

Lecture 14 – Pushdown Automata (PDA)

COSE215: Theory of Computation

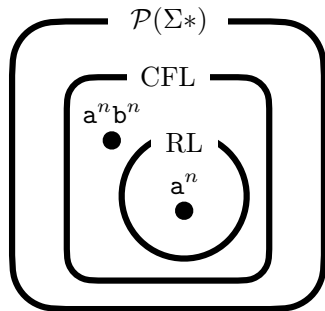
Jihyeok Park



2023 Spring

- A context-free grammar (CFG):

$$G = (V, \Sigma, S, P)$$



Languages	Automata	Grammars
Context-Free Language (CFL)	???	Context-Free Grammar (CFG)
Regular Language (RL)	Finite Automata (FA)	Regular Expression (RE)

1. Pushdown Automata

- Definition

- Transition Diagram

- Pushdown Automata in Scala

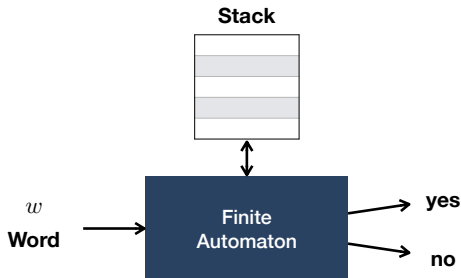
- Configurations and One-Step Moves

- Acceptance by Final States

- Acceptance by Empty Stacks

A **pushdown automaton (PDA)** is an ϵ -NFA with a **stack**:

- In FA, the next state is determined by the current **state** and **symbol**.
- In PDA, the next state is determined by the current **state**, **symbol**, and the **top element of the stack**.



Definition (Pushdown Automata)

A **pushdown automaton (PDA)** is a 7-tuple:

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$$

where

- Q is a finite set of **states**
- Σ is a finite set of **symbols**
- Γ is a finite set of **stack alphabets**
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$ is a **transition function**
- $q_0 \in Q$ is the **initial state**
- $Z \in \Gamma$ is the **initial stack alphabet** (the stack is initially Z)
- $F \subseteq Q$ is a set of **final states**

$$P_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \{X, Z\}, \delta, q_0, Z, \{q_2\})$$

where

$$\delta(q_0, a, Z) = \{(q_0, XZ)\}$$

$$\delta(q_0, b, Z) = \emptyset$$

$$\delta(q_0, \epsilon, Z) = \{(q_1, Z)\}$$

$$\delta(q_1, a, Z) = \emptyset$$

$$\delta(q_1, b, Z) = \emptyset$$

$$\delta(q_1, \epsilon, Z) = \{(q_2, Z)\}$$

$$\delta(q_2, a, Z) = \emptyset$$

$$\delta(q_2, b, Z) = \emptyset$$

$$\delta(q_2, \epsilon, Z) = \emptyset$$

$$\delta(q_0, a, X) = \{(q_0, XX)\}$$

$$\delta(q_0, b, X) = \emptyset$$

$$\delta(q_0, \epsilon, X) = \{(q_1, X)\}$$

$$\delta(q_1, a, X) = \emptyset$$

$$\delta(q_1, b, X) = \{(q_1, \epsilon)\}$$

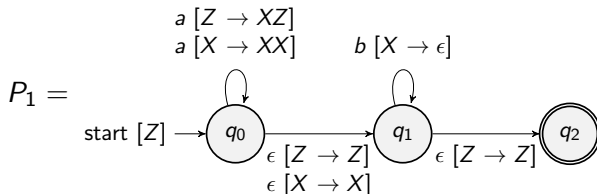
$$\delta(q_1, \epsilon, X) = \emptyset$$

$$\delta(q_2, a, X) = \emptyset$$

$$\delta(q_2, b, X) = \emptyset$$

$$\delta(q_2, \epsilon, X) = \emptyset$$

$$P_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \{X, Z\}, \delta, q_0, Z, \{q_2\})$$



For example,

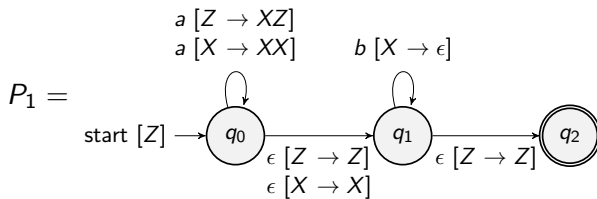
$$\begin{aligned}\delta(q_0, a, Z) &= \{(q_0, XZ)\} \\ \delta(q_0, \epsilon, X) &= \{(q_1, X)\}\end{aligned}$$

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$$

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$$

```
// The type definitions of states, symbols, words, and stack alphabets
type State = Int
type Symbol = Char
type Word = String
type Alphabet = Char

// The definition of PDA
case class PDA(
  states: Set[State],
  symbols: Set[Symbol],
  alphabets: Set[Alphabet],
  trans: Map[(State, Option[Symbol], Alphabet), Set[(State, List[Alphabet])]],
  initState: State,
  initAlphabet: Alphabet,
  finalStates: Set[State],
)
```

```
// An example of PDA
val pdal: PDA = PDA(
  states      = Set(0, 1, 2),          symbols      = Set('a', 'b'),
  alphabets   = Set('X', 'Z'),         trans        = Map(
    (0, Some('a'), 'Z') -> Set((0, List('X', 'Z'))),
    (0, Some('a'), 'X') -> Set((0, List('X', 'X'))),
    (0, None,      'Z') -> Set((1, List('Z'))),
    (0, None,      'X') -> Set((1, List('X'))),
    (1, Some('b'), 'X') -> Set((1, List())),
    (1, None,      'Z') -> Set((2, List('Z'))),
  ).withDefaultValue(Set()),
  initState   = 0,                    initAlphabet  = 'Z',
  finalStates = Set(2),
)
```

Definition (Configurations of PDA)

A **configuration** of a PDA P represents the current status of P . It is defined as a triple (q, w, α) where

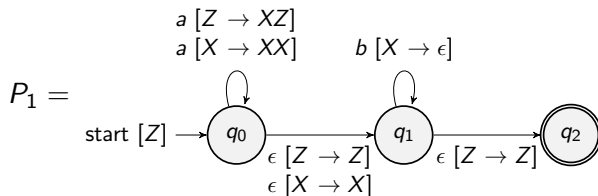
- $q \in Q$: the current state
- $w \in \Sigma^*$: the remaining word
- $\alpha \in \Gamma^*$: the current status of the stack

Definition (One-Step Moves of PDA)

A **one-step move** (\vdash) of a PDA P is a transition from a configuration to another configuration:

$$\begin{aligned}(q, aw, X\beta) &\vdash (p, w, \alpha\beta) && \text{if } (p, \alpha) \in \delta(q, a, X) \\ (q, w, X\beta) &\vdash (p, w, \alpha\beta) && \text{if } (p, \alpha) \in \delta(q, \epsilon, X)\end{aligned}$$

$$\begin{aligned} (q, aw, X\beta) \vdash (p, w, \alpha\beta) & \text{ if } (p, \alpha) \in \delta(q, a, X) \\ (q, w, X\beta) \vdash (p, w, \alpha\beta) & \text{ if } (p, \alpha) \in \delta(q, \epsilon, X) \end{aligned}$$



$$\begin{aligned} (q_0, ab, Z) & \vdash (q_0, b, XZ) & (\because (q_0, XZ) \in \delta(q_0, a, Z)) \\ & \vdash (q_1, b, XZ) & (\because (q_1, X) \in \delta(q_0, \epsilon, X)) \\ & \vdash (q_1, \epsilon, Z) & (\because (q_1, \epsilon) \in \delta(q_1, b, X)) \\ & \vdash (q_2, \epsilon, Z) & (\because (q_2, Z) \in \delta(q_1, \epsilon, Z)) \end{aligned}$$

Definition (Acceptance by Final States)

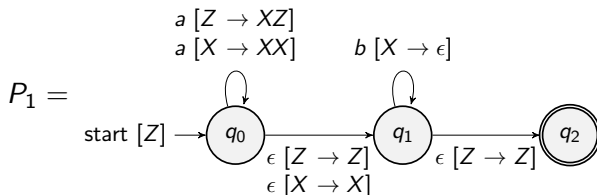
For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **final states** is defined as:

$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$

Definition (Acceptance by Final States)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **final states** is defined as:

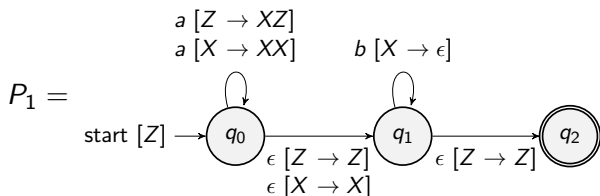
$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$



Definition (Acceptance by Final States)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **final states** is defined as:

$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$

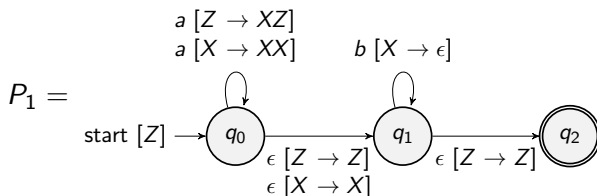


$$(q_0, ab, Z) \vdash^* (q_2, \epsilon, Z) \implies ab \in L_F(P)$$

Definition (Acceptance by Final States)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **final states** is defined as:

$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$

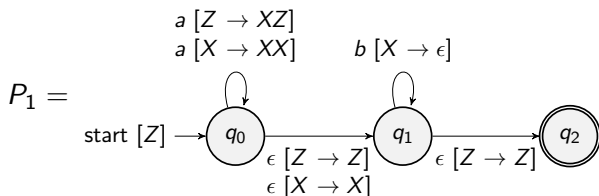


$$\begin{aligned}
 (q_0, ab, Z) &\vdash^* (q_2, \epsilon, Z) &\implies & ab \in L_F(P) \\
 (q_0, aabb, Z) &\vdash^* (q_2, \epsilon, Z) &\implies & aabb \in L_F(P)
 \end{aligned}$$

Definition (Acceptance by Final States)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **final states** is defined as:

$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$



$$\begin{aligned}
 (q_0, ab, Z) \vdash^* (q_2, \epsilon, Z) &\implies ab \in L_F(P) \\
 (q_0, aabb, Z) \vdash^* (q_2, \epsilon, Z) &\implies aabb \in L_F(P)
 \end{aligned}$$

$$L_F(P_1) = \{a^n b^n \mid n \geq 0\}$$

$$L_F(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^*\}$$

```
// The type definition of configurations
type Config = (State, Word, List[Alphabet])

// Configurations reachable from the initial configuration by one-step moves
def reachableConfig(pda: PDA)(init: Config): Set[Config] = ...

// Acceptance by final states
def acceptByFinalState(pda: PDA)(word: Word): Boolean =
  val init: Config = (pda.initState, word, List(pda.initAlphabet))
  reachableConfig(pda)(init).exists(config => {
    val (q, w, xs) = config
    w.isEmpty && pda.finalStates.contains(q)
  })

acceptByFinalState(pda1)("ab")    // true
acceptByFinalState(pda1)("aba")  // false
acceptByFinalState(pda1)("aabb") // true
acceptByFinalState(pda1)("abab") // false
```

Definition (Acceptance by Empty Stacks)

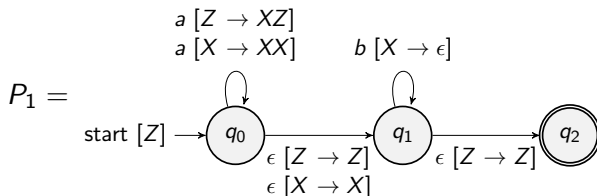
For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$

Definition (Acceptance by Empty Stacks)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

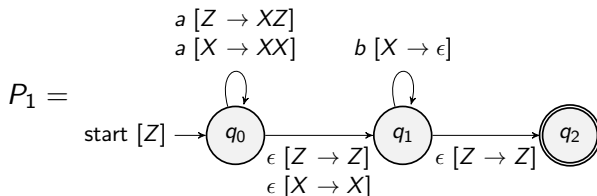
$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$



Definition (Acceptance by Empty Stacks)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$

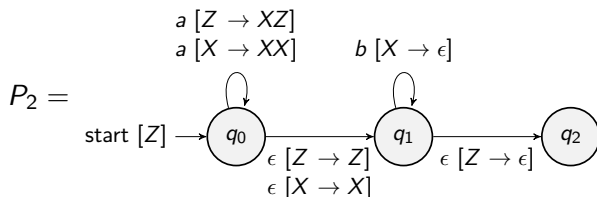


$$L_E(P_1) = \emptyset$$

Definition (Acceptance by Empty Stacks)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

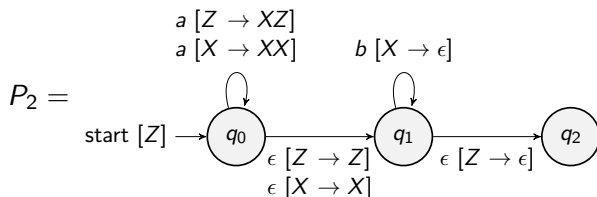
$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$



Definition (Acceptance by Empty Stacks)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$

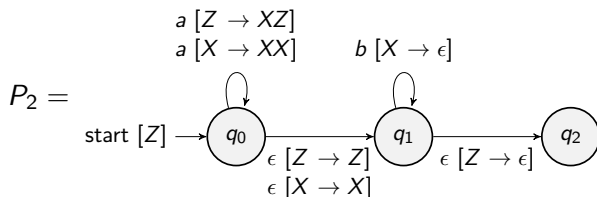


$$(q_0, ab, Z) \vdash^* (q_2, \epsilon, \epsilon) \implies ab \in L_E(P)$$

Definition (Acceptance by Empty Stacks)

For a given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, the language accepted by **empty stacks** is defined as:

$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$



$$(q_0, ab, Z) \vdash^* (q_2, \epsilon, \epsilon) \implies ab \in L_E(P)$$

$$L_E(P_2) = \{a^n b^n \mid n \geq 0\}$$

$$L_E(P) = \{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$$

```
// The type definition of configurations
type Config = (State, Word, List[Alphabet])

// Configurations reachable from the initial configuration by one-step moves
def reachableConfig(pda: PDA)(init: Config): Set[Config] = ...

// Acceptance by empty stacks
def acceptByEmptyStack(pda: PDA)(word: Word): Boolean =
  val init: Config = (pda.initState, word, List(pda.initAlphabet))
  reachableConfig(pda)(init).exists(config => {
    val (q, w, xs) = config
    w.isEmpty && xs.isEmpty
  })

// Another example of PDA
val pda2: PDA = ...

acceptByEmptyStack(pda2)("ab")    // true
acceptByEmptyStack(pda2)("aba")   // false
acceptByEmptyStack(pda2)("aabb")  // true
acceptByEmptyStack(pda2)("abab")  // false
```


1. Pushdown Automata

- Definition

- Transition Diagram

- Pushdown Automata in Scala

- Configurations and One-Step Moves

- Acceptance by Final States

- Acceptance by Empty Stacks

- Examples of Pushdown Automata

Jihyeok Park

jihyeok_park@korea.ac.kr

<https://plrg.korea.ac.kr>