Equivalence of C.F. Grammars and PDAs

We now note that C.F. grammars and PDAs are equivalent characterizations of C.F. languages. I.e. we note

Theorem 1 If \( L = L(G) \) is defined by a c.f. grammar \( G \), then \( L \) is recognized by some PDA \( M \).

Theorem 2 If a language \( L \) is recognized by a PDA \( M \), then \( L = L(G) \) is defined by some c.f. grammar \( G \).

In other words, a c.f. grammar can be simulated by a PDA and vice versa. We only prove Theorem 1 here and that too only via an illustrative example.
Theorem 1, Proof by Example

Suppose the grammar is

\[ S \rightarrow aTbb \mid b \]
\[ T \rightarrow Tab \mid \epsilon. \]

The PDA simulates derivation of a string in \( \{a, b\}^* \) as follows:

- Push \( S \) onto the stack.

\[ \rightarrow \] If the top-stack symbol is a variable, replace it with right hand side of some rule for that variable.

\[ \rightarrow \] If the top-stack symbol is a terminal, match/pop it with the input symbol.

Repeat. Accept if input is exhausted and the stack is empty.
Input: aabbb

\[
\begin{align*}
S &\rightarrow b \\
\text{aabbb} &\rightarrow \text{die} \\
\downarrow S \rightarrow aTbb & \\
\text{aabbb} &\rightarrow \text{die} \\
\downarrow T \rightarrow \lambda & \\
\text{aabbb} &\rightarrow \text{die eventually or continue forever} \\
\downarrow T \rightarrow Tab & \\
\text{aabbb} &\rightarrow \text{Accept!} \\
\downarrow T \rightarrow \lambda & \\
\text{aabbb} &
\end{align*}
\]
The figure shows many possible computation paths of the PDA on input $aabbb$.

- Since there is a choice of two rules to apply for variable $S$ (and also for $T$), the PDA is non-deterministic.

- The accepting computation of the PDA corresponds precisely to the leftmost derivation of the input string.

Exercise. Convince yourself of this conclusion and that this simulation works for every c.f. grammar.

We will henceforth take for granted the equivalence of CFGs and PDAs.
Pumping Lemma for Context Free Languages

Similar to the pumping lemma for regular languages, we have a pumping lemma for c.f.l. languages, used to show that certain languages are not context free.

Lemma (Pumping Lemma for CFLs)

Let $L$ be a c.f.l. There exists an integer $p$ (pumping length) such that the following holds.

Every $s \in L$, $|s| \geq p$ can be written as $s = uvxyz$ such that

1. $uv^ixyz \in L \ \forall i \geq 0$
2. $|vy| \geq 1$
3. $|vxy| \leq p$. 
Proof. Let $G$ be a c.f. grammar for $L$.

Let $n = \# \text{variables in } G$.

$b = \text{maximum length of right hand side of a rule in } G$.

Let $p = b^{n+2}$.

Consider any string $s \in L$, $|s| \geq p$ and consider the parse tree for $s$ that has the minimum number of nodes.

The tree is "b-ary" and since $|s| \geq p = b^{n+2}$, has height at least $n+2$.

A: Start variable

a: Terminal/leaf at depth at least $n+2$. 

Diagram:

- A: Start variable
- B1, B2, B_{n+1}: Internal nodes
- a: Terminal/leaf
- s: String

$n+1$ variables
let a be a terminal symbol that occurs as a leaf of this tree and has depth at least \( n+2 \).

Consider the path upwards from a to the root of the tree (\( = \) start variable) \( A \).
Let \( B_1, B_2, \ldots, B_{n+1} \) be the successive variables (upwards) on this path as shown in the picture. Since the grammar has only \( n \) variables, by Pigeon-Hole Principle, for some \( 1 \leq i < j \leq n+1 \), \( B_i = B_j = R \) is the same variable. We can redraw as:
Let $s = uvxyz$ be the splitting as shown. Clearly, we have derivations

$$A \Rightarrow uRz$$

$$R \Rightarrow vRyz \mid x$$

We now prove the three conclusions of the lemma.

1. $i = 0$  
   \[ A \Rightarrow uRz \Rightarrow uxz \]
   Hence $uxz \in L$.

2. $i > 1$  
   \[ A \Rightarrow uRz \Rightarrow uvRyz \]
   \[ \Rightarrow uRyz \]
   \[ \Rightarrow uvyz \]
   \[ \Rightarrow uvy'yz \]
   Hence $uv^i x^i y^i z \in L$.

3. $|vxy| \leq p$. This is because $vxy$ is
generated by the parse tree rooted at variable \( R \) (that appears "above") and having height at most \( n+1 \). Since the tree is "b-ary", we have

\[ |vxy| \leq b^{n+1} \leq b^{n+2} = p. \]

(3) \( |vy| \geq 1 \). Suppose on the contrary that \( vy \) equals the empty string \( \epsilon \). Then \( s = uxz \) is also generated by the parse tree.

This parse tree has fewer nodes than the one before, contradicting the minimality of the tree before!
Now we'll apply the pumping lemma to show that:

Claim \( L = \{ a^n b^n c^n \mid n \geq 0 \} \) is not context free.

Proof Suppose on the contrary that \( L \) is context free. Let \( p \) be the pumping length and let \( s = a^{5p} b^{5p} c^{5p} \).

Since \( s \in L \) and \( |s| \geq p \), by the pumping lemma, \( s \) can be written as 

\[ s = uvxyz, \quad uv^ixyz \in L \quad \forall i \geq 0 \]

\[ |vy| \geq 1 \]

\[ |vxy| \leq p. \]

Since \( |vxy| \leq p \), the sub-string \( vxy \) can be either contained in the prefix \( a^{5p} \)
or contained in the suffix $b^{5p}c^{5p}$.

**Case 1** \(vxy\) is contained in \(a^{5p}b^{5p}\).

Since \(|vy| \geq 1\), \(vy\) contains \('a'\) or \('b'\) or both.

Thus the string \(uv^ixy^iz\) has more \('a's\) or \('b's\) than \('c's\), for \(i \geq 2\).

\[
\begin{array}{c}
\text{aaa} \ldots \text{a} \text{bbb} \ldots \text{b} \text{ccc} \ldots \text{c} \\
\hline
\text{u} \quad \text{vxy} \\
\text{z}
\end{array}
\]

Hence for \(i \geq 2\), \(uv^ixy^iz \notin L\), contradicting the conclusion of the pumping lemma.

**Case 2** \(vxy\) is contained in \(b^{5p}c^{5p}\).

\[
\begin{array}{c}
\text{aaa} \ldots \text{a} \text{bbb} \ldots \text{b} \text{ccc} \ldots \text{c} \\
\hline
\text{u} \quad \text{vxy} \\
\text{z}
\end{array}
\]

The proof is similar. Now \(uv^2xy^2z\) contains more \('b's\) or \('c's\) than \('a's\).
The contradiction in both cases shows that $L$ is not context free.

Note \[ L = \{ a^n b^n c^n \mid n \geq 0 \} \]
is
\[ L = L_1 \cap L_2 \text{ where} \]
\[ L_1 = \{ a^n b^n c^m \mid n, m \geq 0 \} \]
\[ L_2 = \{ a^m b^n c^n \mid n, m \geq 0 \}. \]

It is easily seen that $L_1, L_2$ are context free. Thus intersection of context free languages need not be context free.

As another application, we show:

Claim \[ L = \{ w \# w \mid w \in \{0,1\}^* \} \quad \Sigma = \{0,1,\#\} \]
is not context free.

Proof Suppose $L$ is context free and
be the pumping length. Consider the string $S = 0^{5p} 1^{5p} # 0^{5p} 1^{5p}$.

Since $s \in L$, $|s| \geq p$, by Pumping Lemma,

$S = uvxyz \text{ s.t. } - uv^ixyz \in L \forall i \geq 0$

$- |vy| \geq 1$

$- |vxy| \leq p$

Since $|vxy| \leq p$ there are 4 cases.

Case 1

$00 \cdots 0 1 1 \cdots 1 \# 00 \cdots 0 1 1 \cdots 1$

$vxy$

I.e. $vxy$ occurs before the $#$ symbol.

Case 2

$00 \cdots 0 1 1 \cdots 1 \# 00 \cdots 0 1 1 \cdots 1$

$vxy$

I.e. $vxy$ occurs after the $#$ symbol.
Case 3

\[ 00\ldots011\ldots1 \# 00\ldots011\ldots1 \]
\[
\begin{array}{c}
v \\
x \\
y
\end{array}
\]

\( \text{V only has } '1' \text{'s, } X \text{ contains } \# \text{ symbol,} \)
\( \text{Y only has } '0' \text{'s.} \)

Case 4

\( \# \text{ is contained in } V \text{ (or } Y) \).

\[ 00\ldots011\ldots1 \# 00\ldots011\ldots1 \]
\[
\begin{array}{c}
v \\
x \\
y
\end{array}
\]

In all the four cases, \( uV^2xy^2z \in L \)
contradicting the conclusion of the Pumping Lemma. The reason is that \( uV^2xy^2z \)

In case 1 has longer length before \( \# \)
than after.

In case 2 has longer length after \( \# \)
than before.
In case 3 has the form

\[ 0 \quad \begin{array}{cccc}
5p & 5p+a & 5p+b & 5p \\
1 & \# & 0 & 1
\end{array} \]

with \((a+b) \geq 1\). Clearly this is not of the form \(W \neq \#W\).

In case 4 has two \# symbols.

The contradiction shows that \(L\) is not context free.

Exercise \(L = \{w\#w \mid w \in \{0,1\}^*\}\) as above, observe that \(L = L_1 \cup L_2 \cup L_3\) where

- \(L_1 = \{0,1\}^* \# \{0,1\}^* \# \{0,1\}^*\) i.e., two or more \# \(L_2 = \{0,1\}^*\) i.e., no \# symbol
- \(L_3 = \{w \# w' \mid w, w' \in \{0,1\}^*, w \neq w'\}\)

Show that \(L_3\) is context free (use PDA!) and hence \(\overline{L}\) is context free (but \(L\) is not as shown). So complement CFL need not be CFL.