Definition

A PDA, \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \), is a 7-tuple consisting of
- a finite set of states, denoted \( Q \)
- a finite set of input symbols, denoted \( \Sigma \)
- a finite stack alphabet, denoted \( \Gamma \)
- a transition function, \( \delta: Q \times \Sigma \times \{\varepsilon}\times \Gamma \rightarrow Q \times \Gamma^* \)
- the start state \( q_0 \in Q \)
- the start symbol \( Z_0 \in \Gamma \),
- a set of final states \( F \subseteq Q \)

Informal Introduction

The pushdown automaton (PDA) is essentially an \( \varepsilon \)-NFA with a stack.

For the transition function, \( \delta: Q \times \Sigma \times \{\varepsilon\} \times \Gamma \rightarrow Q \times \Gamma^* \)
- Informally, for \( q \in Q \), \( a \in \Sigma \), and \( X \in \Gamma \),
  \( \delta(q, a, X) = (p, \gamma) \) where
  - \( p \in Q \) is the next state; and
  - \( \gamma \in \Gamma^* \) is the string of stack symbols that replaces \( X \) at the top of the stack.
  - If \( \gamma = \varepsilon \), then the stack is popped.
  - If \( \gamma = X \), then the stack is unchanged.
  - If \( \gamma = YZ \), then \( X \) is replaced by \( Z \), and \( Y \) is pushed onto the stack.
Definition

Example

The CFL \( L = \{ w w^R \mid w \in (0+1)^* \} \) is represented by the following PDA:

![Diagram of PDA]

Instantaneous Descriptions

The configuration of a PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) is a triple \( (q, w, \gamma) \), where:

- \( q \in Q \) is the (current) state;
- \( w \in \Sigma^* \) is the remaining output; and
- \( \gamma \in \Gamma^* \) is the stack contents. (By convention, the top of the stack is at the left end of \( \gamma \) and the bottom at the right end.)

Such a triple is called an instantaneous description, or ID, of the PDA, \( P \).

Moves of a PDA

The “turnstile” notation for connecting pairs of IDs that represent one or many moves of a PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) is denoted by \( \vdash \), or just \( \vdash \) when \( P \) is understood.

Suppose \( \delta(q, a, X) = (p, \alpha) \). Then, for all strings \( w \in \Sigma^* \) and \( \beta \in \Gamma^* \):

\[(q, aw, X\beta) \vdash (p, w, \alpha\beta)\]

So, by consuming \( a \) (which may be \( \varepsilon \)) from the input and replacing \( X \) on the top of the stack with \( \alpha \), we can go from state \( q \) to state \( p \).

Moves of a PDA

The symbol \( \vdash^* \), or just \( \vdash \) when \( P \) is understood, is used to denote zero or more moves of the PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \).

That is,

- \( I \vdash^* I \), for any ID \( I \).
- \( I \vdash^* J \), if there exists some ID \( K \) such that \( I \vdash K \) and \( K \vdash J \). So, \( I \vdash J \) if there is a sequence of one IDs \( K_1, K_2, \ldots, K_n \) such that \( I = K_1 \), \( J = K_n \), and for all \( 1 \leq i \leq n-1 \), we have \( K_i \vdash K_{i+1} \).
PDA IDs and Moves

Three important principles about IDs and moves:

1. If a sequence of IDs (computation) is legal for a PDA $P$, then the computation formed by adding the same additional input string to the end of the input (second component) in each ID is also legal.

2. If a computation is legal for a PDA $P$, then the computation formed by adding the same additional stack symbols below the stack in each ID is also legal.

3. Theorem 6.5

If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, and $(q, x, \alpha) \xrightarrow{\tau} (p, y, \beta)$, then for any strings $w \in \Sigma^*$ and $\gamma \in \Gamma^*$, it is also true that

$$(q, xw, \alpha \gamma) \xrightarrow{\tau} (p, yw, \beta \gamma).$$

PDA Languages

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.

- $P$ accepts an input string $w \in \Sigma^*$ if $(q_0, w, Z_0) \xrightarrow{\tau} (p, \varepsilon, \varepsilon)$ for any final state $p \in F$ and any stack string $\gamma \in \Gamma^*$. This approach is known as “acceptance by final state” and the set of strings accepted this way is denoted $L(P)$.

- $P$ accepts an input string $w \in \Sigma^*$ if $(q_0, w, Z_0) \xrightarrow{\tau} (q, \varepsilon, \varepsilon)$ for any state $q \in Q$. This approach is also known as “acceptance by empty stack” and the set of strings accepted this way is denoted $N(P)$.
PDA Languages

Theorem 6.9 (From Empty Stack to Final State)

If \( L = N(P_N) \) for some PDA \( P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0, F) \), then there is a PDA \( P_F \) such that \( L = L(P_F) \).

(Constructive) Proof:

1. Construct a PDA \( P_F \) as follows:
   - Introduce new start state \( p_0 \) and new bottom-of-stack marker \( X_0 \).
   - First move of \( P_F \): \( \delta_F(p_0, \varepsilon, X_0) = (q_0, Z_0 X_0) \).
   - Then, \( P_F \) simulates \( P_N \); i.e., give \( P_F \) all the transitions of \( P_N \).

2. Introduce a new final state \( p_f \) for \( P_F \).
3. For every state \( q \in Q \), \( \delta_F(q, \varepsilon, X_0) = (p_f, \varepsilon) \).

Does this really work? For \( w \in Q^* \) where \( w \in N(P_N) \):

\[
(\rho_0, w, X_0) \xrightarrow{\delta_N} (q_0, w, Z_0 X_0) \xrightarrow{\rho_0} (q, \varepsilon, X_0) \xrightarrow{\delta_F} (p_f, \varepsilon, \varepsilon)
\]

So, \( P_F \) accepts \( w \) by final state.

Example

Consider the following PDA, \( P_N = (Q = \{ q \}, \Sigma = \{ i, e \}, \Gamma = \{ Z \}, \delta_N, q, Z, F = \{ q \}) \), which processes sequences of \( \pm \)’s (denoted \( i \)) and \( \pm \)’s (denoted \( e \)) in a C program:

Notice that accepts/recognizes if/else errors by the empty stack.
PDA Languages

**Theorem 6.9** *(From Empty Stack to Final State)*

*Example, continued*

- Construct from $P_F$ a PDA $P_N$ that accepts the same language by *final state*.

- Hence, $P_F = (\{q\}, N(p_0 = p, p_f = r), \Sigma = \{i, e\}, \{Z\} N(X_0), \delta_F, p_0 = p, X_0 \{p = r\})$.

---

**Theorem 6.11** *(From Final State to Empty Stack)*

*(Constructive) Proof:*

- Given $L = L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, then there is a PDA $P_N$ such that $L = N(P_N)$.

- Construction of PDA $P_N$...

- For every state $q \in F$ and any stack symbol $Y \in \Gamma$, $\delta_F(q, e, Y) = (p, e)$.

- For any stack symbol $Y \in \Gamma$, add the transition $\delta_F(p, e, Y) = (p, e)$ to use state $p$ as an auxiliary to keep popping the stack of $P_N$ until it is empty.

- For $w \in Q^*$ where $w \in L(P_F)$ and $q \in F$:

  $(p_0, w, X_0) \rightarrow^* (q_0, w, Z_0, X_0) \rightarrow^* (q, e, X_0) \rightarrow^* (p, e, \epsilon)$.

- So, $P_N$ accepts $w$ by empty stack.
Equivalence of PDAs & CFGs

The following three classes of languages:

- CFLs; i.e. languages defined by CFGs
- Languages accepted by final state by some PDA
- Languages accepted by empty stack by some PDA

are all the same class!

We’ve shown these through Theorems 6.9 and 6.11

From CFGs to PDAs

Let \( L = L(G) \) for some CFG \( G = (V, T, P, S) \).

Idea: Have PDA \( A \) simulate leftmost derivations in \( G \), where a left-sentential form is represented by

- the sequence of input symbols that \( A \) has consumed from its input, followed by
- \( A \)’s stack top

Example: If \((q,abcd,S)\) \(\vdash\) \((q,cd,ABC)\), then the left-sentential form represented is \( abABC \).
Equivalence of PDAs & CFGs

From CFGs to PDAs, continued:

Example:

Then, PDA \( M = (\{q\}, \{0,1\}, \{0,1,A,S\}, \delta, q, S, \emptyset) \), where \( \delta \) is defined by

\[
\begin{align*}
\delta(q, \varepsilon, S) &= \{(q, AS), (q, \varepsilon)\} \\
\delta(q, \varepsilon, A) &= \{(q, 0A1), (q, A1), (q, 01)\} \\
\delta(q, 0, 0) &= \{(q, \varepsilon)\} \\
\delta(q, 1, 1) &= \{(q, \varepsilon)\}
\end{align*}
\]

Theorem 6.13

If PDA \( P \) is constructed from CFG \( G \) by the construction above, then \( N(P) = L(G) \).

Proof:

By induction on the number of steps in the derivation \( S \Rightarrow^* \alpha \) that for any \( x, (q, wx, S) \overset{\text{fin}}{\Rightarrow} (q, x, \beta) \), where

\[
\begin{align*}
\text{if } \beta = \alpha & \quad \text{then } \beta \text{ is the suffix of } \alpha \text{ that begins at the leftmost variable (if there is no variable)} \\
\text{if } \beta \neq \alpha & \quad \text{then } \beta \text{ is the suffix of } \alpha \text{ that begins at the leftmost variable (if there is no variable)}
\end{align*}
\]

Proof detail in textbook ...

Equivalence of PDAs & CFGs

From PDAs to CFGs

Let \( L = N(P) \) for some PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \).

Idea: Units of PDA action have the net effect of popping one symbol from the stack, consuming some input, and making a state change.

For \( q, p \in Q \) and \( X \in \Gamma \), the composite symbol \( [qXp] \) is a single CFG variable that generates exactly those strings \( w \) such that \( P \) can read \( w \) from the input, pop \( X \) (net effect), and go from state \( q \) to state \( p \).

Theorem 6.14

Let \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) be a PDA. Then, there is a CFG \( G \) such that \( L(G) = N(P) \).

(Constructive) Proof

Construct CFG \( G = (V, \Sigma, P, S) \) where

\[
\begin{align*}
V & \text{ consists of} \\
& \quad \text{A the start symbol, } S, \text{ and} \\
& \quad \text{A all symbols of the form } [qXp], \text{ where } q, p \in Q \text{ and } X \in \Gamma
\end{align*}
\]
Equivalence of PDAs & CFGs

(Constructive) Proof continued:

- $P$ consists of
  - $S \rightarrow [q_0Z_0p]$, for all states $p \in Q$.
  - Let $(r, Y_1Y_2w Y_k) \in \delta(q,a,X)$, where
    - $a \in \Sigma \cup \{\varepsilon\}$
    - $k$ can be any number, including 0, in which case $(r, Y_1Y_2w Y_k) = (r, \varepsilon)$

Then for all $r_1, r_2, w, r_k \in Q$, $G$ has the production

$[qX] \rightarrow a[rY_1][r_1Y_2][w[r_1Y_2][r_1][r_1][r_1]]$

Hence, productions in $P$ could be of the form

- popping rule, $[qZp] \rightarrow a$, whenever $(p, \varepsilon) \in \delta(q,a,Z)$;
- one stack symbol, one state replacement rule, $[qZr] \rightarrow a[pYr]$, for all $r \in Q$, whenever $(p, Y) \in \delta(q,a,Z)$; or
- one stack symbol replaced by two rule, $[qZs] \rightarrow a[pXr][rYs]$, for all $r, s \in Q$, whenever $(p, XY) \in \delta(q,a,Z)$.

From PDAs to CFGs, continued:

So, the essence of the Theorem is

$qX \Rightarrow w$ if and only if $(q, w, X) \vdash (p, \varepsilon, \varepsilon)$

Show the above holds by induction

- (If) on number of moves made by PDA
- (Only-if) on number of steps in the derivation

Proof detail in textbook ...

Example: Recall the PDA, $P_N = (Q=\{q\}, \Sigma=\{i, e\}, \Gamma=\{Z\}, \delta_N, q, Z, F=\emptyset)$, which processes sequences of $i$’s (denoted $i$) and $e$’s (denoted $e$) in a C program:

```
Start
q
```

and accepts/recognizes if/else errors by the empty stack.
Equivalence of PDAs & CFGs

From PDAs to CFGs, continued:

Construct CFG $G = (V, \Sigma, P, S)$ where

- $V$ consists of:
  - the start symbol, $S$, and
  - $[qZq]$, the only composite symbol

- $P$ consists of:
  - $S \rightarrow [qZq]$
  - $[qZq] \rightarrow i[qZq][qZq]$, since $(q.ZZ) \in \delta_{\delta}(q,i.Z)$
  - $[qZq] \rightarrow \varepsilon$, since $(q.i) \in \delta_{\delta}(q,i.Z)$

Hence, replacing the composite symbol $[qZq]$ by $A$, the CFG $G$ has the productions:

$S \rightarrow A$
$A \rightarrow iAA | \varepsilon$

Furthermore, CFG $G$ can be simply written as:

$G = (\{S\}, \{i,\varepsilon\}, \{S \rightarrow iSS | \varepsilon\}, S)$

Deterministic PDAs

A PDA, $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, is a deterministic PDA or DPDA if and only if the following conditions are met:

- $|\delta(q,a,X)| \leq 1$ for any state $q \in Q$, input symbol $a \in \Sigma \cup \{\varepsilon\}$, and stack symbol $X \in \Gamma$.
- If $\delta(q,a,X) \neq \emptyset$ for some $a \in \Sigma$, then $\delta(q,\varepsilon,X) = \emptyset$.

Note: Parsers are DPDAs ...

From Figure 6.11 of UATLC, Hopcroft, Motwani, & Ullman, 2001.
Deterministic PDAs

- Regular Languages and DPDAs
  - The DPDAs accept a class of languages that is between the RLs and the CFLs.
- Theorem 6.17
  - If \( L \) is RL, then \( L = L(P) \) for some DPDA \( P \).
- Proof:
  - Let \( A = (Q, \Sigma, \delta_A, q_0, F) \) be a DFA.
  - Construct DPDA \( P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F) \) by defining \( \delta_P(q, a, Z_0) = \{(p, Z_0)\} \) for all states \( p, q \in Q \) such that \( \delta_A(q, a) = p \).

Deterministic PDAs

- A language \( L \) is said to have the *prefix property* if there are no two different strings \( x \) and \( y \) in \( L \) such that \( x \) is a prefix of \( y \).
- Theorem 6.19
  - A language \( L \) is \( N(P) \) for some DPDA \( P \) if and only if \( L \) has the prefix property and \( L = L(P') \) for some DPDA \( P' \).

Equivalence of PDAs & CFGs

- Alternate construction, CFG → PDA
  - Idea: Instead of constructing a PDA with only one state (as in our textbook), create extra temporary states to push additional symbols into the stack. (This models production rules that have a body with more than one symbol.)
  - Given CFG \( G = (V, \Sigma, P, S) \).
  - Construct PDA \( P = (Q, \Sigma, \{V \Sigma \}, \delta, q_s, Z_0, \{q_f\}) \) such that \( Q \) contains start state \( q_s \), final state \( q_f \) and other states to be defined next ...

Deterministic PDAs and CFLs

- The languages accepted by DPDAs by final state properly include the regular languages, but are properly included in the CFLs.

Deterministic PDAs and Ambiguous Grammars

- Theorem 6.20
  - If \( L = N(P) \) for some DPDA \( P \), then \( L \) has an unambiguous CFG \( G \).
- Theorem 6.21
  - If \( L = L(P) \) for some DPDA \( P \), then \( L \) has an unambiguous CFG \( G \).
Equivalence of PDAs & CFGs

Alternate construction, continued:

P’s transition function, \( \delta \), contains:

\[
\delta(q_s, \varepsilon, \varepsilon) = \{(q_f, S)\}
\]

\[
\delta(q_f, a, a) = \{(q_f, \varepsilon)\}, \text{ for all } a \in \Sigma
\]

For every rule \( r_i = (A \rightarrow w_1 w_2 w_k) \) in \( P \), where each \( w_j \in VN \Sigma, 1 \leq j \leq k \), create \( k-1 \) new states \( q_{i,1}, q_{i,2}, \ldots, q_{i,k-1} \)...

Equivalence of PDAs & CFGs

Alternate construction, continued:

Example: Consider CFG \( G = (\{S\}, \{a,b\}, P, S) \), where \( P \) contains \( S \rightarrow aS | aSbS | \varepsilon \).

By the construction in our textbook:

- PDA \( P_1 = (\{q\}, \{a,b\}, \{S,a,b\}, \delta, q, S, \emptyset) \) where
  - \( \delta(q, a, S) = \{(q, aS), (q, aSbS), (q, \varepsilon)\} \)
  - \( \delta(q, a, a) = \{(q, \varepsilon)\} \)
  - \( \delta(q, b, b) = \{(q, \varepsilon)\} \)
- Note that \( L(G) = N(P_1) \).
Equivalence of PDAs & CFGs

Alternate construction, continued:

By alternate construction:

Initially, PDA $P_2 = (\{q_s, q_f\}, \{a, b\}, \{S, a, b\}, \delta, q_s, \varepsilon, \{q_f\})$

\[
\delta(q_s, \varepsilon, \varepsilon) = \{(q_f, S)\}
\]

\[
\delta(q_f, a, a) = \{(q_f, \varepsilon)\}
\]

\[
\delta(q_f, b, b) = \{(q_f, \varepsilon)\}
\]

Alternate construction, continued:

By alternate construction:

For rule $r_1 = (S \rightarrow aS)$ in $P$:

- Create $k-1=1$ new state, $q_{1,1}$.
- Add the following into $\delta$:
  - $\delta(q_f, \varepsilon, S) \ni (q_{1,1}, S)$
  - $\delta(q_{1,1}, \varepsilon, \varepsilon) = \{(q_f, a)\}$

Alternate construction, continued:

By alternate construction:

For rule $r_2 = (S \rightarrow aSB)$ in $P$:

- Create $k-1=3$ new states, $q_{2,1}$, $q_{2,2}$, $q_{2,3}$.
- Add the following into $\delta$:
  - $\delta(q_f, \varepsilon, S) \ni (q_{2,1}, S)$
  - $\delta(q_{2,1}, \varepsilon, \varepsilon) = \{(q_{2,2}, b)\}$
  - $\delta(q_{2,2}, \varepsilon, \varepsilon) = \{(q_{2,3}, S)\}$
  - $\delta(q_{2,3}, \varepsilon, \varepsilon) = \{(q_f, a)\}$

Alternate construction, continued:

By alternate construction:

For rule $r_3 = (S \rightarrow \varepsilon)$ in $P$:

- No new states to add!
- Add the following into $\delta$:
  - $\delta(q_f, \varepsilon, S) \ni (q_f, \varepsilon)$
Equivalence of PDAs & CFGs

Alternate construction, continued.

By alternate construction:

The transition diagram for PDA \( P_2 \) is

Note that \( L(G) = L(P_2) \).