Lecture 23: Repetition in context free languages

20 April 2010

1 Generating new words

We are interested in phenomena of repetition in context free languages. We had seen that regular languages repeat themselves if the strings are sufficiently long. We would like to make a similar statement about regular languages, but unfortunately, while the general statement is correct, the details are somewhat more involved.

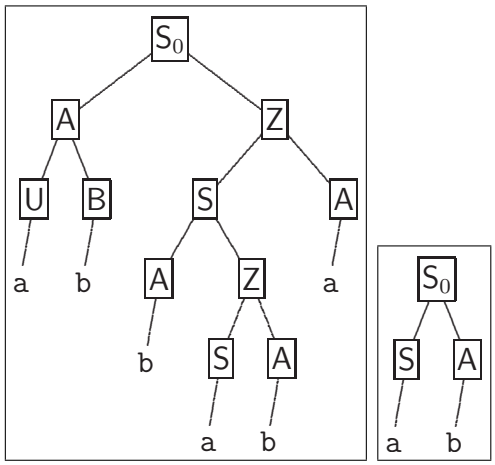
1.1 Example of repetition

As a concrete example, consider the following context-free grammar which is in Chomsky normal form (CNF). (We remind the reader that any context free grammar can be converted into CNF, as such assuming that we have a grammar in CNF form does not restrict our discussion.)

As a concrete example, consider the following grammar from the previous lecture:

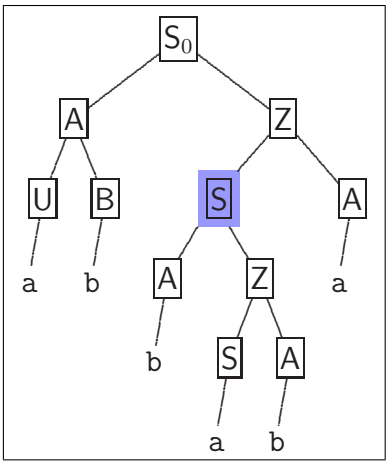
(G5)

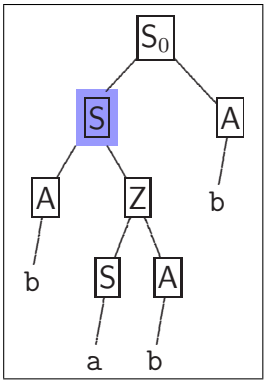
$$
\begin{array}{ll}
\Rightarrow & S_0 \to AZ \mid UB \mid a \mid SA \mid AS \\
 & S \to AZ \mid UB \mid a \mid SA \mid AS \\
 & A \to b \mid AZ \mid UB \mid a \mid SA \mid AS \\
 & B \to b \\
 & U \to a \\
 & Z \to SA
\end{array}
$$



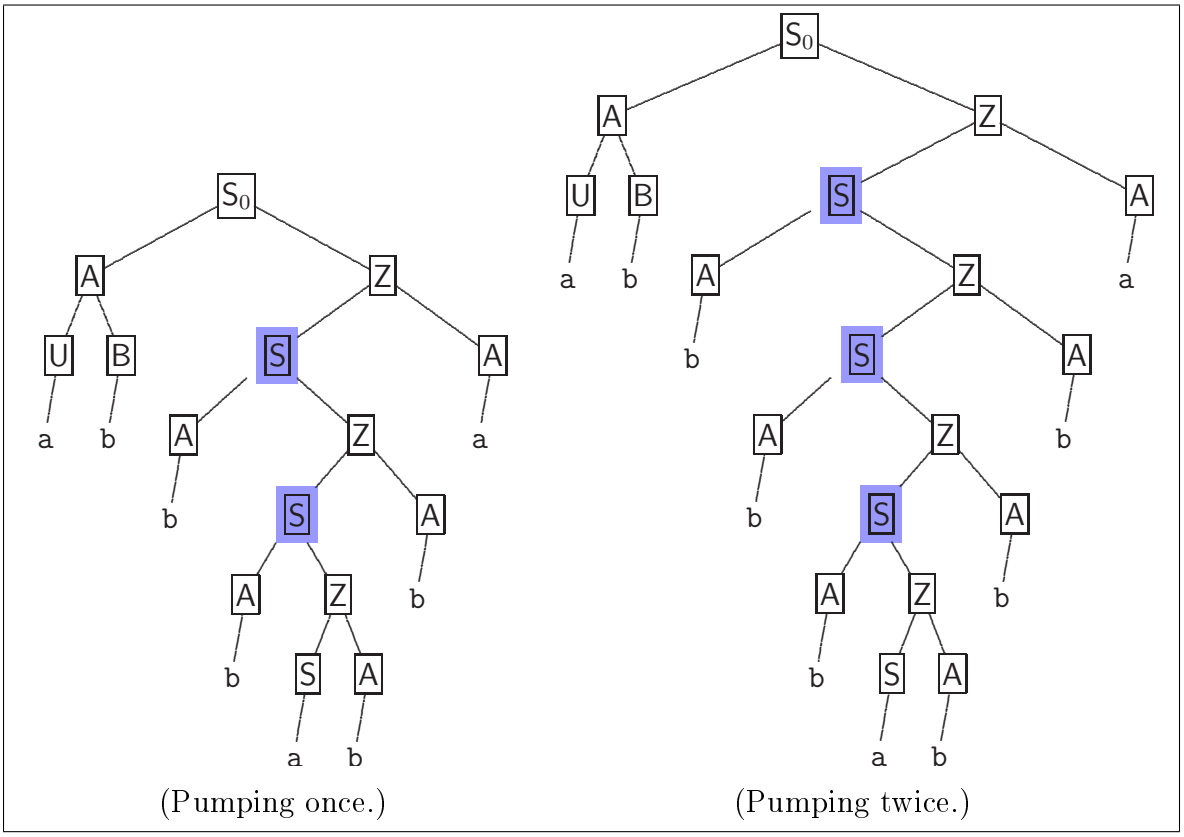
Next, consider the two words created from this grammar, as depicted on the right.

What if we wanted to make the second word longer without thinking too much? Well, both parse trees have subtrees with nodes generated by the variable S. As such, we could just cut the subtree of the first word using the variable S, and replace the subtree of S in the second word by this subtree, which would like the following:





Even more interestingly, we can do this cut and paste on the original tree:



Naturally, we can repeat this pumping operation (cutting and pasting a subtree) as many times as want, see for example Figure 1. In particular, we get that the word $\mathtt{abb}^i\mathtt{ab}^i\mathtt{a}$, for any i, is in the language of the grammar (G5). Notice that unlike the pumping lemma for regular languages, here the repetition happens in two places in the string. We claim that such a repetition (in two places) in the word must happen for any context free language, once we take a word which is sufficiently long.

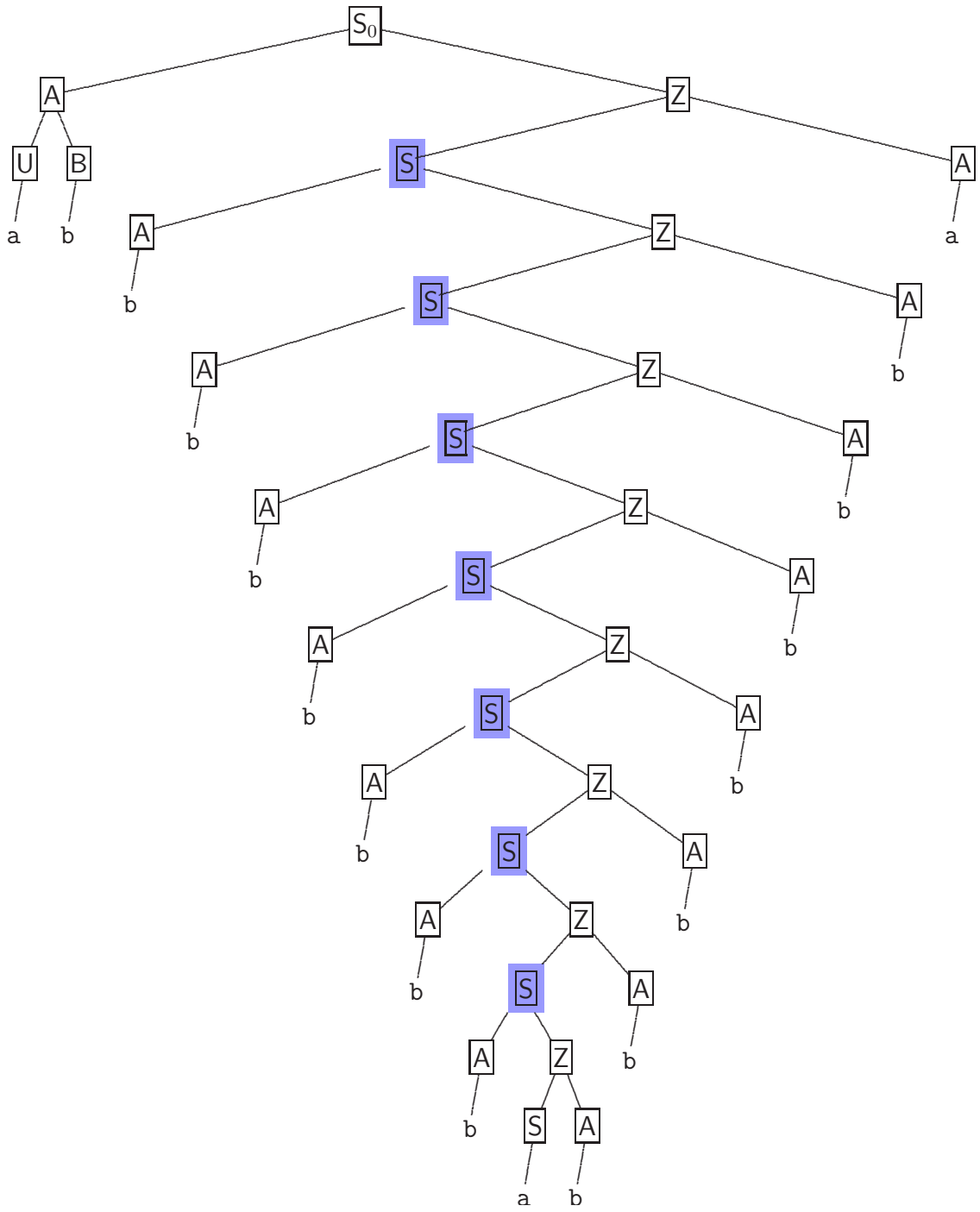


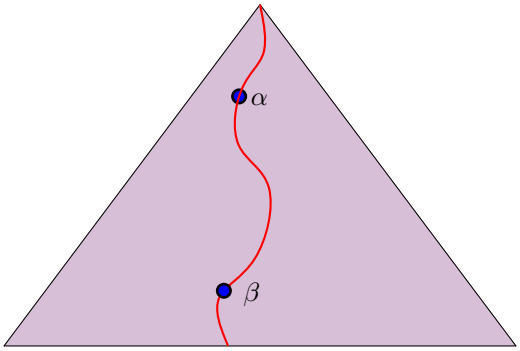
Figure 1: Naturally, one can pump the string as many times as one want, to get a longer and longer string.

2 The pumping lemma for CFG languages

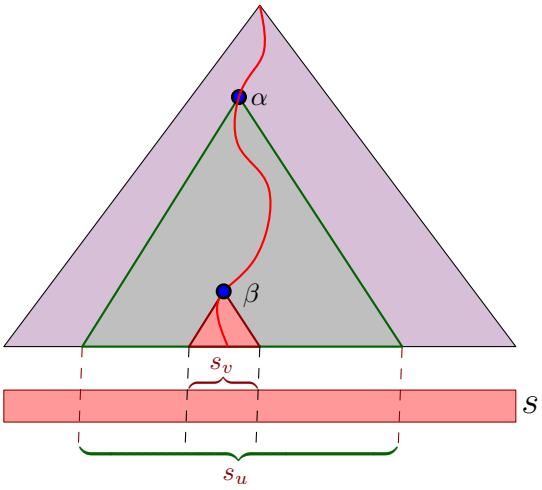
So, assume we are given a context free grammar $\mathcal{G}$ which is in CNF, and it has m variables.

2.1 If a variable repeats

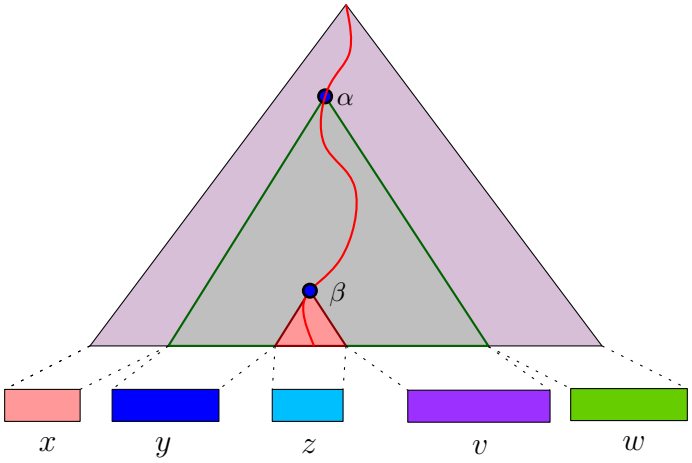
So, assume we have a parsing tree T for a word s (where the underlying grammar is in CNF), and there is a path in T from the root to a leaf, such that a variables repeats twice. So, say nodes α and β have the same variable (say S) stored in them:



The subtrees rooted at α and β corresponds to substrings of s:

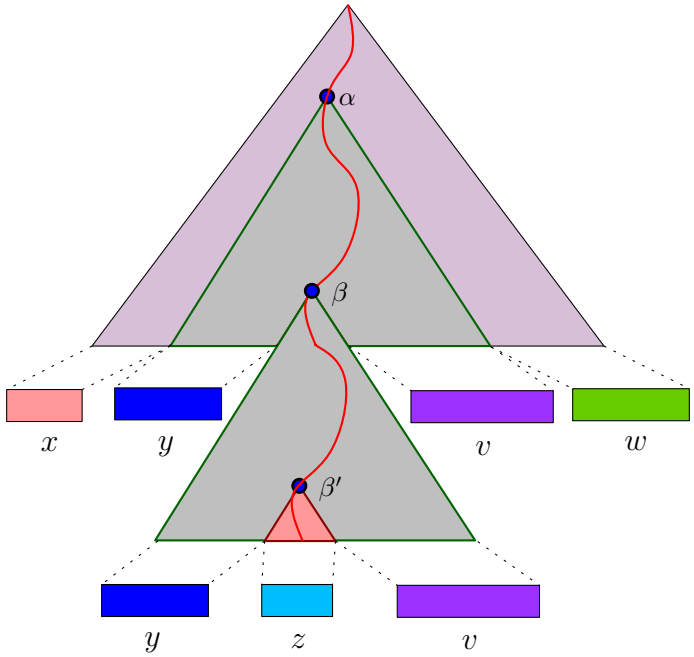


In particular, the substrings s_u and s_v break s into 5 parts:



Namely, s can be written as $s = xyzvw$.

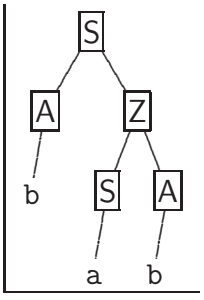
Now, if we copy the subtree rooted at α and copy it to β, we get a new parse tree:



The new tree is much bigger, and the new string it represents is $s = xyyzvvw$. In general, if we do this cut & paste operation $i - 1$ times, we get the string

$$xy^i zv^i w.$$

2.2 How tall the parse tree have to be?



We will refer to a parse generated from a context free grammar in CNF form as a ***CNF tree***. A CNF tree has the special property that the parent of a leaf as a single child which is a terminal. The ***height*** of a tree is the maximum number of edges on a path from the root of the tree to a leaf. Thus, the tree depicted on the right has height 3.

The grammar $\mathcal{G}$ has m variables. As such, if the parse tree T has a path π from the root of length k, and $k > m$ (i.e., the path has k edges), then it must contain at least $m + 1$ variables (the last edge is between a variable and a terminal). As such, by the pigeon hole principle, there must be a repeated variable along π. In particular, a parse tree that does not have a repeated variable have height at most m.

Since $\mathcal{G}$ is in CNF, its a binary tree, and a variable either has two children, or a single child which is a leaf (and that leaf contains a single character of the input). As such, a tree of height at most m, contains at most 2^m leaves[1], and represents as such a string of length at most 2^m.

[1] In fact, a CNF tree of height m can have at most 2^{m-1} leaves (figure out why), but thats a subtlety we will ignore that anyway works in our favor.

We restate the above observation formally for the record.

Observation 2.1 *If a CNF parse tree (or a subtree of such a tree) has height h, then the string it generates is of length at most 2^h.*

Lemma 2.2 *Let $\mathcal{G}$ be a grammar given in Chomsky Normal Form (CNF), and consider a word $s \in \mathsf{L}(\mathcal{G})$, such that $\ell = |s|$ is strictly larger than 2^m (i.e., $\ell > 2^m$). Then, any parse tree T for s (generated by $\mathcal{G}$) must have a path from the root to some leaf with a repeated variable on it.*

Proof: Assume for the sake of contradiction that T has no repeated variable on any path from the root, then the height of T is at most m. But a parse tree of height m for a CNF can generate a string of length at most 2^m. A contradiction, since $\ell = |s| > 2^m$. ■

2.3 Pumping Lemma for CNF grammars

We need the following observation.

Lemma 2.3 (CNF is effective.) *In a CNF parse tree T, if u and v are two nodes, both storing variables in them, and u is an ancestor of v, then the string S_u generated by the subtree of u is strictly longer than the substring S_v generated by the subtree of u. Namely, $|S_u| > |S_v|$. (Of course, S_v is a substring of S_u.)*

Proof: Assume that the node u stores a variable X, and that we had used the rule $\mathsf{X} \rightarrow \mathsf{BC}$ to generate its two children u_L and u_R. Furthermore, assume that u_L and u_R generated the strings S_L and S_R, respectively. The string generated by v must be a substring of either S_L or S_R. However, CNF has the property that no variable[2] can generate the empty word ϵ. As such $|S_R| > 0$ and $|S_L| > 0$.

In particular, assume without loss of generality, that v is in the left subtree of u, and as such S_v is a substring of S_L. We have that

$$|S_v| \leq |S_L| < |S_L| + |S_R| = |S_u| .$$

■

Lemma 2.4 *Let T be a tree, and π be the longest path in the tree realizing the height h of T. Fix $k \geq 0$, and let u be the kth node from the end of π (i.e., u is in distance $h - k$ from the root of T). Then the tree rooted at u has height at most k.*

Proof: Let r be the root of T, and assume, for the sake of contradiction, that T_u (i.e., the subtree rooted at u) has height larger than k, and let σ be the path from u to the leaf γ of T_u realizing this height (i.e., the length of σ is $> k$). Next, consider the path formed by concatenating the path in T from r to u with the path σ. Clearly, this is a new path of length $h - k + |\sigma| > h$ that leads from the root of T into a leaf of T. As such, the height of T is larger than h, which is a contradiction. ■

[2]Except the start variable, but this not relevant here.

Lemma 2.5 (Pumping lemma for Chomsky Normal Form (CNF).) *Let $\mathcal{G}$ be a CNF context-free grammar with m variables in it. Then, given any word $\mathcal{S}$ in $\mathsf{L}(\mathcal{G})$ of length $> 2^m$, one can break $\mathcal{S}$ into 5 substrings $\mathcal{S} = xyzvw$, such that for any $i \geq 0$, we have that xy^izv^iw is a word in $\mathsf{L}(\mathcal{G})$. In addition, the following holds:*

1. *The strings y and v are not both empty (i.e., the pumping is getting us new words).*
2. *$|yzv| \leq 2^m$.*

Proof: Let T be a CNF parse tree for $\mathcal{S}$ (generated by $\mathcal{G}$). Since $\ell = |s| > 2^m$, by Lemma 2.2, there is a path in T from its root to a leaf which has a repeated variable (and its length is longer than m). In fact, let π be the longest path in T from the root to a leaf (i.e., π is the path realizing the height of the tree T). We know that T has more than $m+1$ variables on it and as such it has a repetition.

We need to be a bit careful in picking the two nodes α and β on π to apply the pumping to. In particular, let α be the last node on π such that there is a repeated appearance of the symbol stored in u later in the path. Clearly, the length of the subpath τ of π starting at α till the end of π has at most m symbols on it (because otherwise, there would be another repetition on π). Let β be the node of $\tau \subseteq \pi$ which has repetition of the symbol stored in α.

By Lemma 2.4 the subtree T_α (i.e., the subtree of T rooted at α) has height at most m. As above, T_α and T_β generate two strings $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$, respectively. By Observation 2.1, we have that $|\mathcal{S}_\alpha| \leq 2^m$. By Lemma 2.3, we have that $|\mathcal{S}_\alpha| > |\mathcal{S}_\beta|$. As such, the two substrings $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ breaks $\mathcal{S}$ into 5 substrings $\mathcal{S} = xyzvw$. Here, we have

$$\mathcal{S} = x \overbrace{y \underbrace{z}_{=\mathcal{S}_\beta} v}^{\mathcal{S}_\alpha =} w.$$

As such, we know that $|yv| = |\mathcal{S}_\alpha| - |\mathcal{S}_\beta| > 0$. Namely, the strings y and v are not both empty. Furthermore, $|yzv| = |\mathcal{S}_\alpha| \leq 2^m$.

The remaining task is to show the pumping. Indeed, if we replace T_β by the tree T_α we get a parse tree generating the string xy^2zv^2w. If we repeat this process $i-1$ times, we get the word

$$xy^izv^iw \in \mathsf{L}(\mathcal{G}),$$

for any i, establishing the lemma. ∎

Lemma 2.6 (Pumping lemma for context-free languages.) *If L is a context-free language, then there is a number p (the pumping length) where, if $\mathcal{S}$ is any string in L of length at least p, then $\mathcal{S}$ may be divided into five pieces $\mathcal{S} = xyzvw$ satisfying the conditions:*

1. *for any $i \geq 0$, we have $xy^izv^iw \in L$,*
2. *$|yv| > 0$,*
3. *and $|yzv| \leq p$.*

Proof: Since L is context free it has a CNF grammar $\mathcal{G}$ that generates it. Now, if m is the number of variables in $\mathcal{G}$, then for $p = 2^m + 1$, the lemma follows by Lemma 2.5. ■

We now prove a corollary to the above lemma, which is easy to show, and which is easier to use to show languages are not context-free.

Lemma 2.7 *If L is a context-free language, then there is a number n such that, for any word $\mathcal{S}$ in L of length at least n, $\mathcal{S}$ can be divided into three words $\mathcal{S} = xtw$, where $|t| \leq n$, and there exists a strict contiguous substring t' of t such that $xt'z \in L$.*

Proof: Choose n to be p, where p is the number assured by the pumping lemma for CFLs. Let $\mathcal{S}$ in L of length at least n. Then by the pumping lemma, there is a split $\mathcal{S} = xyzvw$ such that $|yv| > 0$, $|yzv| \leq p$, and for any $i \geq 0$, $xy^izv^iw \in L$. In particular, for $i = 0$, $xzw \in L$.

Now, choose $t = yzv$ and $t' = z$. Then $\mathcal{S} = xtw$, t' is a strict contiguous substring of t (since $|yv| > 0$), $|t| \leq n$, and $xt'w \in L$. ■

Intuitively, the above lemma says that if L is a CFL, then there is an n such that if w is a word in L of length greater than n, then there is a small substring t of z (of length at most n) that can be *contracted* by replacing t with a smaller string t' that is a contiguous substring of t.

3 Languages that are not context-free

3.1 The language $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ is not context-free

Lemma 3.1 *The language $L = \left\{ \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \;\middle|\; n \geq 0 \right\}$ is not context-free.*

Proof: We give two proofs. The first is based on the pumping lemma, and the second based on the corollary to the pumping lemma.
Proof I: Using the pumping lemma:
Assume, for the sake of contradiction, that L is context-free, and apply the Pumping Lemma to it (Lemma 2.6). As such, there exists $p > 0$ such that any word in L longer than p can be pumped. So, consider the word $\mathcal{S} = \mathtt{a}^{p+1}\mathtt{b}^{p+1}\mathtt{c}^{p+1}$. By the pumping lemma, it can be written as $\mathtt{a}^{p+1}\mathtt{b}^{p+1}\mathtt{c}^{p+1} = xyzvw$, where $|yzv| \leq p$.

We claim, that yzv can made out of only two characters. Indeed, if yzv contained all three characters, it would have to contain the string $\mathtt{b}^{p+1}$ as a substring (as $\mathtt{b}^{p+1}$ separates all the appearances of $\mathtt{a}$ from all the appearances of $\mathtt{c}$ in $\mathcal{S}$). This would require that $|yzv| > p$ but we know that $|yzv| \leq p$.

In particular, let $i_\mathtt{a}, i_\mathtt{b}$ and $i_\mathtt{c}$ be the number of $\mathtt{a}$s, $\mathtt{b}$s and $\mathtt{c}$s in the string yv, respectively. All we know is that $i_\mathtt{a} + i_\mathtt{b} + i_\mathtt{c} = |yv| > 0$ and that $i_\mathtt{a} = 0$ or $i_\mathtt{c} = 0$. Namely, $i_\mathtt{a} \neq i_\mathtt{b}$ or $i_\mathtt{b} \neq i_\mathtt{c}$ (the case $i_\mathtt{a} \neq i_\mathtt{c}$ implies one of these two cases). In particular, by the pumping lemma, the word

$$\mathcal{S}_2 = xy^2zv^2w \quad \in \quad L.$$

We have the following:

character	how many times it appears in $\mathcal{S}_2$
$\mathtt{a}$	$p+1+i_{\mathtt{a}}$
$\mathtt{b}$	$p+1+i_{\mathtt{b}}$
$\mathtt{c}$	$p+1+i_{\mathtt{c}}$

If $i_{\mathtt{a}} \neq i_{\mathtt{b}}$ then $\mathcal{S}_2$, by the above table, does not have the same number of $\mathtt{a}$s and $\mathtt{b}$s and as such it is not in L.

If $i_{\mathtt{b}} \neq i_{\mathtt{c}}$ then $\mathcal{S}_2$, by the above table, does not have the same number of $\mathtt{b}$s and $\mathtt{c}$s and as such it is not in L.

In either case, we get that $\mathcal{S}_2 \notin L$, which is a contradiction. Namely, our assumption that L is context-free is false.

Proof II: Using the corollary to pumping lemma:
Assume, for the sake of contradiction, that L is context-free, and apply the corollary to the Pumping Lemma to it (Lemma 2.7). Then there exists $n > 0$ such that any word in L longer than n can be contracted. Consider the word $\mathcal{S} = \mathtt{a}^{n+1}\mathtt{b}^{n+1}\mathtt{c}^{n+1}$. By the corollary to the pumping lemma, it can be written as $\mathtt{a}^{n+1}\mathtt{b}^{n+1}\mathtt{c}^{n+1} = xtw$, where $|t| \leq n$, and there is a strict contiguous substring t' of t such that $xt'w \in L$.

Since $|t| \leq n$, t cannot contain all of the letters a, b, and c. Hence contracting t to t' will reduce the number of letters of at most two kinds (e.g. a and b, or, b and c, etc.) but not the third. Hence $xt'w$ cannot have the same number of a's, b's, and c's, and hence is not in L, which is a contradiction. Hence our assumption must be wrong, and L cannot be context-free. ■

4 Closure properties

4.1 Context-free languages are not closed under intersection

We know that the languages

$$L_1 = \left\{ \mathtt{a}^*\mathtt{b}^n\mathtt{c}^n \;\middle|\; n \geq 0 \right\} \text{ and } L_2 = \left\{ \mathtt{a}^n\mathtt{b}^n\mathtt{c}^* \;\middle|\; n \geq 0 \right\}$$

are context-free (prove this). But

$$L = \left\{ \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \;\middle|\; n \geq 0 \right\} = L_1 \cap L_2$$

is not context-free by Lemma 3.1. We conclude that the intersection of two context-free languages is not necessarily context-free.

Lemma 4.1 *Context-free languages are not closed under intersection.*

4.2 Context-free languages are closed under union

Lemma 4.2 *Context-free languages over Σ are closed under union.*

Let L_1 and L_2 be two context-free languages, and let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be grammars generating these languages, respectively. Assume that $V_1 \cap V_2 = \emptyset$ (if not, rename the variables so that the sets are disjoint). Then it is easy to see that the grammar $G = (V, \Sigma, P, S)$ where $V = V_1 \cup V_2 \cup \{S\}$ and $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$ generates $L_1 \cup L_2$.

Proof: If $w \in L_i$ (where $i = 1$ or $i = 2$), then $S_i \Rightarrow^* w$ in the CFG G_i. Then, $w \in L(G)$ as we can derive $S \Rightarrow S_i \Rightarrow^* w$. Conversely, let $w \in L(G)$, and hence $S \Rightarrow^* w$. Since the only rules involving S are $S \to S_1$ and $S \to S_2$, the first rule used in the derivation of w must use such a rule. So let $S \Rightarrow S_i \Rightarrow^* w$ (where $i = 1$ or $i = 2$). Since the rules that can be used in the derivation from S_i in G are all from G_i, we can derive $S_i \Rightarrow^* w$ in G_i as well, and hence $w \in L_i$. Hence $L(G) = L(G_1) \cup L(G_2)$.

Lemma 4.3 *Context-free languages are not closed under complement.*

Proof: Since intersection can be written using union and complement operations, i.e. $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$, and since CFLs are closed under union, if CFLs were closed under complement, then CFLs will be closed under intersection as well. Since CFLs are not closed under intersection, it follows that they can't be closed under complement.

■