structure):

(c) Transition flattening:

$$M' = M \cup \{ \langle q, \epsilon, x, q', \epsilon \rangle \mid \langle q, \epsilon, \epsilon, q_{x/x}, q \rangle, \langle q_{x/x}, \epsilon, x, q_x, \epsilon \rangle, \langle q_x, q, \epsilon, q', \epsilon \rangle \in M, q' \in F \}$$
8. Implementation of PDA

Since only one expansion/reduction between transitions, we only need four combinations:

a) $(q_{t-1}^{d-1})$

    (no expand + trans + no reduce)

    $q_{t-1}^{d}$

    $x_{t}$

    $q_{t}^{d}$

b) $(q_{t-1}^{d+1})$

    (expand + trans + no reduce)

    $q_{t-1}^{d+1}$

    $x_{t}$

    $q_{t}^{d+1}$
Recognition algorithm:

for each time step \( t \):
   for each previous state \( q_{d-1}^{t} \) and store \( \sigma \) (where \( d-1 \) is the depth of the store):
      a) for each final state \( f_{t}^{d} \) and current state \( q_{t}^{d} \): (expand +) trans
         \[
         V[t, q_{t}^{d} \sigma] = V[t, q_{t}^{d} \sigma] \text{ or } (V[t - 1, q_{t - 1}^{d} \sigma] \text{ and } M[q_{t - 1}^{d}, t, f_{t}^{d}, \epsilon])
         \]
         and
         \[
         M[f_{t}^{d}, q_{t - 1}^{d}, e, q_{t - 1}^{d}, q_{t - 1}^{d - 1}])
         \]
      b) for each previous state at deeper level \( q_{d}^{t} \) and final state \( f_{t}^{d+1} \) and current state \( q_{t}^{d+1} \):
         \[
         V[t, q_{t}^{d+1} q_{t}^{d} \sigma] = V[t, q_{t}^{d+1} q_{t}^{d} \sigma] \text{ or } (V[t - 1, q_{t - 1}^{d+1} q_{t - 1}^{d} \sigma] \text{ and } M[q_{t - 1}^{d+1}, t, f_{t}^{d+1}, \epsilon])
         \]
         and
         \[
         M[f_{t}^{d+1}, q_{t - 1}^{d+1}, e, q_{t - 1}^{d+1}, q_{t - 1}^{d}])
         \]
      c) for each final state \( f_{t}^{d} \) and current state \( q_{t}^{d-1} \): (expand +) trans + reduce
         \[
         V[t, q_{t}^{d-1} \sigma] = V[t, q_{t}^{d-1} \sigma] \text{ or } (V[t - 1, q_{t - 1}^{d-1} q_{t - 1}^{d-1} \sigma] \text{ and } M[q_{t - 1}^{d-1}, t, f_{t}^{d}, \epsilon])
         \]
         and
         \[
         M[f_{t}^{d}, q_{t - 1}^{d}, \epsilon, q_{t - 1}^{d}, q_{t - 1}^{d}]])
         \]
      d) for each previous state at deeper level \( q_{d}^{t} \) and final state \( f_{t}^{d+1} \) and current state \( q_{t}^{d+1} \):
         \[
         V[t, q_{t}^{d+1} q_{t}^{d} \sigma] = V[t, q_{t}^{d+1} q_{t}^{d} \sigma] \text{ or } (V[t - 1, q_{t - 1}^{d+1} q_{t - 1}^{d-1} \sigma] \text{ and } M[q_{t - 1}^{d+1}, t, f_{t}^{d+1}, \epsilon])
         \]
         and
         \[
         M[f_{t}^{d+1}, q_{t - 1}^{d+1}, \epsilon, q_{t - 1}^{d+1}, q_{t - 1}^{d}])
         \]

Correctness can be shown from definition of accepted languages.

But, iterating over all possible infinite stores \( \sigma \) results in exponential complexity.

9. Example:

Transforming the following CFG:
$T \rightarrow V\ T$ (a top-level discourse type)

$V \rightarrow N\ V\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$V\ -aN \rightarrow V\ -aN\ R\ -aN$

$A\ -aN \rightarrow off\ N$

$N \rightarrow N\ A\ -aN$

$N \rightarrow the\ cat$

$N \rightarrow the\ toy$

$N \rightarrow the\ mat$

then recognizing 'the cat hit the toy off the mat' results in the following sequence: