Lecture 14 – Pushdown Automata (PDA)

COSE215: Theory of Computation

Jihyeok Park

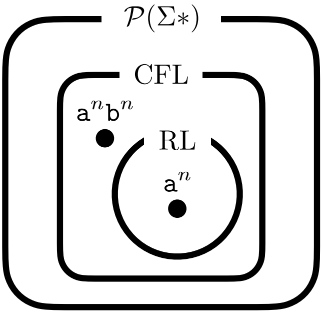
PLRG

2024 Spring

Recall

- A **context-free grammar (CFG)**:

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



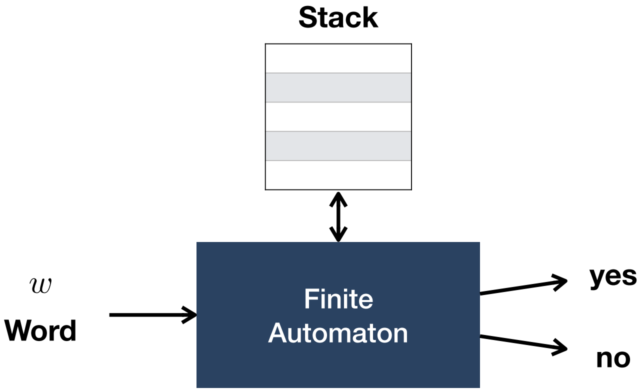
Languages	**Automata**	**Grammars**
Context-Free Language **(CFL)**	**???**	Context-Free Grammar **(CFG)**
Regular Language **(RL)**	Finite Automata **(FA)**	Regular Expression **(RE)**

Contents

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

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

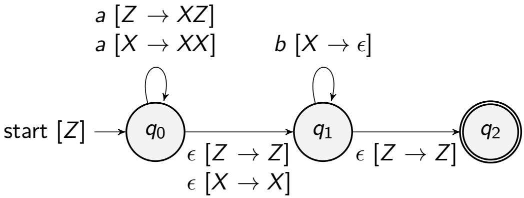
where

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

PLRG

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

$P_1 =$



For example,

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

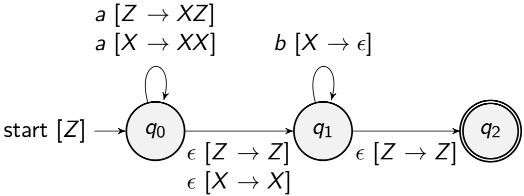
Pushdown Automata in Scala

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

// 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
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

PLRG

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)

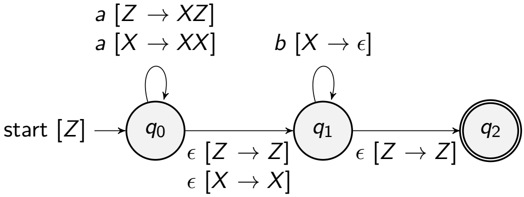
A **one-step move** ($\vdash$) of a PDA P is a transition from a configuration to another configuration:

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

Configurations and One-Step Moves

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

$P_1 =$



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

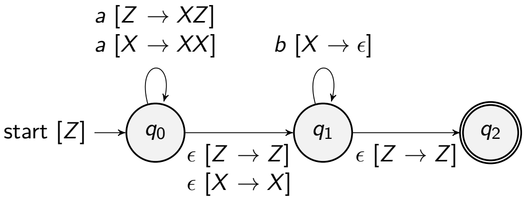
Acceptance by Final States

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:

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

$P_1 =$

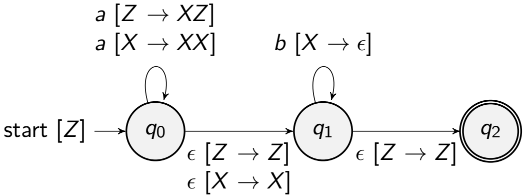


$$(q_0, \mathtt{ab}, Z) \vdash^* (q_2, \epsilon, Z) \implies \mathtt{ab} \in L_F(P)$$
$$(q_0, \mathtt{aabb}, Z) \vdash^* (q_2, \epsilon, Z) \implies \mathtt{aabb} \in L_F(P)$$

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

Acceptance by Final States

$P_1 =$



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

The key idea is to **count** the number of a's using the stack.

1. Start with the stack only having the initial stack alphabet Z.
2. Repeatedly **push** X onto the stack for each a.
3. Repeatedly **pop** X from the stack for each b.
4. Accept when the top of the stack is Z.

See the additional material for the input string aaabbb.[1]

[1] https://plrg.korea.ac.kr/courses/cose215/materials/pda-an-bn-final.pdf

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
case class Config(state: State, word: Word, stack: List[Alphabet])
case class PDA(...):
  // Configurations reachable from the initial configuration
  def reachableConfig(init: Config): Set[Config] = ... // See PDA.scala
  // The initial configuration
  def init(word: Word): Config =
    Config(initState, word, List(initAlphabet))
  // Acceptance by final states
  def acceptByFinalState(word: Word): Boolean =
    reachableConfig(init(word)).exists(config => {
      val Config(q, w, _) = config
      w.isEmpty && finalStates.contains(q)
    })

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

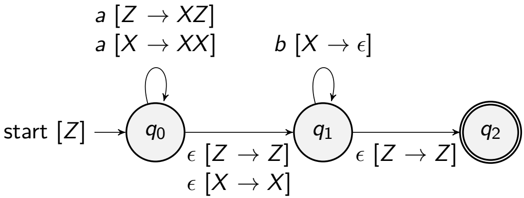
Acceptance by Empty Stacks

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:

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

$P_1 =$



$$L_E(P_1) = \varnothing$$

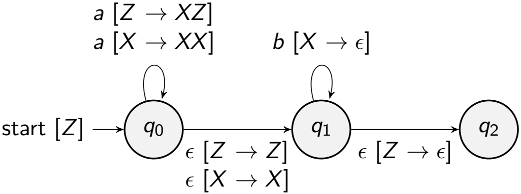
Acceptance by Empty Stacks

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:

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

$P_2 =$



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

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

Acceptance by Empty Stacks

PLRG

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

```scala
// The type definition of configurations
case class Config(state: State, word: Word, stack: List[Alphabet])
case class PDA(...):
  // Configurations reachable from the initial configuration
  def reachableConfig(init: Config): Set[Config] = ... // See PDA.scala
  // The initial configuration
  def init(word: Word): Config =
    Config(initState, word, List(initAlphabet))
  // Acceptance by empty stacks
  def acceptByEmptyStack(word: Word): Boolean =
    reachableConfig(init(word)).exists(config => {
      val Config(_, w, xs) = config
      w.isEmpty && xs.isEmpty
    })

acceptByEmptyStack(pda2)("ab")   // true
acceptByEmptyStack(pda2)("aba")  // 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

- Examples of Pushdown Automata

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