

Properties of Context-Free Languages

Teodor Rus

`rus@cs.uiowa.edu`

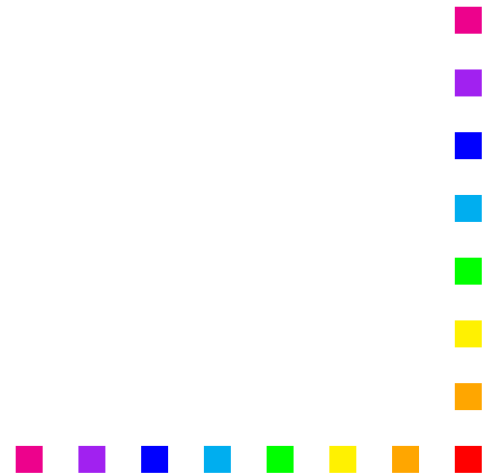
The University of Iowa, Department of Computer Science



Proving context-freeness

There are two mechanisms that may be used to show that a language is not context-free:

1. Use closure operators;
2. Use pumping lemma for context-free languages.



CFL Closure Properties

Theorem: class of CFL is closed under union, concatenation, and star.

Proof: by construction. Consider two CFL, L_1, L_2 specified by the CFG G_1, G_2 where $G_i = (V_i, \Sigma, P_i, S_i)$, $i = 1, 2$, $V_1 \cap V_2 = \emptyset$.

1. $G_{\cup} = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$, where S is a new nonterminal symbol, specifies $L_1 \cup L_2$
2. $G_{\circ} = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$, where S is a new nonterminal symbol, specifies $L_1 \circ L_2$
3. $G_{*} = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow S_1 S | \epsilon\}, S)$, where S is a new nonterminal symbol, specifies L_1^* .



Corollary

Each language that differ from a context-free language by a regular language is context-free

Proof: by construction. If L is context-free and L' differ from L by a regular language then either $L' = L \cup R$ or $L' = L \setminus R$

1. If $L' = L \cup R$ then by CFL closure for union L' is a CFL
2. If $L' = L \setminus R$ then $L' = L \cap \bar{R}$ by the definition of set-subtraction operator.
3. Since \bar{R} is regular (complement of a regular language is regular) it follows by the the solution of problem 2.18 that L' is a CFL.



Homomorphism

Let Σ and Δ be two alphabets.

A homomorphism is a function $h : \Sigma \rightarrow \Delta^*$ assigning words in Δ^* to symbols in Σ

Extension: $h : \Sigma \rightarrow \Delta^*$ can be extended to $h : \Sigma^* \rightarrow \Delta^*$ by: $h(\epsilon) = \epsilon$ and for all $x = ay$ with $a \in \Sigma$, $h(ay) = h(a) \circ h(y)$

h may be extended to operate on languages:
for $L \subseteq \Sigma^*$, $h(L) = \{h(x) \mid x \in L\}$



Substitution

Consider again two alphabets, Σ and Δ .

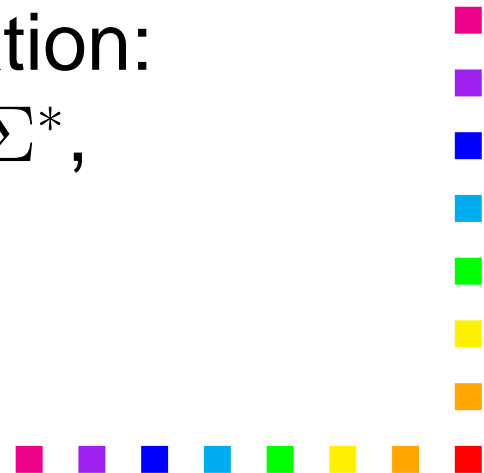
A substitution is a function $\sigma : \Sigma \rightarrow \mathcal{P}(\Delta^*)$

A substitution assigns to each letter of Σ a (possible infinite) set of words over Δ .

Extension: by element-wise application:

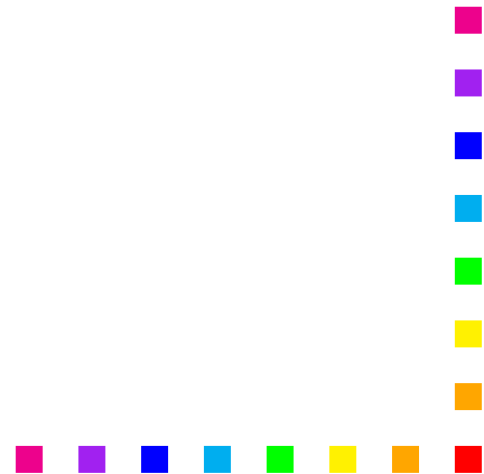
$\sigma(\epsilon) = \epsilon$; for all $x = ay$, $a \in \Sigma$, $y \in \Sigma^*$,

$\sigma(x) = \sigma(a) \circ \sigma(y)$



Note

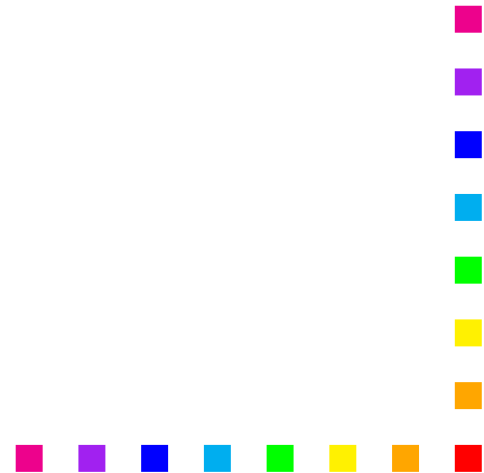
A homomorphism $h : \Sigma \rightarrow \Delta^*$ is a particular case of a substitution $\sigma : \Sigma \rightarrow \mathcal{P}(\Delta^*)$ where $|\sigma(x)| = 1$ for all $x \in \Sigma$.



Operations on languages

For $L \subseteq \Sigma^*$, $L = \{x_1, x_2, x_3, \dots\}$:

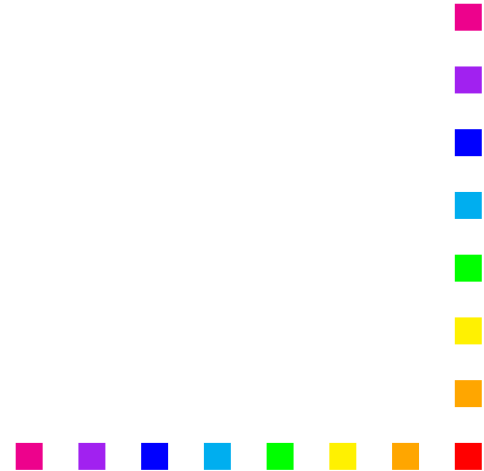
1. $\sigma(L) = \sigma(x_1) \cup \sigma(x_2) \cup \sigma(x_3) \cup \dots$
2. σ is regular if $\sigma(a)$ is regular for each $a \in \Sigma$;
3. σ is context-free if $\sigma(a)$ is a CFL for each $a \in \Sigma$.



Fact

For each $X, Y \subseteq \Sigma^*$, and for each substitution (homomorphism) $\sigma : \Sigma \rightarrow \Delta^*$:

1. $\sigma(X \cup Y) = \sigma(X) \cup \sigma(Y)$
2. $\sigma(X \circ Y) = \sigma(X) \circ \sigma(Y)$
3. $\sigma(X^*) = (\sigma(X))^*$

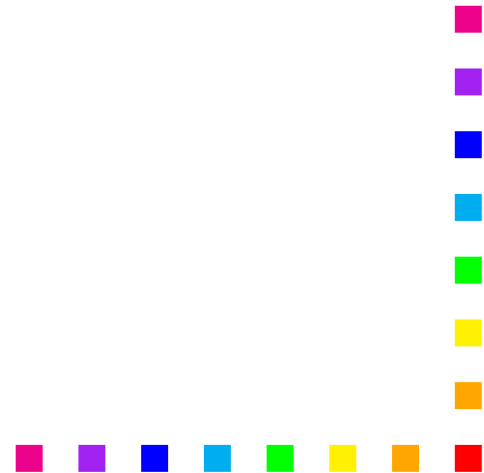


Theorem 2.39 (Fleck)

The class of CFL is closed under substitution and homomorphism.

Proof: by construction.

Consider a CFL $L \subseteq \Sigma^*$ and $\sigma : \Sigma \rightarrow \mathcal{P}(\Delta^*)$ a CF substitution.



Proof, continuation

1. There exist a CFG $G = (V, \Sigma, P, S)$ and for each $a \in \Sigma$, $G_a = (V_a, \Delta, P_a, S_a)$ with $L(G) = L$, and $L(G_a) = \sigma(a)$
2. We may assume without loss of generality that $V \cap (\cup_{a \in \Sigma} V_a) = \emptyset$ and $\cap_{a \in \Sigma} V_a = \emptyset$
3. Construct $G' = (V', \Delta, P', S)$, where $V' = V \cup (\cup_{a \in \Sigma} V_a)$, $P' = P_\sigma \cup (\cup_{a \in \Sigma} P_a)$ where P_σ consists of P where each terminal $a \in \Sigma$ is replaced by S_a .
4. G' is a CFG and derivations initiated in P_σ are strings in $L(G)$ where each terminal $a \in \Sigma$ is replaced by S_a .
5. Derivations in (4) may be continued with rules in P_a , $a \in \Sigma$ thus obtaining $L(G') = \sigma(L(G)) = \sigma(L)$



Theorem 2.40 (Fleck)

If $L \subseteq \Sigma^*$ is a CFL and $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ is a transducer then $M(L) = \{\rho(q_0, x) \mid x \in L\} \subseteq \Delta^*$ is a CFL.

Proof idea: construct PDA B to accept $M(L)$ from DGSM M and PDA A that accepts L

1. Embed both M and A as sub-machines of B
2. Generates hypothetical input sequences for A and M
3. Monitor computations of M and A so that when A accepts an input and M 's output for it is B 's input, B accepts



Most general CFL

- Dyck languages are the most general CFL and are defined over alphabets with an even number of letters that are explicitly paired;
- For $k > 0$ we assume an alphabet where pairing is explicitly given: $\Delta_k = \{(1)_1, (2)_2, \dots, (k)_k\}$
- Dyck language D_k over Δ_k is the language of the CFG with one non-terminal, X , and $k + 1$ productions, $X \rightarrow \epsilon$ and $X \rightarrow X(iX)_iX$, for $i = 1, 2, \dots, k$.

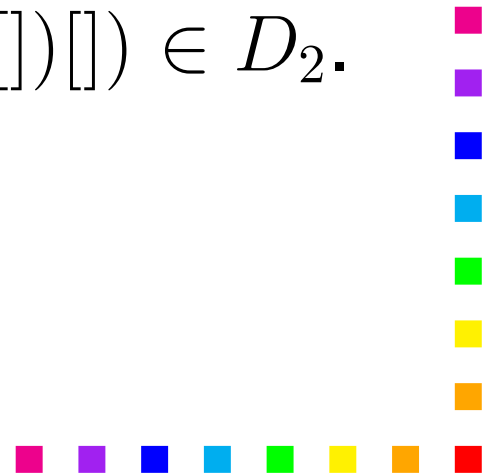


Intuitive interpretation

- Given k pairs of different parentheses, the Dyck language is the collection of all well-formed parenthesis sequences.

Example:

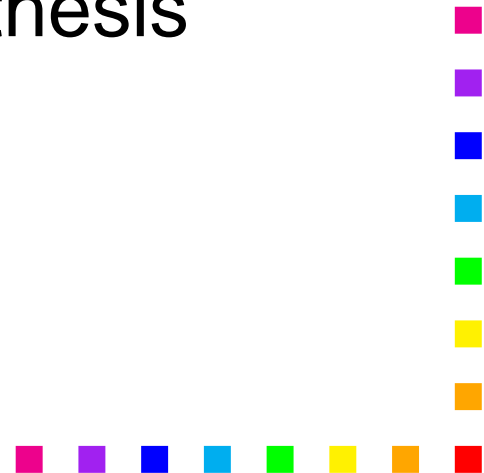
for $k = 2$, $\Delta_2 = \{ (,), [,] \}$ and $()[[()]]$, $(([])) \in D_2$.



Well-formed string

The property that characterizes a well-formed parenthesis string:

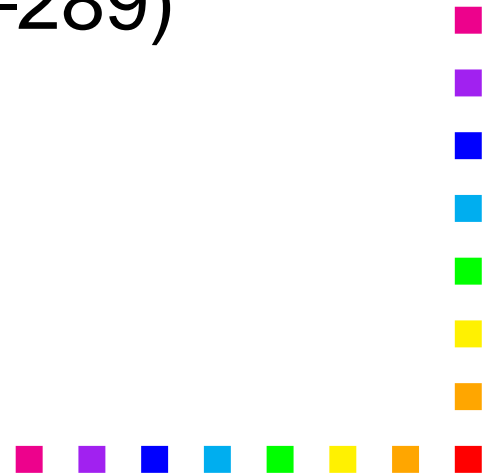
1. each left parenthesis in string is eventually followed by a matching right parenthesis;
2. each matched left and right parenthesis encloses a well-formed string.



Theorem 2.41 (Fleck)

A language L is a CFL iff there is a Dyck language $D \subseteq \Sigma^*$, a regular language $R \subseteq \Sigma^*$, and a homomorphism $h : \Sigma \rightarrow \Sigma^*$ so that $L = h(D \cap R)$.

Proof: by construction (see Fleck 288–289)



Using closure operators

We illustrate here the usage of closure operators solving the problem 2.18 from the textbook.

Problem 2.18 Let C be a context free language and R be a regular language.

1. Prove that $C \cap R$ is context-free.
2. Use the result at (1) to show that the language $L = \{w \mid w \in \{a, b, c\}^* \text{ contains an equal number of } a\text{'s, } b\text{'s, and } c\text{'s}\}$ is not a context-free language.



Part 1

Let C be context-free and R be regular. Consider

$P = (Q_C, \Sigma, \Gamma, \delta_C, q_C^0, F_C)$ the PDA that recognizes C and

$D = (Q_R, \Sigma, \delta_R, q_R^0, F_R)$ the DFA that recognizes R .

Construct $P' = (Q', \Sigma, \Gamma, \delta', q', F')$ that recognizes $C \cap R$ as follows:

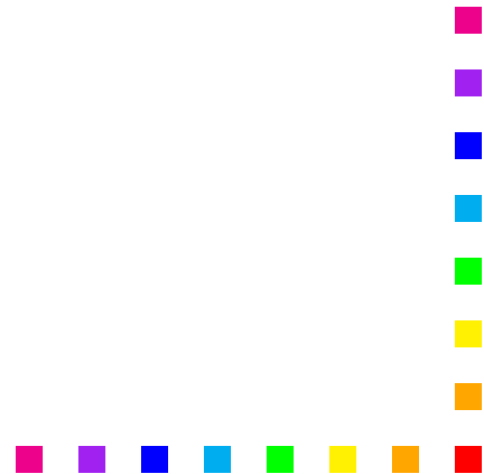
1. $Q' = Q_C \times Q_R$
2. P' does whatever P does, keeping track of the states of D ,
i.e., $\forall q_C \in Q_C, q_R \in Q_R, a \in \Sigma_\epsilon, s \in \Gamma_\epsilon$,
$$\delta'((q_C, q_R), a, s) = (\delta_C(q_C, a, s), \delta_R(q_R, a))$$
3. $q' = (q_C^0, q_R^0)$,
4. $F' = F_C \times F_D$ (i.e., P' stops only at a state $q \in F_C \times F_R$).



Conclusion

Since P' recognizes $C \cap R$ it follows that $C \cap R$ is context-free

Note: this is a model for the construction of a machine that simulates the working of other (two in our case) machines.



Part 2

Consider the regular language $R = a^*b^*c^*$ and let $L' = L \cap R$. If L would be context-free then according to Part 1 we would have L' a context-free language. But we know that $L' = \{a^n b^n c^n \mid n \geq 0\}$ is not a context-free language. Hence, L is not a context-free language.

Note: we will use pumping lemma for CFL to prove that

$L' = \{a^n b^n c^n \mid n \geq 0\}$ is not context free.



Pumping lemma for CFL

- The second mechanism for proving that a given language is not context-free is pumping-lemma for CFL;
- This mechanism is similar to the pumping lemma used for proving that a given language is not regular;
- Here we present a similar lemma for context-free languages.



Informal

- Pumping lemma for context-free languages states that every CFL has a specific value called *pumping length* such that all longer strings in the language can be pumped;
- However, the meaning of pumping is a bit more complex than in case of regular languages;
- Here pumping means that a string can be divided into five parts so that the second and fourth parts may be repeated any number of times and the resulting string is in the language.



Pumping lemma for CFL

Theorem 2.34: If A is a context-free language, then there is a number p (the pumping length) where, if $s \in A$ and $|s| \geq p$, then s may be divided into five pieces, $s = uvxyz$ satisfying the conditions:

1. For each $i \geq 0$, $uv^i xy^i z \in A$
2. $|vy| > 0$
3. $|vxy| \leq p$



Interpretation

- Condition 1 states that length of strings in A can be unlimited but have a fixed structure $uv^i xy^i z, i \geq 0$;
- Condition 2 says that in the structure $uvxyz$ either v or y is not empty; otherwise theorem would be trivially true ($\forall i \geq 0, u\epsilon^i x \epsilon^i z = uxz$);
- Condition 3 states that pieces v, x, y together have length at most p (this is useful for proving that some languages are not context-free).



Proof idea

Let A be a CFL and G be the CFG generating A . We must show that any sufficiently long $s \in A$ can be pumped and remains in A .

Proving steps:

1. Because $s \in A$, s is derivable from G and has a derivation tree D_s ;
2. Derivation tree D_s of s must be very tall because s is very long;
3. This means that D_s contains some long path from start variable at the root to one of terminal symbols at a leaf;
4. On this long path some variable symbol X must be repeated because of pigeonhole principle.



Observations

1. Repetition of X in D_s allows us to replace the subtree under the second occurrence of X with the subtree under the first occurrence of X and still get a legal derivation tree;
2. Hence, we may cut s into five pieces, as shown in Figures 1, and we may repeat the second and fourth piece and obtain a string $uv^i xy^i z \in A$, for any $i \geq 0$.



Very tall trees

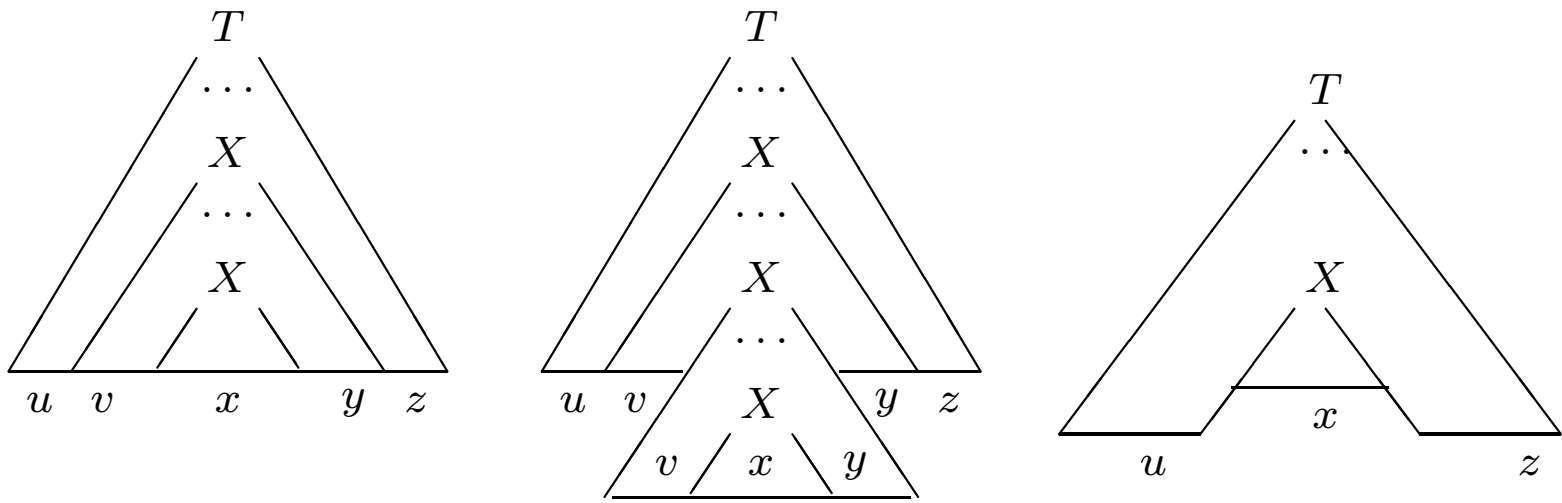


Figure 1: Tall derivation trees



Formal Proof

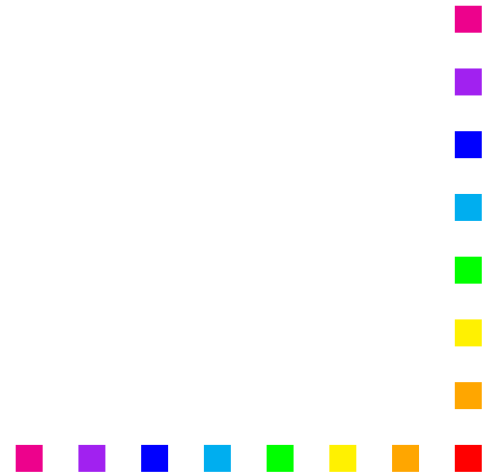
Let $G = (V, \Sigma, R, S)$ be a CFG that generates A .
Let $b \geq 2$ be the maximum number of symbols in $rhs(r)$, $r \in R$, i.e., $b = \max\{|rhs(r)| \mid r \in R\}$.

- In any derivation tree D using G we know that a node cannot have more than b children; i.e., in a derivation starting with S :
at most b leaves are in step 1;
at most b^2 leaves in step 2;
...
at most b^h in step h ;
- If the height of D is at most h the length of the string generated is at most b^h ;



Choosing p

If $|V|$ is the number of variables in G ,
then we set p to $b^{|V|+1}$.



Observation

- If $s \in A \wedge |s| \geq p$, D_s must be at least $|V| + 1$ high because $b \geq 2$ and thus $b^{|V|+1} \geq b^{|V|} + 1$.

Conclusion: the derivation tree of any string from A of length at least p requires a derivation tree of height at least $|V| + 1$.



Formal proof, continuation

Let $s \in A$, $|s| \geq p$. We show how to pump s .

- Let D_s be the derivation tree of s ; if s has several derivation trees we choose D_s to be the tree with the smallest number of nodes;
- The longest path in D_s has at least $|V| + 1$ nodes labeled by variables because only the leaf is a terminal;
- Since there are only $|V|$ variables in V , some variable X appear more than once on the longest path in D_s .

Assume that X is the closest variables to the leaf that repeat on this path, Figure 2.



Repeating X in D_S

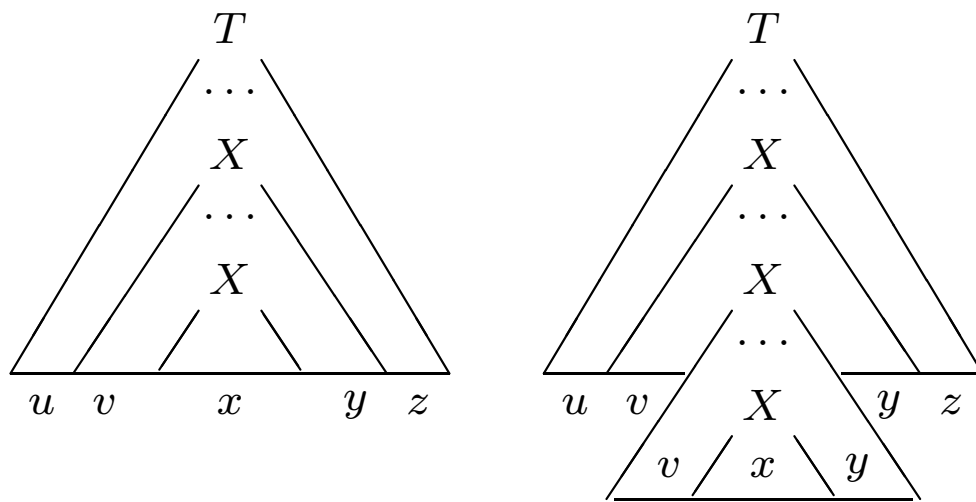


Figure 2: Repeating nonterminal



Formal proof, continuation

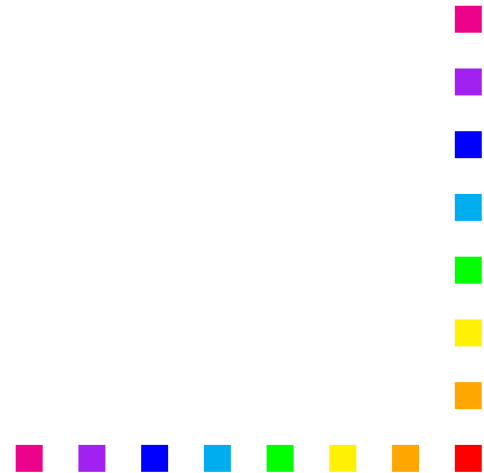
Now we divide s into $uvxyz$ according to Figure 2.

- Each occurrence of X has a subtree that generates a portion of s ;
- The upper occurrence of X has the larger subtree and generates vxy , whereas the lower occurrence generates just x ;
- Since both trees mentioned above are generated by X we may substitute one for the other and still obtain a valid derivation tree;
- Replacing the smaller tree by the larger repeatedly, gives derivation trees for the string $uv^i xy^i z$, at each $i > 1$;
- Replacing the larger tree by the smaller one generates the string $uv^{i-1} xy^{i-1} z$, at each $i > 1$.



Condition 1

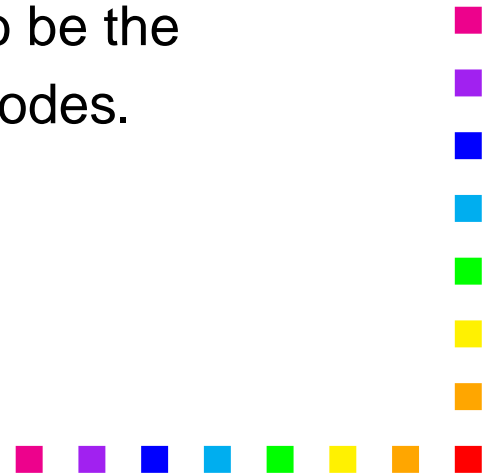
$uv^i xy^i z \in A$ for $i \geq 0$ is thus established.



Condition 2

To get condition 2 we must ensure that not both v and y are ϵ .

- If both v and y were ϵ the derivation tree obtained by substituting the smaller tree for the larger tree would have fewer nodes than D_s and would still generate s ;
- This is impossible because we have chosen D_s to be the derivation tree for s with the smallest number of nodes.



Condition 3

Now we need to be sure that $|vxy| \leq p$.

- In the derivation tree D_s the upper occurrence of X generates vxy ;
- We chose X so that both occurrences fall within the bottom $|V| + 1$ variables on the path and we chose the longest path in D_s ;
- Hence, the subtree where X generates vxy is at most $|V| + 1$ high;
- A tree of this height can generate a string of length at most $b^{|V|+1} = p$.



Example 1

Use pumping lemma to show that the language $B = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

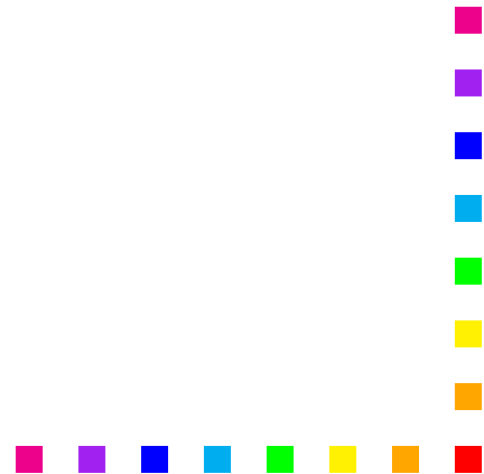
Proof: assume that B is context-free to obtain a contradiction.

- Let p be the pumping length for B that is guaranteed to exist by pumping lemma;
- Consider the string $s = a^p b^p c^p \in B$ of length at least p ;
- The pumping lemma states that s can be pumped and here we show that it cannot be pumped.



Contradiction

We show that no matter how we divide s into $uvxyz$ one of the three conditions of the pumping lemma for CFL is violated.



Checking condition 2

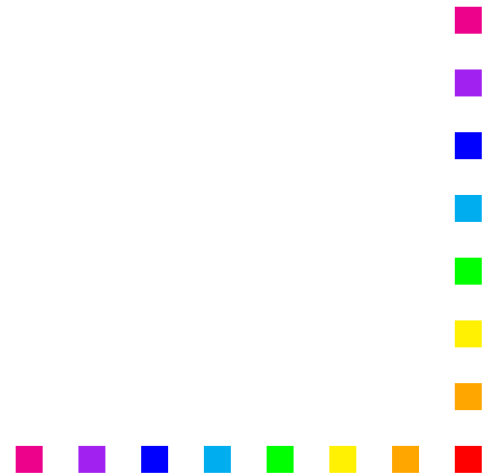
Condition 2 stipulates that either v or y is not empty. There are two cases to examine:

1. When both v and y contain only one type of symbols (a,b,c) v does not contain both a's and b's or both b's and c's; the same hold for y . In this case uv^2xy^2z cannot contain equal number of a's, b's and c's., hence it cannot be in B which violates condition 1 of the lemma;
2. When either v or y contain more than one type of symbols (a,b,c) uv^2xy^2z may contain equal numbers of a's, b's, c's but they don't come in the right order. Hence, it cannot be in B either.



Fact

One of the cases enumerated before must occur. Because both cases result in contradiction, a contradiction is unavoidable so the assumption " B is a CFL" must be false.



Example 2

Use pumping lemma to show that $C = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$ is not a context-free language.

Proof: assume that C is CFL and try to obtain a contradiction. Let p be the pumping length and $s = a^p b^p c^p$. We will try to pump it down and pump it up.

Let $s = uvxyz \in C$ and again consider two cases

1. Both v and y contain only one of the symbols a, b, c ;
2. Either v or y contain more than one of symbols a, b, c .



Case 1

Because v or y contains one symbol, one of the symbols a, b, or c does not appear in v or y .

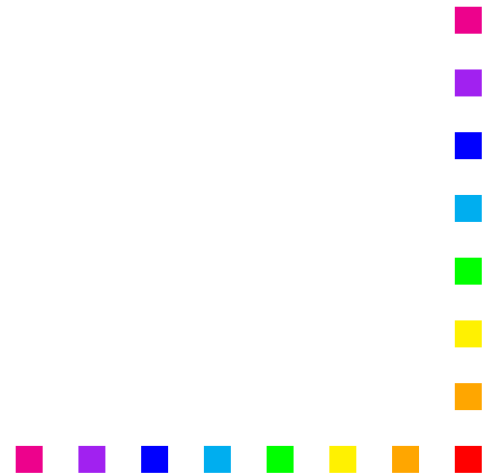
1. The a's do not appear in v and y . Pumping s down we get the string $uv^0xy^0z = uxz$ which contains the same number of a's as s but it contain fewer b's or fewer c's. Therefore $uxz \notin C$.
2. The b's do not appear in v and y . Since not both v, y may be ϵ a's or c's must appear in v or y . If a appears, uv^2xy^2z contains more a's than b's so $uv^2xy^2z \notin C$; if c appears, uv^0xy^0z contains more b's than c's so $uv^0xy^0z \notin C$. Either way we obtain a contradiction.
3. The c's do not appear. Then uv^2xy^2z contains more a's or b's than c's so it is not in C , and a contradiction occurs.



Case 2

When either v or y contain more than one of a, b, c , then uv^2xy^2z will not contain the symbols a, b, c in the correct order. Hence it cannot be a member of C and a contradiction occurs.

Conclusion : s cannot be pumped in violation of pumping lemma and C is not a CFL.



Example 3

Use pumping lemma to show that

$D = \{ww \mid w \in \{0, 1\}^*\}$ is not a CFL

Proof: assume that D is CFL and try to obtain a contradiction. Let p be the pumping length given by pumping lemma. However, here the choosing of s is less obvious.

1. Try $s = 0^p 1 0^p 1$. If we may chose: $u = 0^{p-1}$, $v = 0$, $x = 1$, $y = 0$, $z = 0^{p-1} 1$ we can see that s can be pumped.
2. Another candidate is $s = 0^p 1^p 0^p 1^p$. We shows that this string cannot be pumped using condition 3 of the pumping lemma.

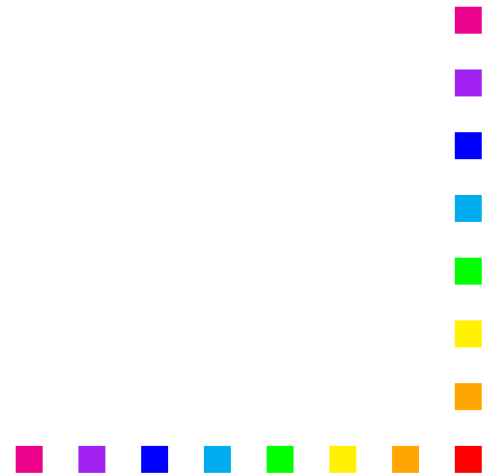


Using $s = 0^p 1^p 0^p 1^p$

- Assume that s can be pumped and set $s = uvxyz$ where $|vxy| \leq p$.
- String vxy must contain the midpoint of s . If vxy would occur only in the first half of s , pumping uv^2xy^2z moves a 1 into the first position of the second half and so it cannot be of the form ww .
- Similarly if vxy is in the second part of s , pumping uv^2xy^2z moves a 0 into the last position of the first half of s so it cannot be of the form ww .
- If vxy contains the midpoint of s , when we pump s down to $uv^0xy^0z = uxz$, it has the form $0^p 1^i 0^j 1^p$ where i and j cannot be both p because then both v and y would have to be empty thus violating $|vy| > 0$. Hence this string is not of the form ww either.

Conclusion

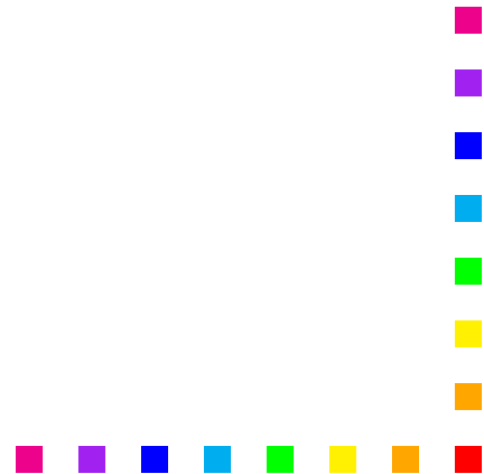
s cannot be pumped and thus D is not a CFL.



Example 4

$L = \{a^p b^q c^p d^q \mid p, q \geq 0\}$ is not context free.

Note: A PDA would try to match $a^p b^q$ with $c^p d^q$. Because this need two comparisons that interfere with one another, they cannot be accomplished with a stack.



Using pumping lemma

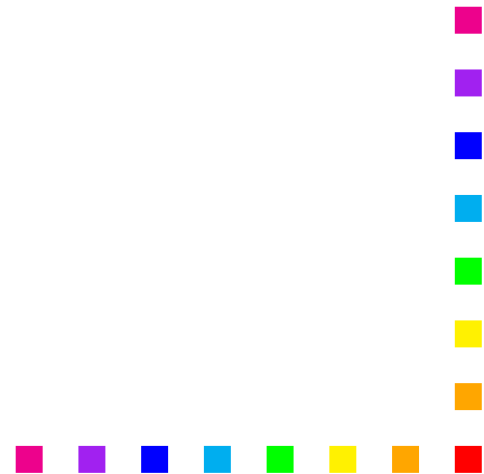
Assume that L is CF and let p be its pumping length. Consider the string $s = a^p b^p c^p d^p \in L$, $|s| = 4p \geq p$, Then by the pumping lemma there exists u, v, x, y, z such that $s = uvxyz$ and $uv^i xy^i z \in L$ for any $i \geq 0$, $|vy| \geq 1$, and $|vxy| \leq p$. Trying to pump s we examine the following cases:

- v or y contain two or more different letters.
- v or y contain just one letter.



Case 1: 2 or more letters

1. Suppose that v contains both a's and b's, $v = a^r b^s$, $r, s \geq 1$. Then $s = uv^2xy^2z = ua^r b^s a^r b^s xy^2z$ and by pumping lemma, $s \in L$.
2. But note, s has a' following b' contradicting the definition of L .
3. Similarly, assuming that y contains 2 or more letters we reach a similar contradiction.



v or y contain one letter only

1. Suppose $v \in a^*$. Since $|vxy| \leq p$, $y \in a^* \cup b^*$. Then $uv^2xy^2x \in L$ by the pumping lemma. But it has either more a's than c's or more b's than d's or both, since $|vy| \geq 1$, contradicting definition of L ;
2. Suppose $v \in b^*$. Since $|vxy| \leq p$, $y \in b^* \cup c^*$. Then $uv^2xy^2x \in L$ by the pumping lemma. But it has either more b's than d's or more c's than a's or both, since $|vy| \geq 1$, again contradicting the definition of L ;
3. Suppose that $v \in c^*$. This can be analyzed as above and the same contradiction is generated.

Conclusion: L cannot be CF.

