Pushdown Automata

Raju Hazari

Department of Computer Science and Engineering
National Institute of Technology Calicut

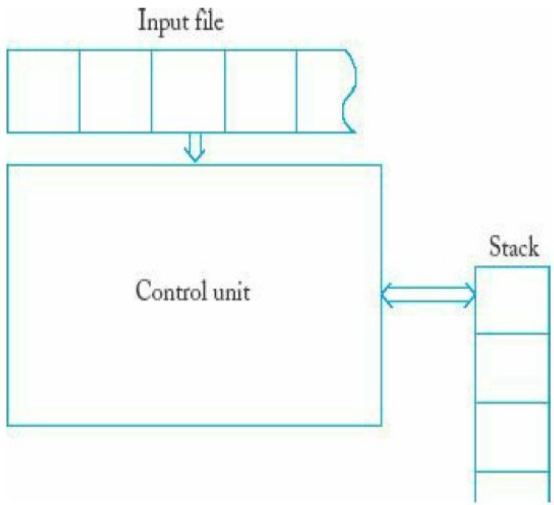
November 9, 2024

Pushdown Automata

- The accepting device corresponding to context-free grammar is pushdown automata.

Pushdown Automata

- The accepting device corresponding to context-free grammar is pushdown automata.



Schematic representation of a pushdown automaton

Pushdown Automata

- Pushdown automata has an input file, a finite control and a stack or pushdown store.
- Each move of the control unit reads a symbol from the input file, while at the same time changing the contents of the stack through the usual stack operations.
- Each move of the control unit is determined by the current input symbol as well as by the symbol currently on top of the stack.
- The result of the move is a new state of the control unit and a change in the top of the stack.

Pushdown Automata

- A **Non-deterministic Pushdown Automata** is defined by the sep-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_o, z, F),$$

where

  - Q is a finite set of **internal states** of the control unit,
  - Σ is the **input alphabet**,
  - Γ is a finite set of symbols called the **stack alphabet**,
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to$ set of finite subsets of $Q \times \Gamma^*$ is the transition function,
  - $q_0 \in Q$ is the **initial state**,
  - $z \in \Gamma$ is the **stack start symbol**,
  - $F \subseteq Q$ is the set of final states

- The mappings are like: $\delta(p, a, z)$ contains (q, γ), where $p, q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $z \in \Gamma$, $\gamma \in \Gamma^*$.

Pushdown Automata

- **Example :** Consider an PDA with

$$\begin{aligned}
&Q = \{q_0, q_1, q_2, q_3\},\\
&\Sigma = \{a, b\},\\
&\Gamma = \{0, 1\},\\
&z = 0,\\
&F = \{q_3\}
\end{aligned}$$

  with initial state q_0 and

$$\begin{aligned}
&\delta(q_0, a, 0) = \{(q_1, 10), (q_3, \epsilon)\},\\
&\delta(q_0, \epsilon, 0) = \{(q_3, \epsilon)\},\\
&\delta(q_1, a, 1) = \{(q_1, 11)\},\\
&\delta(q_1, b, 1) = \{(q_2, \epsilon)\},\\
&\delta(q_2, b, 1) = \{(q_2, \epsilon)\},\\
&\delta(q_2, \epsilon, 0) = \{(q_3, \epsilon)\}
\end{aligned}$$

- The language accepted by the automata :

Pushdown Automata

- **Example :** Consider an PDA with

$$
\begin{aligned}
&Q = \{q_0, q_1, q_2, q_3\},\\
&\Sigma = \{a, b\},\\
&\Gamma = \{0, 1\},\\
&z = 0,\\
&F = \{q_3\}
\end{aligned}
$$

  with initial state q_0 and

$$
\begin{aligned}
&\delta(q_0, a, 0) = \{(q_1, 10), (q_3, \epsilon)\},\\
&\delta(q_0, \epsilon, 0) = \{(q_3, \epsilon)\},\\
&\delta(q_1, a, 1) = \{(q_1, 11)\},\\
&\delta(q_1, b, 1) = \{(q_2, \epsilon)\},\\
&\delta(q_2, b, 1) = \{(q_2, \epsilon)\},\\
&\delta(q_2, \epsilon, 0) = \{(q_3, \epsilon)\}
\end{aligned}
$$

- The language accepted by the automata :
  $L = \{a^n b^n : n \geq 0\} \cup \{a\}$

Graphical Notation for PDA's

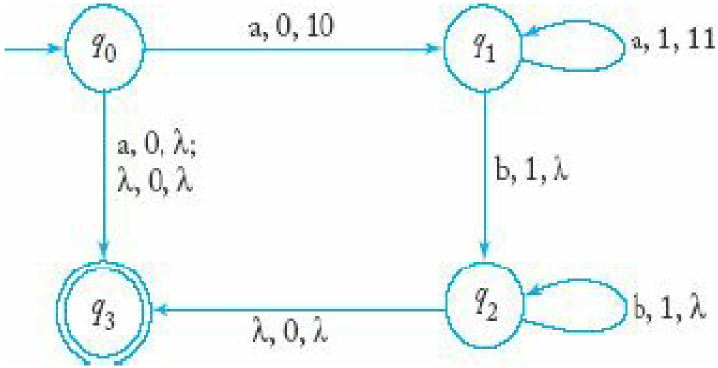
We can also use transition graphs to represent PDA.

- The nodes corresponds to the states of the PDA.
- An arrow labeled Start indicates the start state, and doubly circled states are accepting state.
- In this representation we label the edges of the graph with three things : the current input symbol, the symbol at the top of the stack, and the string that replaces the top of the stack.

Graphical Notation for PDA's

We can also use transition graphs to represent PDA.

- The nodes corresponds to the states of the PDA.
- An arrow labeled Start indicates the start state, and doubly circled states are accepting state.
- In this representation we label the edges of the graph with three things : the current input symbol, the symbol at the top of the stack, and the string that replaces the top of the stack.



- The only thing that the diagram does not tell us is which stack symbol is the start symbol.

Instantaneous Description of a PDA

- The PDA goes from configuration to configuration, in response to input symbols (or sometimes ϵ), but unlike the finite automaton, where the state is the only thing that we need to know about the automaton, the PDA's configuration involves both the state and the contents of the stack.

Instantaneous Description of a PDA

- The PDA goes from configuration to configuration, in response to input symbols (or sometimes ϵ), but unlike the finite automaton, where the state is the only thing that we need to know about the automaton, the PDA's configuration involves both the state and the contents of the stack.

- How we represent the configuration of a PDA ?
  - We shall represent the configuration of a PDA by a triple (q, w, γ), where q is the current state, w is the remaining input, and γ is the stack contents. Conventionally, the top of the stack at the left end of γ and the bottom at the right end.
  - Such a triple is called an **instantaneous description**, or ID, of the pushdown automaton.
  - Let $(q, aw, X\beta)$ is an ID, that is the machine is in state q, and portion of the input remaining is aw, X is the top of the stack under you have β. Suppose $\delta(q, a, X)$ contains (p, α). Then for all strings w in Σ^* and β in Γ^* : $(q, aw, X\beta) \vdash (p, w, \alpha\beta)$

The Languages Accepted by a PDA

- We have assumed that a PDA accepts its input by consuming it and entering an accepting state. We call this approach "**acceptance by final state**".
- There is a second approach to defined the language of a PDA that has important applications. We may also define for any PDA the language "**accepted by empty stack**" , that is, the set of strings that cause the PDA to empty its stack, starting from the initial ID.
- These two methods are equivalent, in the sense that a language L has a PDA that accepts it by final state if and only if L has a PDA that accepts it by empty stack.
- However, for a given PDA M, the languages that M accepts by final state and by empty stack are usually different.

The Languages Accepted by a PDA

- **Acceptance by Final State :**

  Let $M = (Q, \Sigma, \Gamma, \delta, q_o, z, F)$ be a PDA. Then $L(M)$, the language accepted by M by final state, is

  $$L(M) = \{w | w \in \Sigma^*, (q_0, w, z) \vdash^* (q_f, \epsilon, \gamma)\}$$

  for some state q_f in F and any stack string γ. That is, starting in the initial ID with w waiting on the input, M consumes w from the input and enters an accepting state. The contents of the stack at that time is irrelevant.

The Languages Accepted by a PDA

- **Acceptance by Final State :**

  Let $M = (Q, \Sigma, \Gamma, \delta, q_o, z, F)$ be a PDA. Then $L(M)$, the language accepted by M by final state, is

  $$L(M) = \{w | w \in \Sigma^*, (q_0, w, z) \vdash^* (q_f, \epsilon, \gamma)\}$$

  for some state q_f in F and any stack string γ. That is, starting in the initial ID with w waiting on the input, M consumes w from the input and enters an accepting state. The contents of the stack at that time is irrelevant.

- **Acceptance by Empty Stack :**

  Let $M = (Q, \Sigma, \Gamma, \delta, q_o, z, F)$ be a PDA. Then $N(M)$, the language accepted by M by empty stack, is

  $$N(M) = \{w | w \in \Sigma^*, (q_0, w, z) \vdash^* (q, \epsilon, \epsilon)\}$$

  for any state q. That is, $N(M)$ is the set of inputs w that M can consume and at the same time empty its stack.

The Languages Accepted by a PDA

- Example 1: Construct an PDA for accepting the language
$$L = \{wcw^R : w \in \{a, b\}^+\}$$

- Here, we consider acceptance by empty stack

$$\text{Let } M = (\{q_1, q_2\}, \{0, 1, c\}, \{r, b, g\}, \delta, q_1, r, \phi)$$

$$\begin{array}{ll}
\delta(q_1, 0, r) = \{(q_1, br)\}, & \delta(q_1, 0, g) = \{(q_1, bg)\}, \\
\delta(q_1, 1, r) = \{(q_1, gr)\}, & \delta(q_1, 1, g) = \{(q_1, gg)\}, \\
\delta(q_1, c, r) = \{(q_2, r)\}, & \delta(q_1, c, g) = \{(q_2, g)\}, \\
\delta(q_1, 0, b) = \{(q_1, bb)\}, & \delta(q_2, 0, b) = \{(q_2, \epsilon)\}, \\
\delta(q_1, 1, b) = \{(q_1, gb)\}, & \delta(q_2, 1, g) = \{(q_2, \epsilon)\}, \\
\delta(q_1, c, b) = \{(q_2, b)\}, & \delta(q_2, \epsilon, r) = \{(q_2, \epsilon)\}
\end{array}$$

The Languages Accepted by a PDA

- Example 1: Construct an PDA for accepting the language
$$L = \{wcw^R : w \in \{a, b\}^+\}$$

- Here, we consider acceptance by empty stack

$$\text{Let } M = (\{q_1, q_2\}, \{0, 1, c\}, \{r, b, g\}, \delta, q_1, r, \phi)$$

$$\begin{aligned}
&\delta(q_1, 0, r) = \{(q_1, br)\}, && \delta(q_1, 0, g) = \{(q_1, bg)\},\\
&\delta(q_1, 1, r) = \{(q_1, gr)\}, && \delta(q_1, 1, g) = \{(q_1, gg)\},\\
&\delta(q_1, c, r) = \{(q_2, r)\}, && \delta(q_1, c, g) = \{(q_2, g)\},\\
&\delta(q_1, 0, b) = \{(q_1, bb)\}, && \delta(q_2, 0, b) = \{(q_2, \epsilon)\},\\
&\delta(q_1, 1, b) = \{(q_1, gb)\}, && \delta(q_2, 1, g) = \{(q_2, \epsilon)\},\\
&\delta(q_1, c, b) = \{(q_2, b)\}, && \delta(q_2, \epsilon, r) = \{(q_2, \epsilon)\}
\end{aligned}$$

- Consider the string 110c011, how this string accepted by the PDA

$(q_1, 110c011, r) \vdash (q_1, 10c011, gr) \vdash (q_1, 0c011, ggr) \vdash$
$(q_1, c011, bggr) \vdash (q_2, 011, bggr) \vdash (q_2, 11, ggr) \vdash (q_2, 1, gr) \vdash$
$(q_2, \epsilon, r) \vdash (q_2, \epsilon, \epsilon)$

The Languages Accepted by a PDA

- Example 2: Construct an PDA for accepting the language
$$L = \{ww^R : w \in \{a, b\}^+\}$$

The Languages Accepted by a PDA

- Example 2: Construct an PDA for accepting the language
$$L = \{ww^R : w \in \{a, b\}^+\}$$
- Here, we consider acceptance by empty stack

Let $M = (\{q_1, q_2\}, \{0, 1\}, \{r, b, g\}, \delta, q_1, r, \phi)$

$$\begin{aligned}
\delta(q_1, 0, r) &= \{(q_1, br)\}, \\
\delta(q_1, 1, r) &= \{(q_1, gr)\}, \\
\delta(q_1, 0, b) &= \{(q_1, bb), (q_2, \epsilon)\}, \\
\delta(q_1, 1, b) &= \{(q_1, gb)\}, \\
\delta(q_1, 0, g) &= \{(q_1, bg)\}, \\
\delta(q_1, 1, g) &= \{(q_1, gg), (q_2, \epsilon)\}, \\
\delta(q_2, 0, b) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, 1, g) &= \{q_2, \epsilon\}, \\
\delta(q_2, \epsilon, r) &= \{(q_2, \epsilon)\}
\end{aligned}$$

The Languages Accepted by a PDA

$$(q_1, 110011, r)$$

The Languages Accepted by a PDA

$$\begin{array}{c} (q_1, 110011, r) \\ \Big\downarrow \\ (q_1, 10011, gr) \end{array}$$

The Languages Accepted by a PDA

$$
\begin{tikzcd}
 & (q_1, 110011, r) \arrow[d] & \\
 & (q_1, 10011, gr) \arrow[dl] \arrow[dr] & \\
(q_1, 0011, ggr) & & (q_2, 0011, r)
\end{tikzcd}
$$

The Languages Accepted by a PDA

$$
\begin{tikzcd}
 & (q_1, 110011, r) \arrow[d] & \\
 & (q_1, 10011, gr) \arrow[dl] \arrow[dr] & \\
(q_1, 0011, ggr) & & (q_2, 0011, r) \arrow[d] \\
 & & (q_2, 0011, \epsilon)
\end{tikzcd}
$$

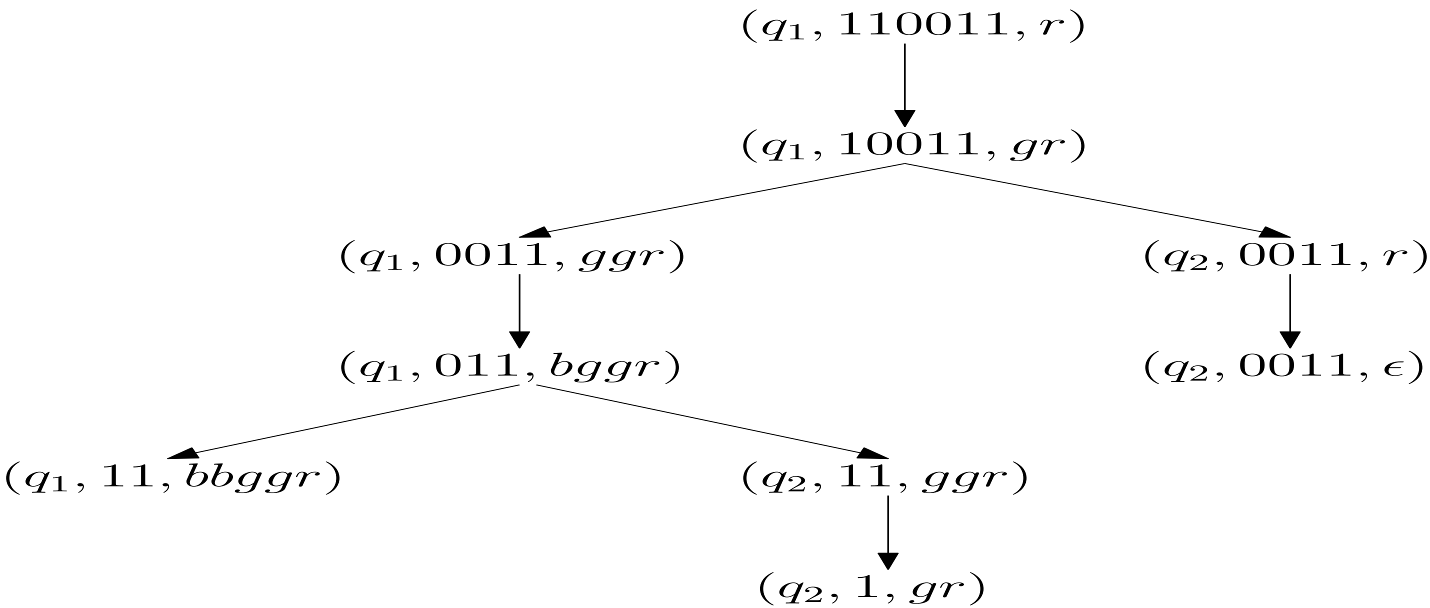
The Languages Accepted by a PDA

$$
\begin{tikzcd}
& (q_1, 110011, r) \arrow[d] & \\
& (q_1, 10011, gr) \arrow[dl] \arrow[dr] & \\
(q_1, 0011, ggr) \arrow[d] & & (q_2, 0011, r) \arrow[d] \\
(q_1, 011, bggr) & & (q_2, 0011, \epsilon)
\end{tikzcd}
$$

The Languages Accepted by a PDA

$$
\begin{tikzcd}
 & & (q_1, 110011, r) \arrow[d] & \\
 & & (q_1, 10011, gr) \arrow[dl] \arrow[dr] & \\
 & (q_1, 0011, ggr) \arrow[d] & & (q_2, 0011, r) \arrow[d] \\
 & (q_1, 011, bggr) \arrow[dl] \arrow[dr] & & (q_2, 0011, \epsilon) \\
(q_1, 11, bbggr) & & (q_2, 11, ggr) &
\end{tikzcd}
$$

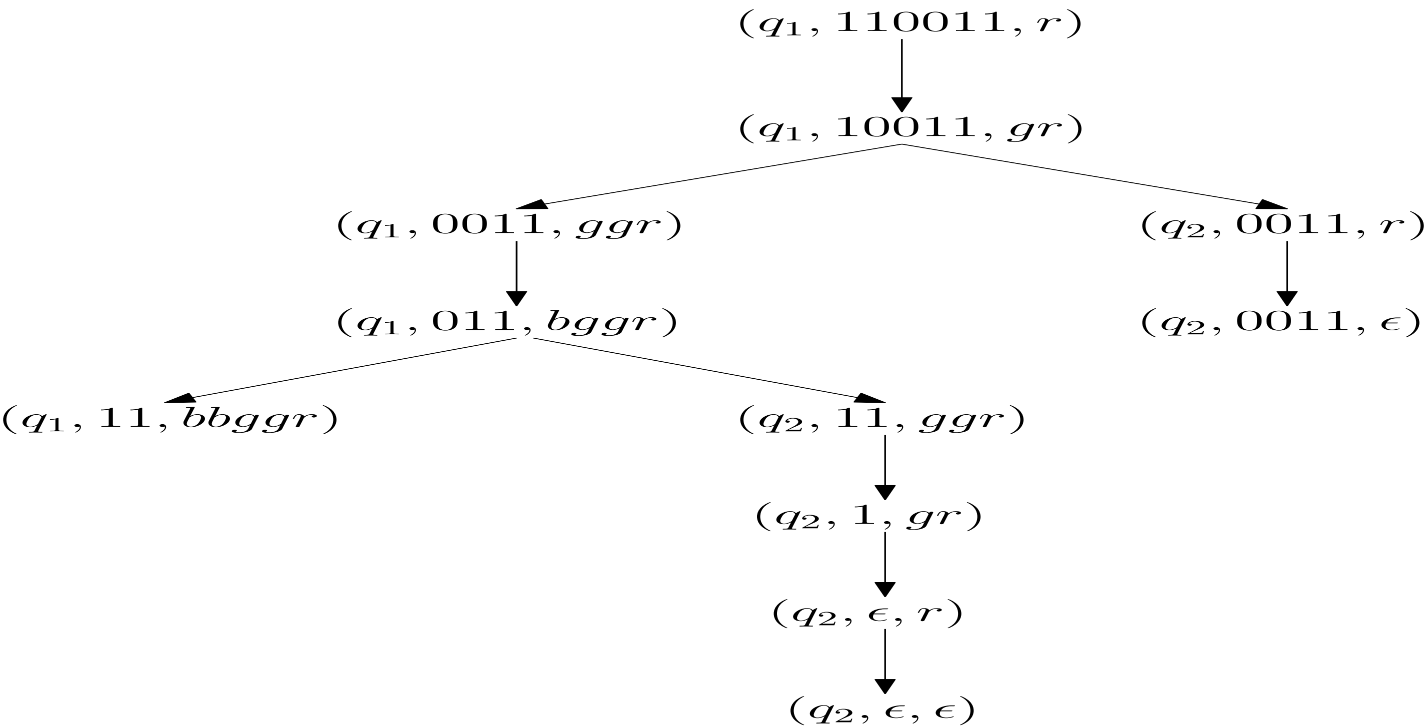
The Languages Accepted by a PDA



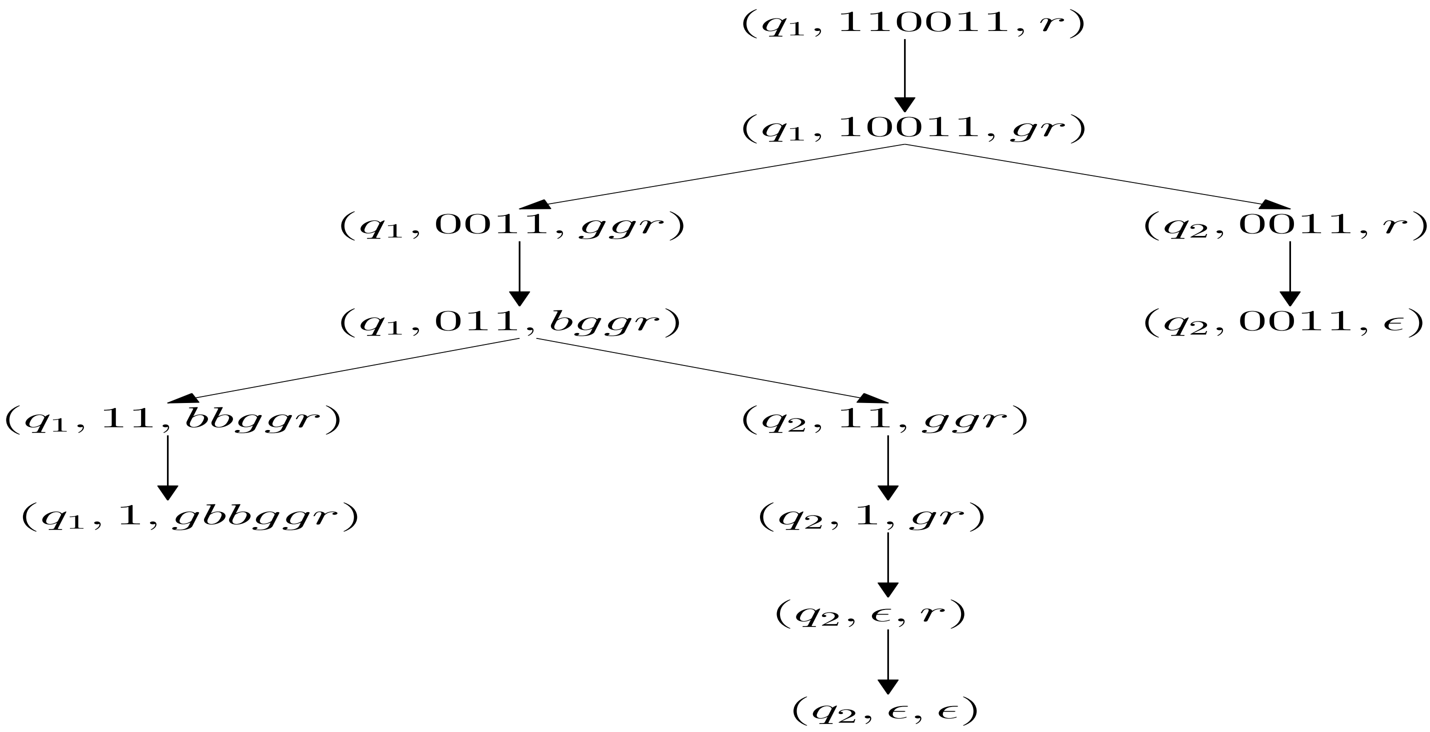
The Languages Accepted by a PDA

$$
\begin{tikzcd}[column sep=small]
 & & (q_1, 110011, r) \arrow[d] & \\
 & & (q_1, 10011, gr) \arrow[dl] \arrow[dr] & \\
 & (q_1, 0011, ggr) \arrow[d] & & (q_2, 0011, r) \arrow[d] \\
 & (q_1, 011, bggr) \arrow[dl] \arrow[dr] & & (q_2, 0011, \epsilon) \\
(q_1, 11, bbggr) & & (q_2, 11, ggr) \arrow[d] & \\
 & & (q_2, 1, gr) \arrow[d] & \\
 & & (q_2, \epsilon, r) &
\end{tikzcd}
$$

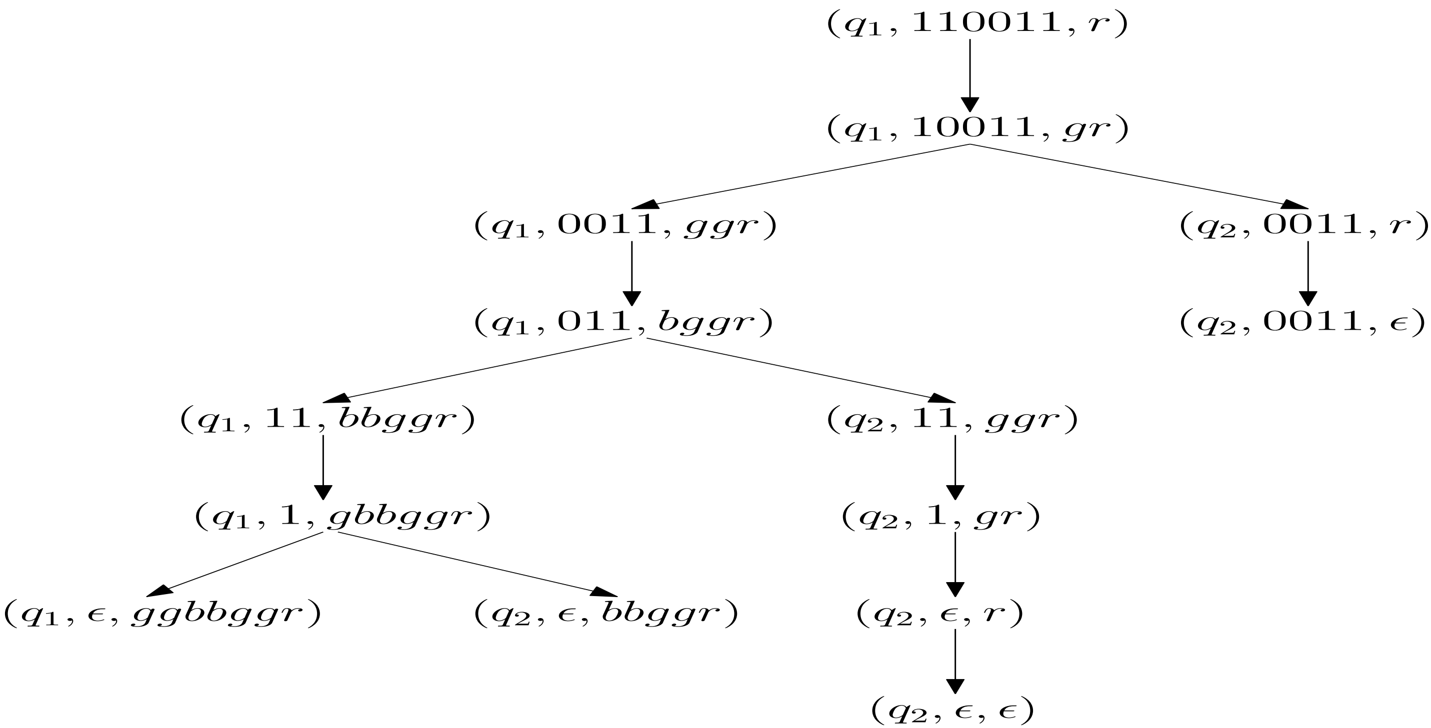
The Languages Accepted by a PDA



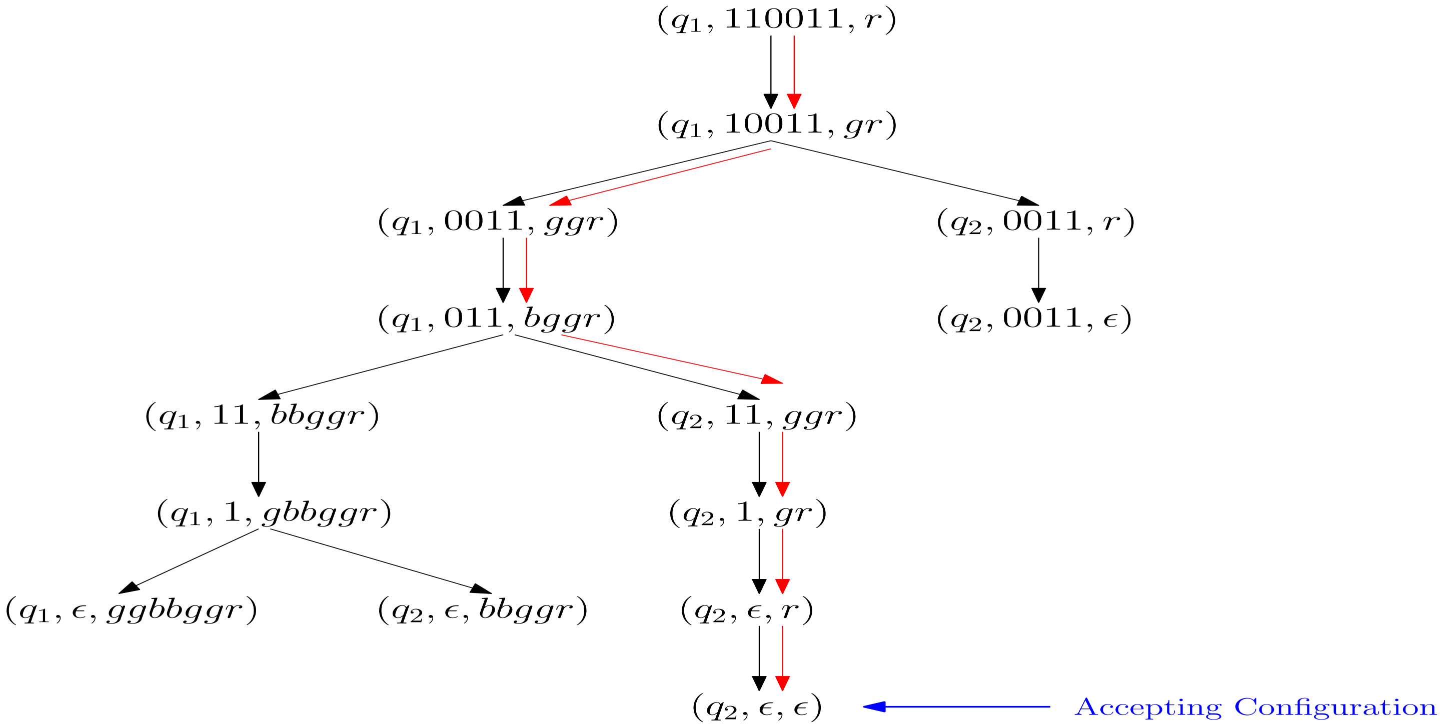
The Languages Accepted by a PDA



The Languages Accepted by a PDA



The Languages Accepted by a PDA



Equivalence of Acceptance by final state and Acceptance by empty stack

- **Theorem :** Let L be $L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Then there is a PDA P_N such that $L = N(P_N)$.

- **Proof :** The construction is
$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0, \phi)$$
where δ_N is defined by:

  1. $\delta_N(p_0, \epsilon, X_0)$ contains (q_0, Z_0X_0). We start by pushing the start symbol of P_F onto the stack and going to the start state of P_F.
  2. For all states q in Q, input symbols a in Σ or $a = \epsilon$, and Y in Γ, $\delta_N(q, a, Y)$ contains every pair that is in $\delta_F(q, a, Y)$. That is, P_N simulates P_F.
  3. For all accepting states q in F and stack symbols Y in Γ or $Y = X_0$, $\delta_N(q, \epsilon, Y)$ contains (p, ϵ). By the rule, whenever P_F accepts, P_N can start emptying its stack without consuming any more input.
  4. For all stack symbols Y in Γ or $Y = X_0$, $\delta_N(p, \epsilon, Y)$ contains (p, ϵ). Once in state p, which only occurs when P_F has accepted, P_N pops every symbol on its stack, until the stack is empty. No further input is consumed.

Equivalence of Acceptance by final state and Acceptance by empty stack

Now, we must prove that w is in $N(P_N)$ if and only if w is in $L(P_F)$.

- **If Part :** Suppose $(q_0, w, Z_0) \vdash^*_{P_F} (q, \epsilon, \alpha)$ for some accepting state q and stack string α. Using the fact that every transaction of P_F is a move of P_N, we know that $(q_0, w, Z_0X_0) \vdash^*_{P_N} (q, \epsilon, \alpha X_0)$. Then P_N can do the following :
$$(p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0) \vdash^*_{P_N} (q, \epsilon, \alpha X_0) \vdash^*_{P_N} (p, \epsilon, \epsilon)$$
The first move is by rule(1) of the construction of P_N, while the last sequence of moves is by rules (3) and (4). Thus, w is accepted by P_N, by empty stack.

- **Only-if Part :** The only way P_N can empty its stack is by entering state p, since X_0 is sitting at the bottom of stack and X_0 is not a symbol on which P_F has any moves. The only way P_N can enter state p is if the simulated P_F enters an accepting state. The first move of P_N is surely the move given in rule(1). Thus, every accepting computation of P_N looks like
$$(p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0) \vdash^*_{P_N} (q, \epsilon, \alpha X_0) \vdash^*_{P_N} (p, \epsilon, \epsilon)$$
where q is an accepting state of P_F.

  Moreover, between ID's (q_0, w, Z_0X_0) and $(q, \epsilon, \alpha X_0)$, all the moves are moves of P_F. In particular, X_0 was never the top stack symbol prior to reaching ID $(q, \epsilon, \alpha X_0)$. Thus, we conclude that the same computation can occur in P_F, without the X_0 on the stack; that is, $(q_0, w, Z_0) \vdash^*_{P_F} (q, \epsilon, \alpha)$. Now, we see that P_F accepts w by final state, so w is in $L(P_F)$.

Equivalence of Acceptance by empty stack and Acceptance by final state

- **Theorem :** If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0, \phi)$, then there is a PDA P_F such that $L = L(P_F)$.
- **Proof :** The specification of P_F is as follows :

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

where δ_F is defined by:

1. $\delta_F(p_0, \epsilon, X_0)$ contains (q_0, Z_0X_0). In its start state, P_F makes a spontaneous transition to the start state of P_N, pushing its start symbol Z_0 onto the stack.
2. For all states q in Q, inputs a in Σ or $a = \epsilon$, and stack symbols Y in Γ, $\delta_F(q, a, Y)$ contains all the pairs in $\delta_N(q, a, Y)$.
3. In addition to rule(2), $\delta_F(q, \epsilon, X_0)$ contains (p_f, ϵ) for every state q in Q.

Equivalence of Acceptance by empty stack and Acceptance by final state

We must show that w is in $L(P_F)$ if and only if w is in $N(P_N)$.

- **If Part :** We are given that $(q_0, w, Z_0) \vdash^*_{P_N} (q, \epsilon, \epsilon)$ for some state q. Insert X_0 at the bottom of the stack and conclude $(q_0, w, Z_0X_0) \vdash^*_{P_N} (q, \epsilon, X_0)$. Since by rule(2), P_F has all the moves of P_N, we may also conclude that $(q_0, w, Z_0X_0) \vdash^*_{P_F} (q, \epsilon, X_0)$. If we put this sequence of moves together with the initial and final moves from rules(1) and (3) above, we get:

  $$(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash^*_{P_F} (q, \epsilon, X_0) \vdash (p_f, \epsilon, \epsilon)$$

  Thus, P_F accepts w by final state.

- **Only-if Part :** The converse requires only that we observe the additional transitions of rules(1) and (3) give us very limited ways to accept w by final state. We must use rule(3) at the last step, and we can only use that rule if the stack of P_F contains only X_0. No X_0's ever appear on the stack except at the bottommost position. Further, rule(1) is only used at the first step, and it must be used at the first step.

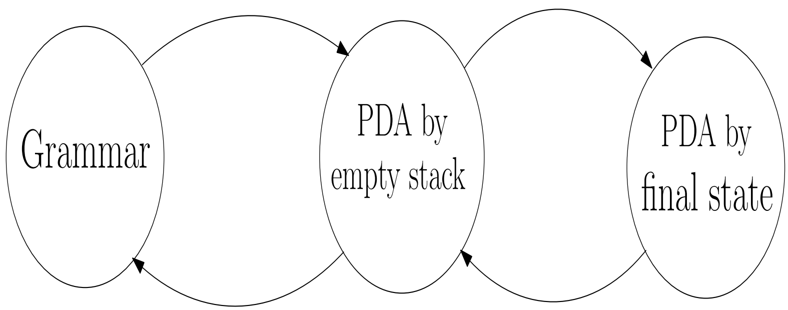
  Thus, any computation of P_F that accepts w must look like

  $$(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash^*_{P_F} (q, \epsilon, X_0) \vdash (p_f, \epsilon, \epsilon)$$

  Moreover, the middle of the computation— all but the first and last steps—must also be a computation of P_N with X_0 below the stack. The reason is that, except for the first and last steps, P_F cannot use any transition that is not also a transition of P_N, and X_0 cannot be exposed at the next step. We conclude that $(q_0, w, Z_0) \vdash^*_{P_N} (q, \epsilon, \epsilon)$. That is, w is in $N(P_N)$.

Equivalence of PDA's and CFG's

- We can prove that the following three classes of languages are all the same class.
  - The context-free languages, i.e., the languages defined by CFG's.
  - The languages that are accepted by final state by some PDA.
  - The languages that are accepted by empty stack by some PDA.



Equivalence of PDA's and CFG's

Given a context-free grammar, how to construct an equivalent pushdown automata ?

- Two proofs are possible, in one we can assume that the context-free grammar is in Greibach normal form. In another proof no such an assumption is necessary.
- Let L be generated by a CFG $G = (N, T, P, S)$. Then L can be accepted by a PDA M by empty stack.

$$M = (\{q\}, T, N \cup T, \delta, q, S, \phi)$$

  where transition function δ is defined by:

  1. For each non-terminal A,

     $\delta(q, \epsilon, A)$ contains (q, β), if $A \to \beta$ is in P, where $\beta \in (N \cup T)^*$
  2. For each terminal a, $\delta(q, a, a)$ contains (q, ϵ)

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

to a PDA that accepts the same language by empty stack.

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \rightarrow aSb,$$
$$S \rightarrow ab$$

  to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

$(q, aaabbb, S)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :
$$S \to aSb,$$
$$S \to ab$$
to a PDA that accepts the same language by empty stack.
  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:
    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)
- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :
$$S \rightarrow aSb,$$
$$S \rightarrow ab$$
to a PDA that accepts the same language by empty stack.
  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:
    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)
- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \rightarrow aSb,$$
$$S \rightarrow ab$$

  to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :
$$S \to aSb,$$
$$S \to ab$$
to a PDA that accepts the same language by empty stack.
  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:
    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)
- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
  $\vdash (q, abbb, Sbb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

$(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
$\vdash (q, abbb, Sbb) \vdash (q, abbb, abbb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

  to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
  $\vdash (q, abbb, Sbb) \vdash (q, abbb, abbb) \vdash (q, bbb, bbb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:
    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

$(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
$\vdash (q, abbb, Sbb) \vdash (q, abbb, abbb) \vdash (q, bbb, bbb) \vdash (q, bb, bb)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :

$$S \to aSb,$$
$$S \to ab$$

  to a PDA that accepts the same language by empty stack.

  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:

    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)

- The language generated by the grammar is equal number of a's followed by an equal number of b's

  $(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
  $\vdash (q, abbb, Sbb) \vdash (q, abbb, abbb) \vdash (q, bbb, bbb) \vdash (q, bb, bb)$
  $\vdash (q, b, b)$

Equivalence of PDA's and CFG's

- **Example :** Convert the grammar G :
$$S \to aSb,$$
$$S \to ab$$
to a PDA that accepts the same language by empty stack.
  - Construct a PDA $M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$

    where the transition function δ is defined by:
    - ★ $\delta(q, \epsilon, S)$ contain (q, aSb)
    - ★ $\delta(q, \epsilon, S)$ contain (q, ab)
    - ★ $\delta(q, a, a)$ contain (q, ϵ)
    - ★ $\delta(q, b, b)$ contain (q, ϵ)
- The language generated by the grammar is equal number of a's followed by an equal number of b's

$(q, aaabbb, S) \vdash (q, aaabbb, aSb) \vdash (q, aabbb, Sb) \vdash (q, aabbb, aSbb)$
$\vdash (q, abbb, Sbb) \vdash (q, abbb, abbb) \vdash (q, bbb, bbb) \vdash (q, bb, bb)$
$\vdash (q, b, b) \vdash (q, \epsilon, \epsilon)$

Equivalence of PDA's and CFG's

Given a PDA by empty stack, how to construct an equivalent context-free grammar ?

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$ be a PDA accepting a context-free language L by empty stack. Construct $G = (N, T, P, S)$ such that $L(G) = N(M)$
  - $N = \{S\} \cup \{[q, A, p]\} \qquad |q, p \in Q, A \in \Gamma$
    - ⋆ In the triples, first and third component are states in the PDA and middle component is a pushdown symbol.
    - ⋆ If there is k states and m symbols then the number of non-terminals will be $k^2m + 1$
  - $T = \Sigma$, $S = S$ and P :
  - For all states p, G has the production $S \to [q_0, Z_0, p]$
  - Let $\delta(q, a, X)$ contain the pair $(r, Y_1 Y_2 \cdots Y_k)$, where $a \in \Sigma \cup \{\epsilon\}$, k can be any number, including 0, in which case the pair is (r, ϵ).

    Then for all lists of states $r_1, r_2, \cdots, r_k$, G has the production $[q, X, r_k] \to a[r, Y_1, r_1][r_1, Y_2, r_2] \cdots [r_{k-1}, Y_k, r_k]$
  - Suppose, the rule of the form $\delta(q, a, X)$ contain (r, ϵ) then

$$[q, X, r] \to a$$

Equivalence of PDA's and CFG's

- **Example :** Convert the PDA $M = (\{q_0, q_1\}, \{0, 1\}, \{Z_0, X\}, \delta, q_0, Z_0, \phi)$ to a CFG, if δ is given by:

$$
\begin{aligned}
&\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\} \\
&\delta(q_0, 0, X) = \{(q_0, XX)\} \\
&\delta(q_0, 1, X) = \{(q_1, \epsilon)\} \\
&\delta(q_1, 1, X) = \{(q_1, \epsilon)\} \\
&\delta(q_1, \epsilon, X) = \{(q_1, \epsilon)\} \\
&\delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}
\end{aligned}
$$

  - The language accepted by PDA is $0^m 1^n$ where $m \geq n$, $m, n \geq 1$
  - $N = \{S, [q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1], [q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1]\}$
  - $T = \{0, 1\}$
  - S is the start symbol

Equivalence of PDA's and CFG's

- The production rules are:

$$S \to [q_0, Z_0, q_0], \qquad S \to [q_0, Z_0, q_1]$$

$[q_0, X, q_1] \to 1$, $[q_1, X, q_1] \to 1$, $[q_1, X, q_1] \to \epsilon$, $[q_1, Z_0, q_1] \to \epsilon$

$$\begin{aligned}
&[q_0, Z_0, q_0] \to 0[q_0, X, q_0][q_0, Z_0, q_0]\\
&[q_0, Z_0, q_0] \to 0[q_0, X, q_1][q_1, Z_0, q_0]\\
&[q_0, Z_0, q_1] \to 0[q_0, X, q_0][q_0, Z_0, q_1]\\
&[q_0, Z_0, q_1] \to 0[q_0, X, q_1][q_1, Z_0, q_1]\\
&[q_0, X, q_0] \to 0[q_0, X, q_0][q_0, X, q_0]\\
&[q_0, X, q_0] \to 0[q_0, X, q_1][q_1, X, q_0]\\
&[q_0, X, q_1] \to 0[q_0, X, q_0][q_0, X, q_1]\\
&[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]
\end{aligned}$$

Now, we replaced $[q_0, Z_0, q_0]$, $[q_0, Z_0, q_1]$, $[q_1, Z_0, q_0]$, $[q_1, Z_0, q_1]$, $[q_0, X, q_0]$, $[q_0, X, q_1]$, $[q_1, X, q_0]$, $[q_1, X, q_1]$ with A, B, C, D, E, F, G, H respectively.

Equivalence of PDA's and CFG's

$$
\begin{array}{ll}
S \rightarrow A, & B \rightarrow 0EB, \\
S \rightarrow B, & B \rightarrow 0FD, \\
F \rightarrow 1, & E \rightarrow 0EE, \\
H \rightarrow 1, & E \rightarrow 0FG, \\
H \rightarrow \epsilon, & F \rightarrow 0EF, \\
D \rightarrow \epsilon, & F \rightarrow 0FH \\
A \rightarrow 0EA, & \\
A \rightarrow 0FC, &
\end{array}
$$

Now, we can remove the useless non-terminals. After removing the useless non-terminals, the final rule sets

$$
\begin{array}{l}
S \rightarrow B, \\
B \rightarrow 0FD, \\
F \rightarrow 0FH, \\
F \rightarrow 1, \\
H \rightarrow 1, \\
H \rightarrow \epsilon, \\
D \rightarrow \epsilon
\end{array}
$$

Deterministic Pushdown Automata (DPDA)

- A PDA is deterministic if there is never a choice of move in any situation.
- **Definition :** A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ is said to be deterministic (a deterministic PDA or DPDA), if and only if the following conditions are met:
  1. $\delta(q, a, X)$ has at most one member for any q in Q, a in Σ or $a = \epsilon$, and X in Γ.
  2. If $\delta(q, \epsilon, X)$ is nonempty, then $\delta(q, a, X)$ is empty for all a in Σ.

  - The first condition is that for any given input symbol and any stack top, at most one move can be made.
  - The second condition is that when ϵ-move is possible for some configuration, no input-consuming alternative is available.
- $DPDA \subseteq CFL$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

$(q_0, aaabbb, 0)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$
\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\},\\
\delta(q_1, a, 1) &= \{(q_1, 11)\},\\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\},\\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\},\\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}
$$

  $(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned} \delta(q_0, a, 0) &= \{(q_1, 10)\}, \\ \delta(q_1, a, 1) &= \{(q_1, 11)\}, \\ \delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\ \delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\ \delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\} \end{aligned}$$

$(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

$(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110) \vdash (q_1, bbb, 1110)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$
\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}
$$

  $(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110) \vdash (q_1, bbb, 1110) \vdash (q_2, bb, 110)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

$(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110) \vdash (q_1, bbb, 1110)$
$\vdash (q_2, bb, 110) \vdash (q_2, b, 10)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

$(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110) \vdash (q_1, bbb, 1110)$
$\vdash (q_2, bb, 110) \vdash (q_2, b, 10) \vdash (q_2, \epsilon, 0)$

Deterministic Pushdown Automata (DPDA)

- A language L is said to be a **deterministic context-free language** if and only if there exists a DPDA M such that $L = L(M)$.

- **Example :** The language $L = \{a^n b^n : n \geq 0\}$ is a deterministic context-free language.
  The PDA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \epsilon)\}, \\
\delta(q_2, \epsilon, 0) &= \{(q_0, \epsilon)\}
\end{aligned}$$

$(q_0, aaabbb, 0) \vdash (q_1, aabbb, 10) \vdash (q_1, abbb, 110) \vdash (q_1, bbb, 1110)$
$\vdash (q_2, bb, 110) \vdash (q_2, b, 10) \vdash (q_2, \epsilon, 0) \vdash (q_0, \epsilon, \epsilon)$

The Pumping Lemma for Context-Free Languages

- We develop a tool, called "**pumping lemma for context-free languages**", for showing that certain languages are not context-free.
- The "**pumping lemma for context-free languages**" says that in any sufficiently long string in a CFL, it is possible to find at most two short, nearby substrings, that we can "pump" in tandem. That is, we may repeat both of the strings i times, for any integer i, and the resultant string will still be in the language.

The Pumping Lemma for Context-Free Languages

- **Theorem : (The pumping lemma for context-free languages)** Let L be a CFL. Then there exists a constant n such that if z is any string in L such that $|z|$ is at least n, then we write $z = uvwxy$, subject to the following conditions:

   1. $|vwx| \leq n$. That is, the middle portion is not too long.
   2. $vx \neq \epsilon$ or $|vx| \geq 1$. Since v and x are the pieces to be "pumped", this condition says that at least one of the strings we pump must not be empty.
   3. For all $i \geq 0$, uv^iwx^iy is in L. That is, the two strings v and x may be "pumped" any number of times, including 0, and the resulting string will still be a member of L.

The Pumping Lemma for Context-Free Languages

- **Example :** Let L be the language $\{0^n 1^n 2^n \mid n \geq 1\}$. Suppose L be a context-free language. Then there is an integer n given to us by the pumping lemma. Let us pick $z = 0^n 1^n 2^n$.

  Suppose the "adversary" break z as $z = uvwxy$, where $|vwx| \leq n$ and v and x are not both ϵ. Then we know that vwx cannot involve both 0's and 2's, since the last 0 and the first 2 are separated by $n + 1$ positions. We shall prove that L contains some string known not to be in L, thus contradicting the assumption that L is a CFL. The cases are as follows :

  - vwx has no 2's. Then vx consists of only 0's and 1's, and has at least one of these symbols. Then uwy, which would have to be in L by the pumping lemma, has n 2's, but has fewer that n 0's or fewer that n 1's, or both. It therefore does not belong in L, and we conclude L is not a CFL in this case.
  - vwx has no 0's. Similarly, uwy has n 0's, but fewer 1's or fewer 2's. It therefore is not in L.

  Whichever case holds, we conclude that L has a string we know not to be in L. This contradiction allows us to conclude that our assumption was wrong; L is not a CFL.

Closure Properties of Context-Free Languages

- **Theorem 1 :** The context-free languages are closed under the following operation :
  1. Union
  2. Concatenation
  3. Closure(*), and positive closure($^+$)
  4. Homomorphism

Closure Properties of Context-Free Languages

- **Theorem 1 :** The context-free languages are closed under the following operation :
  1. Union
  2. Concatenation
  3. Closure(*), and positive closure($^+$)
  4. Homomorphism

- **Theorem 2 :** The family of context-free languages is not closed under intersection and complementation.

Closure Properties of Context-Free Languages

- **Theorem 1 :** The context-free languages are closed under the following operation :
  1. Union
  2. Concatenation
  3. Closure(*), and positive closure($^+$)
  4. Homomorphism

- **Theorem 2 :** The family of context-free languages is not closed under intersection and complementation.

- **Theorem 3 :** Let L_1 be a context-free language and L_2 be a regular language. Then $L_1 \cap L_2$ is context-free.

Closure Properties of Context-Free Languages

- **Theorem 1 :** The context-free languages are closed under the following operation :
  1. Union
  2. Concatenation
  3. Closure(*), and positive closure($^+$)
  4. Homomorphism

- **Theorem 2 :** The family of context-free languages is not closed under intersection and complementation.

- **Theorem 3 :** Let L_1 be a context-free language and L_2 be a regular language. Then $L_1 \cap L_2$ is context-free.

- **Theorem 4 :** If L is a CFL, then so is L^R.

Closure Properties of Context-Free Languages

- **Theorem 5 :** The following are true about a CFL's L, L_1, and L_2, and a regular language R.
  1. $L - R$ is a context-free language
  2. $\overline{L}$ is not necessarily a context-free language.
  3. $L_1 - L_2$ is not necessarily context-free.
- **Theorem 6 :** Let L be a CFL and h a homomorphism. Then $h^{-1}(L)$ is a CFL.