Chapter 7.2

Theorem Given NPDA M that accepts by final state, ∃ NPDA M' that accepts by empty stack s.t. L(M)=L(M').

- **Proof** (sketch)
  M=(Q,Σ,Γ,δ,q_0,z,F)
  Construct M'=(Q',Σ,Γ',δ',q_s,z',F')

Theorem Given NPDA M that accepts by empty stack, ∃ NPDA M' that accepts by final state.

- **Proof:** (sketch)
  M=(Q,Σ,Γ,δ,q_0,z,F)
  Construct M'=(Q',Σ,Γ',δ',q_s,z',F')

Theorem For any CFL L not containing λ, $\exists$ an NPDA M s.t. L=L(M).

- **Proof** (sketch)
  Given (λ-free) CFL L.
  $\Rightarrow$ $\exists$ CFG G such that L=L(G).
  $\Rightarrow$ $\exists$ G' in GNF, s.t. L(G)=L(G').
  G'=(V,T,S,P). All productions in P are of the form:

Example: Let G'=(V,T,S,P), P=

$$
\begin{aligned}
&S \rightarrow aSA \mid aAA \mid b \\
&A \rightarrow bBBB \\
&B \rightarrow b
\end{aligned}
$$

Theorem Given a NPDA M, ∃ a NPDA M' s.t. all transitions have the form $\delta(q_i$,a,A)=$\{c_1, c_2, \ldots c_n\}$ where

$$\begin{aligned} & c_i = (q_j, \lambda) \\ \text{or} \quad & c_i = (q_j, \text{BC}) \end{aligned}$$

Each move either increases or decreases stack contents by a single symbol.

- **Proof** (sketch)

Theorem If L=L(M) for some NPDA M, then L is a CFL.

- **Proof:** Given NPDA M.

  First, construct an equivalent NPDA M that will be easier to work with. Construct M' such that

  1. accepts if stack is empty
  2. each move increases or decreases stack content by a single symbol. (can only push 2 variables or no variables with each transition)

  M'=(Q,Σ,Γ,δ,q_0,z,F)

  Construct G=(V,Σ,S,P) where

  V=$\{(q_i c q_j)|q_i, q_j \in Q, c \in \Gamma\}$

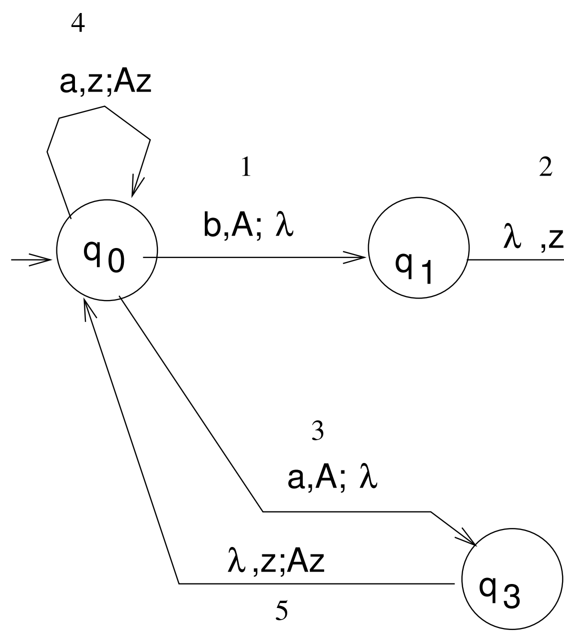
  $(q_i c q_j)$ represents "starting at state q_i the stack contents are cw, $w \in \Gamma^*$, some path is followed to state q_j and the contents of the stack are now w".

  Goal: $(q_0 z q_f)$ which will be the start symbol in the grammar.

  Meaning: We start in state q_0 with z on the stack and process the input tape. Eventually we will reach the final state q_f and the stack will be empty. (Along the way we may push symbols on the stack, but these symbols will be popped from the stack).

Example:

L(M)={aa^*b}, M=(Q,Σ,Γ,δ,q_0,z,F), Q={q_0, q_1, q_2, q_3}, Σ={a, b},Γ={A, z},F={}. M accepts by empty stack.



Construct the grammar G=(V,T,S,P),

V={$(q_0Aq_0), (q_0zq_0), (q_0Aq_1), (q_0zq_1), \ldots$}

T=Σ

S=(q_0zq_2)

P=

From transition 1 $(q_0Aq_1) \rightarrow b$

From transition 2 $(q_1zq_2) \rightarrow \lambda$

From transition 3 $(q_0Aq_3) \rightarrow a$

From transition 4

$$
\begin{aligned}
(q_0zq_0) \rightarrow\ & a(q_0Aq_0)(q_0zq_0)| \\
& a(q_0Aq_1)(q_1zq_0)| \\
& a(q_0Aq_2)(q_2zq_0)| \\
& a(q_0Aq_3)(q_3zq_0) \\
(q_0zq_1) \rightarrow\ & a(q_0Aq_0)(q_0zq_1)| \\
& a(q_0Aq_1)(q_1zq_1)| \\
& a(q_0Aq_2)(q_2zq_1)| \\
& a(q_0Aq_3)(q_3zq_1) \\
(q_0zq_2) \rightarrow\ & a(q_0Aq_0)(q_0zq_2)| \\
& a(q_0Aq_1)(q_1zq_2)| \\
& a(q_0Aq_2)(q_2zq_2)| \\
& a(q_0Aq_3)(q_3zq_2) \\
(q_0zq_3) \rightarrow\ & a(q_0Aq_0)(q_0zq_3)| \\
& a(q_0Aq_1)(q_1zq_3)| \\
& a(q_0Aq_2)(q_2zq_3)| \\
& a(q_0Aq_3)(q_3zq_3)
\end{aligned}
$$

From transition 5

$$
\begin{aligned}
(q_3zq_0) \rightarrow\ & (q_0Aq_0)(q_0zq_0)| \\
& (q_0Aq_1)(q_1zq_0)| \\
& (q_0Aq_2)(q_2zq_0)| \\
& (q_0Aq_3)(q_3zq_0) \\
(q_3zq_1) \rightarrow\ & (q_0Aq_0)(q_0zq_1)| \\
& (q_0Aq_1)(q_1zq_1)| \\
& (q_0Aq_2)(q_2zq_1)| \\
& (q_0Aq_3)(q_3zq_1) \\
(q_3zq_2) \rightarrow\ & (q_0Aq_0)(q_0zq_2)| \\
& (q_0Aq_1)(q_1zq_2)| \\
& (q_0Aq_2)(q_2zq_2)| \\
& (q_0Aq_3)(q_3zq_2) \\
(q_3zq_3) \rightarrow\ & (q_0Aq_0)(q_0zq_3)| \\
& (q_0Aq_1)(q_1zq_3)| \\
& (q_0Aq_2)(q_2zq_3)| \\
& (q_0Aq_3)(q_3zq_3)
\end{aligned}
$$

Recognizing aaab in M:

$$
\begin{aligned}
(q_0, aaab, z) \ & \vdash (q_0, aab, Az) \\
& \vdash (q_3, ab, z) \\
& \vdash (q_0, ab, Az) \\
& \vdash (q_3, b, z) \\
& \vdash (q_0, b, Az) \\
& \vdash (q_1, \lambda, z) \\
& \vdash (q_2, \lambda, \lambda)
\end{aligned}
$$

Derivation of string aaab in G:

$$
\begin{aligned}
(q_0zq_2) \ & \Rightarrow a(q_0Aq_3)(q_3zq_2) \\
& \Rightarrow aa(q_3zq_2) \\
& \Rightarrow aa(q_0Aq_3)(q_3zq_2) \\
& \Rightarrow aaa(q_3zq_2) \\
& \Rightarrow aaa(q_0Aq_1)(q_1zq_2) \\
& \Rightarrow aaab(q_1zq_2) \\
& \Rightarrow aaab
\end{aligned}
$$

Chapter 7.3

Definition: A PDA M=(Q,Σ,Γ,δ,q_0,z,F) is *deterministic* if for every $q \in$Q, $a \in \Sigma \cup \{\lambda\}$, $b \in \Gamma$

1. $\delta(q, a, b)$ contains at most 1 element
2. if $\delta(q, \lambda, b) \neq \emptyset$ then $\delta(q, c, b)$=$\emptyset$ for all $c \in \Sigma$

Definition: L is DCFL iff $\exists$ DPDA M s.t. L=L(M).

Examples:

1. Previous pda for $\{a^n b^n | n \geq 0\}$ is deterministic.
2. Previous pda for $\{a^n b^m c^{n+m} | n, m > 0\}$ is deterministic.
3. Previous pda for $\{ww^R | w \in \Sigma^+\}$,$\Sigma = \{a, b\}$ is nondeterministic.

Note: There are CFL's that are not deterministic.

L=$\{a^n b^n | n \geq 1\} \cup \{a^n b^{2n} | n \geq 1\}$ is a CFL and not a DCFL.

- **Proof:** $L = \{a^n b^n : n \geq 1\} \cup \{a^n b^{2n} : n \geq 1\}$

  It is easy to construct a NPDA for $\{a^n b^n : n \geq 1\}$ and a NPDA for $\{a^n b^{2n} : n \geq 1\}$. These two can be joined together by a new start state and λ-transitions to create a NPDA for L. Thus, L is CFL.

  Now show L is not a DCFL. Assume that there is a deterministic PDA M such that $L = L(M)$. We will construct a PDA that recognizes a language that is not a CFL and derive a contradiction.

  Construct a PDA M' as follows:

  1. Create two copies of M: M_1 and M_2. The same state in M_1 and M_2 are called cousins.
  2. Remove accept status from accept states in M_1, remove initial status from initial state in M_2. In our new PDA, we will start in M_1 and accept in M_2.
  3. Outgoing arcs from old accept states in M_1, change to end up in the cousin of its destination in M_2. This joins M_1 and M_2 into one PDA. There must be an outgoing arc since you must recognize both $a^n b^n$ and $a^n b^{2n}$. After reading n b's, must accept if no more b's and continue if there are more b's.
  4. Modify all transitions that read a b and have their destinations in M_2 to read a c.

  This is the construction of our new PDA.

  When we read $a^n b^n$ and end up in an old accept state in M_1, then we will transfer to M_2 and read the rest of $a^n b^{2n}$. Only the b's in M_2 have been replaced by c's, so the new machine accepts $a^n b^n c^n$.

  The language accepted by our new PDA is $a^n b^n c^n$. But this is not a CFL. Contradiction! Thus there is no deterministic PDA M such that $L(M) = L$. Q.E.D.