Lecture 14 – Pushdown Automata (PDA)

COSE215: Theory of Computation

Jihyeok Park

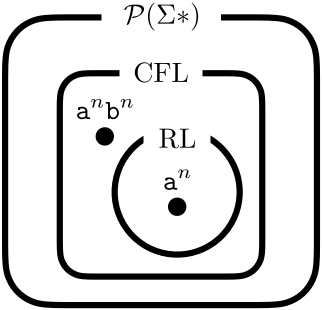
PLRG

2023 Spring

Recall

- 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)**

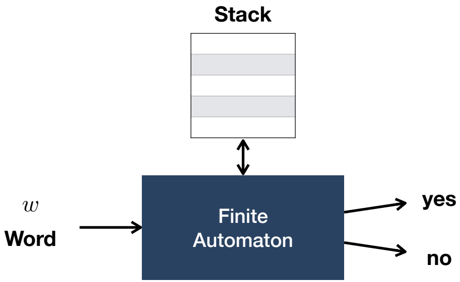
Contents

1. Pushdown Automata
   - Definition
   - Transition Diagram
   - Pushdown Automata in Scala
   - Configurations and One-Step Moves
   - Acceptance by Final States
   - Acceptance by Empty Stacks

Pushdown Automata

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 of Pushdown Automata

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

Definition of Pushdown Automata – Example

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

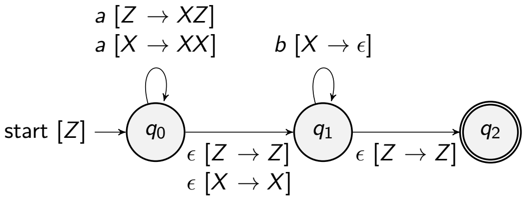
where

$$
\begin{array}{ll}
\delta(q_0, \mathtt{a}, Z) = \{(q_0, XZ)\} & \delta(q_0, \mathtt{a}, X) = \{(q_0, XX)\} \\
\delta(q_0, \mathtt{b}, Z) = \varnothing & \delta(q_0, \mathtt{b}, X) = \varnothing \\
\delta(q_0, \epsilon, Z) = \{(q_1, Z)\} & \delta(q_0, \epsilon, X) = \{(q_1, X)\} \\
\delta(q_1, \mathtt{a}, Z) = \varnothing & \delta(q_1, \mathtt{a}, X) = \varnothing \\
\delta(q_1, \mathtt{b}, Z) = \varnothing & \delta(q_1, \mathtt{b}, X) = \{(q_1, \epsilon)\} \\
\delta(q_1, \epsilon, Z) = \{(q_2, Z)\} & \delta(q_1, \epsilon, X) = \varnothing \\
\delta(q_2, \mathtt{a}, Z) = \varnothing & \delta(q_2, \mathtt{a}, X) = \varnothing \\
\delta(q_2, \mathtt{b}, Z) = \varnothing & \delta(q_2, \mathtt{b}, X) = \varnothing \\
\delta(q_2, \epsilon, Z) = \varnothing & \delta(q_2, \epsilon, X) = \varnothing
\end{array}
$$

Transition Diagram

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

$P_1 =$



For example,

$$\begin{array}{rcl} \delta(q_0, \mathtt{a}, Z) & = & \{(q_0, XZ)\} \\ \delta(q_0, \mathtt{a}, X) & = & \{(q_0, XX)\} \end{array}$$

Pushdown Automata in Scala

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

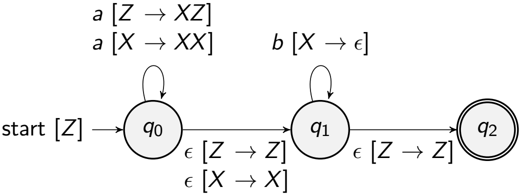
$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$$

```scala
// 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],
)
```

Pushdown Automata in Scala – Example

$P_1 =$




```scala
// An example of PDA
val pda1: 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),
)
```

Configurations and One-Step Moves

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)

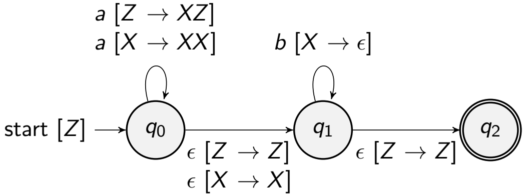
A **one-step move** ($\vdash$) of a PDA P is a transition from a configuration to another configuration:

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

Configurations and One-Step Moves

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

$P_1 =$



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

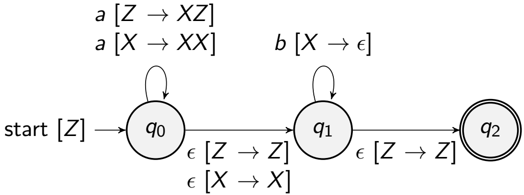
Acceptance by Final States

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:

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

$P_1 =$



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

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

Acceptance by Final States

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

```scala
// 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
acceptByEmptyStack(pda1)("abab") // false
```

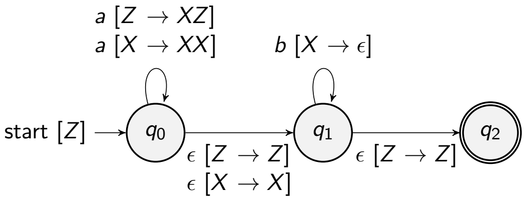
Acceptance by Empty Stacks

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:

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

$P_1 =$



$$L_E(P_1) = \varnothing$$

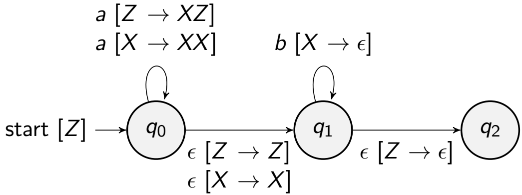
Acceptance by Empty Stacks

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:

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

$P_2 =$



$$(q_0, \texttt{ab}, Z) \vdash^* (q_2, \epsilon, \epsilon) \quad \Longrightarrow \quad \texttt{ab} \in L_E(P)$$

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

Acceptance by Empty Stacks

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

```
// 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
```

Summary

1. Pushdown Automata
    - Definition
    - Transition Diagram
    - Pushdown Automata in Scala
    - Configurations and One-Step Moves
    - Acceptance by Final States
    - Acceptance by Empty Stacks

Next Lecture

PLRG

- Examples of Pushdown Automata

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr