**Introduction and Motivation**

In this lecture we present the *Pumping Lemma* for *Context Free Languages*. This lemma enables us to prove that some languages are not CFL and hence are not recognizable by any PDA.

**The Pumping Lemma**

Let $A$ be a context free language. There exists a number $p$ such that for every $w \in A$, if $|w| \geq p$ then $w$ may be divided into five parts, $w = uvxyz$ satisfying:

1. for each $i \geq 0$, it holds that $uv^ixy^iz \in A$.
2. $|vy| > 0$.
3. $|vxy| \leq p$.

*Note*: Without req. 2 the Theorem is trivial.

**Proof Idea**

If $w$ is “long enough” (to be precisely defined later) it has a large parse tree which has a “long enough” path $\alpha$ from its root to one of its leaves.

Under these conditions, some variable on $\alpha$ should appear *twice*. This enables pumping of $w$ as demonstrated in the next slide:
**Proof Idea**

Pumping up  
\[ R \]

Pumping down  
\[ R \]

Let \( T \) be a binary tree.

The 0-th level of \( T \) has \( 1 = 2^0 \) nodes.

The 1-th level of \( T \) has at most \( 2 = 2^1 \) nodes.

... 

The \( i \)-th level of \( T \) has at most \( 2^i \) nodes.

If \( T' \) is a \( b \)-ari tree then its \( i \)-th level has at most \( 2^i \) nodes.

**Reminder**

Let \( T \) be a binary tree.

The 0-th level of \( T \) has \( 1 = 2^0 \) nodes.

The 1-th level of \( T \) has at most \( 2 = 2^1 \) nodes.

... 

The \( i \)-th level of \( T \) has at most \( 2^i \) nodes.

If \( T' \) is a \( b \)-ari tree then its \( i \)-th level has at most \( b^i \) nodes.

**The Proof**

Let \( G \) be a grammar for the language \( L \). Let \( b \) be the maximum number of symbols (variables and constants) in the right hand side of a rule of \( G \). (Assume \( b \geq 2 \)). In any parse tree, \( T \), for generating \( w \) from \( G \), a node of \( T \) may have no more than \( b \) children.

If the height of \( T \) is \( h \) then \( |w| \leq b^h \).

**The Proof (cont.)**

If the height of \( T \) is \( h \) then \( |w| \leq b^h \). Conversely, if \( |w| > b^h \) then the height of \( T \) is at least \( h + 1 \).

Assume that \( G \) has \( |V| \) variables. Then we set \( p = b^{|V|+1} \).

**Conclusion:** For any \( w \in L \), if \( |w| \geq p \), then the height of any parse tree of \( w \) is at least \( |V|+1 \).
The Proof (cont.)

To see how pumping works let \( \tau \) be the parse tree of \( w \) with a **minimal number of nodes**. The height of the tree \( \tau \), is at least \(|V|+1\), so it has a path, \( \alpha \) with at least \(|V|+2\) nodes, from its root until some leaf. The path \( \alpha \) has at least \(|V|+1\) variables and a single terminal.

The Proof (cont.)

Since \( G \) has \(|V|\) variables and \( \alpha \) has at least \(|V|+1\) variables, there exists a variable, \( R \), that repeats itself among the \(|V|+1\) lowest variables of \( \alpha \), as depicted in the following picture:

The Proof (cont.)

Each occurrence of \( R \) has a sub-tree rooted at it:
Let \( vxy \) be the word generated by the upper occurrence of \( R \) and let \( x \) be the word generated by the lower occurrence of \( R \).

The Proof (cont.)

Since both sub-trees are generated by the same variable, each of these sub-trees can be replaced by another. This tree is obtained from \( \tau \) by substituting the upper sub-tree at the lower occurrence of \( R \).
The Proof (cont.)

The word generated is $w^2xy^2z$, and since it is generated by a parse tree of $G$ we get $w^2xy^2z \in A$. Additional substitutions of the upper sub-tree at the lower occurrence of $R$, yield the conclusion $w^i xy^i z \in A$ for each $i > 0$.

The Proof (cont.)

Substitution of the lower sub-tree at the upper occurrence of $R$ yields this pars tree whose generated word is $uv^0 xy^0 z = uxz$. Since once again this is a legitimate parse tree we get $uxz \in A$.

The Proof (cont.)

To see that $|vy| > 0$, assume that this is the situation. In this case, this tree is a parse tree for $w$ with less nodes then $\tau$, in contradiction with the choice of $\tau$ as a parse tree for $w$ with a minimal number of nodes.

The Proof (cont.)

In order to show that $|vxy| \leq p$ recall that we chose $R$ so that both its occurrences fall within the bottom $|V|+1$ nodes of the path $\alpha$, where $\alpha$ is the longest path of the tree so the height of the red sub-tree is at most $|V|+1$ and the number of its leaves is at most $b^{|V|+1} = p$. 
Now we use the pumping lemma for CFL to show that the language $L = \{a^nb^n c^n | n \geq 0\}$ is not CFL.

Assume towards a contradiction that $L$ is CFL and let $p$ be the pumping constant. Consider $w = a^pb^pc^p$. Obviously $w \in L$.

By the pumping lemma, there exist a partition $w = uvxyz$ where $|vy| > 0$, $|vx| \leq p$ and for each $i$, it holds that $uv^i xy^i z \in L$.

Case 1: Both $v$ and $y$ contain one symbol each:
Together they may hold 2 symbols, so in $uv^2xy^2z$, the third symbol appears less than the other two.

Case 2: Either $v$ or $y$ contain two symbols:
In this case, the word $uv^2xy^2z$ has more than three blocks of identical letters: In other words: $uv^2xy^2z \notin a^+b^+c^+$, Q.E.D.

quod erat demonstrandum (Wiktionary) which was to be proved; which was to be demonstrated. Abbreviation: QED

Some weeks ago we started our quest to find out “What can be computed and what cannot?”
So far we identified two classes: RL-s and CFL-s and found some examples which do not belong in neither class.
Discussion

This is what we got so far:

RL-s Ex: \( \{ a^n \mid n \geq 0 \} \)

CFL-s Ex: \( \{ a^n b^n \mid n \geq 0 \} \)

Non CFL-s Ex: \( \{ a^n b^n c^n \mid n \geq 0 \} \)

Moreover: Our most complex example, namely, the language \( L = \{ a^n b^n c^n \mid n \geq 0 \} \) is easily recognizable by your everyday computer, so we did not get so far yet. Our next attempt to grasp the essence of “What’s Computable?” are Turing Machines. Some surprises are awaiting…

Recap

In this lecture we introduced and proved the Pumping Lemma for CFL-s.

Using this lemma we managed to prove that the fairly simple language \( L = \{ a^n b^n c^n \mid n \geq 0 \} \), is not CFL.

The next step is to define Turing Machines.