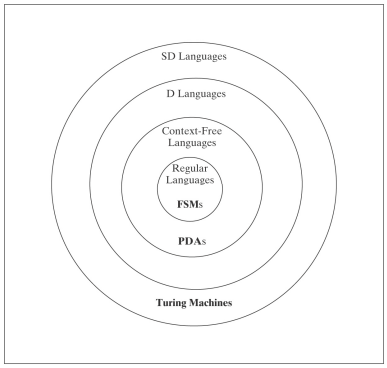
Context-Free and Noncontext-Free Languages

Chapter 13

Languages and Machines



Languages That Are and Are Not Context-Free

a^*b^* is regular.

$A^nB^n = \{a^nb^n : n \geq 0\}$ is context-free but not regular.

$A^nB^nC^n = \{a^nb^nc^n : n \geq 0\}$ is not context-free.

- Intuitively because a PDA's stack cannot count all three of the letter regions and compare them

The Regular and the CF Languages

Theorem: The regular languages are a proper subset of the context-free languages.

Proof: In two parts:

- Every regular language is CF.
- There exists at least one language that is CF but not regular.

The Regular and the CF Languages

Lemma: Every regular language is CFL.

Proof: Every FSM is (trivially) a PDA:

Given an FSM $M = (K, \Sigma, \Delta, s, A)$ and elements of δ of the form: (*p*, *c*, *q*)
old state, input, new state

Construct a PDA $M' = (K, \Sigma, \{\varnothing\}, \Delta, s, A)$. Each (*p*, *c*, *q*) becomes:

((*p*, *c*, ε), (*q*, ε))
old state, input, don't look at stack; new state; don't push on stack

In other words, **we just don't use the stack**.

There Exists at Least One Language that is CF but Not Regular

Lemma: There exists at least one language that is CF but not regular

Proof: $\{a^nb^n, n \geq 0\}$ is context-free but not regular.

So the regular languages are a proper subset of the context-free languages.

How Many Context-Free Languages Are There?

Theorem: There is a countably infinite number of CFLs.

Proof:

- Upper bound: we can lexicographically enumerate all the CFGs.
- Lower bound: every regular language is context-free and there is a countably infinite number of regular languages
- So, there is at most and at least a countably infinite number of context-free langauges.

- A set is **countably infinite** if its elements can be put in one-to-one correspondence with the set of natural numbers.
  - In other words, one can count off all elements in the set in such a way that, even though the counting will take forever, you will get to any particular element in a finite amount of time.
- A set is **uncountably infinite** if it contains so many elements that they cannot be put in one-to-one correspondence with the set of natural numbers.
  - In other words, there is no way that one can count off all elements in the set in such a way that, even though the counting will take forever, you will get to any particular element in a finite amount of time.

How Many Context-Free Languages Are There?

There is an uncountable number of languages (theorem 2.3).

Thus there are more languages than there are context-free languages.

So there must exist some languages that are not context-free.

Example: $\{a^n b^n c^n : n \geq 0\}$

Showing that *L* is Context-Free

Using Closure properties

Showing that *L* is Context-Free

Techniques for showing that a language *L* is context-free:

Methods are listed below

1. Exhibit a context-free grammar for *L*.
2. Exhibit a PDA for *L*.
3. Use the closure properties of context-free languages.
   - Unfortunately, there are fewer closure theorems than regular languages.

Closure Theorems for Context-Free Languages

The context-free languages are closed under:

- Union (natural, combination of CF grammar rules, combination of PDAs by episilon-transitions …)
- Concatenation
- Kleene star
- Reverse
- Letter substitution

Proof: The context-free languages are closed under Union

The context-free languages are closed under union: If L_1 and L_2 are context-free languages, then there exist context-free grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. If necessary, rename the nonterminals of G_1 and G_2 so that the two sets are disjoint and so that neither includes the symbol S. We will build a new grammar G such that $L(G) = L(G_1) \cup L(G_2)$. G will contain all the rules of both G_1 and G_2. We add to G a new start symbol, S, and two new rules, $S \rightarrow S_1$ and $S \rightarrow S_2$. The two new rules allow G to generate a string iff at least one of G_1 or G_2 generates it. So $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$.

Proof: The context-free languages are closed under Concatenation

The context-free languages are closed under concatenation: If L_1 and L_2 are context-free languages, then there exist context-free grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. If necessary, rename the nonterminals of G_1 and G_2 so that the two sets are disjoint and so that neither includes the symbol S. We will build a new grammar G such that $L(G) = L(G_1)L(G_2)$. G will contain all the rules of both G_1 and G_2. We add to G a new start symbol, S, and one new rule, $S \rightarrow S_1 S_2$. So $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}, S)$.

Proof: The context-free languages are closed under Kleene Star

The context-free languages are closed under Kleene star: If L_1 is a context-free language, then there exists a context-free grammar $G_1 = (V_1, \Sigma_1, R_1, S_1)$ such that $L_1 = L(G_1)$. If necessary, rename the nonterminals of G_1 so that V_1 does not include the symbol S. We will build a new grammar G such that $L(G) = L(G_1)^*$. G will contain all the rules of G_1. We add to G a new start symbol, S, and two new rules $S \rightarrow \varepsilon$ and $S \rightarrow S S_1$. So $G = (V_1 \cup \{S\}, \Sigma_1, R_1 \cup \{S \rightarrow \varepsilon, S \rightarrow S S_1\}, S)$.

Proof: The context-free languages are closed under Reverse

The context-free languages are closed under reverse: Recall that $L^R = \{w \in \Sigma^* : w = x^R \text{ for some } x \in L\}$. If L is a context-free language, then it is generated by some Chomsky normal form grammar $G = (V, \Sigma, R, S)$. Every rule in G is of the form $X \rightarrow BC$ or $X \rightarrow a$, where X, B, and C are elements of $V - \Sigma$ and $a \in \Sigma$. In the latter case $L(X) = \{a\}$. $\{a\}^R = \{a\}$. In the former case, $L(X) = L(B)L(C)$. By Theorem 2.4, $(L(B)L(C))^R = L(C)^R L(B)^R$. So we construct, from G, a new grammar G', such that $L(G') = L^R$. $G' = (V_G, \Sigma_G, R', S_G)$, where R' is constructed as follows:

- For every rule in G of the form $X \rightarrow BC$, add to R' the rule $X \rightarrow CB$.
- For every rule in G of the form $X \rightarrow a$, add to R' the rule $X \rightarrow a$.

Proof: The context-free languages are closed under Letter Substitution

- Proof is left as an exercise

The context-free languages are closed under letter substitution, defined as follows: Consider any two alphabets, Σ_1 and Σ_2. Let *sub* be any function from Σ_1 to Σ_2^*. Then *letsub* is a letter substitution function from L_1 to L_2 iff *letsub* $(L_1) = \{w \in \Sigma_2^* : \exists y \in L_1 (w = y$ except that every character c of y has been replaced by *sub* $(c))\}$. We leave the proof of this as an exercise.

What About Intersection?

The context-free languages are **not** closed under intersection:

The proof is by counterexample. Let:

$L_1 = \{a^n b^n c^m : n, m \geq 0\}$ /* equal a's and b's.
$L_2 = \{a^m b^n c^n : n, m \geq 0\}$ /* equal b's and c's.

Both L_1 and L_2 are context-free, since there exist straightforward context-free grammars for them.

But now consider:

$L = L_1 \cap L_2$
$= \{a^n b^n c^n : n \geq 0\}$

What About Complement?

The context-free languages are **not** closed under complement:

Closure under complement implies closure under intersection, since:

$$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$$

The context-free languages are closed under union, so if they were closed under complement, they would be closed under intersection (which they are not).

What About difference?

The context-free languages are **not** closed under difference:

The context-free languages are not closed under difference (subtraction): Given any language L,

$$\neg L = \Sigma^* - L.$$

Σ^* is context-free. So, if the context-free languages were closed under difference, the complement of any context-free language would necessarily be context-free. But we just showed that that is not so.

THEOREM 13.7 Closure Under Intersection With the Regular Languages

Theorem: The context-free languages are closed under intersection with the regular languages.

Proof: The proof is by construction. If L_1 is context-free, then there exists some PDA $M_1 = (K_1, \Sigma, \Gamma_1, \Delta_1, s_1, A_1)$ that accepts it. If L_2 is regular then there exists a DFSM $M_2 = (K_2, \Sigma, \delta, s_2, A_2)$ that accepts it. We construct a new PDA, M_3 that accepts $L_1 \cap L_2$. M_3 will work by simulating the parallel execution of M_1 and M_2. The states of M_3 will be ordered pairs of states of M_1 and M_2. As each input character is read, M_3 will simulate both M_1 and M_2 moving appropriately to a new state. M_3 will have a single stack, which will be controlled by M_1. The only slightly tricky thing is that M_1 may contain ε-transitions. So M_3 will have to allow M_1 to follow them while M_2 just stays in the same state and waits until the next input symbol is read.

$M_3 = (K_1 \times K_2, \Sigma, \Gamma_1, \Delta_3, (s_1, s_2), A_1 \times A_2)$, where Δ_3 is built as follows:

- For each transition $((q_1, a, \beta), (p_1, \gamma))$ in Δ_1, and each transition $((q_2, a,), p_2)$ in δ, add to Δ_3 the transition: $(((q_1, q_2), a, \beta), ((p_1, p_2), \gamma))$.
- For each transition $((q_1, \varepsilon, \beta), (p_1, \gamma))$ in Δ_1, and each state q_2 in K_2, add to Δ_3 the transition: $(((q_1, q_2), \varepsilon, \beta), ((p_1, q_2), \gamma))$.

THEOREM 13.8 Closure Under Difference with the Regular Languages

Theorem: The difference $(L_1 - L_2)$ between a context-free language L_1 and a regular language L_2 is context-free.

Proof: $L_1 - L_2 = L_1 \cap \neg L_2$. If L_2 is regular, then, since the regular languages are closed under complement, $\neg L_2$ is also regular. Since L_1 is context-free, by Theorem 13.7, $L_1 \cap \neg L_2$ is context-free.

EXAMPLE 13.5 Using Closure Theorems to Prove A Language Context-Free

Consider the perhaps contrived language $L = \{a^n b^n : n \geq 0 \text{ and } n \neq 1776\}$. Another way to describe L is that it is $\{a^n b^n : n \geq 0\} - \{a^{1776} b^{1776}\}$. $A^n B^n = \{a^n b^n : n \geq 0\}$ is context-free. We have shown both a simple grammar that generates it and a simple PDA that accepts it. $\{a^{1776} b^{1776}\}$ is finite and thus regular. So, by Theorem 13.8, L is context free.

Why are the Context-Free Languages Not Closed under Complement, Intersection and Subtraction But the Regular Languages Are?

Given an NDFSM M_1, build an FSM M_2 such that $L(M_2) = \neg L(M_1)$:

1. From M_1, construct an equivalent deterministic FSM M', using *ndfsmtodfsm*.
2. If M' is described with an implied dead state, add the dead state and all required transitions to it.
3. Begin building M_2 by setting it equal to M'. Then swap the accepting and the nonaccepting states. So:

$$M_2 = (K_{M'}, \Sigma, \delta_{M'}, s_{M'}, K_{M'} - A_{M'}).$$

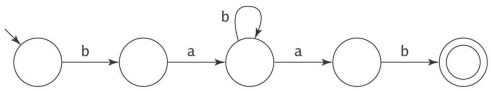
We could do the same thing for CF languages if we could do step 1, but we can't.

The need for nondeterminism is the key.

Showing that *L* is Not Context-Free

Showing that *L* is Not Context-Free

Remember the pumping argument for regular languages:

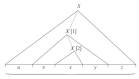


Proof by contradiction: If *L* were context-free, then it would possess certain properties. But it does not posses those properties. Therefore, *L* is not context-free.

Context-free pumping theorem is based on the structure of parse trees.

The Context-Free Pumping Theorem

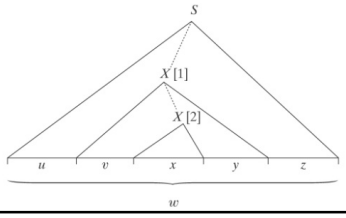
If *L* is a context-free language, then $\exists k \geq 1$, such that
$\forall$ strings $w \in L$, where $|w| \geq k$,
$\exists$ *u*, *v*, *x*, *y*, *z*, such that: $w = uvxyz$,and
$vy \neq \varepsilon$, and
$|vxy| \leq k$, and
$\forall q \geq 0$, $uv^q xy^q z$ is in *L*.



Proof: *L* is generated by some CFG $G = (V, \Sigma, R, S)$ with *n* nonterminal symbols and branching factor *b*. Let *k* be b^{n+1}. The longest string that can be generated by *G* with no repeated nonterminals in the resulting parse tree has length b^n. Assuming that $b \geq 2$, it must be the case that $b^{n+1} > b^n$. So let *w* be any string in $L(G)$ where $|w| \geq k$. Let *T* be any smallest parse tree for *w*. *T* must have height at least $n + 1$. Choose some path in *T* of length at least $n + 1$. Let *X* be the bottom-most repeated nonterminal along that path. Then *w* can be rewritten as *uvxyz*. The tree rooted at [1] has height at most $n + 1$. Thus its yield, *vxy*, has length less than or equal to b^{n+1}, which is *k*. $vy \neq \varepsilon$ since if *vy* were ε then there would be a smaller parse tree for *w* and we chose *T* so that that wasn't so. *uxz* must be in *L* because $rule_2$ could have been used immediately at [1]. For any $q \geq 1$, $uv^q xy^q z$ must be in *L* because $rule_1$ could have been used *q* times before finally using $rule_2$.

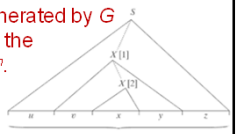
The Context-Free Pumping Theorem

If *L* is a context-free language, then $\exists k \geq 1$, such that
$\forall$ strings $w \in L$, where $|w| \geq k$,
$\exists$ *u*, *v*, *x*, *y*, *z*, such that:
$w = uvxyz$,
$vy \neq \varepsilon$,
$|vxy| \leq k$, and
$\forall q \geq 0$, $uv^q xy^q z$ is in *L*.



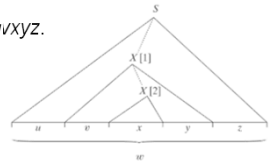
Consider an arbitrary parse tree, as shown in Figure 13.1 The ***height*** of a tree is the length of the longest path from the root to any leaf. The ***branching factor*** of a tree is the largest number of daughters of any node in the tree. The ***yield*** of a tree is the ordered sequence of its leaf nodes.

- *L* is generated by some CFG $G = (V, \Sigma, R, S)$ with *n* nonterminal symbols and branching factor *b*.
- The longest string that can be generated by *G* with no repeated nonterminals in the resulting parse tree has length b^n.



- Let *k* be b^{n+1}.
- Assuming that $b \geq 2$, it must be the case that $b^{n+1} > b^n$.
- So let *w* be any string in $L(G)$ where $|w| \geq k$.
- Let *T* be any smallest parse tree for *w*.
- *T* must have height at least $n + 1$.
- Choose some path in *T* of length at least $n + 1$.
- Let *X* be the bottom-most repeated nonterminal along that path.
- Then *w* can be rewritten as *uvxyz*.

- Then *w* can be rewritten as *uvxyz*.



- The tree rooted at [1] has height at most $n + 1$.
- Thus its yield, *vxy*, has length less than or equal to b^{n+1}, which is *k*.
- $vy \neq \varepsilon$ since if *vy* were ε then there would be a smaller parse tree for *w* and we chose *T* so that that wasn't so.
- *uxz* must be in *L* because $rule_2$ could have been used immediately at [1].
- For any $q \geq 1$, $uv^q xy^q z$ must be in *L* because $rule_1$ could have been used *q* times before finally using $rule_2$.

- So, if *L is* a context-free language, every "long" string in *L* must he pumpable.
- If there is even one long string in *L* that is not pumpable then L is not context-free.

Regular vs CF Pumping Theorems

Similarities:

- We choose *w*, the string to be pumped.
- We choose a value for *q* that shows that *w* isn't pumpable.
- We may apply closure theorems before we start.

Differences:

- Two regions, *v* and *y*, must be pumped in tandem.
- We don't know anything about where in the strings *v* and *y* will fall. All we know is that they are reasonably "close together", i.e., $|vxy| \le k$.
- Either *v* or *y* could be empty, although not both.

An Example of Pumping: $A^nB^nC^n$

$A^nB^nC^n = \{a^n b^n c^n, n \ge 0\}$

Choose $w = a^k b^k c^k$
1 | 2 | 3

If either *v* or *y* spans regions, then let *q* = 2 (i.e., pump in once).
The resulting string will have letters out of order and thus not be in $A^nB^nC^n$.

If both *v* and *y* each contain only one distinct character then set *q* to 2.
Additional copies of at most two different characters are added, leaving the third unchanged.
There are no longer equal numbers of the three letters, so the resulting string is not in $A^nB^nC^n$.

1. To prove $a^nb^nc^n$ is not CFL

- Consider w = $a^kb^kc^k \in$ L
- Since |w| > k, there are u, v, w, x, y such that w = uvxyz, $|vxy| \le k$, $vy \ne \varepsilon$ and $uv^qxy^qz \in$ L for all q ≥ 0.
- Since |vxy| ≤ k, vxz cannot contain all three of the symbols a, b, c, because there are q b's.
- So vwx either does not have any a's or does not have any b's or does not have any c's.
- Then uv^0wx^0y = uwy contains more a's than either b's or c's. Hence uwy ∉ L, which is a violation of conditions of Pumping theomem.
- So given language is not context free

2. The Language of Strings with n^2 a's is not CFL

- Let L = $\{a^{n^2} : n \ge 0\}$.
- We use the pumping theorem to show that L is not CFL
- If it were, then there would exist some *k* such that any string *w*, where **|w| ≥ *k***, must **satisfy the conditions** of the theorem.
- We show one string ***w* that does not**.
- Let n (in the definition of L) be k^2. So $n^2 = k^4$ and $w = a^{k^4}$
- For *w* to satisfy the conditions of the Pumping Theorem, there must be some *u, v, x, y,* and *z* such that

  w = *uvxyz*, $vy \ne \varepsilon$.

  $|vxy| \le k$. and

  ∀ *q* ≥ 0, uv^qxy^qz is in L
- We show that no such *u, v, x, y,* and *z* exist.

2. The Language of Strings with n^2 a's is not CFL

- Since *w* contains only a's, ***vy* = a^p**, for some nonzero *p*.
- Set *q* to 2. The resulting string, **s = a^{k^4+p}**, which must be in *L*. But it isn't because it is too short.
  - If a^{k^4}, which contains $(k^2)^2$ a's, is in *L*, then the next longer element of *L* contains $(k^2 + 1)^2 = k^4 + 2k^2 + 1$ a's.
  - So there are no strings in *L* with length between k^4 and $k^4 + 2k^2 + 1$.
  - But |s| = $k^4 + p$. So, for *s* to be in *L* , *p* = |*vy*| would have to be at least $2k^2 + 1$.
  - But |*vxy*| ≤ *k, so p* can't be that large.
- Thus s is not in *L*. There is no way to divide *w* into *uvxyz* such that all the conditions of the Pumping Theorem are met. So *L* is not context-free.

3. Dividing the String w Into Regions

- Let L = {$a^n b^m a^n$: m,n ≥ 0, m≥n}.
- We use the pumping theorem to show that L is not CFL
- If it were, then there would exist some *k* such that any string *w,* where **|w| ≥ *k*,** must **satisfy the conditions** of the theorem.
- We show one string ***w* that does not**.
- Let w = $a^k b^k c^k$
- For w to satisfy the conditions of the Pumping Theorem, there must be some u, v, x, y. and z, such that w = uvxyz, vy ≠ ε. lvxy|≤ k. and ∀ q ≥ 0 ($uv^q xy^q z$ is in L).
- We show that no such u, v, x, y, and z exist.
- Imagine w divided into three regions as follows:

```
aaa ... aaabbb ... bbbaaa ...aaa
|   1   |   2   |   3   |
```

```
aaa ... aaabbb ... bbbaaa ...aaa
|   1   |   2   |   3   |
```

- If either *v* or *y* crosses regions, then set *q* to 2 (thus pumping in once). The resulting string will have letters out of order and so not be in L.
- So in all the remaining cases we assume that *v* and y each falls within a single region.
- (1,1): Both *v* and y fall in region 1. Set *q to* 2. In the resulting string, the **first group of a's is longer than the second group** of a's. So, the string is not in *L*.
- (2. 2): Both *v* and y fall in region 2. Set *q* to 2. In the resulting string, **the b region is longer than** either of the a regions. So, the string is not in L
- (3, 3): Both *v* and y fall in region 3. Set *q* to 0. The same argument as for (1,1).

```
aaa ... aaabbb ... bbbaaa ...aaa
|   1   |   2   |   3   |
```

- (1, 2): Nonempty *v* falls in region 1 and nonempty *y* in region 2. Set *q* to 2. In the resulting string, the **first group of a's is longer than the second group of a's**. So, the string is not in *L*.
- (2, 3): Nonempty v falls in region 2 and nonempty *y* in region 3. Set *q* to 2. In the resulting string the **second group of a's is longer than the first group of a's**. So, the string is not in *L*.
- (1, 3): Nonempty *v* falls in region 1 and nonempty *y* in region 3. If this were allowed by the other conditions of the Pumping Theorem, we could pump in a's and still produce strings in *L*. But i **we pumped out, we would violate the requirement that the a regions be at least as long as the b region**. More importantly, this case violates the requirement that |vxy| ≤ *k* . So it need not be considered.
- There is no way to divide w into *uvxyz* such that all the conditions of the Pumping Theorem are met. So *L* is not context-free.

4. Prove L = {wcw, w is in {a,b}* } if not CFL

- Let L = {wcw; w belongs to {a,b}*.
- We use the pumping theorem to show that L is not CFL
- If it were, then there would exist some *k* such that any string *w,* where **|w| ≥ *k*,** must **satisfy the conditions** of the theorem.
- We show one string ***w* that does not**.
- Let w = $a^k b^k c a^k b^k$
- For w to satisfy the conditions of the Pumping Theorem, there must be some u, v, x, y. and z, such that w = uvxyz, vy ≠ ε. lvxy|≤ k. and ∀ q ≥ 0 ($uv^q xy^q z$ is in L).
- We show that no such u, v, x, y, and z exist.
- Imagine w divided into three regions as follows:

```
aaa ... aaabbb ... bbbcaaa ... aaabbb ... bbb
|   1   |   2   |3|   4   |   5   |
```

```
aaa ... aaabbb ... bbbcaaa ... aaabbb ... bbb
|   1   |   2   |3|   4   |   5   |
```

Call the part before the c the left side and the part after the c the right side. We consider all the cases for where *v* and *y* could fall and show that in none of them are all the conditions of the theorem met:

- If either *v* or *y* overlaps region 3, set *q* to 0. The resulting string will no longer contain a c and so is not in WcW.
- If both *v* and *y* occur before region 3 or they both occur after region 3, then set *q* to 2. One side will be longer than the other and so the resulting string is not in WcW.
- If either *v* or *y* overlaps region 1, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 4. But any *v*, *y* pair that did that would violate the requirement that $|vxy| \leq k$.
- If either *v* or *y* overlaps region 2, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 5. But any *v*, *y* pair that did that would violate the requirement that $|vxy| \leq k$.

There is no way to divide *w* into *uvxyz* such that all the conditions of the Pumping Theorem are met. So WcW is not context-free.

Using the Pumping Theorem in Conjunction with the Closure Properties

EXAMPLE 13.6 WW is Not Context-Free

Let WW = $\{ww : w \in \{a, b\}^*\}$. WW is similar to WcW = $\{wcw : w \in \{a, b\}^*\}$, except that there is no longer a middle marker. Because, like WcW, it contains cross-serial dependencies, it is not context-free. We could try proving that by using the Pumping Theorem alone. Here are some attempts, using various choices for w:

- Let $w = (ab)^{2k}$. If $v = \varepsilon$ and $y =$ ab, pumping works fine.
- Let $w = a^k b a^k b$. If $v =$ a and is in the first group of a's and $y =$ a and is in the second group of a's, pumping works fine.
- Let $w = a^k b^k a^k b^k$. Now the constraint that $|vxy| \leq k$ prevents v and y from both being in the two a regions or the two b regions. This choice of w will lead to a successful Pumping Theorem proof. But there are four regions in w and we must consider all the ways in which v and y could overlap those regions, including all those in which either or both of v and y occur on a region boundary. While it is possible to write out all those possibilities and show, one at a time, that every one of them violates at least one condition of the Pumping Theorem, there is an easier way.

If WW were context-free, then $L' =$ WW $\cap$ a*b*a*b* would also be context-free. But it isn't, which we can show using the Pumping Theorem. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = a^k b^k a^k b^k$, where k is the constant from the Pumping Theorem. For w to satisfy the conditions of the Pumping Theorem, there must be some u, v, x, y, and z, such that $w = uvxyz$, $vy \neq \varepsilon$, $|vxy| \leq k$, and $\forall q \geq 0$ ($uv^q xy^q z$ is in L'). We show that no such u, v, x, y, and z exist. Imagine w divided into four regions as follows:

```
aaa ... aaabbb ... bbbaaa ... aaabbb ... bbb
|   1   |   2   |   3   |   4   |
```

We consider all the cases for where v and y could fall and show that in none of them are all the conditions of the theorem met:

- If either v or y overlaps more than one region, set q to 2. The resulting string will not be in a*b*a*b* and so is not in L'.
- If $|vy|$ is not even then set q to 2. The resulting string will have odd length and so not be in L'. We assume in all the other cases that $|vy|$ is even.
- (1, 1), (2, 2), (1, 2): Set q to 2. The boundary between the first half and the second half will shift into the first b region. So the second half will start with a b, while the first half still starts with an a. So the resulting string is not in L'.


```
aaa ... aaabbb ... bbbaaa ... aaabbb ... bbb
|   1   |   2   |   3   |   4   |
```


- (3, 3), (4, 4), (3, 4): Set q to 2. This time the boundary shifts into the second a region. The first half will end with an a while the second half still ends with a b. So the resulting string is not in L'.
- (2, 3): Set q to 2. If $|v| \neq |y|$ then the boundary moves and, as argued above, the resulting string is not in L'. If $|v| = |y|$ then the first half contains more b's and the second half contains more a's. Since they are no longer the same, the resulting string is not in L'.
- (1, 3), (1, 4), and (2, 4) violate the requirement that $|vxy| \leq k$.

There is no way to divide w into $uvxyz$ such that all the conditions of the Pumping Theorem are met. So L' is not context-free. So neither is WW.

EXAMPLE 13.7 A Simple Arithmetic Language is Not Context-Free

Let $L = \{x \# y = z : x, y, z \in \{0,1\}^*$ and, if x, y and z are viewed as positive binary numbers without leading zeros, then $xy = z^R\}$. For example, 100#111 = 00111 $\in L$. (We do this example instead of the more natural one in which we require that $xy = z$ because it seems as though it might be more likely to be context-free. As we'll see, however, even this simpler variant is not.)

If L were context-free, then $L' = L \cap 10^*\#1^* = 0^*1^*$ would also be context-free. But it isn't, which we can show using the Pumping Theorem. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = 10^k \#1^k = 0^k 1^k$, where k is the constant from the Pumping Theorem. Note that $w \in L$ because $10^k \cdot 1^k = 1^k 0^k$.

For w to satisfy the conditions of the Pumping Theorem, there must be some u, v, x, y, and z, such that $w = uvxyz$, $vy \neq \varepsilon$, $|vxy| \leq M$, and $\forall q \geq 0$ ($uv^q xy^q z$ is in L). We show that no such u, v, x, y, and z exist. Imagine w divided into seven regions as follows:

```
1 000 ... 000 # 111 ... 111 = 000 ... 000111 ... 111
|1|   2    |3|   4    |5|   6    |   7   |
```

We consider all the cases for where v and y could fall and show that in none of them are all the conditions of the theorem met:


```
1 000 ... 000 # 111 ... 111 = 000 ... 000111 ... 111
|1|   2    |3|   4    |5|   6    |   7   |
```


- If either v or y overlaps region 1, 3, or 5 then set q to 0. The resulting string will not be in 10*#1* = 0*1* and so is not in L'.
- If either v or y contains the boundary between 6 and 7, set q to 2. The resulting string will not be in 10*#1* = 0*1* and so is not in L'. So the only cases left to consider are those where v and y each occur within a single region.
- (2, 2), (4, 4), (2, 4): Set q to 2. Because there are no leading zeros, changing the left side of the string changes its value. But the right side doesn't change to match. So the resulting string is not in L'.
- (6, 6), (7, 7), (6, 7): Set q to 2. The right side of the equality statement changes value but the left side doesn't. So the resulting string is not in L'.
- (4, 6): Note that, because of the first argument to the multiplication, the number of 1's in the second argument must equal the number of 1's after the =. Set q to 2. The number of 1's in the second argument changed but the number of 1's in the result did not. So the resulting string is not in L'.
- (2, 6), (2, 7), and (4, 7) violate the requirement that $|vxy| \leq k$.

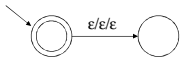
There is no way to divide w into $uvxyz$ such that all the conditions of the Pumping Theorem are met. So L is not context-free.

DETERMINISTIC CFL

Deterministic PDAs

A PDA M is ***deterministic*** iff:

- Δ_M contains no pairs of transitions that compete with each other, and
- Whenever M is in an accepting configuration it has no available moves.



M can choose between accepting and taking the ε-transition, so it is not deterministic.

Deterministic CFLs

A language L is ***deterministic context-free*** iff $L\$$ can be accepted by some deterministic PDA.

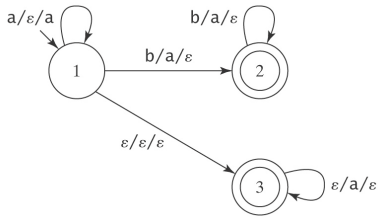
$: end of string marker Why $?

Let $L = a^* \cup \{a^n b^n : n > 0\}$.

- When it begins reading a's, M must push them onto the stack in case there are going to be b's it runs out of input without seeing b's, it needs a way to pop out the a's from the stack before it accepts.
  - recall how "accept" was defined
- Add $ to make it easier to build DPDAs, it does not add power (to allow building a PDA for L that was not context-free)
- There exist CLFs that are not deterministic, e.g., $L = \{a^i b^j c^k, i \neq j \text{ or } j \neq k\}$.

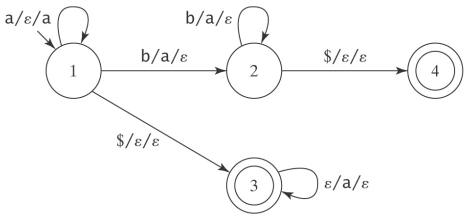
An NDPDA for L

$L = a^* \cup \{a^n b^n : n > 0\}$.



A DPDA for $L\$$

$L = a^* \cup \{a^n b^n : n > 0\}$.



DCFLs are Closed Under Complement

Proof by construction.

$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$.

What about intersection and union?

DCFLs are Not Closed Under Union

$L_1 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i \neq j\}$. (a DCFL)
$L_2 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } j \neq k\}$. (a DCFL)

$L' = L_1 \cup L_2$.
$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } (i \neq j) \text{ or } (j \neq k)\}$.

$L'' = \neg L'$.
$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j = k\} \cup$
$\{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$.

$L''' = L'' \cap a^* b^* c^*$.
$= \{a^n b^n c^n, n \geq 0\}$.

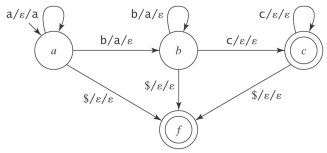
L''' is not even CF, much less DCF.

DCFLs are Not Closed Under Intersection

$L_1 = \{a^i b^j c^k,\ i, j, k \geq 0 \text{ and } i = j\}$.
$L_2 = \{a^i b^j c^k,\ i, j, k \geq 0 \text{ and } j = k\}$.

$L' = L_1 \cap L_2$
=

L_1 and L_2 are deterministic context-free:



Nondeterministic CFLs

Theorem: There exist CFLs that are not deterministic.

Proof: By example. Let $L = \{a^i b^j c^k,\ i \neq j \text{ or } j \neq k\}$. L is CF. If L is DCF then so is:

$L' = \neg L$
$= \{a^i b^j c^k,\ i, j, k \geq 0 \text{ and } i = j = k\} \cup$
$\{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$.

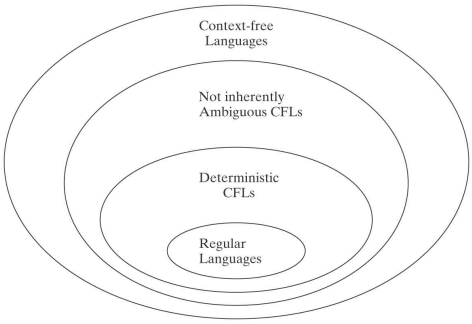
But then so is:

$L'' = L' \cap a^* b^* c^*$.
$= \{a^n b^n c^n,\ n \geq 0\}$.

But it isn't. So L is context-free but not deterministic context-free.

This simple fact poses a real problem for the designers of efficient context-free parsers.

The CFL Hierarchy



Algorithms and Decision Procedures for Context-Free Languages

Chapter 14

Decidability of CFLs

Theorem: Given a context free language L and a string w, there exists a decision procedure that answers the questions, "is w in L?"

Two approaches:
- Find a context-free grammar to generate it
- Find a PDA to accept it

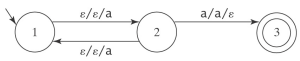
Using a Grammar

decideCFLusingGrammar(L: CFL, w: string) =

1. If given a PDA, build G so that $L(G) = L(M)$.
2. If $w = \varepsilon$ then if S_G is nullable then accept, else reject.
3. If $w \neq \varepsilon$ then:
   3.1 Construct G' in Chomsky normal form such that $L(G') = L(G) - \{\varepsilon\}$.
   3.2 If G derives w, it does so in $2 \cdot |w| - 1$ steps. Try all derivations in G of $2 \cdot |w| - 1$ steps. If one of them derives w, accept. Otherwise reject.

Using a PDA

Problem:



Theorem: Given any context-free language grammar $G = (V, \Sigma, R, S)$, there exists a PDA M such that $L(M) = L(G) - \{\varepsilon\}$ and M contains no transitions of the form $((q_1, \varepsilon, \alpha), (q_2, \beta))$.
In other words, every transition reads exactly one input character.

Greibach Normal Form: All rules are of the following form:
$X \rightarrow a\ A$, where $a \in \Sigma$ and $A \in (V - \Sigma)^*$.

No need to push the a and then immediately pop it.

So $M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

1. The start-up transitions:
   For each rule $S \rightarrow cs_2 \ldots s_n$, the transition:
   $((p, c, \varepsilon), (q, s_2 \ldots s_n))$.

2. For each rule $X \rightarrow cs_2 \ldots s_n$ (where $c \in \Sigma$ and s_2 through s_n are elements of $V - \Sigma$), the transition:

   $((q, c, X), (q, s_2 \ldots s_n))$

A PDA Without ε-Transitions Must Halt

Consider the execution of M on input w:

- Each individual path of M must halt within $|w|$ steps.
- The total number of paths pursued by M must be less than or equal to $P = B^{|w|}$, where B is the maximum number of competing transitions from any state in M.
- The total number of steps that will be executed by all paths of M is bounded by $P * |w|$.

So all paths must eventually halt.

• **The same for NDFSM**

An Algorithm to Decide Whether *M* Accepts *w*

decideCFLusingPDA(*L*: CFL, *w*: string) =
1. If *L* is specified as a PDA, use *PDAtoCFG* to construct a grammar *G* such that $L(G) = L(M)$.
2. If *L* is specified as a grammar *G*, simply use *G*.
3. If $w = \varepsilon$ then if S_G is nullable then accept, otherwise reject.
4. If $w \neq \varepsilon$ then:
   4.1 From *G*, construct *G'* such that $L(G') = L(G) - \{\varepsilon\}$ and *G'* is in Greibach normal form.
   4.2 From *G'* construct a PDA *M* such that $L(M) = L(G')$ and *M'* has no ε-transitions.
   4.3 All paths of *M* are guaranteed to halt within a finite number of steps. So run *M* on *w*. Accept if it accepts and reject otherwise.

What can you say about PDA's halting behavior?
• A PDA *M* must halt ?
• For an arbitrary PDA *M*, there exists *M'* that halts and $L(M') = L(M)$?

Decidability of Emptiness and Finiteness

Theorem: Given a context free language *L*, there exists a decision procedure that answers the following questions:

1. Given a context-free language *L*, is $L = \varnothing$?
2. Given a context-free language *L*, is *L* infinite?

1. Given a context-free language *L*, is $L = \varnothing$?

decideCFLempty(*G*: context-free grammar) =
1. Let $G' = removeunproductive(G)$.
2. If *S* is not present in *G'* then return *True* else return *False*.

Given a context-free language *L*, is *L* infinite?

decideCFLinfinite(*G*: context-free grammar) =

1. Lexicographically enumerate all strings in Σ^* of length greater than b^n and less than or equal to $b^{n+1} + b^n$.
2. If, for any such string w, *decideCFL*(L, w) returns *True* then return *True*. *L* is infinite.
3. If, for all such strings w, *decideCFL*(L, w) returns *False* then return *False*. *L* is not infinite.

Equivalence of DCFLs

Theorem: Given two *deterministic* context-free languages L_1 and L_2, there exists a decision procedure to determine whether $L_1 = L_2$.

- This claim was not proved until 1997

The Undecidable Questions about CFLs

- Is $L = \Sigma^*$?
- Is the complement of *L* context-free?
- Is *L* regular?
- Is $L_1 = L_2$?
- Is $L_1 \subseteq L_2$?
- Is $L_1 \cap L_2 = \emptyset$?
- Is *L* inherently ambiguous?
- Is *G* ambiguous?

CFL Summary

- The theory of CFL is not as tidy as the theory of RL
- Interesting subsets, DCFL, and not inherently ambiguous CFL, are only proper subset of CFL
- Not closed under many common operations
- No algorithm for minimizing PDAs
- No fast recognition algorithm that works on arbitrary CFL
- No decision procedure for many important questions
- Yet substantial effort has been invested in CFLs as they are useful

RLs vs. CFLs

Regular Languages	Context-Free Languages
● regular exprs. ● or ● regular grammars ● = DFSMs ● recognize ● minimize FSMs	● context-free grammars ● = NDPDAs ● parse ● find unambiguous grammars ● reduce nondeterminism in PDAs ● find efficient parsers
● closed under: ♦ concatenation ♦ union ♦ Kleene star ♦ complement ♦ intersection	● closed under: ♦ concatenation ♦ union ♦ Kleene star ♦ intersection w/ reg. langs
● pumping theorem ● D = ND	● pumping theorem ● D ≠ ND