Lecture 13 — Pushdown Automata (2)

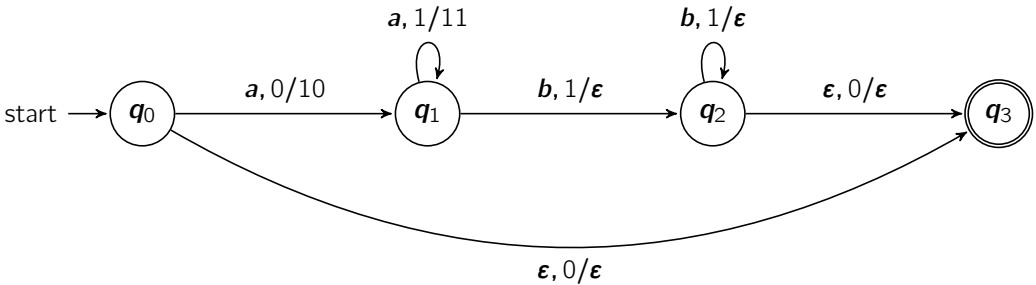
CSE205: Theory of Computation

Minseok Jeon

2026 Fall

Exercise 1: $L = \{a^n b^n \mid n \geq 0\}$

$$P = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_3\})$$



Each a pushes a 1 (the first one onto the start symbol 0); each b pops a 1. When the start symbol 0 is exposed again the input is balanced and we accept. The bottom edge accepts ε (the case $n = 0$).

Exercise 2: Equal Numbers of a's and b's

$L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$

- **Idea.** On each a push a counter 0; on each b pop one.

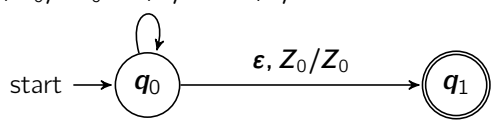
$$(abab, Z_0) \to (bab, 0Z_0) \to (ab, Z_0) \to (b, 0Z_0) \to (\varepsilon, Z_0)$$

- When a prefix has more b's than a's, push a *negative* counter 1 for the b's to be matched later.

$$(bbaa, Z_0) \to (baa, 1Z_0) \to (aa, 11Z_0) \to (a, 1Z_0) \to (\varepsilon, Z_0)$$

$$P = (\{q_0, q_1\}, \{a, b\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_1\})$$

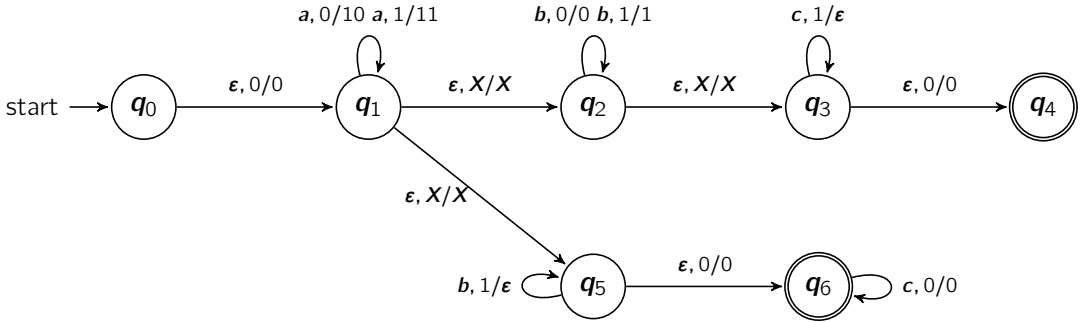
$a, Z_0/0Z_0$ $\quad a, 0/00$ $\quad b, 0/\varepsilon$
$b, Z_0/1Z_0$ $\quad b, 1/11$ $\quad a, 1/\varepsilon$



Exercise 3: $L = \{a^i b^j c^k \mid i = j \vee i = k\}$

$L = \{a^i b^j c^k \mid i, j, k \geq 0 \ \wedge \ (i = j \vee i = k)\}$. Guess nondeterministically which case holds: $L_1 = \{a^i b^j c^k \mid i = j\}$ or $L_2 = \{a^i b^j c^k \mid i = k\}$.

$$P = (\{q_0, \ldots, q_6\}, \{a, b, c\}, \{0, 1\}, \delta, q_0, 0, \{q_4, q_6\})$$



$X \in \{0, 1\}$. Top ($i = k$): skip b's, pop on c's. Bottom ($i = j$): pop on b's, skip c's.

Configurations of a PDA

A *configuration* (or *instantaneous description*, ID) of a PDA records its state and stack contents together with the remaining input.
An ID is a triple (q, w, γ), where

- q is the current state,
- w is the remaining input, and
- γ is the stack contents (top at the left).

Move relation

Suppose $(q, aw, X\beta)$ is a configuration and $(p, \alpha) \in \delta(q, a, X)$, where $a \in \Sigma \cup \{\varepsilon\}$. Then the PDA moves in one step to $(p, w, \alpha\beta)$:

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

We write $\vdash^*$ for the reflexive-transitive closure of $\vdash$ (zero or more moves).

The Language of a PDA: Acceptance by Final State

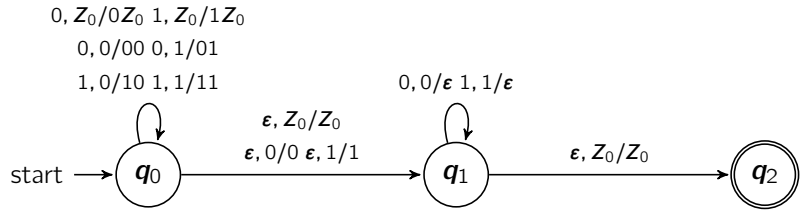
Definition (Acceptance by Final State)

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The language of P accepted by final state is

$$L(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha) \text{ for some } q \in F,\ \alpha \in \Gamma^*\}.$$

The PDA accepts w if, after reading all of w, it can reach a final state — regardless of what remains on the stack.

Example: Acceptance by Final State



This PDA accepts the even-length palindromes over $\{0, 1\}$. For example, it accepts 1111 because

$$(q_0, 1111, Z_0) \vdash^* (q_2, \varepsilon, Z_0).$$

Another Definition: Acceptance by Empty Stack

Definition (Acceptance by Empty Stack)

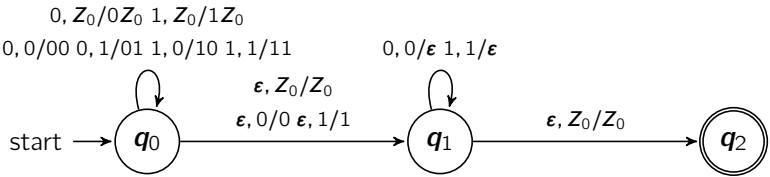
Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The language of P accepted by empty stack is

$$N(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon) \text{ for some } q \in Q\}.$$

The PDA accepts w if it can consume all of w and *empty its stack*. Final states are irrelevant, so we drop the F component:

$$(Q, \Sigma, \Gamma, \delta, q_0, Z_0)$$

Examples: $L(P)$ vs. $N(P)$



With the final-state edge $\boldsymbol{\varepsilon}, Z_0/Z_0$ to q_2:

- $L(P) = \{ww^R \mid w \in \{0,1\}^*\}$, the even-length palindromes.
- $N(P) = \emptyset$, since Z_0 is never popped and the stack is never emptied.

If instead the last edge popped Z_0 (i.e. $\boldsymbol{\varepsilon}, Z_0/\boldsymbol{\varepsilon}$ into q_2), then $N(P) = \{ww^R \mid w \in \{0,1\}^*\}$ as well.

Equivalence of Final State and Empty Stack

Theorem (Equivalence)

For any language L, there is a PDA P_F with $L = L(P_F)$ if and only if there is a PDA P_N with $L = N(P_N)$.

The theorem follows from two constructions:

Lemma (Empty Stack $\Rightarrow$ Final State)

For any PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, there is a PDA P_F with $N(P_N) = L(P_F)$.

Lemma (Final State $\Rightarrow$ Empty Stack)

For any PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, there is a PDA P_N with $N(P_N) = L(P_F)$.

From Empty Stack to Final State

Given $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, define

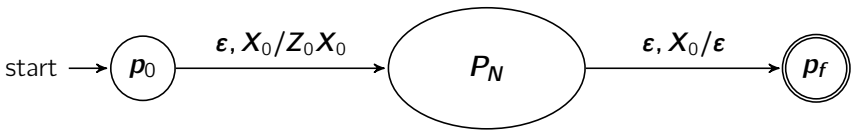
$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

where:

1. $\delta_F(p_0, \varepsilon, X_0) = \{(q_0, Z_0 X_0)\}$
   (push Z_0 above a fresh bottom marker X_0, then start P_N).
2. For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $Y \in \Gamma$: $\delta_F(q, a, Y) \supseteq \delta_N(q, a, Y)$ (simulate P_N).
3. For all $q \in Q$: $\delta_F(q, \varepsilon, X_0)$ contains (p_f, ε).

X_0 becomes exposed exactly when P_N would have emptied its stack; P_F then moves to the final state p_f. Hence $w \in L(P_F)$ iff $w \in N(P_N)$.

From Empty Stack to Final State: Construction Diagram



From any state of P_N, once the original stack is empty the marker X_0 is on top, and an ε-move carries the machine into the new accepting state p_f.

From Final State to Empty Stack

Given $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, define

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

where:

1. $\delta_N(p_0, \varepsilon, X_0) = \{(q_0, Z_0 X_0)\}$.
2. For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $Y \in \Gamma$: $\delta_N(q, a, Y) \supseteq \delta_F(q, a, Y)$.
3. For all $q \in F$ and $Y \in \Gamma \cup \{X_0\}$: $\delta_N(q, \varepsilon, Y)$ contains (p, ε).
4. For all $Y \in \Gamma \cup \{X_0\}$: $\delta_N(p, \varepsilon, Y) = \{(p, \varepsilon)\}$.

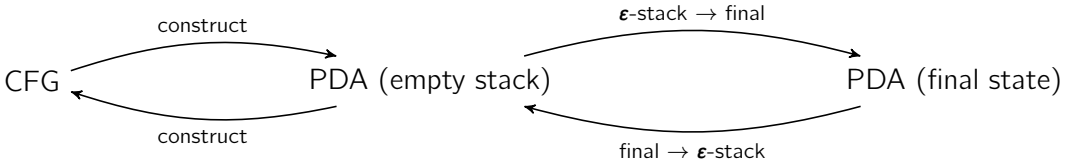
Once P_F reaches a final state, P_N may enter the "erasing" state p and pop the entire stack (including X_0), emptying it.

Equivalence of PDA's and CFG's

Theorem

The following three classes of languages are all the same:

1. *The context-free languages, i.e. the languages defined by CFGs.*
2. *The languages accepted by final state by some PDA.*
3. *The languages accepted by empty stack by some PDA.*



From CFG to PDA

Given a CFG $G = (V, T, P, S)$, build a PDA (accepting by empty stack) with a single state that simulates leftmost derivations on its stack:

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

where:

- For each variable $A \in V$,

  $$\delta(q, \varepsilon, A) = \{(q, \beta) \mid (A \to \beta) \text{ is in } G\}.$$

  (Replace a variable on top by the right-hand side of one of its rules.)
- For each terminal $a \in T$,

  $$\delta(q, a, a) = \{(q, \varepsilon)\}.$$

  (Match a terminal on top against the input and pop it.)

The stack always holds the unmatched suffix of a sentential form; the PDA empties it exactly on the strings G derives.

Example: CFG to PDA (Balanced Parentheses)

$$G = (\{B\}, \{(,)\}, R, B), \qquad B \to BB \mid (B) \mid \varepsilon$$

The corresponding single-state PDA $P = (\{q\}, \{(,)\}, \{B, (,)\}, \delta, q, B)$ has:

$$\begin{aligned} \delta(q, \varepsilon, B) &= \{(q, BB),\ (q, (B)),\ (q, \varepsilon)\} \\ \delta(q, (, () &= \{(q, \varepsilon)\} \\ \delta(q,),)) &= \{(q, \varepsilon)\} \end{aligned}$$

For example, the input () is accepted via

$$(q, (), B) \vdash (q, (), (B)) \vdash (q,), B)) \vdash (q,),)) \vdash (q, \varepsilon, \varepsilon).$$

From PDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ accepting by empty stack, build $G = (V, \Sigma, R, S)$ with variables

$$V = \{S\} \cup \{[pXq] \mid p, q \in Q,\ X \in \Gamma\}.$$

$[pXq]$ generates exactly the strings w with $(p, w, X) \vdash^* (q, \varepsilon, \varepsilon)$: going from p to q, P reads w and its net effect is to pop X.

The rules R are:

1. For every $p \in Q$: $\quad S \to [q_0 Z_0 p]$.
2. For every $(r, Y_1 Y_2 \cdots Y_k) \in \delta(q, a, X)$ with $a \in \Sigma \cup \{\varepsilon\}$, and every choice of states $r_1, \ldots, r_k \in Q$:

   $$[qXr_k] \ \to\ a\,[rY_1r_1][r_1Y_2r_2]\cdots[r_{k-1}Y_kr_k].$$

   If $k = 0$ (the move pops X), the rule is simply $[qXr] \to a$.

Then $L(G) = N(P)$.

Example: PDA to CFG

A one-state PDA $P = (\{q\}, \{i, e\}, \{Z\}, \delta, q, Z)$ that accepts by empty stack:

$$\begin{aligned} \delta(q, i, Z) &= \{(q, ZZ)\} \qquad \text{(an } \texttt{if} \text{ pushes one more } Z\text{)} \\ \delta(q, e, Z) &= \{(q, \varepsilon)\} \qquad \text{(an } \texttt{else} \text{ pops one } Z\text{)} \end{aligned}$$

With one state and one stack symbol, the only triple is $[qZq]$:

- $S \to [qZq]$
- From $\delta(q, i, Z) \ni (q, ZZ)$ with $r_1 = q$: $\quad [qZq] \to i\,[qZq][qZq]$
- From $\delta(q, e, Z) \ni (q, \varepsilon)$: $\quad [qZq] \to e$

Writing A for $[qZq]$:

$$S \to A, \qquad A \to iAA \mid e$$

For example, iee is derived by $S \Rightarrow A \Rightarrow iAA \Rightarrow ieA \Rightarrow iee$.

Deterministic Pushdown Automata

Definition (DPDA)

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a *deterministic* PDA (DPDA) if it makes at most one move at a time:

1. $|\delta(q, a, X)| \leq 1$ for every $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$.
2. If $\delta(q, a, X) \neq \emptyset$ for some $a \in \Sigma$, then $\delta(q, \varepsilon, X) = \emptyset$ (no choice between reading and an ε-move).

Definition

A language L is a *deterministic context-free language* (DCFL) iff $L = L(P)$ for some DPDA P.

DCFLs: Where They Sit

- **Fact 1.** DCFLs include some non-regular CFLs. For example,

$$L = \{a^n b^n \mid n \geq 0\}$$

  is a deterministic context-free language (we never need to guess).

- **Fact 2.** DCFLs do *not* include all CFLs. For example,

$$L = \{ww^R \mid w \in \{a, b\}^*\}$$

  is context-free but *not* deterministic: a DPDA cannot guess the midpoint of the string.

Regular Languages and DCFLs

Fact 3. DCFLs include all regular languages.

Theorem

If L is regular, then $L = L(P)$ for some DPDA P.

Proof.

Let $A = (Q, \Sigma, \delta_A, q_0, F)$ be a DFA for L. Construct the DPDA

$$P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$$

where $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$ whenever $\delta_A(q, a) = p$. The stack is never used; P tracks the DFA's state. Then $(q_0, w, Z_0) \vdash^* (p, \varepsilon, Z_0)$ iff $\hat{\delta}_A(q_0, w) = p$, so $L(P) = L(A) = L$. □

DPDA's and Ambiguous Grammars

Fact 4. Every DCFL has an unambiguous grammar.

Theorem

If $L = L(P)$ for some DPDA P, then L has an unambiguous grammar.

Fact 5. DCFLs do *not* include all unambiguously-generated CFLs. The language

$$L = \{ww^R \mid w \in \{a, b\}^*\}$$

has the unambiguous grammar

$$S \rightarrow aSa \mid bSb \mid \varepsilon$$

yet is *not* a DPDA language (by Fact 2).

Summary

- A PDA is a finite automaton with a stack.
- A PDA is strictly more powerful than an FA: PDAs recognize exactly the context-free languages.
  - Acceptance by final state and by empty stack are equivalent.
  - PDAs and CFGs define the same class of languages.
- Still limited: e.g. $\{ww \mid w \in \Sigma^*\}$ is not context-free.
- DPDAs sit strictly between FAs and PDAs:

$$\text{RL} \subsetneq \text{DCFL} \subsetneq \text{CFL}.$$

A broader question. What if we give an FA other external storage — a queue, two stacks, or random-access memory? Does that increase its language-recognizing power?