COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

March 15, 2023

Lecture 19: CFGs to PDAs and back

Equivalence of NPDA and CFG

Theorem

$L = L(G)$ for some context-free grammar G if and only if it is accepted by some NPDA.

NPDA

Definition

A non-deterministic pushdown automaton (NPDA)
$A = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$, where

Q:	set of states	Σ:	input alphabet
Γ:	stack alphabet	q_0:	start state
$\perp$:	start symbol	F:	set of final states

$\delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*$.

Understanding δ

For $q \in Q, a \in \Sigma$ and $X \in \Gamma$, if $\delta(q, a, X) = (p, \gamma)$,
then p is the new state and γ replaces X in the stack.

if $\gamma = \epsilon$ then X is popped.
if $\gamma = X$ then X stays unchanges on the top of the stack.
if $\gamma = \gamma_1\gamma_2 \dots \gamma_k$ then X is replaced by γ_k
and $\gamma_1\gamma_2 \dots \gamma_{k-1}$ are pushed on top of that.

Example

Consider the grammar:

Example

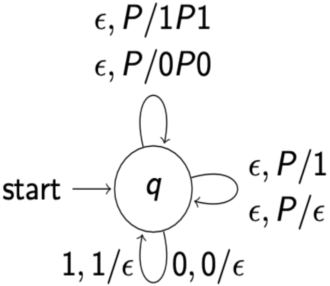
Consider the grammar:

$$
\begin{array}{ll}
P \to 0P0 & P \to 1P1 \\
P \to \varepsilon & P \to 1
\end{array}
$$

Example

Consider the grammar:

$$P \to 0P0 \quad P \to 1P1$$
$$P \to \varepsilon \quad P \to 1$$



$$A_G = (\{q\}, \{0,1\}, \{0,1,P\}, \delta, q, P, \varnothing)$$

Equivalence of NPDAs and CFGs

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Proof.

- Assume CFG is in the Chomsky normal form.
- Push S_0 on the stack and make it the current variable.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Proof.

- Assume CFG is in the Chomsky normal form.
- Push S_0 on the stack and make it the current variable.
- Push non-deterministically one of the strings in the right hand side of the rule generated from the current variable on the stack.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Proof.

- Assume CFG is in the Chomsky normal form.
- Push S_0 on the stack and make it the current variable.
- Push non-deterministically one of the strings in the right hand side of the rule generated from the current variable on the stack.

  e.g. $A \to BC \mid DE$ then non-deterministically choose either BC or DE and depending on the choice, say it is BC, push the string BC on the stack with B on the top of the stack.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Proof.

- Assume CFG is in the Chomsky normal form.
- Push S_0 on the stack and make it the current variable.
- Push non-deterministically one of the strings in the right hand side of the rule generated from the current variable on the stack.

  e.g. $A \to BC \mid DE$ then non-deterministically choose either BC or DE and depending on the choice, say it is BC, push the string BC on the stack with B on the top of the stack.
- If the the top is a terminal, then match it off with the input bit,
- If the top of the stack is $\perp$ then accept else make that the new current variable.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Proof.

- Assume CFG is in the Chomsky normal form.
- Push S_0 on the stack and make it the current variable.
- Push non-deterministically one of the strings in the right hand side of the rule generated from the current variable on the stack.

  e.g. $A \to BC \mid DE$ then non-deterministically choose either BC or DE and depending on the choice, say it is BC, push the string BC on the stack with B on the top of the stack.
- If the the top is a terminal, then match it off with the input bit,
- If the top of the stack is $\perp$ then accept else make that the new current variable.

Repeat the above procedure. $\square$

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)
- $\Sigma = T$ (Input Alphabet is set of terminals)

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)
- $\Sigma = T$ (Input Alphabet is set of terminals)
- $\Gamma = V \cup T$ (Stack alphabet is terminals and non-terminals)

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)
- $\Sigma = T$ (Input Alphabet is set of terminals)
- $\Gamma = V \cup T$ (Stack alphabet is terminals and non-terminals)
- $\perp = S$ (stack bottom is start symbol of CFG)
- $F = \varnothing$ (Acceptance by empty stack)

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)
- $\Sigma = T$ (Input Alphabet is set of terminals)
- $\Gamma = V \cup T$ (Stack alphabet is terminals and non-terminals)
- $\perp = S$ (stack bottom is start symbol of CFG)
- $F = \varnothing$ (Acceptance by empty stack)
- δ is defined as:

$$\delta(q, \epsilon, B) := \{(q, \beta) \mid B \to \beta \text{ in } P\}$$

$$\delta(q, a, a) := \{(q, \epsilon)\}$$

CFG to NPDA

Theorem

If $L = L(G)$ for some context-free grammar G then it is accepted by some NPDA.

Let $G = (V, T, P, S)$ then $A_G = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ where

- $Q = \{q\} = q_0$ (Single state)
- $\Sigma = T$ (Input Alphabet is set of terminals)
- $\Gamma = V \cup T$ (Stack alphabet is terminals and non-terminals)
- $\perp = S$ (stack bottom is start symbol of CFG)
- $F = \varnothing$ (Acceptance by empty stack)
- δ is defined as:

$$\delta(q, \epsilon, B) := \{(q, \beta) \mid B \to \beta \text{ in } P\}$$

$$\delta(q, a, a) := \{(q, \epsilon)\}$$

Guess production rule and push on to the stack and verify guess while popping.

NPDA to CFG

Theorem

$L = L(M)$ *for some NPDA* M *then there is some grammar* G *such that* $L(G) = L(M)$.

NPDA to CFG

Theorem

$L = L(M)$ for some NPDA M then there is some grammar G such that $L(G) = L(M)$.

Proof.

- Want: Given PDA P need CFG G_P that generates all strings P accepts

NPDA to CFG

Theorem

$L = L(M)$ for some NPDA M then there is some grammar G such that $L(G) = L(M)$.

Proof.

- Want: Given PDA P need CFG G_P that generates all strings P accepts
- G should generate a string if that string causes PDA to go from start to accept state.

NPDA to CFG

Theorem

$L = L(M)$ for some NPDA M then there is some grammar G such that $L(G) = L(M)$.

Proof.

- Want: Given PDA P need CFG G_P that generates all strings P accepts
- G should generate a string if that string causes PDA to go from start to accept state.
- Idea: Design a CFG that for each pair of states p, q in P, have a variable $A_{p,q}$ which generates all strings that can take P from p (with empty stack) to q (with empty stack).

NPDA to CFG

Theorem

$L = L(M)$ for some NPDA M then there is some grammar G such that $L(G) = L(M)$.

Proof.

- Want: Given PDA P need CFG G_P that generates all strings P accepts
- G should generate a string if that string causes PDA to go from start to accept state.
- Idea: Design a CFG that for each pair of states p, q in P, have a variable $A_{p,q}$ which generates all strings that can take P from p (with empty stack) to q (with empty stack).
- Modify P so that
  - It has single accept state.
  - It empties its stack before accepting.
  - Each transition either pushes a symbol or pops a symbol (not both).

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ).

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$
- $S = A_{q_0, q_F}$

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$
- $S = A_{q_0, q_F}$
- P has transitions of the following form:
  - $A_{q,q} \to \epsilon$ for all $q \in Q$;

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$
- $S = A_{q_0, q_F}$
- P has transitions of the following form:
  - $A_{q,q} \to \epsilon$ for all $q \in Q$;
  - $A_{p,q} \to A_{p,r} A_{r,q}$ for all $p, q, r \in Q$,

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$
- $S = A_{q_0,q_F}$
- P has transitions of the following form:
  - $A_{q,q} \to \epsilon$ for all $q \in Q$;
  - $A_{p,q} \to A_{p,r}A_{r,q}$ for all $p, q, r \in Q$,
  - $A_{p,q} \to aA_{r,s}b$ if $\delta(p, a, \epsilon)$ contains (r, X) and $\delta(s, b, X)$ contains (q, ϵ).

NPDA to CFG

Given a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \perp, q_F)$ with restriction that every transition is either pushes a symbol or pops a symbol form the stack, i.e. $\delta(q, a, X)$ contains either (q_0, YX) or (q_0, ϵ). Consider the grammar $G_p = (V, T, P, S)$ such that

- $V = \{A_{p,q} : p, q \in Q\}$
- $T = \Sigma$
- $S = A_{q_0, q_F}$
- P has transitions of the following form:
  - $A_{q,q} \to \epsilon$ for all $q \in Q$;
  - $A_{p,q} \to A_{p,r} A_{r,q}$ for all $p, q, r \in Q$,
  - $A_{p,q} \to a A_{r,s} b$ if $\delta(p, a, \epsilon)$ contains (r, X) and $\delta(s, b, X)$ contains (q, ϵ).

Lemma

$L(G_p) = L(P)$.

PDA to CFG

Lemma

If $A_{p,q} \Longrightarrow^* x$ *then* x *can bring the PDA* P *from state* p *on empty stack to state* q *on empty stack.*

PDA to CFG

Lemma

If $A_{p,q} \Longrightarrow^* x$ then x can bring the PDA P from state p on empty stack to state q on empty stack.

Proof (by induction on number of steps in derivation of x from $A_{p,q}$.)

PDA to CFG

Lemma

If $A_{p,q} \Longrightarrow^* x$ then x can bring the PDA P from state p on empty stack to state q on empty stack.

Proof (by induction on number of steps in derivation of x from $A_{p,q}$.)

- **Base case.** If $A_{p,q} \Longrightarrow^* x$ in one step, then the only rule that can generate a variable free string in one step is $A_{p,p} \rightarrow \epsilon$.

PDA to CFG

Lemma

If $A_{p,q} \Longrightarrow^* x$ then x can bring the PDA P from state p on empty stack to state q on empty stack.

Proof (by induction on number of steps in derivation of x from $A_{p,q}$.)

- **Base case.** If $A_{p,q} \Longrightarrow^* x$ in one step, then the only rule that can generate a variable free string in one step is $A_{p,p} \rightarrow \epsilon$.
- **Inductive step.** If $A_{p,q} \Longrightarrow^* x$ in $n+1$ steps. The first step in the derivation must be $A_{p,q} \rightarrow A_{p,r} A_{r,q}$ or $A_{p,q} \rightarrow a A_{r,s} b$.

PDA to CFG

Lemma

If $A_{p,q} \Longrightarrow^* x$ then x can bring the PDA P from state p on empty stack to state q on empty stack.

Proof (by induction on number of steps in derivation of x from $A_{p,q}$.)

- **Base case.** If $A_{p,q} \Longrightarrow^* x$ in one step, then the only rule that can generate a variable free string in one step is $A_{p,p} \to \epsilon$.
- **Inductive step.** If $A_{p,q} \Longrightarrow^* x$ in $n+1$ steps. The first step in the derivation must be $A_{p,q} \to A_{p,r} A_{r,q}$ or $A_{p,q} \to a A_{r,s} b$.
  - If it is $A_{p,q} \to A_{p,r} A_{r,q}$, then the string x can be broken into two parts $x_1 x_2$ such that $A_{p,r} \Longrightarrow^* x_1$ and $A_{r,q} \Longrightarrow^* x_2$ in at most n steps. The claim easily follows in this case.
  - If it is $A_{p,q} \to a A_{r,s} b$, then the string x can be broken as ayb such that $A_{r,s} \Longrightarrow^* y$ in n steps. Notice that from p on reading a the PDA pushes a symbol X to stack, while it pops X in state s and goes to q.

$\square$