4.10: The Pumping Lemma for Context-free Languages

Let L be the language $\{\, 0^n 1^n 2^n \mid n \in \mathbb{N} \,\}$.

Question: is L context-free? I.e., is there a grammar that generates L?

Answer:

4.10: The Pumping Lemma for Context-free Languages

Let L be the language $\{\, 0^n 1^n 2^n \mid n \in \mathbb{N} \,\}$.

Question: is L context-free? I.e., is there a grammar that generates L?

Answer: No. Intuitively, although it's easy to keep the 0's and 1's matched, or to keep the 1's and 2's matched, or to keep the 0's and 2's matched, there is no way to keep all three symbols matched simultaneously.

Introduction

In this section, we will study the pumping lemma for context-free languages, which can be used to show that many languages are not context-free.

We will use the pumping lemma to prove that L is not context-free, and then we will prove the lemma.

Building on this result, we'll be able to show that the context-free languages are not closed under intersection, complementation or set-difference.

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L*,*

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L, there is a $n \in \mathbb{N}$ such that,

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L*, there is a* $n \in \mathbb{N}$ *such that, for all* $z \in \mathbf{Str}$*, if* $z \in L$ *and* $|z| \geq n$*, then*

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L*, there is a* $n \in \mathbb{N}$ *such that, for all* $z \in \textbf{Str}$*, if* $z \in L$ *and* $|z| \geq n$*, then there are* $u, v, w, x, y \in \textbf{Str}$ *such that* $z = uvwxy$ *and*

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L, there is a $n \in \mathbb{N}$ such that, for all $z \in \textbf{Str}$, if $z \in L$ and $|z| \geq n$, then there are $u, v, w, x, y \in \textbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$;

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L, there is a $n \in \mathbb{N}$ such that, for all $z \in \mathbf{Str}$, if $z \in L$ and $|z| \geq n$, then there are $u, v, w, x, y \in \mathbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$;

(2) $vx \neq \%$; *and*

Statement, Application and Proof of Pumping Lemma

Lemma 4.10.1 (Pumping Lemma for Context Free Languages)
For all context-free languages L, there is a $n \in \mathbb{N}$ such that, for all $z \in \mathbf{Str}$, if $z \in L$ and $|z| \geq n$, then there are $u, v, w, x, y \in \mathbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$;

(2) $vx \neq \%$; *and*

(3) $uv^iwx^iy \in L$, *for all* $i \in \mathbb{N}$.

Example Use of Pumping Lemma

Before proving the pumping lemma, let's see how it can be used to show that $L = \{ 0^n 1^n 2^n \mid n \in \mathbb{N} \}$ is not context-free. Suppose, toward a contradiction that L is context-free. Thus there is an $n \in \mathbb{N}$ with the property of the lemma. Let $z =$

Example Use of Pumping Lemma

Before proving the pumping lemma, let's see how it can be used to show that $L = \{\, 0^n 1^n 2^n \mid n \in \mathbb{N} \,\}$ is not context-free. Suppose, toward a contradiction that L is context-free. Thus there is an $n \in \mathbb{N}$ with the property of the lemma. Let $z = 0^n 1^n 2^n$. Since $z \in L$ and $|z| = 3n \geq n$, we have that there are $u, v, w, x, y \in \textbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$;

(2) $vx \neq \%$; and

(3) $uv^i wx^i y \in L$, for all $i \in \mathbb{N}$.

Since $0^n 1^n 2^n = z = uvwxy$, (1) tells us that vwx

Example Use of Pumping Lemma

Before proving the pumping lemma, let's see how it can be used to show that $L = \{ 0^n 1^n 2^n \mid n \in \mathbb{N} \}$ is not context-free. Suppose, toward a contradiction that L is context-free. Thus there is an $n \in \mathbb{N}$ with the property of the lemma. Let $z = 0^n 1^n 2^n$. Since $z \in L$ and $|z| = 3n \geq n$, we have that there are $u, v, w, x, y \in \textbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$;

(2) $vx \neq \%$; and

(3) $uv^i wx^i y \in L$, for all $i \in \mathbb{N}$.

Since $0^n 1^n 2^n = z = uvwxy$, (1) tells us that vwx doesn't contain both a 0 and a 2. Thus, either vwx has no 0's, or vwx has no 2's, so that there are two cases to consider.

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

- has n 0's; and

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

- has n 0's; and
- either has less than n 1's or has less than n 2's.

But (3) tells us that

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

- has n 0's; and
- either has less than n 1's or has less than n 2's.

But (3) tells us that $uwy = uv^0wx^0y \in L$, so that uwy has an equal number of 0's, 1's and 2's—contradiction.

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

- has n 0's; and
- either has less than n 1's or has less than n 2's.

But (3) tells us that $uwy = uv^0wx^0y \in L$, so that uwy has an equal number of 0's, 1's and 2's—contradiction.

The case where vwx has no 2's is similar.

Example

Suppose vwx has no 0's. Thus vx has no 0's. By (2), we have that vx contains a 1 or a 2. Thus uwy:

- has n 0's; and
- either has less than n 1's or has less than n 2's.

But (3) tells us that $uwy = uv^0wx^0y \in L$, so that uwy has an equal number of 0's, 1's and 2's—contradiction.

The case where vwx has no 2's is similar.

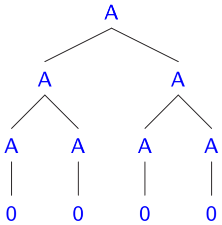
Since we obtained a contradiction in both cases, we have an overall contradiction. Thus L is not context-free.

A Fact About CNF Grammars

When we prove the pumping lemma for context-free languages, we will make use of a fact about grammars in Chomsky Normal Form.

Suppose G is a grammar in CNF and that $w \in (\textbf{alphabet}\ G)^*$ is the yield of a valid parse tree pt for G whose root label is a variable.

For instance, if G is the grammar with variable A and productions A → AA and A → 0, then w could be 0000 and pt could be the following tree of height 3:



CNF Fact

Generalizing from this example, we can see that if pt has height 3, $|w|$ will never be greater than $4 = 2^2 = 2^{3-1}$.

Question: how can we express an upper bound for $|w|$ as a function of the height k of pt?

Answer: $|w| \leq$

CNF Fact

Generalizing from this example, we can see that if pt has height 3, $|w|$ will never be greater than $4 = 2^2 = 2^{3-1}$.

Question: how can we express an upper bound for $|w|$ as a function of the height k of pt?

Answer: $|w| \leq 2^{k-1}$.

We can prove this by induction on

CNF Fact

Generalizing from this example, we can see that if pt has height 3, $|w|$ will never be greater than $4 = 2^2 = 2^{3-1}$.

Question: how can we express an upper bound for $|w|$ as a function of the height k of pt?

Answer: $|w| \leq 2^{k-1}$.

We can prove this by induction on pt.

Proof of Pumping Lemma

Proof. Suppose L is a context-free language.

Proof of Pumping Lemma

Proof. Suppose L is a context-free language. By the results of the preceding section, there is a grammar G in Chomsky Normal Form such that $L(G) = L - \{\%\}$. Let

Proof of Pumping Lemma

Proof. Suppose L is a context-free language. By the results of the preceding section, there is a grammar G in Chomsky Normal Form such that $L(G) = L - \{\%\}$. Let $k = |Q_G|$ and $n = 2^k$.

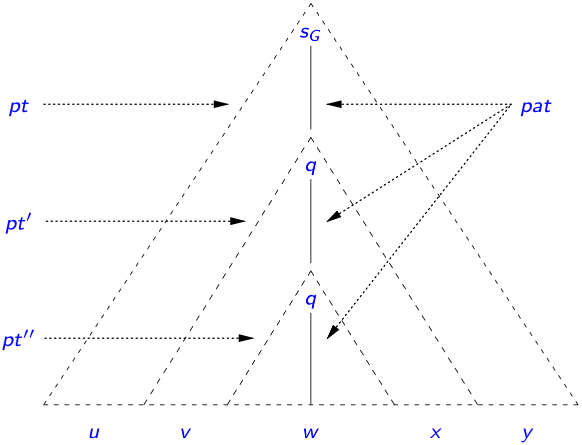
Proof of Pumping Lemma

Proof. Suppose L is a context-free language. By the results of the preceding section, there is a grammar G in Chomsky Normal Form such that $L(G) = L - \{\%\}$. Let $k = |Q_G|$ and $n = 2^k$. Suppose $z \in Str$, $z \in L$ and $|z| \geq n$. Since $n \geq 2$, we have that $z \neq \%$. Thus $z \in L - \{\%\} = L(G)$, so that there is a parse tree pt such that pt is valid for G, $\textbf{rootLabel}\, pt = s_G$ and $\textbf{yield}\, pt = z$. By our fact about CNF grammars, we have that the height of pt is at least

Proof of Pumping Lemma

Proof. Suppose L is a context-free language. By the results of the preceding section, there is a grammar G in Chomsky Normal Form such that $L(G) = L - \{\%\}$. Let $k = |Q_G|$ and $n = 2^k$. Suppose $z \in Str$, $z \in L$ and $|z| \geq n$. Since $n \geq 2$, we have that $z \neq \%$. Thus $z \in L - \{\%\} = L(G)$, so that there is a parse tree pt such that pt is valid for G, $\textbf{rootLabel}\, pt = s_G$ and $\textbf{yield}\, pt = z$. By our fact about CNF grammars, we have that the height of pt is at least $k + 1$. (If pt's height were only k, then $|z| \leq 2^{k-1} < n$, which is impossible.)

Proof (cont.). The rest of the proof can be visualized using the diagram



Proof

Proof (cont.). Let *pat* be a valid path for *pt* whose length is equal to the height of *pt*. Thus the length of *pat* is at least $k + 1$, so that the path visits at least variables

Proof

Proof (cont.). Let pat be a valid path for pt whose length is equal to the height of pt. Thus the length of pat is at least $k+1$, so that the path visits at least $k+1$ variables, with the consequence that at least one variable must be visited twice.

Proof

Proof (cont.). Let pat be a valid path for pt whose length is equal to the height of pt. Thus the length of pat is at least $k+1$, so that the path visits at least $k+1$ variables, with the consequence that at least one variable must be visited twice. Working from the last variable visited upwards, we look for the first repetition of variables. Suppose q is this repeated variable, and let pat' and pat'' be the initial parts of pat that take us to the upper and lower occurrences of q, respectively.

Proof

Proof (cont.). Let pat be a valid path for pt whose length is equal to the height of pt. Thus the length of pat is at least $k+1$, so that the path visits at least $k+1$ variables, with the consequence that at least one variable must be visited twice. Working from the last variable visited upwards, we look for the first repetition of variables. Suppose q is this repeated variable, and let pat' and pat'' be the initial parts of pat that take us to the upper and lower occurrences of q, respectively.

Let pt' and pt'' be the subtrees of pt at positions pat' and pat'', i.e., the positions of the upper and lower occurrences of q, respectively.

Proof

Proof (cont.). Let pat be a valid path for pt whose length is equal to the height of pt. Thus the length of pat is at least $k + 1$, so that the path visits at least $k + 1$ variables, with the consequence that at least one variable must be visited twice. Working from the last variable visited upwards, we look for the first repetition of variables. Suppose q is this repeated variable, and let pat' and pat'' be the initial parts of pat that take us to the upper and lower occurrences of q, respectively.

Let pt' and pt'' be the subtrees of pt at positions pat' and pat'', i.e., the positions of the upper and lower occurrences of q, respectively.

Consider the tree formed from pt by replacing the subtree at position pat' by q. This tree has yield

Proof

Proof (cont.). Let pat be a valid path for pt whose length is equal to the height of pt. Thus the length of pat is at least $k + 1$, so that the path visits at least $k + 1$ variables, with the consequence that at least one variable must be visited twice. Working from the last variable visited upwards, we look for the first repetition of variables. Suppose q is this repeated variable, and let pat' and pat'' be the initial parts of pat that take us to the upper and lower occurrences of q, respectively.

Let pt' and pt'' be the subtrees of pt at positions pat' and pat'', i.e., the positions of the upper and lower occurrences of q, respectively.

Consider the tree formed from pt by replacing the subtree at position pat' by q. This tree has yield uqy, for unique strings u and y.

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Furthermore, since $|pat|$ is the height of pt, the length of the path formed by removing pat' from pat will be the height of

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Furthermore, since $|pat|$ is the height of pt, the length of the path formed by removing pat' from pat will be the height of pt'. But we know that this length is at most $k+1$, because

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Furthermore, since $|pat|$ is the height of pt, the length of the path formed by removing pat' from pat will be the height of pt'. But we know that this length is at most $k + 1$, because, when working upwards through the variables visited by pat, we stopped as soon as we found a repetition of variables. Thus the height of pt' is at most $k + 1$.

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Furthermore, since $|pat|$ is the height of pt, the length of the path formed by removing pat' from pat will be the height of pt'. But we know that this length is at most $k+1$, because, when working upwards through the variables visited by pat, we stopped as soon as we found a repetition of variables. Thus the height of pt' is at most $k+1$.

Let w be the yield of pt''. Thus vwx is the yield of pt', so that $z = uvwxy$ is the yield of pt. Because the height of pt' is at most $k+1$, our fact about valid parse trees of grammars in CNF, tells us that $|vwx| \leq$

Proof

Proof (cont.). Consider the tree formed from pt' by replacing the subtree pt'' by q. More precisely, form the path pat''' by removing pat' from the beginning of pat''. Then replace the subtree of pt' at position pat''' by q. This tree has yield vqx, for unique strings v and x.

Furthermore, since $|pat|$ is the height of pt, the length of the path formed by removing pat' from pat will be the height of pt'. But we know that this length is at most $k+1$, because, when working upwards through the variables visited by pat, we stopped as soon as we found a repetition of variables. Thus the height of pt' is at most $k+1$.

Let w be the yield of pt''. Thus vwx is the yield of pt', so that $z = uvwxy$ is the yield of pt. Because the height of pt' is at most $k+1$, our fact about valid parse trees of grammars in CNF, tells us that $|vwx| \leq 2^{(k+1)-1} = 2^k = n$, showing that part (1) holds.

Proof

Proof (cont.). Because G is in CNF, pt', which has q as its root label, has two children. The child whose root node isn't visited by pat''' will have a non-empty yield, and this yield will be a prefix of , if this child is the left child, and will be a suffix of , if this child is the right child.

Proof

Proof (cont.). Because G is in CNF, pt', which has q as its root label, has two children. The child whose root node isn't visited by pat''' will have a non-empty yield, and this yield will be a prefix of v, if this child is the left child, and will be a suffix of x, if this child is the right child.

Proof

Proof (cont.). Because G is in CNF, pt', which has q as its root label, has two children. The child whose root node isn't visited by pat''' will have a non-empty yield, and this yield will be a prefix of v, if this child is the left child, and will be a suffix of x, if this child is the right child. Thus $vx \neq \%$, showing that part (2) holds.

Proof

Proof (cont.). Because G is in CNF, pt', which has q as its root label, has two children. The child whose root node isn't visited by pat''' will have a non-empty yield, and this yield will be a prefix of v, if this child is the left child, and will be a suffix of x, if this child is the right child. Thus $vx \neq \%$, showing that part (2) holds.

It remains to show part (3), i.e., that $uv^iwx^iy \in L(G) \subseteq L$, for all $i \in \mathbb{N}$. We define a valid parse tree pt_i for G, with root label q and yield v^iwx^i, by recursion on $i \in \mathbb{N}$. We let pt_0 be pt''. Then, if $i \in \mathbb{N}$, we form pt_{i+1} from pt' by replacing the subtree at position pat''' by pt_i.

Proof

Proof (cont.). Because G is in CNF, pt', which has q as its root label, has two children. The child whose root node isn't visited by pat''' will have a non-empty yield, and this yield will be a prefix of v, if this child is the left child, and will be a suffix of x, if this child is the right child. Thus $vx \neq \%$, showing that part (2) holds.

It remains to show part (3), i.e., that $uv^iwx^iy \in L(G) \subseteq L$, for all $i \in \mathbb{N}$. We define a valid parse tree pt_i for G, with root label q and yield v^iwx^i, by recursion on $i \in \mathbb{N}$. We let pt_0 be pt''. Then, if $i \in \mathbb{N}$, we form pt_{i+1} from pt' by replacing the subtree at position pat''' by pt_i.

Suppose $i \in \mathbb{N}$. Then the parse tree formed from pt by replacing the subtree at position pat' by pt_i is valid for G, has root label s_G, and has yield uv^iwx^iy, showing that $uv^iwx^iy \in L(G)$. □

Forlan Implementation of Pumping Lemma

The textbook describes the implementation in Forlan of the idea behind the Pumping Lemma.

Consequences of Pumping Lemma

Suppose

$$L = \{\, 0^n 1^n 2^n \mid n \in \mathbb{N} \,\},$$
$$A = \{\, 0^n 1^n 2^m \mid n, m \in \mathbb{N} \,\}, \text{ and}$$
$$B = \{\, 0^n 1^m 2^m \mid n, m \in \mathbb{N} \,\}.$$

Of course, L is not context-free.

Question: are A and B context-free?

Answer:

Consequences of Pumping Lemma

Suppose

$$L = \{ 0^n 1^n 2^n \mid n \in \mathbb{N} \},$$
$$A = \{ 0^n 1^n 2^m \mid n, m \in \mathbb{N} \}, \text{ and}$$
$$B = \{ 0^n 1^m 2^m \mid n, m \in \mathbb{N} \}.$$

Of course, L is not context-free.

Question: are A and B context-free?

Answer: yes, it's easy to find grammars that generate them.

Question: is $A \cap B$ context-free?

Answer:

Consequences of Pumping Lemma

Suppose

$$L = \{\, 0^n 1^n 2^n \mid n \in \mathbb{N} \,\},$$
$$A = \{\, 0^n 1^n 2^m \mid n, m \in \mathbb{N} \,\}, \text{ and}$$
$$B = \{\, 0^n 1^m 2^m \mid n, m \in \mathbb{N} \,\}.$$

Of course, L is not context-free.

Question: are A and B context-free?

Answer: yes, it's easy to find grammars that generate them.

Question: is $A \cap B$ context-free?

Answer: no—$A \cap B = L$, and L is not context-free.

Thus the context-free languages are not closed under intersection.

Consequences

Question: is $\{0,1,2\}^* - A$ context-free?

Answer:

Consequences

Question: is $\{0,1,2\}^* - A$ context-free?

Answer: yes, since this language is the union of the context-free languages

$$\{0,1,2\}^* - \{0\}^*\{1\}^*\{2\}^*$$

and

$$\{\, 0^{n_1} 1^{n_2} 2^m \mid n_1, n_2, m \in \mathbb{N} \text{ and } n_1 \neq n_2 \,\},$$

(the first of these languages is regular), and the context-free languages are closed under union.

Similarly, we have that $\{0,1,2\}^* - B$ is context-free.

Consequences

Let

$$C = (\{0,1,2\}^* - A) \cup (\{0,1,2\}^* - B).$$

Thus C is a context-free subset of $\{0,1,2\}^*$. Since $A, B \subseteq \{0,1,2\}^*$, it is easy to show that

$$\begin{aligned} A \cap B &= \{0,1,2\}^* - ((\{0,1,2\}^* - A) \cup (\{0,1,2\}^* - B)) \\ &= \{0,1,2\}^* - C. \end{aligned}$$

Consequences

Let

$$C = (\{0,1,2\}^* - A) \cup (\{0,1,2\}^* - B).$$

Thus C is a context-free subset of $\{0,1,2\}^*$. Since $A, B \subseteq \{0,1,2\}^*$, it is easy to show that

$$\begin{aligned} A \cap B &= \{0,1,2\}^* - ((\{0,1,2\}^* - A) \cup (\{0,1,2\}^* - B)) \\ &= \{0,1,2\}^* - C. \end{aligned}$$

Thus

$$\{0,1,2\}^* - C = A \cap B = L$$

is not context-free. Thus the context-free languages are not closed under complementation or set difference.