# CS 302.1 - Automata Theory

Lecture 08


Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)

Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

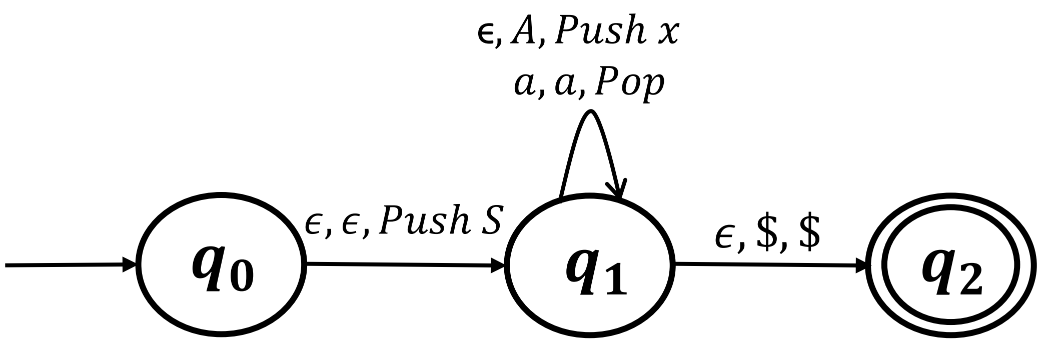

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

# Quick Recap

Pushdown Automata and CFLs are equivalent

**CFLs ⇒ Pushdown Automata**




### Deterministic Pushdown Automata (DPDA)

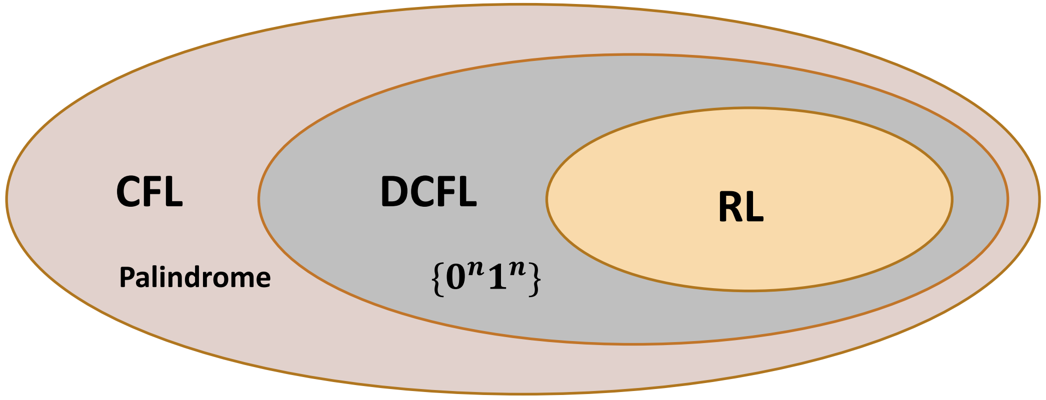
For every $q \in Q, a \in \Sigma$ and $x \in \Gamma$, **exactly one** of the following values is non-empty

$$\boldsymbol{\delta(q, a, x), \delta(q, a, \epsilon), \delta(q, \epsilon, x)} \textbf{ and } \boldsymbol{\delta(q, \epsilon, \epsilon)}$$

- PDAs are more powerful that DPDAs
- Language recognized by DPDA : DCFL

$\boldsymbol{L = \{w | w}$ **is a Palindrome**$\boldsymbol{\}}$ $\boldsymbol{L \subseteq CFL}$ but not $\boldsymbol{DCFL}$

- $\boldsymbol{DCFL \subseteq CFL}$




$$\boldsymbol{(RL \equiv Regular\ Grammar \equiv Regular\ Expressions \equiv NFA \equiv DFA) \subseteq (DCFL \equiv DPDA) \subseteq (CFL \equiv CFG \equiv PDA)}$$

# Pumping Lemma for CFLs

Recall that so far, we have seen the following:

- $L$ is a context-free language.
- $L$ is generated by a Context Free Grammar (CFG) from which any $w \in L$ can be **derived**.
- The derivation of any CFG can be represented by **parse trees**.
- Any CFG can be expressed in Chomsky Normal Form (CNF): the number of steps required to derive any $w \in L$: $2|w| - 1$
- There exists a Pushdown Automata $P$ such that $\mathcal{L}(P) = L$.

# Pumping Lemma for CFLs

Recall that so far, we have seen the following:

- $L$ is a context-free language.
- $L$ is generated by a Context Free Grammar (CFG) from which any $w \in L$ can be **derived**.
- The derivation of any CFG can be represented by **parse trees**.
- Any CFG can be expressed in Chomsky Normal Form (CNF): the number of steps required to derive any $w \in L$: $2|w| - 1$
- There exists a Pushdown Automata $P$ such that $\mathcal{L}(P) = L$.

- Not all languages are context free.
- Just like in the case of Regular languages, the pumping lemma helps us identify non-CFLs.
- **All CFLs satisfy the conditions of the pumping lemma**: If any language $L$ fails to do so, it is not Context-Free.
- The principle of the Pumping Lemma for CFLs is similar to that of Regular Languages

# Pumping Lemma for CFLs

Recall that so far, we have seen the following:

- $L$ is a context-free language.
- $L$ is generated by a Context Free Grammar (CFG) from which any $w \in L$ can be **derived**.
- The derivation of any CFG can be represented by **parse trees**.
- Any CFG can be expressed in Chomsky Normal Form (CNF): the number of steps required to derive any $w \in L$: $2|w| - 1$
- There exists a Pushdown Automata $P$ such that $\mathcal{L}(P) = L$.

- Not all languages are context free.
- Just like in the case of Regular languages, the pumping lemma helps us identify non-CFLs.
- **All CFLs satisfy the conditions of the pumping lemma**: If any language $L$ fails to do so, it is not Context-Free.
- The principle of the Pumping Lemma for CFLs is similar to that of Regular Languages
- In **order to recognize very long strings** in a given CFL $L$, the **model of computation (CFGs/parse-trees) must repeat some steps of the computation**

# Pumping Lemma for CFLs

Recall that so far, we have seen the following:

- $L$ is a context-free language.
- $L$ is generated by a Context Free Grammar (CFG) from which any $w \in L$ can be **derived**.
- The derivation of any CFG can be represented by **parse trees**.
- Any CFG can be expressed in Chomsky Normal Form (CNF): the number of steps required to derive any $w \in L$: $2|w| - 1$
- There exists a Pushdown Automata $P$ such that $\mathcal{L}(P) = L$.

- Not all languages are context free.
- Just like in the case of Regular languages, the pumping lemma helps us identify non-CFLs.
- **All CFLs satisfy the conditions of the pumping lemma**: If any language $L$ fails to do so, it is not Context-Free.
- The principle of the Pumping Lemma for CFLs is similar to that of Regular Languages
- In **order to recognize very long strings** in a given CFL $L$, the **model of computation (CFGs/parse-trees) must repeat some steps of the computation**
- These steps can be **repeated any number of times (pumped) to produce longer and longer strings all of which belong to $\boldsymbol{L}$**.
- Conversely if **this does not hold, $\boldsymbol{L}$ is not CFL.**

# Pumping Lemma for CFLs

**Example:**

$$\begin{aligned} \boldsymbol{A} &\rightarrow \boldsymbol{BC}|\mathbf{0} \\ \boldsymbol{B} &\rightarrow \boldsymbol{BA}|\mathbf{1}|\boldsymbol{CC} \\ \boldsymbol{C} &\rightarrow \boldsymbol{AB}|\mathbf{0} \end{aligned}$$

No of variables $|V| = 3$.
Consider a derivation of $w = 11100001$

# Pumping Lemma for CFLs

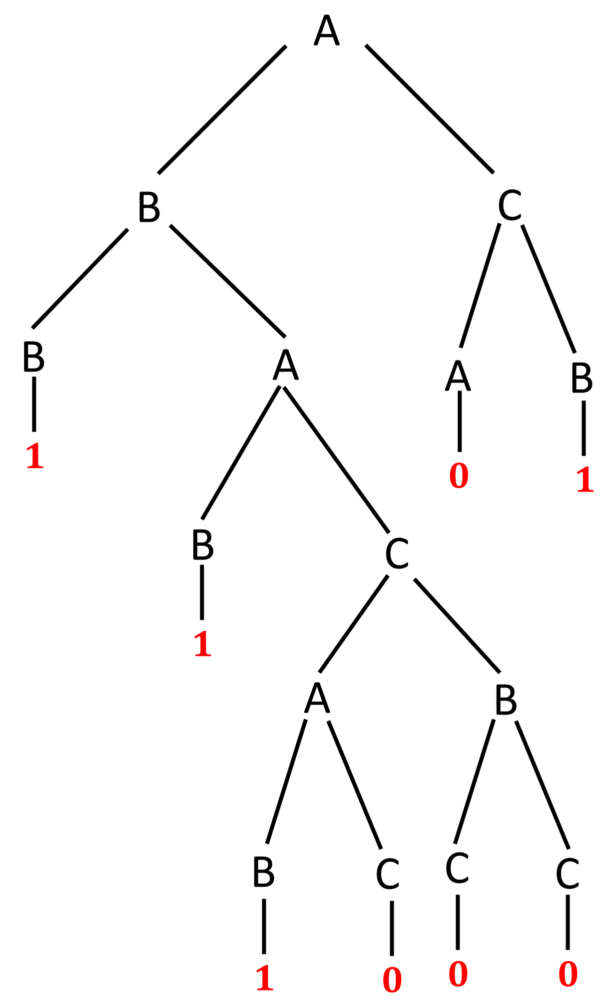
**Example:**

$$A \to BC|0$$
$$B \to BA|1|CC$$
$$C \to AB|0$$

No of variables $|V| = 3$.

Consider a derivation of $w = 11100001$

Consider the longest path in the parse tree.

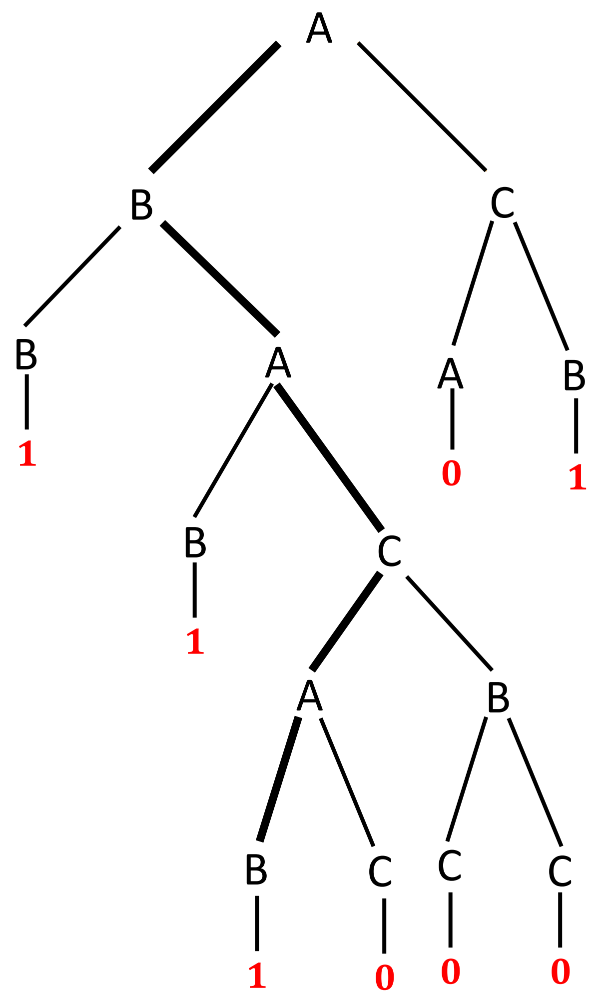


# Pumping Lemma for CFLs

**Example:**

$$A \rightarrow BC|0$$
$$B \rightarrow BA|1|CC$$
$$C \rightarrow AB|0$$

- No of variables $|V| = 3$
- Consider a derivation of $w = 11100001$
- Consider the longest path in the parse tree.
- Longest path length = 5, which is larger than $|V|$.
- There exists at least one variable that is repeated.
- For example: $A$ – mark it.

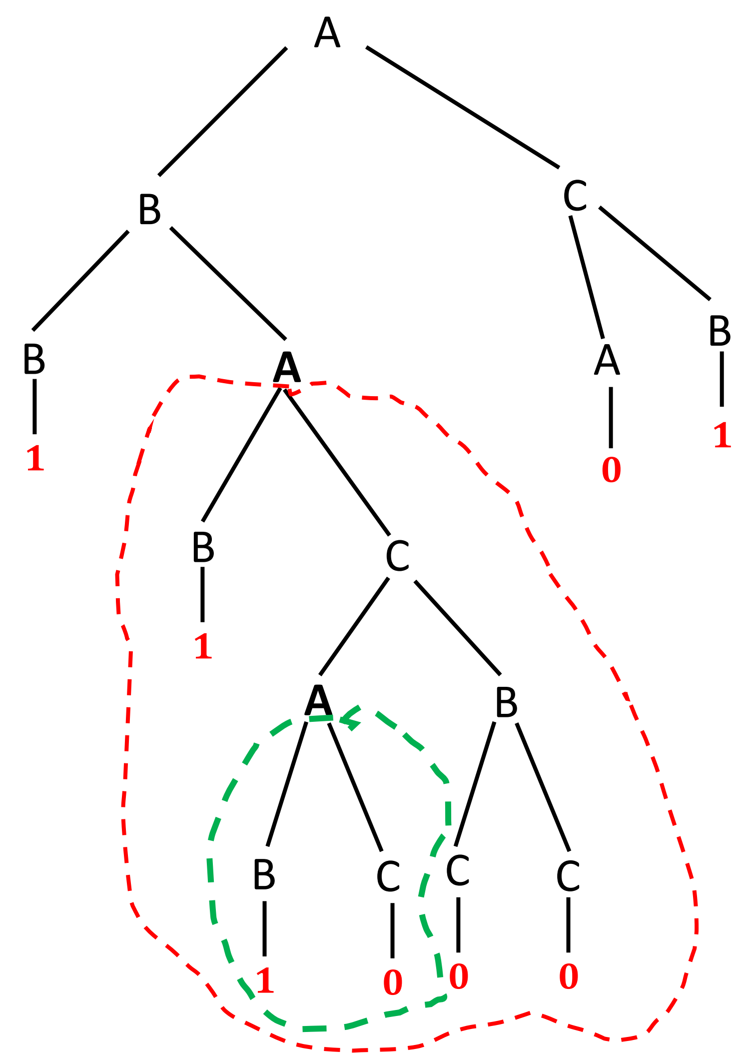


# Pumping Lemma for CFLs

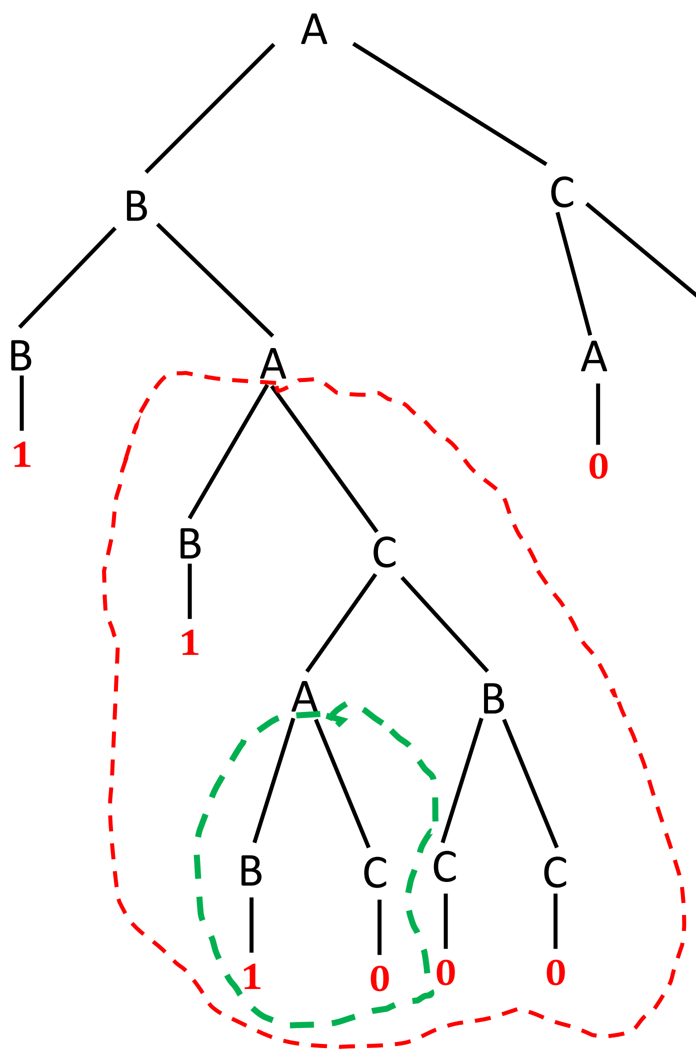
**Example:**

$$A \rightarrow BC|0$$
$$B \rightarrow BA|1|CC$$
$$C \rightarrow AB|0$$

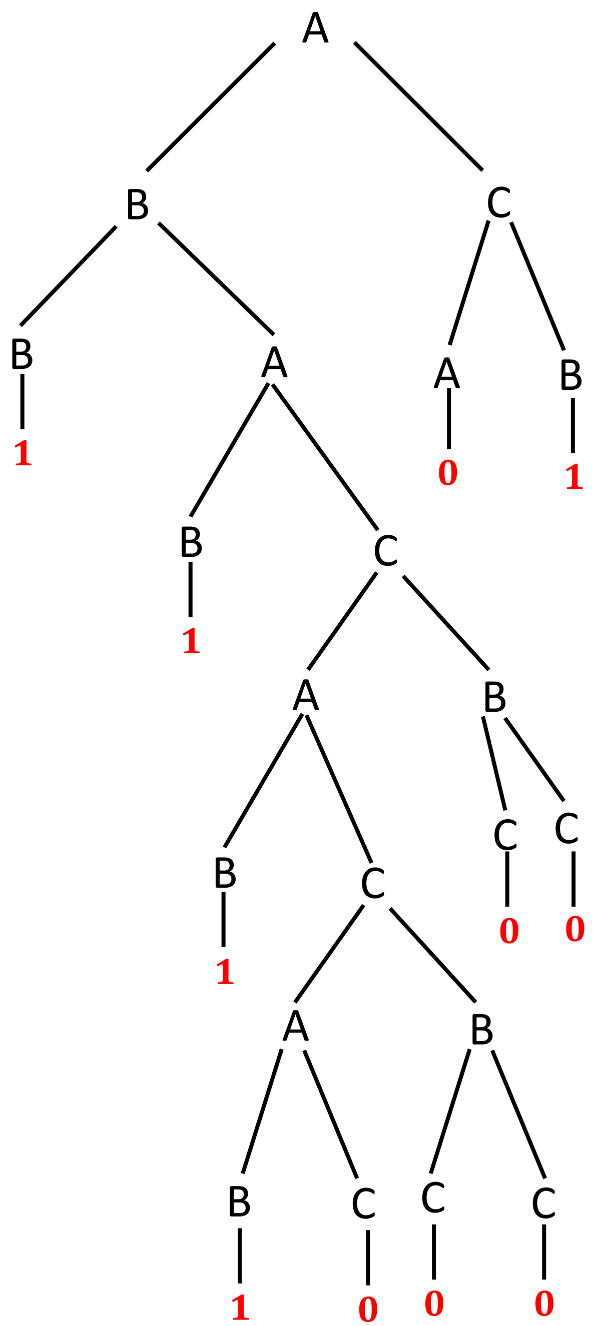
- No of variables $|V| = 3$
- Consider a derivation of $w = 11100001$
- Consider the longest path in the parse tree.
- Longest path length = 5, which is larger than $|V|$.
- There exists at least one variable that is repeated.
- For example: $A$ – mark it.
- Replace the bottom most appearance of the variable $A$ with the subtree in the middle rooted at $A$ to obtain a new tree.



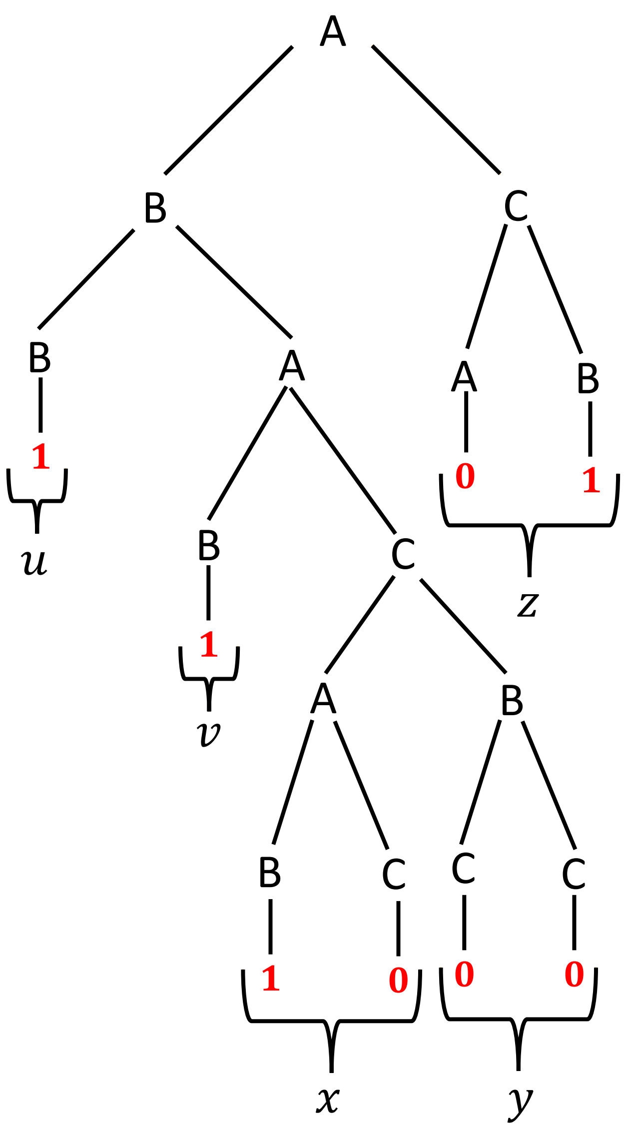
# Pumping Lemma for CFLs




- No of variables $|V| = 3$
- Consider the longest path in the parse tree.
- Longest path length = 5, which is larger than $|V|$.
- There exists at least one variable that is repeated.
- For example: $A$ – mark it.
- Replace the bottom most appearance of the variable $A$ with the subtree in the middle rooted at $A$ to obtain a new tree.



# Pumping Lemma for CFLs

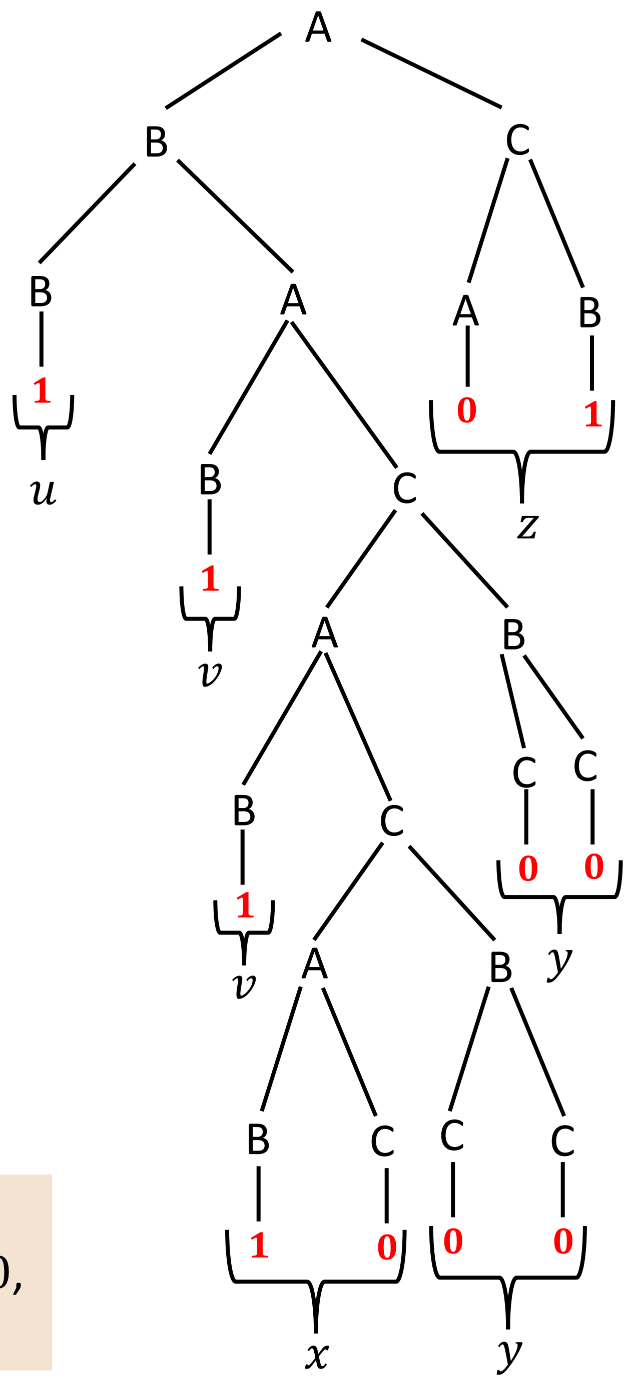



For the tree in the left, the input string $w$ can be split into five parts: $\boldsymbol{w = uvxyz}$

| | **u** | **v** | **x** | **y** | **z** |
|---|---|---|---|---|---|
| L Tree | 1 | 1 | 10 | 00 | 01 |

| | **u** | **vv** | **x** | **yy** | **z** |
|---|---|---|---|---|---|
| R Tree | 1 | 11 | 10 | 0000 | 01 |

By the substitution mentioned in the previous slide, we can keep pumping in $v$ and $y$ to get new strings of the form $\boldsymbol{w = uv^ixy^iz}$ $(i \geq 0)$, and any such $w \in L$ as it is a valid derivation.




**Other conditions:**
$|vy| \geq 1$, $v, y$ cannot be both $\epsilon$
$|vxy| \leq p$

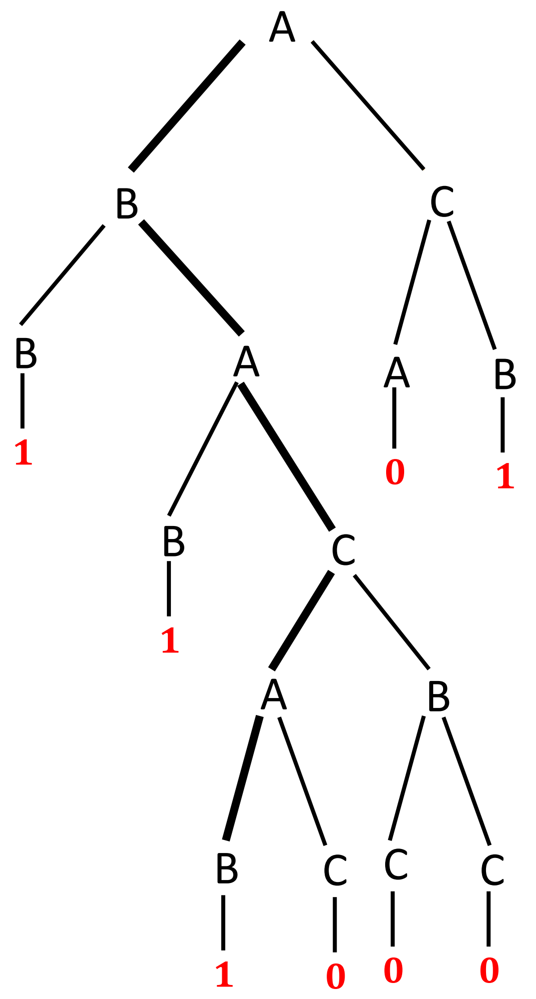
In fact **if $\boldsymbol{L}$ is a CFL**, $\exists p$ such that $\forall w \in L$ of length |w| $\geq p$, we can split $w = uvxyz$, such that $\forall i \geq 0$,
$\boldsymbol{w = uv^ixy^iz \in L}$

# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.

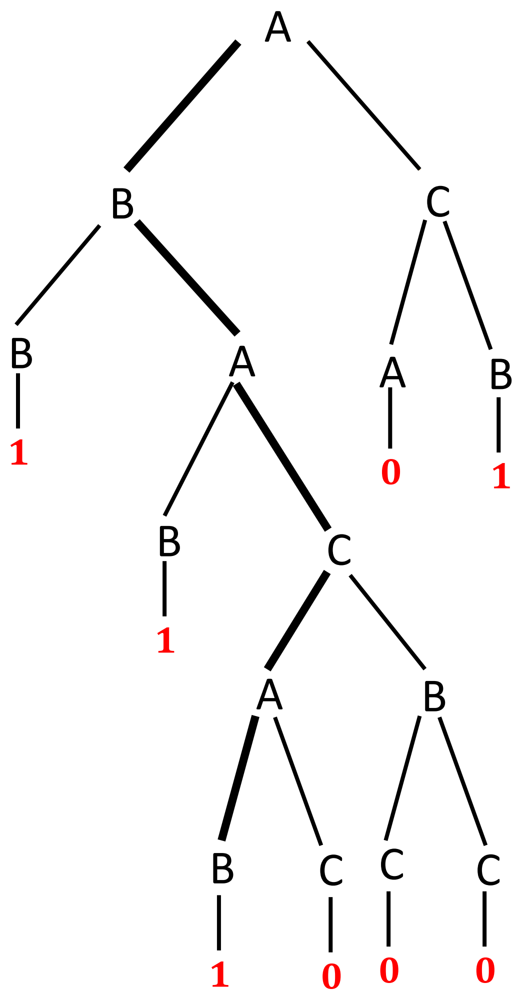


# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.



# Pumping Lemma for CFLs

**Properties of parse trees:**

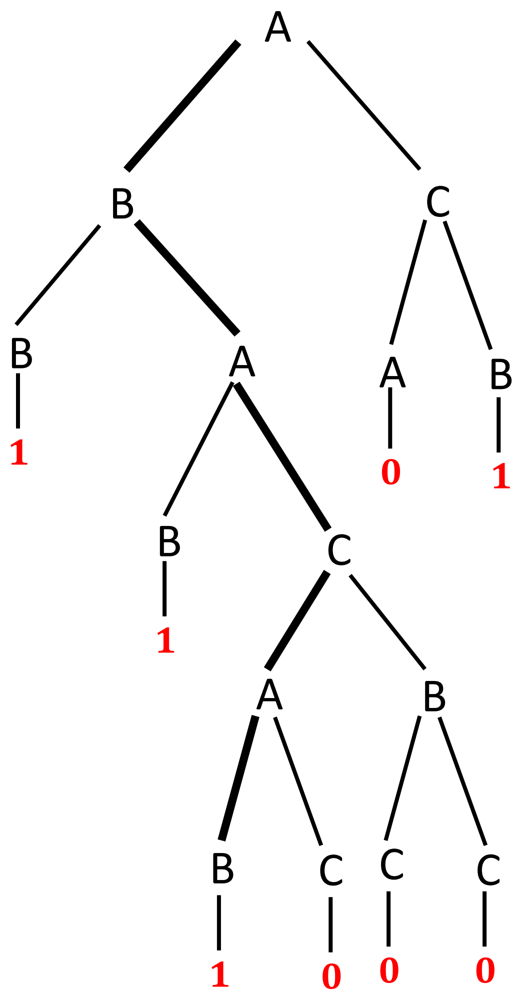
Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.

**Example:**

$$\begin{aligned} \boldsymbol{A} &\to \boldsymbol{BCDE}|\mathbf{0} \\ \boldsymbol{B} &\to \boldsymbol{BA}|\mathbf{1}|\boldsymbol{CC} \\ \boldsymbol{C} &\to \boldsymbol{AB}|\mathbf{0} \\ \boldsymbol{D} &\to \mathbf{01}|\boldsymbol{EA} \end{aligned}$$

$d = ??$



# Pumping Lemma for CFLs

**Properties of parse trees:**

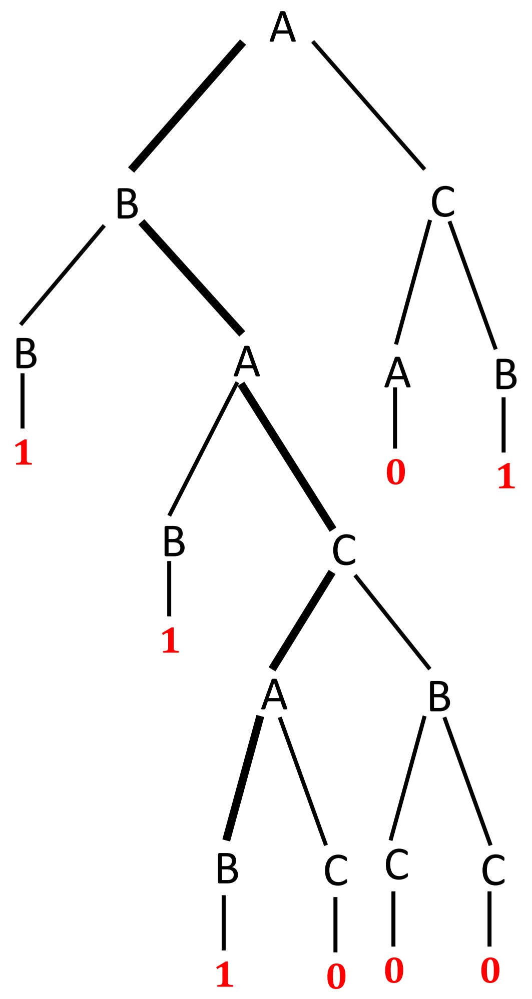
Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.

**Example:**

$$\begin{aligned} \boldsymbol{A} &\rightarrow \boldsymbol{BCDE}|\mathbf{0} \\ \boldsymbol{B} &\rightarrow \boldsymbol{BA}|\mathbf{1}|\boldsymbol{CC} \\ \boldsymbol{C} &\rightarrow \boldsymbol{AB}|\mathbf{0} \\ \boldsymbol{D} &\rightarrow \mathbf{01}|\boldsymbol{EA} \end{aligned}$$

$d = 4$

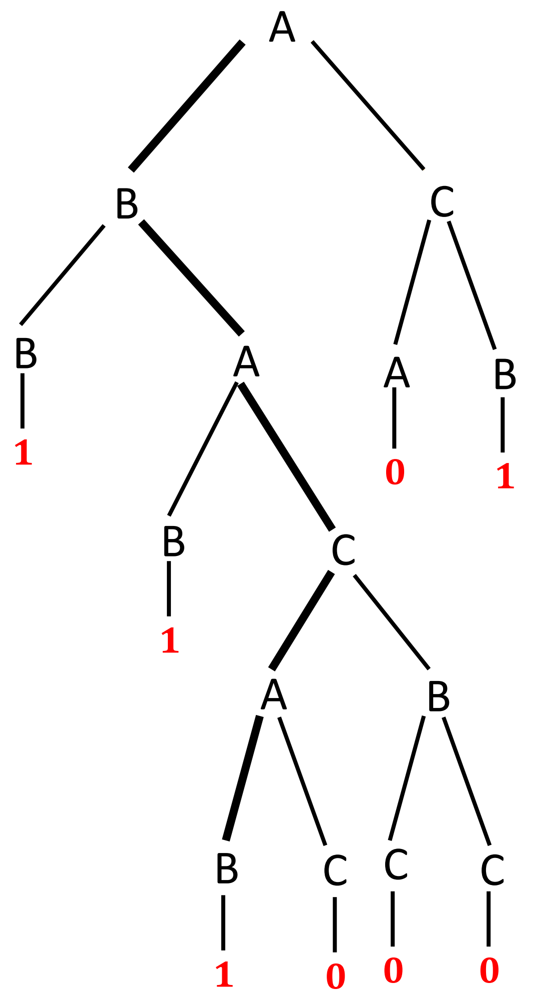


# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.
- This results in a $d$-ary parse tree.

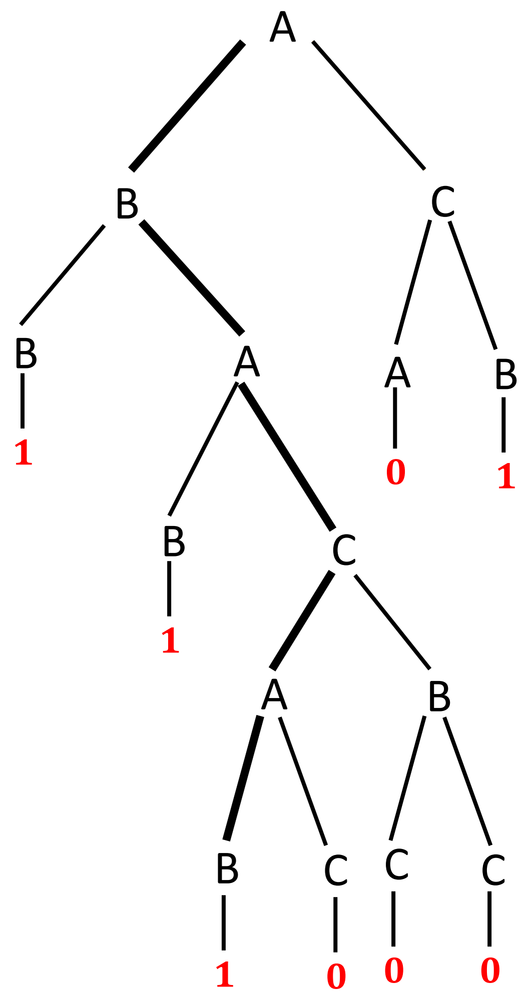


# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.
- This results in a $d$-ary parse tree.
- Any $T_w^G$ has at **level $\boldsymbol{l}$**, at most $\boldsymbol{d^l}$ **nodes**. Thus, any $T_w^G$ of height $h$ has at most $d^h$ terminals.

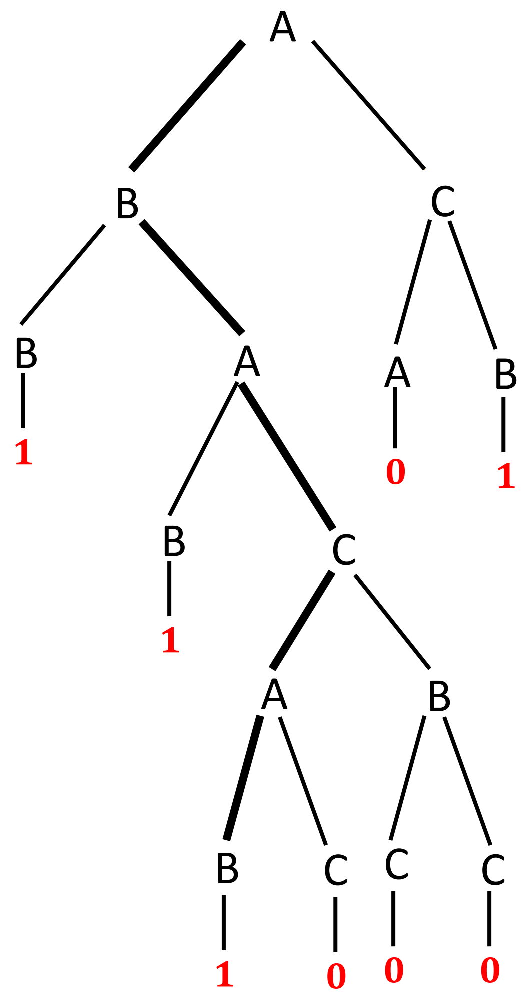


# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.
- This results in a $d$-ary parse tree.
- Any $T_w^G$ has at **level $\boldsymbol{l}$**, at most **$\boldsymbol{d^l}$ nodes**. Thus, any $T_w^G$ of height $h$ has at most $d^h$ terminals.




- Let |V| be the total number of variables in the Grammar $G$.

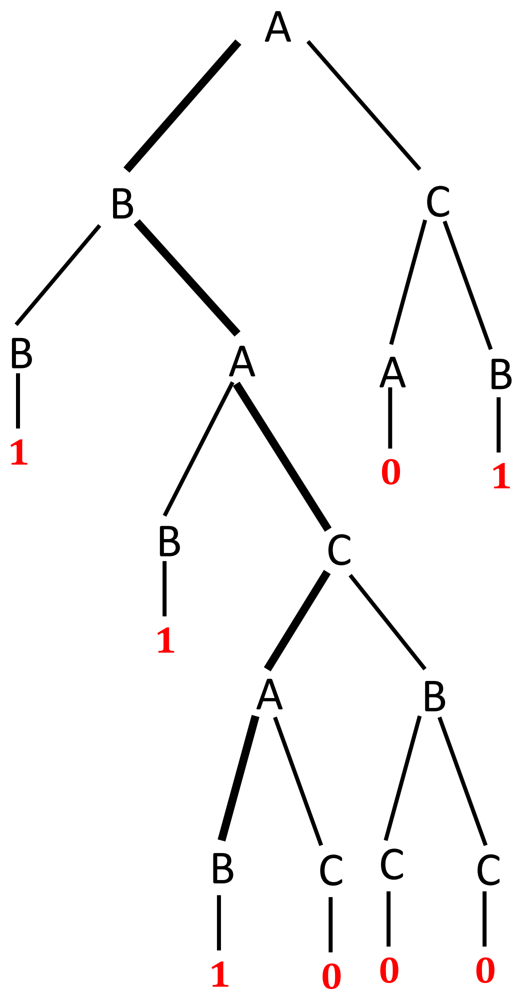
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.

# Pumping Lemma for CFLs

**Properties of parse trees:**

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$. Then:

- A path from the root to a leaf is sequence of variables and ends in a terminal or $\epsilon$.
- Let $d$ be the maximum number of variables in the RHS of any rule of $G$.
- For example: If $G$ is in CNF, $d = 2$.
- This results in a $d$-ary parse tree.
- Any $T_w^G$ has at **level $\boldsymbol{l}$**, at most **$\boldsymbol{d^l}$ nodes**. Thus, any $T_w^G$ of height $h$ has at most $d^h$ terminals.




- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is at least $|V| + 1$ (containing at least $|V| + 1$ variables).

# Pumping Lemma for CFLs

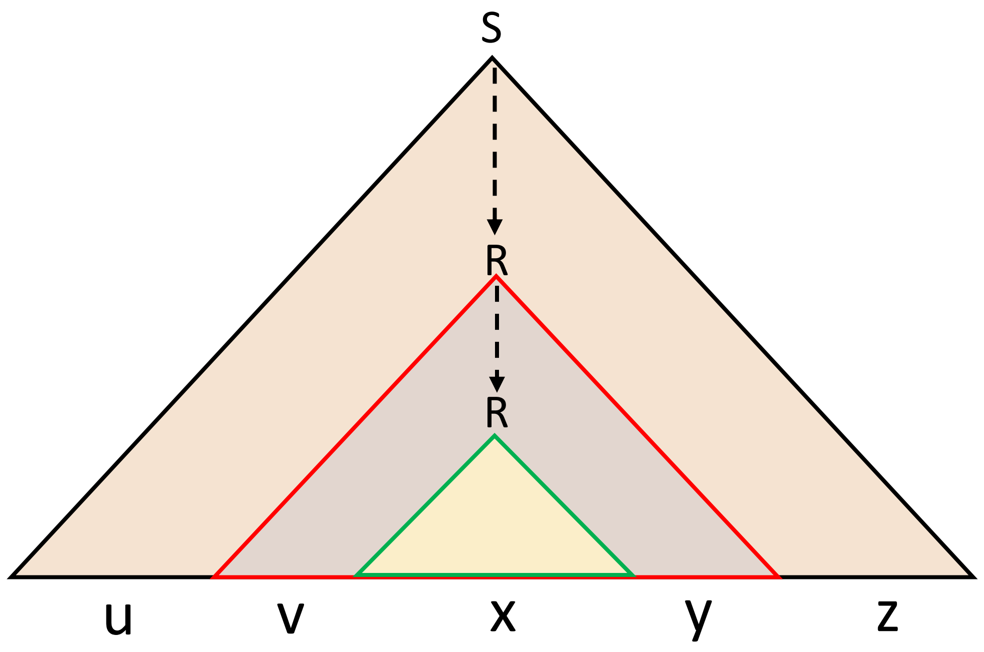
Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height $\geq$ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is $\geq |V| + 1$.
- Consider the lowest $|V| + 1$ variables in that path.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated.

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height $\geq$ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is at least $|V| + 1$.
- Consider the lowest $|V| + 1$ variables in that path.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated.

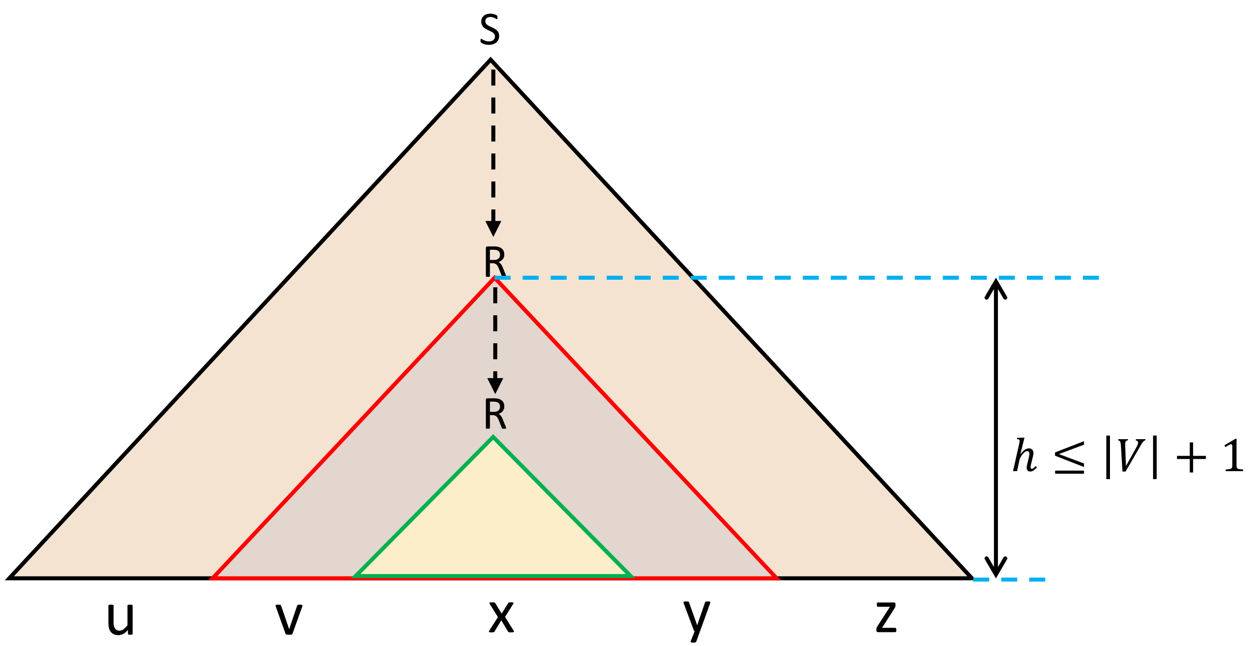



Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- Consider the longest path in the parse tree.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated



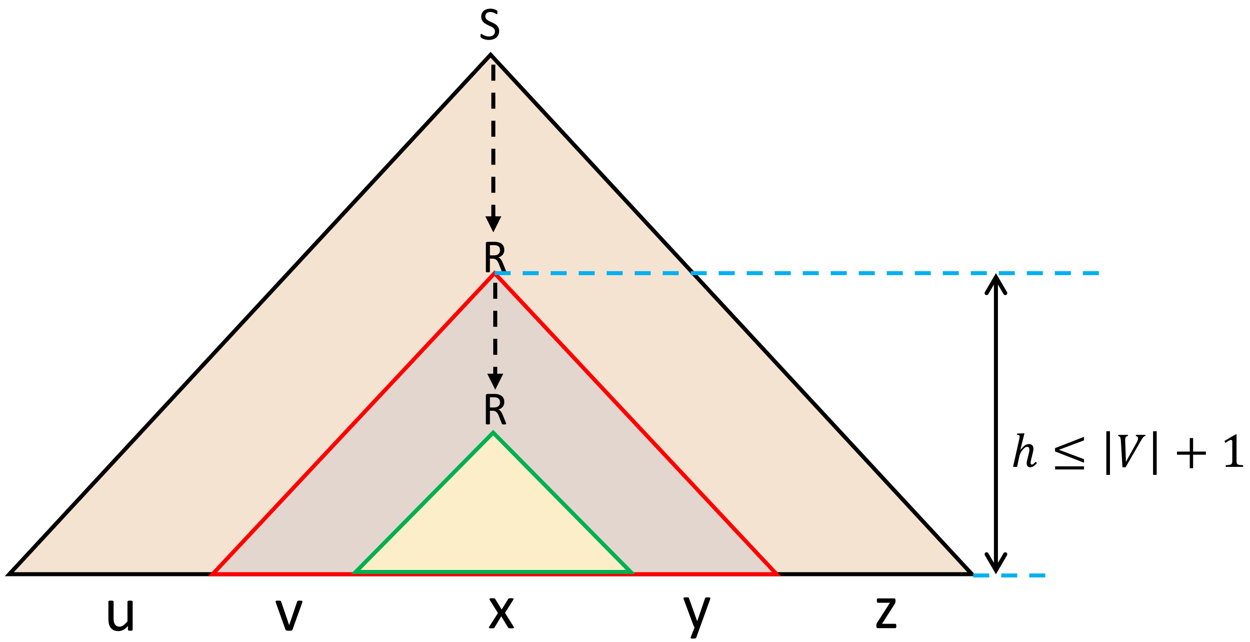

Then any string $w$ such that $|w| \geq p$, can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- Consider the lowest $|V| + 1$ variables in the longest path of length $\geq |V| + 1$.



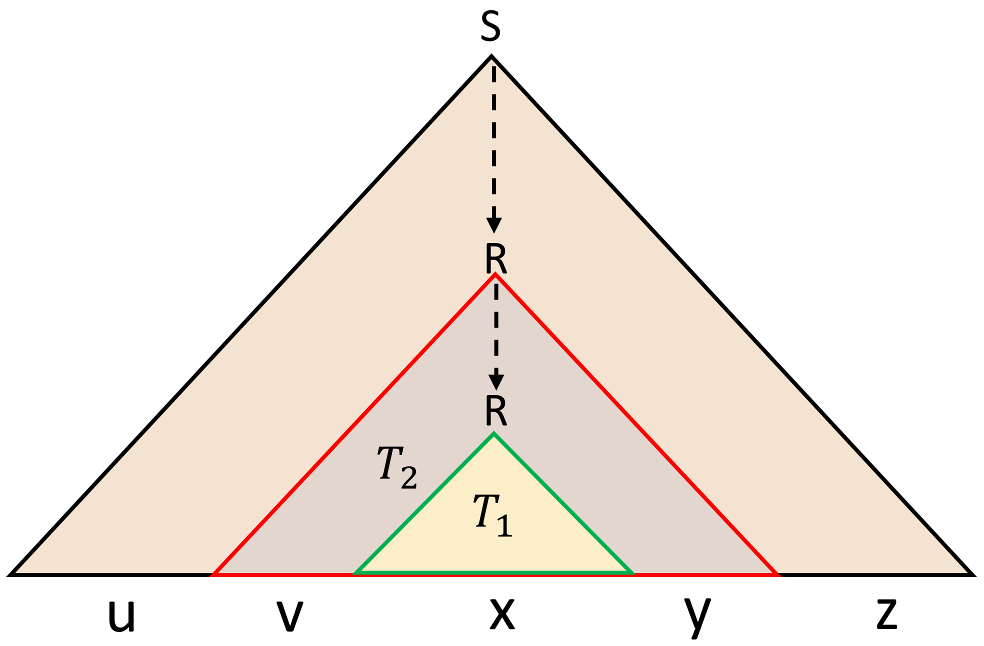

Then any string $w$ such that $|w| \geq p$, can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$ - the uppermost $R$ falls within the bottom $|V| + 1$ variables in the longest path and so the length of the string it can generate is $l \leq d^{|V|+1} = p$.

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- Consider the lowest $|V| + 1$ variables in the longest path of $h \geq |V| + 1$.

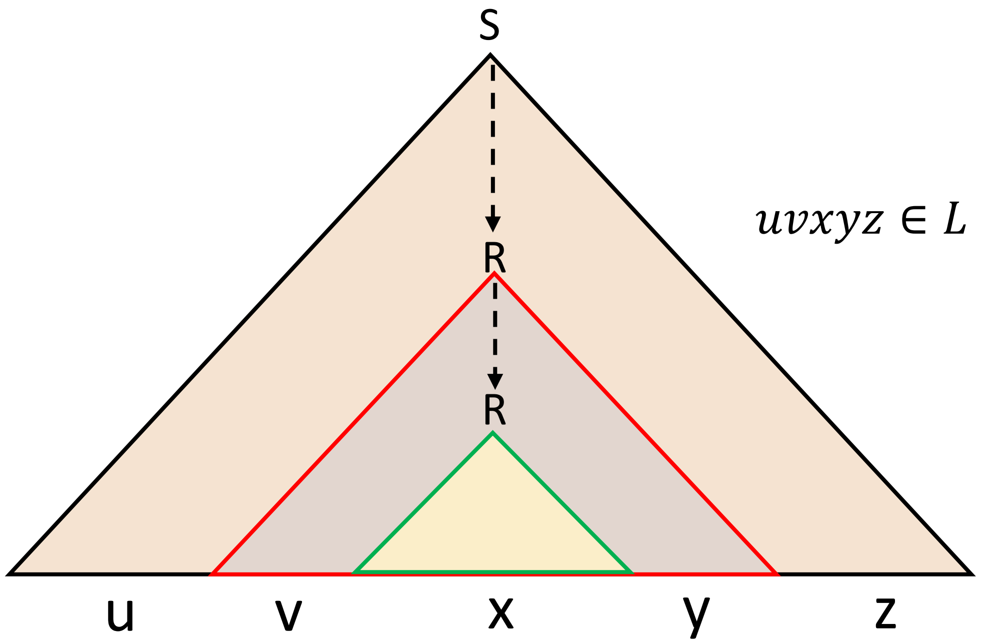



Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

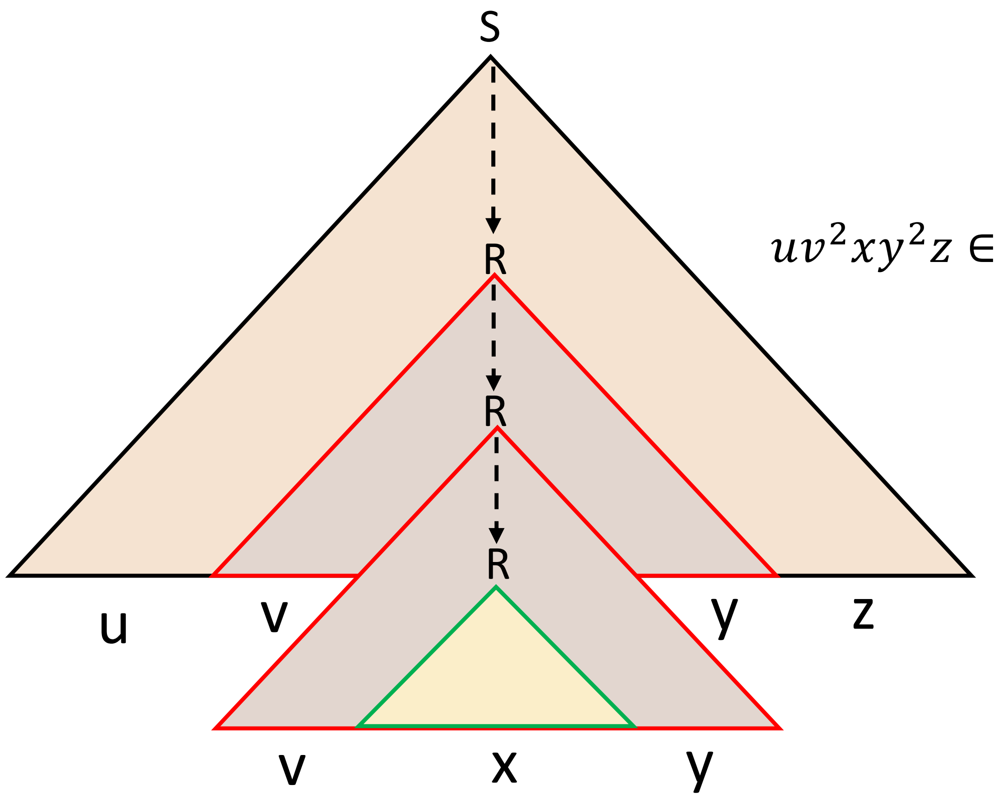
- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i > 0$ – Replace the subtree $T_1$ with the subtree $T_2$.

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.




$uvxyz \in L$



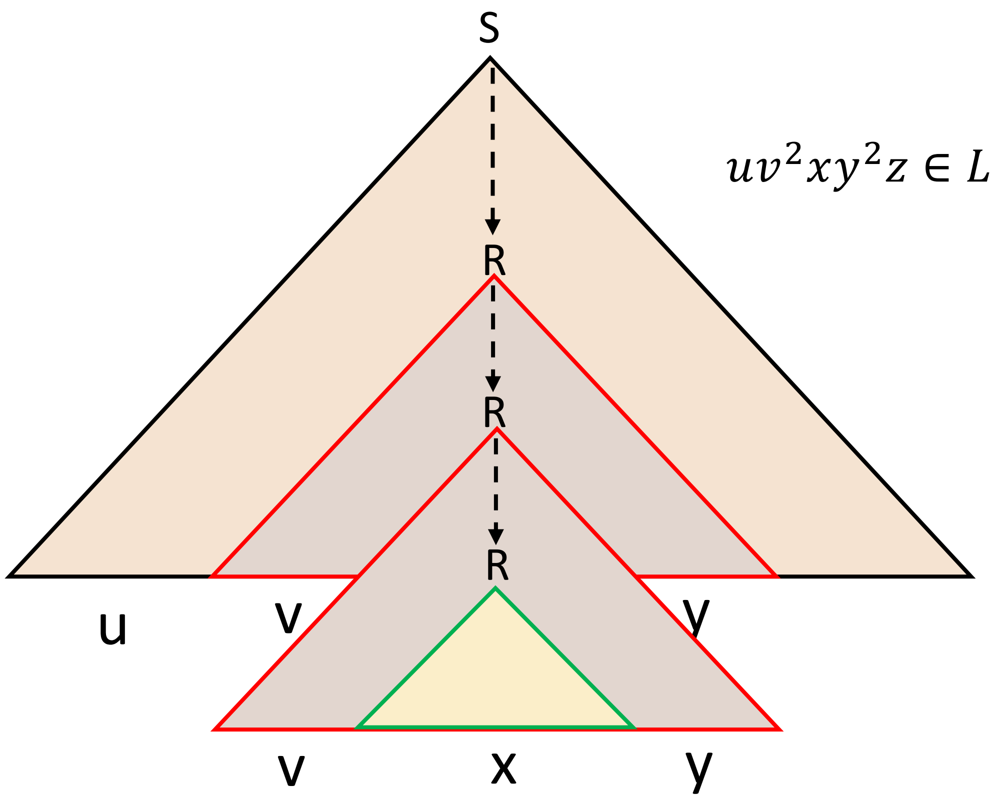

$uv^2xy^2z \in L$

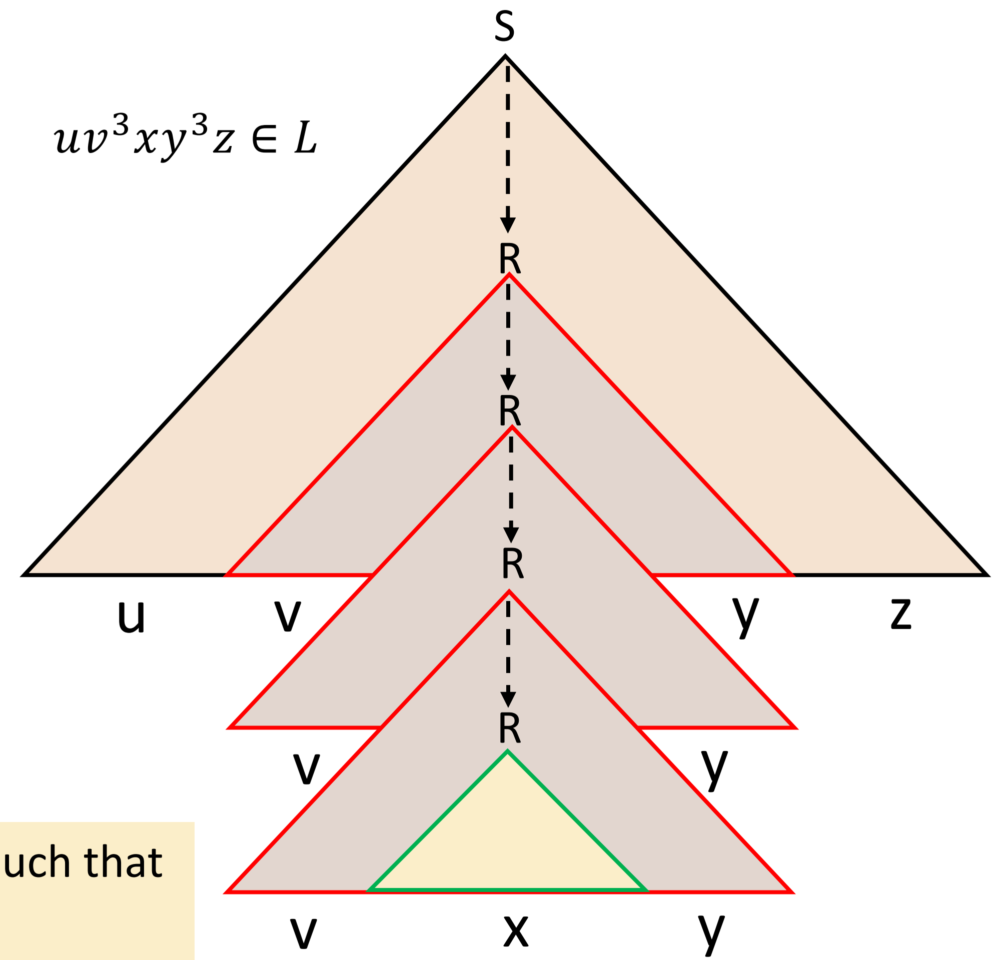
Then any string $w$, such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i > 0$ – Replace the subtree $T_1$ with the subtree $T_2$.

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.



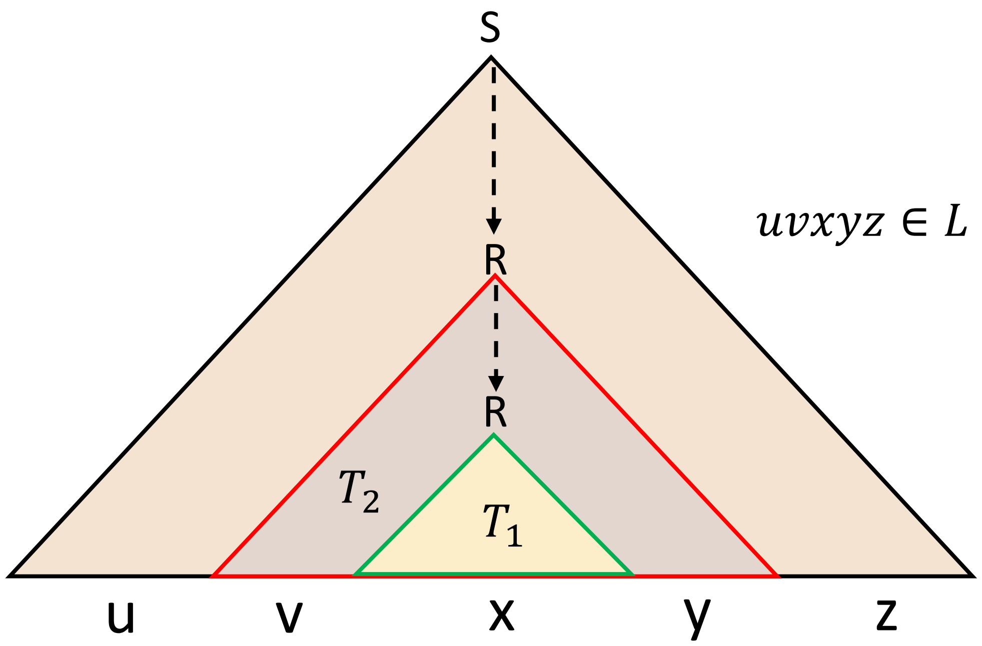




Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i > 0$

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.




$uvxyz \in L$

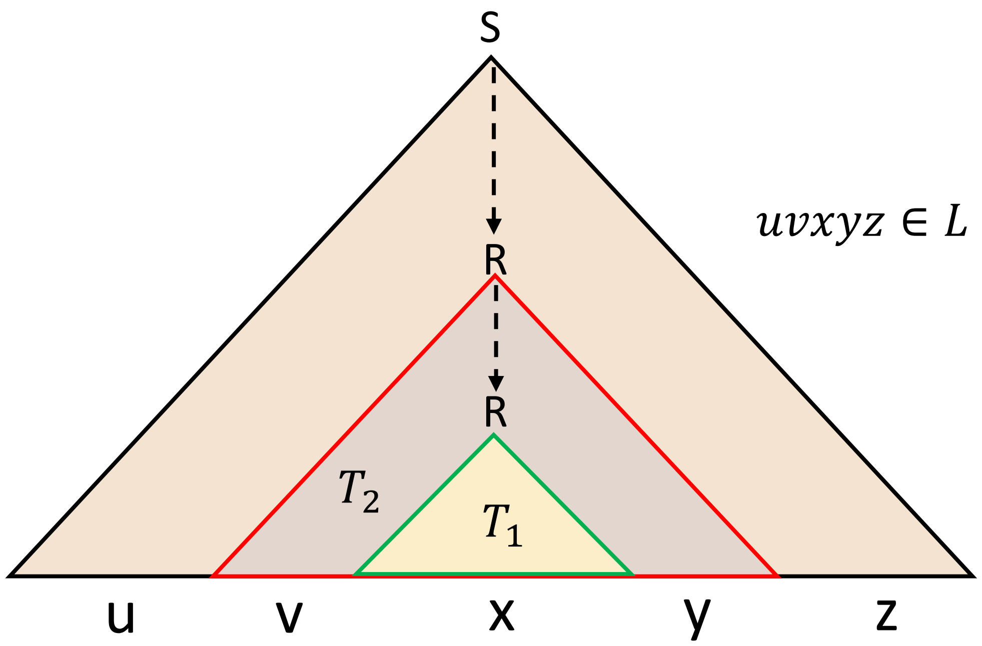
$uv^0xy^0z \in L$

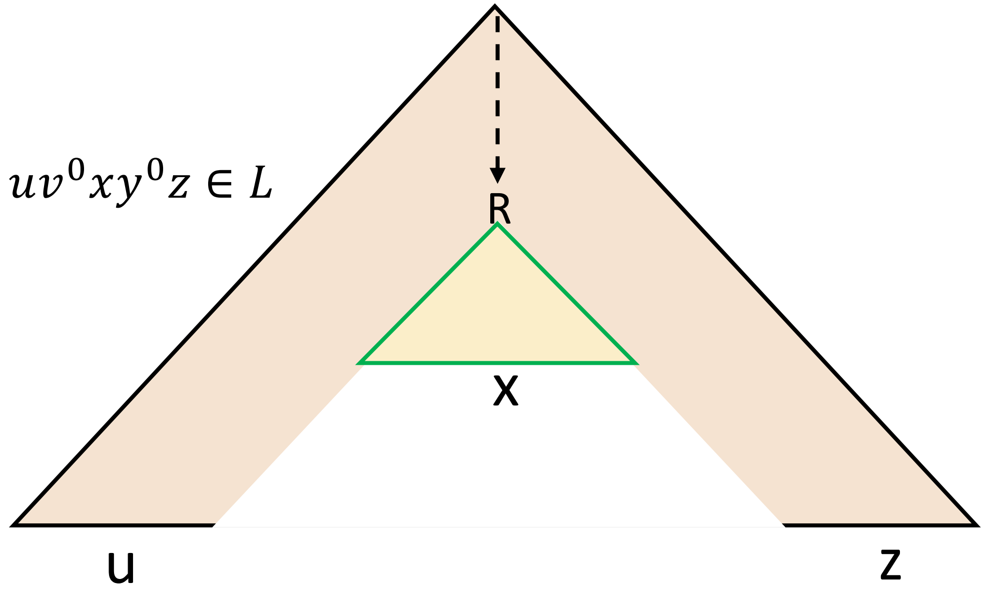
Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$ – for the $i = 0$ case, **replace the subtree $T_2$ with the subtree $T_1$**

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.



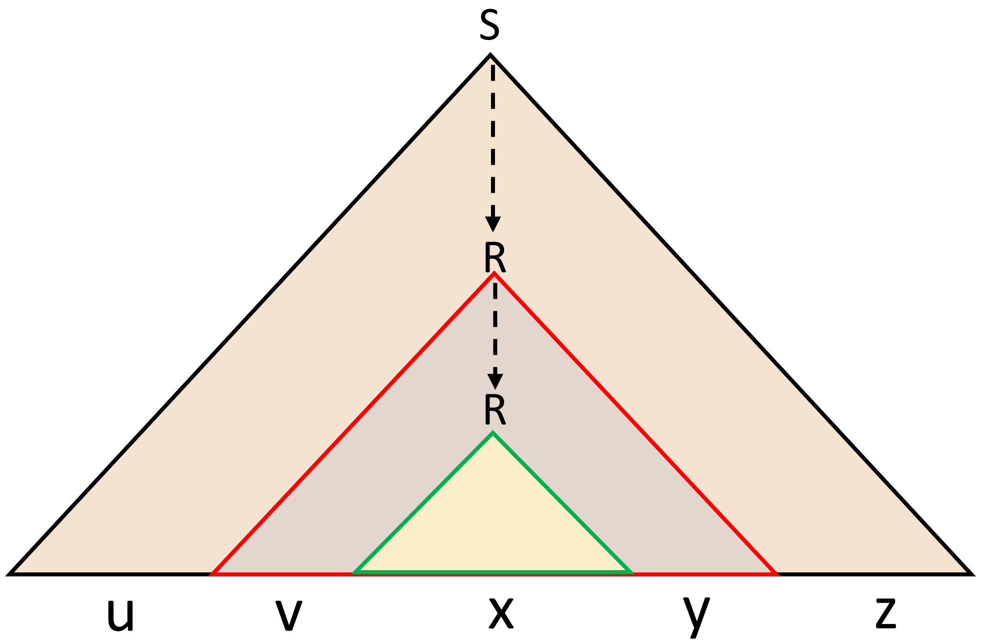

Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$ – for the $i = 0$ case, replace the subtree $T_2$ with the subtree $T_1$

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is at least $|V| + 1$.
- Consider the lowest $|V| + 1$ variables in that path.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated



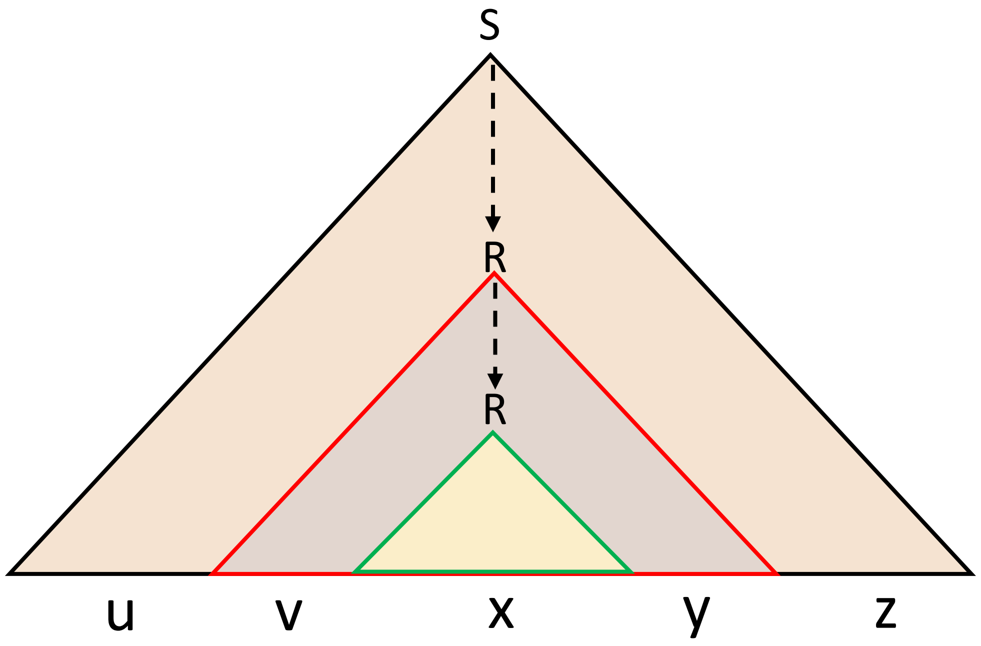

Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider a parse tree $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height $\geq$ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is at least $|V| + 1$
- Consider the lowest $|V| + 1$ variables in that path.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated




Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$

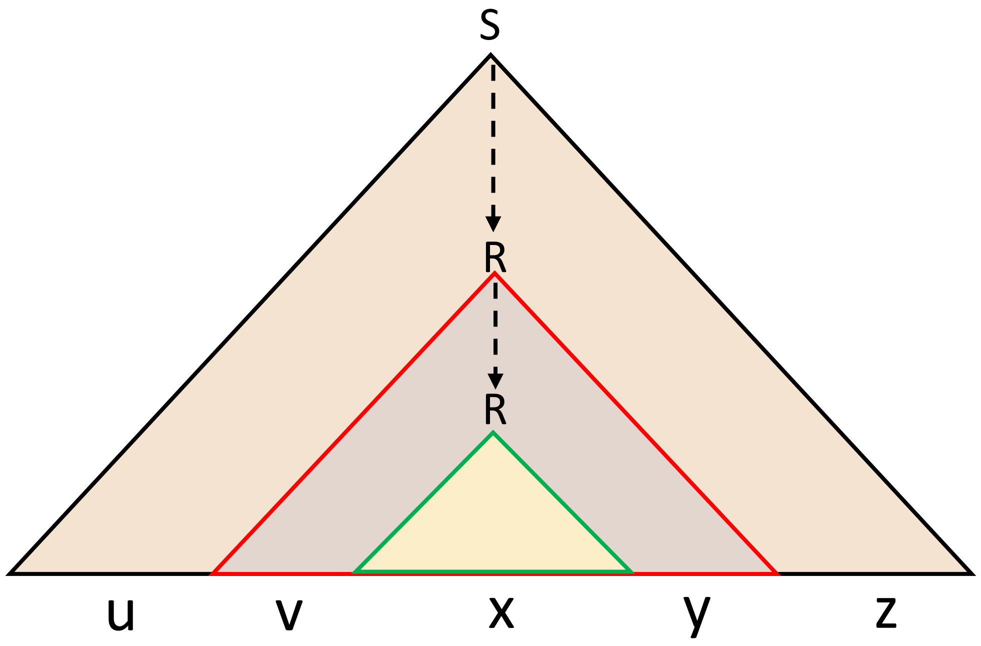
What if $G$ is ambiguous? More than one parse tree generates $w$.

Pick the one with the smallest number of nodes. So $T_w^G$ is the smallest parse tree generating $w$.

# Pumping Lemma for CFLs

Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider the **smallest parse tree** $T_w^G$ of $G$ that yields $w$.

- Let |V| be the total number of variables in the Grammar $G$.
- If $w \in L$ such that $|w| = p = d^{|V|+1}$, the underlying parse tree would have a height ≥ |V|+1.
- The longest path from the Start Variable $S$ to a terminal is at least $|V| + 1$.
- Consider the lowest $|V| + 1$ variables in that path.
- By the pigeonhole principle, within the lowest $|V| + 1$ variables, at least one variable is repeated




Then any string $w$ such that $|w| \geq p$ can be partitioned as $w = uvxyz$ such that

- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$

$T_w^G$ is the smallest parse tree generating $w$.

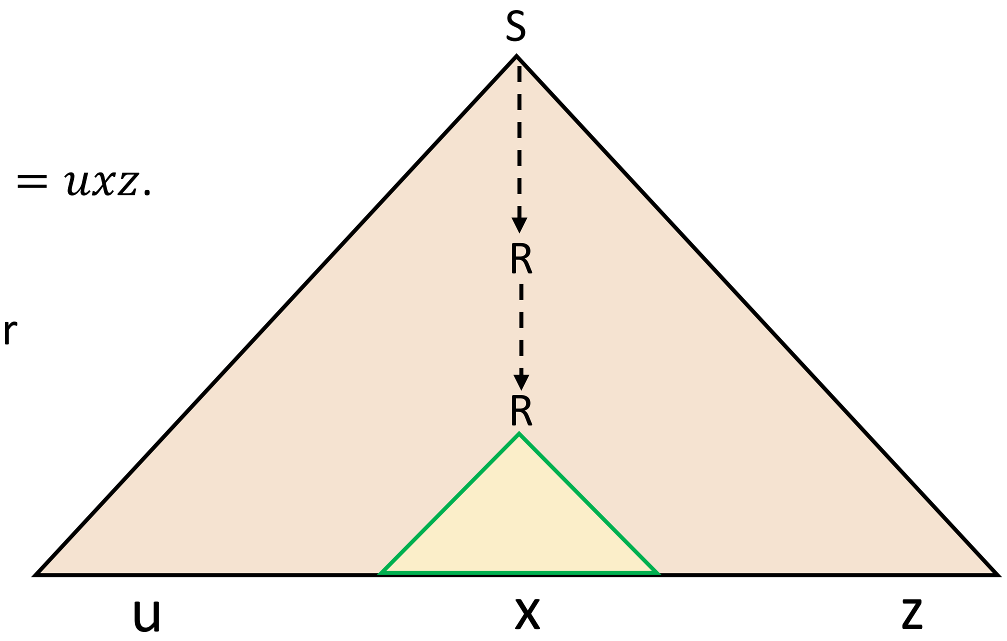
**This leads to an additional condition!**

**$v, y$ cannot be both empty, i.e. $|vy| \geq 1$**

# Pumping Lemma for CFLs

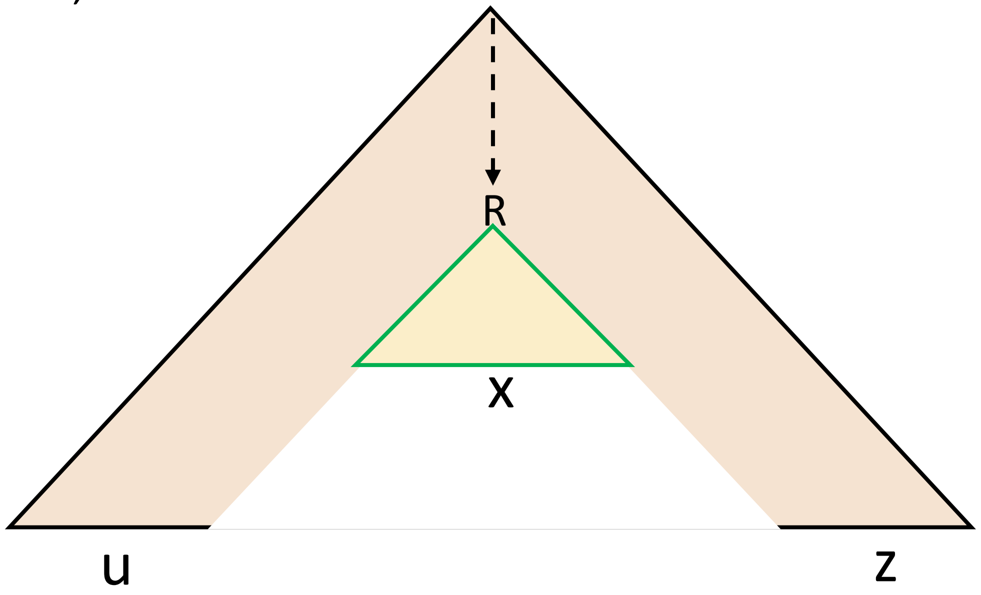
Let $L$ be a CFL and $G$ be such that $L = \mathcal{L}(G)$ and $w \in L$. Consider the **smallest parse tree** $T_w^G$ of $G$ that yields $w$.

**$v, y$ cannot be both empty, i.e. $|vy| \geq 1$**




**Proof by contradiction:** Let us assume that they were both empty, i.e. $w = uxz$.
Then $T_w^G$ would look like this.

However, if we substitute the smaller subtree rooted at $R$ with the higher subtree, we obtain




The parse tree to the left generates $w$ and has fewer nodes which is a **contradiction**!!

# Pumping Lemma for CFLs

**Putting things together:**

**Pumping Lemma for CFL:** If $L$ is Context Free, then there exists $p > 0$ (pumping length), such that, for any $w \in L$ of length $|w| \geq p$, $w$ can be split into five parts, i.e.

$$w = uvxyz$$

satisfying the following conditions:

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^ixy^iz \in L,\ \forall i \geq 0$

We have proved this in the previous slides.

# Pumping Lemma for CFLs

**Pumping Lemma for CFL: IF** $L$ is Context Free, **THEN** there exists $p > 0$ (pumping length), such that, for any $w \in L$ of length $|w| \geq p$, $w$ can be split into five parts, i.e.

$$w = uvxyz$$

satisfying the following conditions:

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^ixy^iz \in L, \ \forall i \geq 0$

**Note:** $(A \Rightarrow B) \equiv (\neg B) \Rightarrow (\neg A)$

IF $L$ is Context Free, THEN conditions of Pumping Lemma are Satisfied

$\equiv$

IF conditions of Pumping Lemma are NOT satisfied THEN $L$ is NOT Context Free

In order to prove that a language is not Context Free, assume that it is Context Free and obtain a contradiction.

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

Note that $|w| = 3p$ $(\geq p)$. The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^i xy^i z \in L, \forall i \geq 0$

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

Note that $|w| = 3p$ $(\geq p)$. The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^i xy^i z \in L, \forall i \geq 0$

First observe that in $w = 0^p 1^p 2^p$, the last $0$ and the first $2$ is separated by $p$ $1$'s. As $|vxy| \leq p$, the substring $vxy$ has at most two distinct symbols. Now consider the string $w' = uv^2 xy^2 z$. What are the possibilities of $vxy$?

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

Note that $|w| = 3p$ $(\geq p)$. The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^i xy^i z \in L, \forall i \geq 0$

First observe that in $w = 0^p 1^p 2^p$, the last $0$ and the first $2$ is separated by $p$ 1's. As $|vxy| \leq p$, the substring $vxy$ has at most two distinct symbols. Now consider the string $w' = uv^2 xy^2 z$. What are the possibilities of $vxy$?

- $vxy = 0^k \; or \; 1^k \; or \; 2^k, k \leq p$: Then $w' \notin L$. (E.g: If $vxy = 0^k$, then, $w'$ will have more $0's$ than $1's$ and $2's$.)

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

Note that $|w| = 3p$ ($\geq p$). The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^i xy^i z \in L, \forall i \geq 0$

First observe that in $w = 0^p 1^p 2^p$, the last $0$ and the first $2$ is separated by $p$ 1's. As $|vxy| \leq p$, the substring $vxy$ has at most two distinct symbols. Now consider the string $w' = uv^2 xy^2 z$. What are the possibilities of $vxy$?

- $vxy = 0^k$ *or* $1^k$ *or* $2^k, k \leq p$: Then $w' \notin L$. (E.g: If $vxy = 0^k$, then, $w'$ will have more $0's$ than $1's$ and $2's$.)

- $vxy = 0^m 1^n$ *or* $1^m 2^n, m + n \leq p$: Again, $w' \notin L$.

# Non Context Free Languages

$\boldsymbol{L = \{0^n1^n2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p1^p2^p \in L$.

Note that $|w| = 3p\ (\geq p)$. The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^ixy^iz \in L, \forall i \geq 0$

First observe that in $w = 0^p1^p2^p$, the last $0$ and the first $2$ is separated by $p$ 1's. As $|vxy| \leq p$, the substring $vxy$ has at most two distinct symbols. Now consider the string $w' = uv^2xy^2z$. What are the possibilities of $vxy$?

- $vxy = 0^k\ or\ 1^k\ or\ 2^k, k \leq p$: Then $w' \notin L$. (E.g: If $vxy = 0^k$, then, $w'$ will have more $0's$ than $1's$ and $2's$.)

- $vxy = 0^m1^n\ or\ 1^m2^n, m + n \leq p$: Again, $w' \notin L$. (E.g: If $vxy = 0^m1^n$, then $w'$ will have less $2'$s than the other two symbols)

# Non Context Free Languages

$\boldsymbol{L = \{0^n 1^n 2^n | n \geq 0\}}$ is not Context-Free.

**Proof:** We shall prove this by contradiction. Let $L$ be a CFL and so it must satisfy the conditions of the Pumping Lemma. Let $p$ be the pumping length and so $w = 0^p 1^p 2^p \in L$.

Note that $|w| = 3p\ (\geq p)$. The pumping lemma states that $w$ can be split into $w = uvxyz$ such that

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^i xy^i z \in L, \forall i \geq 0$

First observe that in $w = 0^p 1^p 2^p$, the last $0$ and the first $2$ is separated by $p$ 1's. As $|vxy| \leq p$, the substring $vxy$ has at most two distinct symbols. Now consider the string $w' = uv^2xy^2z$. What are the possibilities of $vxy$?

- $vxy = 0^k$ *or* $1^k$ *or* $2^k, k \leq p$: Then $w' \notin L$. (E.g: If $vxy = 0^k$, then, $w'$ will have more $0's$ than $1's$ and $2's$.)

- $vxy = 0^m 1^n$ *or* $1^m 2^n, m + n \leq p$: Again, $w' \notin L$. (E.g: If $vxy = 0^m 1^n$, then $w'$ will have less $2'$s than the other two symbols)

Both cases lead to a contradiction. Hence, $L \notin CFL$.

# Non Context Free Languages

$L = \{0^n 1^n 2^n | n \geq 0\}$ is not Context-Free.

Other examples:

- $L = \{ww | w \in \{0, 1\}^*\}$
- $L = \{a^p | p \text{ is prime}\}$
- $L = \{0^n 1^{n^2} | n \geq 0\}$

………………………………………………………………

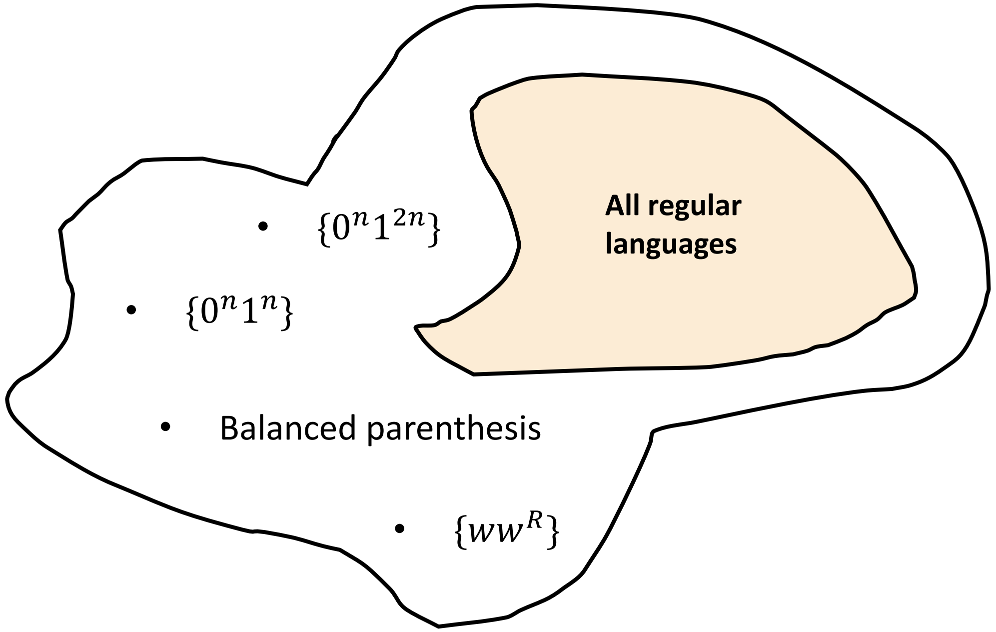
Recommend you to use Pumping Lemma and check that they are indeed not Context Free

# Closure properties of CFL

Now that we know that there are languages that are not Context Free – let us investigate the closure properties of CFLs.

**Recall** what we mean by the statement "**CFLs are closed under some operation**"

- We pick up points within the set of all CFLs (say $L_1$ and $L_2$)
- Perform *set operations* such as Union, concatenation, Star, intersection, compliment etc on them.
- Observe whether the resulting language still belongs to the set of all CFLs.
- If so, we say, CFLs are **closed** under that operation otherwise we say CFLs are not closed under than operation.




**Set of all CFLs**

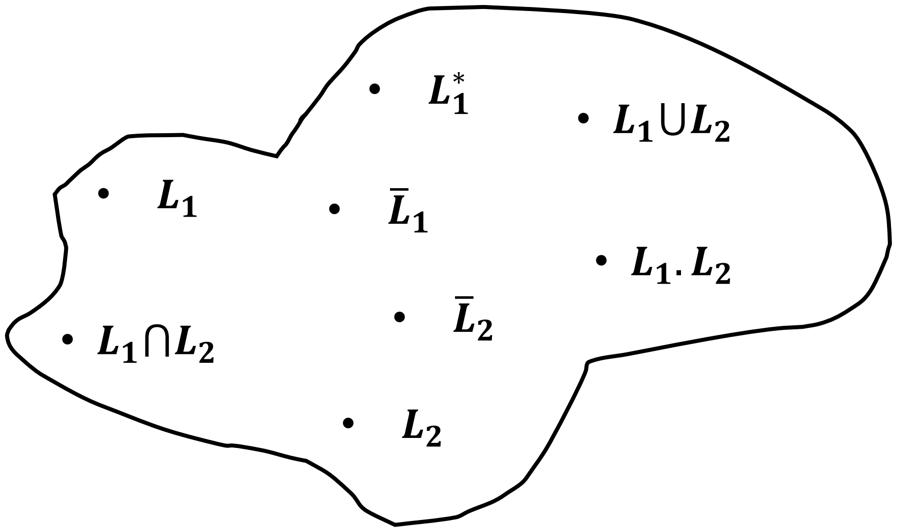
# Closure properties of CFL

**Some operations:** Let $L_1$ and $L_2$ be languages.

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Intersection:** $L_1 \cap L_2 = \{x | x \in L_1 \text{ and } x \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0 \text{ and each } x_i \in L\}$
- **Complementation:** $\bar{L} = \{x | x \notin L\}$

**Recall that for Regular languages:** RL are closed under

- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**




**Set of all regular Languages**

**What about CFLs?**

# Closure properties of CFL

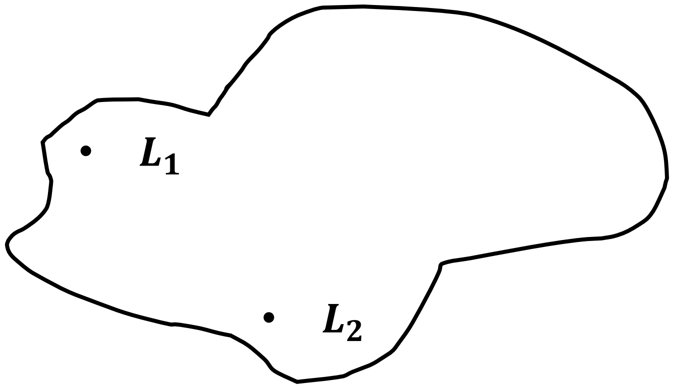
**Q:** Is the set of all CFLs **closed under union**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cup L_2$ also a CFL?

**Proof:** Suppose $G_1$ and $G_2$ be grammars such that $L(G_1) = L_1$ and $L(G_2) = L_2$.

Suppose:

Rules of $G_1$: $S_1 \rightarrow \ldots$
Rules of $G_2$: $S_2 \rightarrow \ldots$




**Set of all CFLs**

# Closure properties of CFL

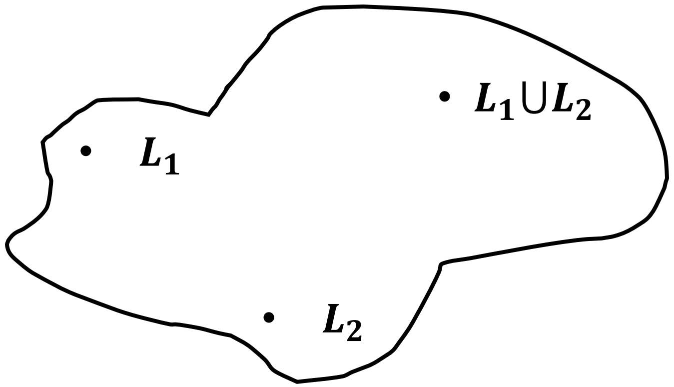
**Q:** Is the set of all CFLs **closed under union**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cup L_2$ also a CFL?

**Proof:** Suppose $G_1$ and $G_2$ be grammars such that $L(G_1) = L_1$ and $L(G_2) = L_2$.

Suppose:

Rules of $G_1$: $S_1 \rightarrow \ldots$
Rules of $G_2$: $S_2 \rightarrow \ldots$




**Set of all CFLs**

Also suppose that the rules of $G_1$ and $G_2$ have different variables.

Then the grammar for $L_1 \cup L_2$ contains all the variables of $G_1$ and $G_2$, all the terminals of $G_1$ and $G_2$. Additionally,

Add a new start symbol $S$ and the rules of $G_1 \cup G_2$ :

$$S \rightarrow S_1 | S_2$$

followed by rules of $G_1$ and rules of $G_2$. So CFLs are closed under union.

# Closure properties of CFL

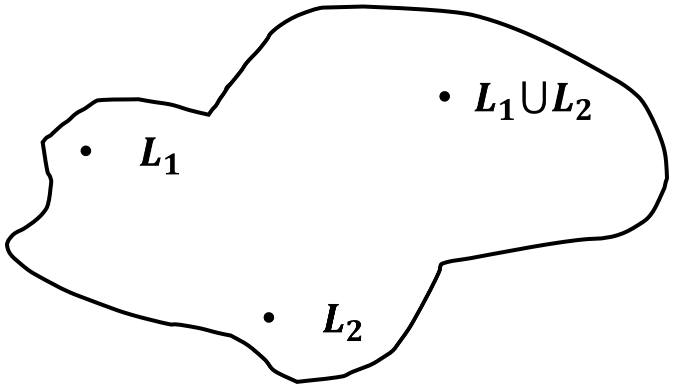
**Q:** Is the set of all CFLs **closed under Concatenation**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1.L_2$ also a CFL?

**Proof:** Suppose $G_1$ and $G_2$ be grammars such that $L(G_1) = L_1$ and $L(G_2) = L_2$.

Suppose:

Rules of $G_1$: $S_1 \rightarrow \ldots$
Rules of $G_2$: $S_2 \rightarrow \ldots$



Set of all CFLs

Also suppose that the rules of $G_1$ and $G_2$ have different variables. Then define $G'$ such that $L(G') = L_1.L_2$, as the grammar containing all the variables of $G_1$ and $G_2$, all the terminals of $G_1$ and $G_2$, with a new start symbol $S$. The new rules:

$$S \rightarrow S_1.S_2$$

followed by rules of $G_1$ and rules of $G_2$.

So CFLs are closed under concatenation.

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under Concatenation**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1.L_2$ also a CFL?

**Proof:** Suppose $G_1$ and $G_2$ be grammars such that $L(G_1) = L_1$ and $L(G_2) = L_2$.

Suppose:

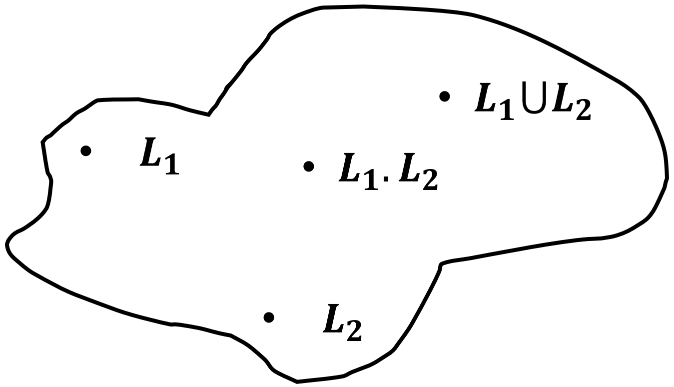
Rules of $G_1$: $S_1 \rightarrow$…
Rules of $G_2$: $S_2 \rightarrow$…

Also suppose that the rules of $G_1$ and $G_2$ have different variables. Then define $G'$ such that $L(G') = L_1.L_2$, as the grammar containing all the variables of $G_1$ and $G_2$, all the terminals of $G_1$ and $G_2$, with a new start symbol $S$. The new rules:

$$S \rightarrow S_1.S_2$$

followed by rules of $G_1$ and rules of $G_2$.

So CFLs are closed under concatenation.




Set of all CFLs

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under Star**?

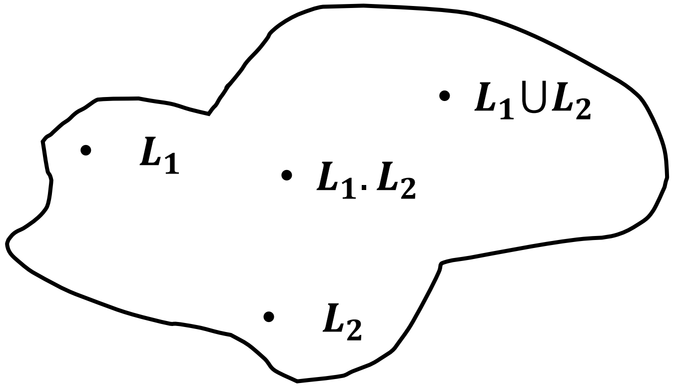
Suppose $L$ is a CFL. Is $L^*$ also a CFL?

**Proof:** Suppose $G$ be a grammar such that $L(G) = L_1$

Suppose:

Rules of G: $S_1 \rightarrow \ldots$

Then the grammar $G'$ such that $L(G) = L^*$ is the same as $G$ with a new start symbol and the additional rules




**Set of all CFLs**

# Closure properties of CFL

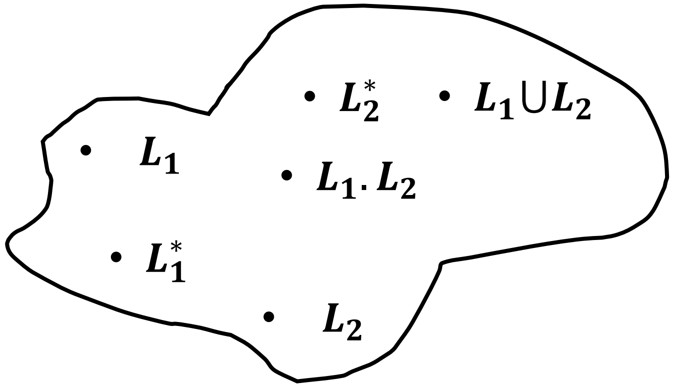
**Q:** Is the set of all CFLs **closed under Star**?

Suppose $L$ is a CFL. Is $L^*$ also a CFL?

**Proof:** Suppose $G$ be a grammar such that $L(G) = L_1$

Suppose:

Rules of G: $S_1 \rightarrow \ldots$



**Set of all CFLs**

Then the grammar $G'$ such that $L(G) = L^*$ is the same as $G$ with a new start symbol and the additional rules

$$S \rightarrow S_1 S | \epsilon$$

So CFLs are closed under Star.

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under intersection**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cap L_2$ also a CFL?

**Proof:** We will prove that CFLs are NOT closed under intersection by using this simple counterexample. Let

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under intersection**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cap L_2$ also a CFL?

**Proof:** We will prove that CFLs are NOT closed under intersection by using this simple counterexample. Let

$\boldsymbol{L_1 = \{0^n 1^n 2^m | m, n \geq 1\}}$ and $\boldsymbol{L_2 = \{0^m 1^n 2^n | l, n \geq 1\}}$

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under intersection**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cap L_2$ also a CFL?

**Proof:** We will prove that CFLs are NOT closed under intersection by using this simple counterexample. Let

$\boldsymbol{L_1 = \{0^n 1^n 2^m | m, n \geq 1\}}$ and $\boldsymbol{L_2 = \{0^m 1^n 2^n | l, n \geq 1\}}$

Note that $L_1, L_2 \in$ CFL – each of them are concatenation of two CFLs.

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under intersection**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cap L_2$ also a CFL?

**Proof:** We will prove that CFLs are NOT closed under intersection by using this simple counterexample. Let

$\boldsymbol{L_1} = \{\boldsymbol{0^n 1^n 2^m} | \boldsymbol{m, n} \geq \boldsymbol{1}\}$ and $\boldsymbol{L_2} = \{\boldsymbol{0^m 1^n 2^n} | \boldsymbol{l, n} \geq \boldsymbol{1}\}$

Note that $L_1, L_2 \in$ CFL – each of them are concatenation of two CFLs.

E.g: $\mathrm{L}_1$ is a concatenation of $\{0^n 1^n | n \geq 1\}$ and $\{2^m | m \geq 1\}$ and the rules of the corresponding grammar are

$$S \to AB$$
$$A \to 0A1|01$$
$$B \to 2B|2$$

What is $L_1 \cap L_2$?

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under intersection**?

Suppose $L_1$ and $L_2$ are CFLs. Is $L = L_1 \cap L_2$ also a CFL?

**Proof:** We will prove that CFLs are NOT closed under intersection by using this simple counterexample. Let

$\boldsymbol{L_1 = \{0^n 1^n 2^m | m, n \geq 1\}}$ and $\boldsymbol{L_2 = \{0^m 1^n 2^n | m, n \geq 1\}}$

Note that $L_1, L_2 \in$ CFL – each of them are concatenation of two CFLs. E.g: $\mathrm{L}_1$ is a concatenation of $\{0^n 1^n | n \geq 1\}$ and $\{2^m | m \geq 1\}$ and the rules of the corresponding grammar are

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow 0A1|01 \\
B &\rightarrow 2B|2
\end{aligned}
$$

$\boldsymbol{L_1 \cap L_2 = \{0^n 1^n 2^n | n \geq 1\}}$ which is not a CFL.

Hence **CFLs are NOT closed under intersection!**

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under complementation**?

Suppose $L$ is a CFL. Is $\overline{L}$ also a CFL?

**Proof: ??????????**

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under complementation**?

Suppose $L$ is a CFL. Is $\bar{L}$ also a CFL?

**Proof:** Let us assume that CFLs are closed under complementation. Then if $L_1$ and $L_2$ are context free, then $\boldsymbol{\bar{L}_1}$ and $\boldsymbol{\bar{L}_2}$ are also context free. This would imply that

$$\boldsymbol{\bar{L}_1 \cup \bar{L}_2 \in CFL}$$

# Closure properties of CFL

**Q:** Is the set of all CFLs **closed under complementation**?

Suppose $L$ is a CFL. Is $\bar{L}$ also a CFL?

**Proof:** Let us assume that CFLs are closed under complementation. Then if $L_1$ and $L_2$ are context free, then $\boldsymbol{\bar{L}_1}$ and $\boldsymbol{\bar{L}_2}$ are also context free. This would imply that

$$\boldsymbol{\bar{L}_1 \cup \bar{L}_2 \in CFL}$$

Finally, this would imply $\boldsymbol{\overline{\overline{L_1} \cup \overline{L_2}} \in CFL}$. However,

$$\boldsymbol{L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}}$$

But this would imply $\boldsymbol{L_1 \cap L_2 \in CFL}$, which is a contradiction.

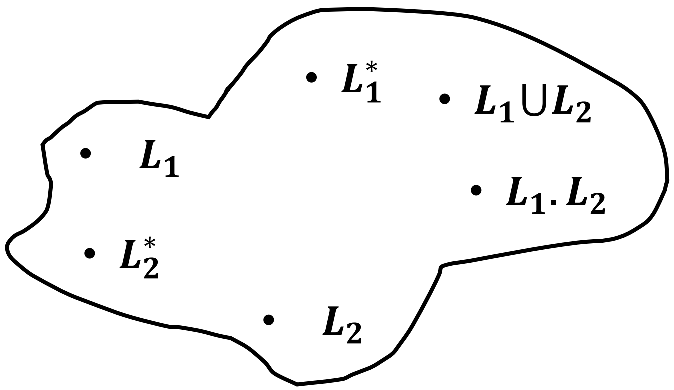
Thus CFLs are NOT closed under complementation.

# Closure properties of CFL

**Recall that for Regular languages:**

RLs are closed under

- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**




**Set of all CFLs**

**For CFLs:**

CFLs are closed under

- **Union**
- **Star**
- **Concatenation**

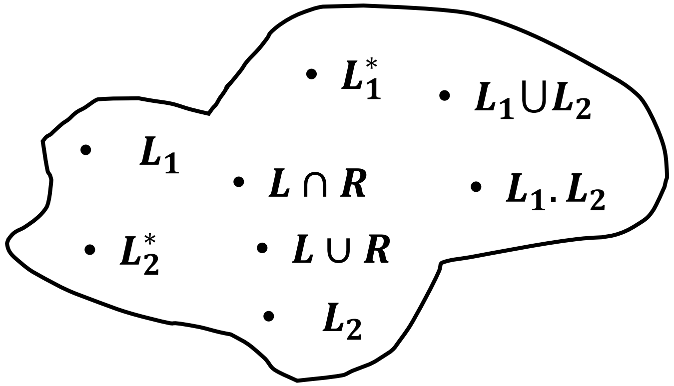
CFLs are NOT closed under

- **Complementation**
- **Intersection**

# Closure properties of CFL

**Recall that for Regular languages:**

RLs are closed under
- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**




**Set of all CFLs**

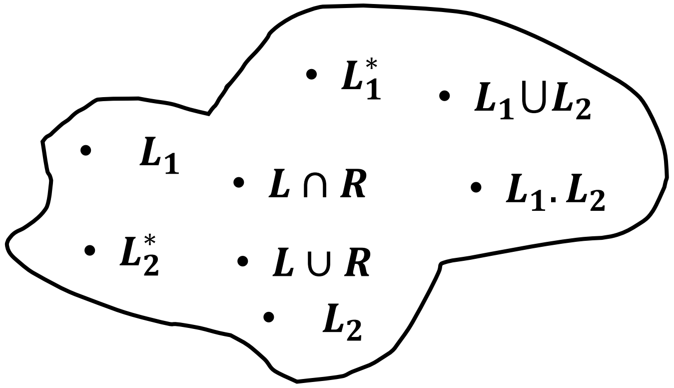
**For CFLs:**

CFLs are closed under
- **Union**
- **Star**
- **Concatenation**

CFLs are NOT closed under
- **Complementation**
- **Intersection**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

# Closure properties of CFL




**Set of all CFLs**

- CFLs are closed under **Union, Star, Concatenation**
- CFLs are NOT closed under **Complementation, Intersection**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

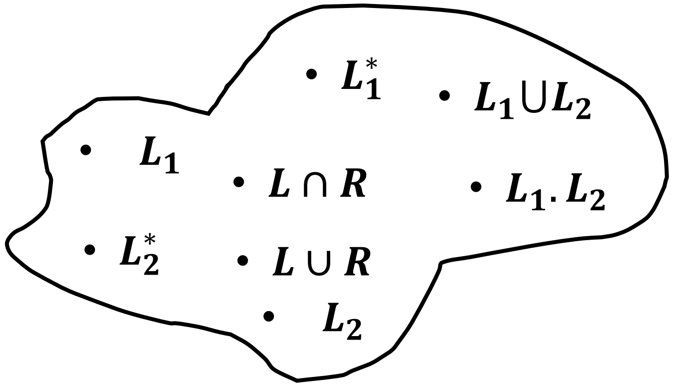
**Proof intuition:** Construct a **Product PDA**.

If the states of the PDA $P$: $Q = (q_1, q_2, \cdots, q_m)$ and DFA $D$: $Q' = (d_1, d_2, \cdots, d_n)$, then states of **Product PDA $X$:**

$Q = \{(q, d), \forall q \in Q, \forall d \in Q'\}$ **Start state:** $(q_1, d_1)$

# Closure properties of CFL

- CFLs are closed under **Union, Star, Concatenation**
- CFLs are NOT closed under **Complementation, Intersection**




**Set of all CFLs**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

**Proof intuition:** Construct a **Product PDA**.

If the states of the PDA $P$: $Q = (q_1, q_2, \cdots, q_m)$ and DFA $D$**:** $Q' = (d_1, d_2, \cdots, d_n)$, then states of **Product PDA $X$:**

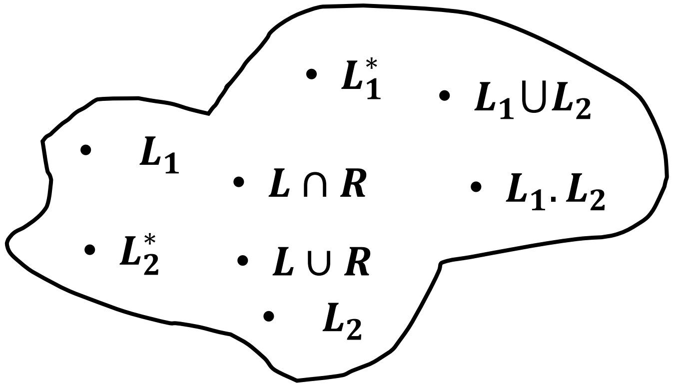
$Q = \{(q, d), \forall q \in Q, \forall d \in Q'\}$ **Start state:** $(q_1, d_1)$

If $\delta(q_i, a, b) = (q_j, c)$ and $\delta(d_k, a) = d_l$, then for $X$: $\delta\big((q_i, d_k), a, b\big) = (\big(q_j, d_l\big), c)$.

**So $X$ is a PDA.**

# Closure properties of CFL

- CFLs are closed under **Union, Star, Concatenation**
- CFLs are NOT closed under **Complementation, Intersection**



**Set of all CFLs**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

**Proof intuition:** Construct a **Product PDA**.

If the states of the PDA $P$: $Q = (q_1, q_2, \cdots, q_m)$ and DFA $D$**:** $Q' = (d_1, d_2, \cdots, d_n)$, then states of **Product PDA $X$:**

$$Q = \{(q, d), \forall q \in Q, \forall d \in Q'\}$$

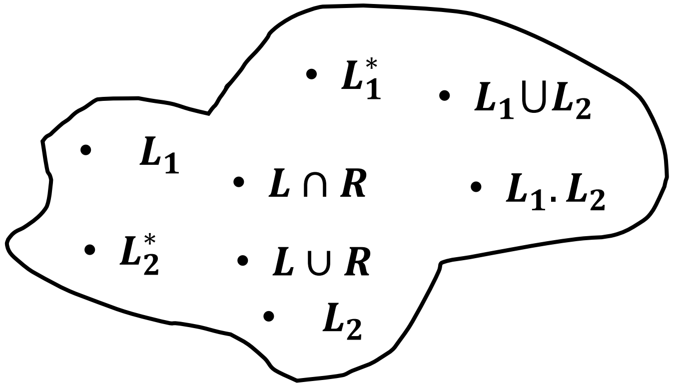
**Start state:** $(q_1, d_1)$

If $\delta(q_i, a, b) = (q_j, c)$ and $\delta(d_k, a) = d_l$, then for $X$: $\delta\big((q_i, d_k), a, b\big) = (\big(q_j, d_l\big), c)$.
If $\delta(q_i, \epsilon, b) = (q_j, c)$ and $\delta(d_k, \epsilon) = \Phi$, then for $X$: $\delta\big((q_i, d_k), \epsilon, b\big) = (\big(q_j, d_k\big), c)$.

- $\boldsymbol{L(X) = L(P) \cap L(R)}$ if the final state, say $(q_r, d_s)$ is such that $q_r$ and $d_s$ **are both final states of $P$ AND D respectively.**

# Closure properties of CFL

- CFLs are closed under **Union, Star, Concatenation**
- CFLs are NOT closed under **Complementation, Intersection**




**Set of all CFLs**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

**Proof intuition:** Construct a **Product PDA**.

If the states of the PDA $P$: $Q = (q_1, q_2, \cdots, q_m)$ and DFA $D$**:** $Q' = (d_1, d_2, \cdots, d_n)$, then states of **Product PDA $X$:**

$$Q = \{(q, d), \forall q \in Q, \forall d \in Q'\}$$

**Start state:** $(q_1, d_1)$

If $\delta(q_i, a, b) = (q_j, c)$ and $\delta(d_k, a) = d_l$, then for $X$: $\delta\big((q_i, d_k), a, b\big) = (\big(q_j, d_l\big), c)$.
If $\delta(q_i, \epsilon, b) = (q_j, c)$ and $\delta(d_k, \epsilon) = \Phi$, then for $X$: $\delta\big((q_i, d_k), \epsilon, b\big) = (\big(q_j, d_k\big), c)$.

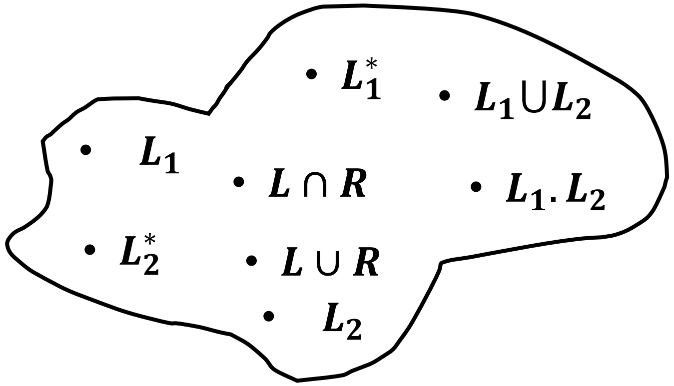
- $\boldsymbol{L(X) = L(P) \cap L(R)}$ if the final state, say $(q_r, d_s)$ is such that $q_r$ and $d_s$ **are both final states of $P$ AND D respectively.**
- $\boldsymbol{L(X) = L(P) \cup L(R)}$ if the final state, say $(q_r, d_s)$ is such that **EITHER** $q_r$ **or** $d_s$ **are final states of $P$ OR D respectively.**

# Closure properties of CFL

**Recall that for Regular languages:**

RL are closed under

- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**




**Set of all CFLs**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

**For CFLs:**

CFLs are closed under

- **Union**
- **Star**
- **Concatenation**

CFLs are NOT closed under

- **Complementation**
- **Intersection**

**For DCFLs**

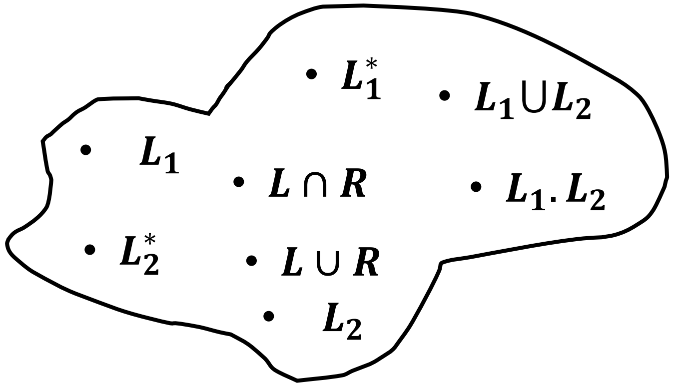
- **NOT closed Union** - construct a counter example
- **Closed under complementation** - construct a "toggled" DPDA (a bit of extra work is needed to take care of the dead states)
- **NOT closed under intersection** - use the first two to prove this

# Closure properties of CFL

**Recall that for Regular languages:**

RL are closed under

- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**




**Set of all CFLs**

**For CFLs:**

CFLs are closed under

- **Union**
- **Star**
- **Concatenation**

CFLs are NOT closed under

- **Complementation**
- **Intersection**

**If $L$ is a CFL and $R$ is a regular language then**
**$L \cap R$ is a CFL.**
**$L \cup R$ is a CFL.**

**For DCFLs**

- **NOT closed Union**
- **Closed under complementation**
- **NOT closed under intersection**

**Next lecture:**

- **Turing Machine**

# Thank You!