# **Motivating the Pumping Lemma**

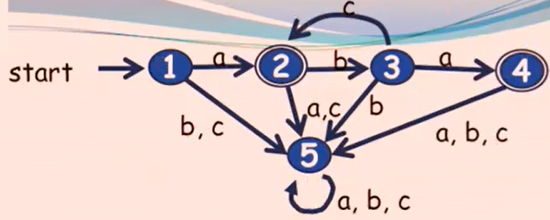
If you visit *n* states, your string has *n* - 1 letters. This only applies if there are no loops in the states and just straightforward arrows.

* You DO visit that many states
* Some visits may be repeats to the same state, the DFA doesn’t need to have that many

**General Thoughts**

* Suppose you have a DFA called *D*
* Suppose the language *D* recognizes a language called *L*
* Let *m* be the number of states in *D*
* Let *s* be a string in *D* that has at least *m* letters in it
* Since *s* has *m* letters in it, it visits *m +* 1 states
* Thus *D* has *m* states
* One state must be visited twice (by pigeonhole)

**Theorem:** If a DFA recognizes an infinite language, then it must have a loop in it.



**A Weird Motivating Procedure**

Take the DFA to the right and run through string abcba.

The state visits are as follows: 1, 2, 3, 2, 3, 4.

What is the first state visited twice? 2

Split the string into 3 sections by drawing a line through the first repeated state: So, in this case, a would be in section *x*, b and c would be in section *y*, and b and a would be in section *z*.

* It is possible for sections *x* and *z* to be equal to lambda (^). *y* can never be lambda because the closest splits will have at least something inside it.
* |*x|* + |*y*| ≤ *m* (the number of letters in *x* and *y* are less than or equal to the number of letters in the string), and |*x*| + |*y*| + |*z*| ≥ *m*

If you want to prove that a language *L* is not regular, use proof by contradiction:

1. Assume *L* is regular
2. Pick any string in *L* whose length is ≥ *m*
3. Show that it cannot be “pumped” to give more strings in the language (you can’t keep adding *y* parts to it)
4. This is a contradiction, so it must not be regular

If the DFA has *m* states, any string in the language that is at least *m* chars long will have to visit at least one state twice. So, *xz*, *xyz*, etc. would be in the language.

# **The Actual Pumping Lemma**

## **Pumping Lemma (informal)**

If you have a regular language, there’s a DFA for that.

Pretend you know exactly how many states are in that DFA (which is the number *m*). Every string in the language that has at least *m* characters long can be “pumped” to make more strings in the language.

Clever hack: If you can just find one string in the language at least *m* characters long that can’t be pumped, then the pumping lemma tells you that the language isn’t regular.

## **Pumping Lemma (formal)**

Let *L* be an infinite regular language over the alphabet A.

Then, there is an integer *m* (the number of states in the DFA) greater than 0 such that any string *s* where s ∊ L and |*s*| ≥ *m* (so there’s a duplicate state), there exists strings *x*, *y*, z ∊ A\*, where *y* ≠ ^ such that:

* s = xyz
* | xy | ≤ *m*
* xy*k*z ∊ L for all *k* ≥ 0

**Outline that {a*n*b*n* | n ∊ N} is not regular**

* Let L equal the language.
* Assume L is regular.
* There’s a number *m* for which pumping lemma applies, such that (three bullets above).
* Since this is true for all strings in L, if you can find just 1 example it’s not true for, it’s a contradiction.

**Actual Proof**

Let L = {a*n*b*n* | n ∊ N} and assume L is regular.

Let m be the number from the pumping lemma.

Let s = a*m*b*m*.

Note that s ∊ L, and |s| ≥ m, so the pumping lemma applies. (The second statement here is true because *s* has 2 times *m* letters, because there are *m* a’s and *m* b’s.)

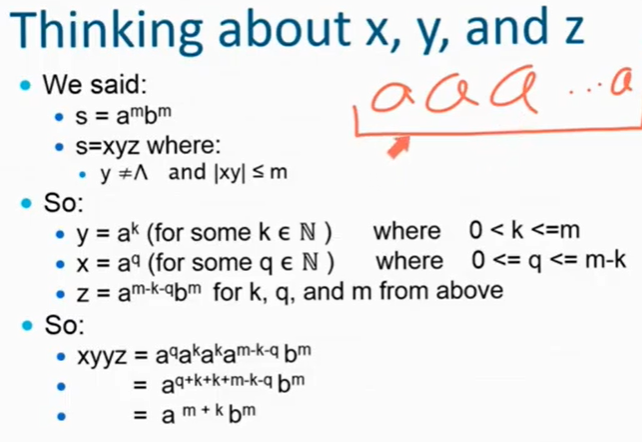
By the pumping lemma, s = xyz where y ≠ ^ and |xy| ≤ m.

Since |xy| ≤ m, xy must consist of just a’s.

So, y must be one or more a’s (since y ≠ ^), so suppose |y| = k (so y = a*k*)

By the pumping lemma, xyyz ∊ L

What is xyyz?



This means that there are *k* a’s, and *m* a’s. However, according to the problem, there are only *m* b’s. BUT, we’ve established by pumping lemma that the string is in the language! But according to this, it ISN’T! So this is a contradiction, thus this language isn’t regular.

QED.

**How to choose a good string**

Definitely remember that it has to be *m* characters long! Pick something that you knew you could break when pumped!

**Example:** Prove that the language L = {All palindromes over {a, b} of odd length} isn’t regular.

Note that at least *m* char long pumping will “mess up” palindromes. Let’s choose *s* to be a*m*ba*m*. This ensures that it is a palindrome, as well as the fact that it has at least *m* characters in the string.

From this, you know that *y* (the pumping part) is all a’s. This is because if you make two splits (like in the motivating example) where there are repeating parts, you will have those at the beginning, and *y* will have one or more a’s in there.

So, let y = a*k* for some 1 ≤ *k* ≤ *m*. You know that 1 is less than or equal to *k* because there is at least *a* in the string to be at least *m* characters long, and because *y* isn’t empty. The number of a’s in *y* is also less than *m* because the size of xy is less than or equal to *m*.

(The official statement for the structure of *y* would be: *y* must be all a’s, so y = a*k* where

1 ≤ *k* ≤ *m.*)

Notice that for the string you chose, |s| = 2m + 1, and thus is greater than *m*.

Let’s say xyyz ∊ L. What is xyyz? Well, according to the string you chose,

y = a*k* for some 0 < *k* ≤ *m*, because *y* can’t be lambda (same as 1 ≤) .

x = a*w* for some 0 ≤ *w* ≤ *m - k,* and *x* CAN be lambda, *m* - *k* is the number of a’s left over

z = a*m-w-k*ba*m*, representing the rest of the string

Thus, xyyz = a*w*a*k*a*k*a*m-w-k*ba*m*. Cancel out the exponents properly to get a*k + m*ba*m*. HOWEVER, this means that xyyz is NOT in the language, because the language is palindromes and having more a’s on one side than the other is not acceptable. By the pumping lemma, it SHOULD be in the language, so this is a contradiction; thus the assumption is false/language is not regular.

QED.

**Example:** Prove that the language L = {All palindromes over {a, b} of even length} isn’t regular.

Let s = a*m*bba*m*, thus giving a length of 2m + 2, guaranteeing that it’s even length + it’s greater than *m*.

y = a*k* for some 0 < *k* ≤ *m*, because *y* can’t be lambda (same as 1 ≤) .

x = a*p* for some 0 ≤ *p* ≤ *m - k,* and *x* CAN be lambda, *m* - *k* is the number of a’s left over

z = a*m-p-k*bba*m*, representing the rest of the string

Let’s choose to do xz this time, which is equal to a*p*a*m-p-k*bba*p*, which can be reduced to a*m=k*bba*m*, which is clearly not a palindrome since *m* is not equal to *m* - *k*.

**Example:** Prove that the language L = {a*n*b*k* | n, k ∊ N and n ≥ k} is not regular.

Let s = a*m*b*m*, thus giving a length of 2m and greater than *m*.

y = a*c* for some 0 < *c* ≤ *m*, because *y* can’t be lambda (same as 1 ≤) .

x = a*p* for some 0 ≤ *p* ≤ *m - c,* and *x* CAN be lambda, *m* - *c* is the number of a’s left over

z = a*m-c-p*b*m*, representing the rest of the string

However, when doing the work using xyyz, it doesn’t contradict anything since the m + c is greater than m (which fits the language definition)! So, you must use xz, which is equal to a*p*a*m-c-p*b*m*, which reduces to a*m-c*b*m*, which breaks the rule that *n* must be greater than or equal to *k*. This is okay because you just need to find one counterexample-ish.

This is a contradiction, and thus the language isn’t regular.

QED.

**Example:** Prove that language L = {ab*n*a*n* | n ∊ N} isn’t regular.

*Reference the document in file about this problem.* *It states:* “In most of our proofs, we had it easy - we had no choice about *y*. But in this proof, *y* could have these 3 different options. In order to show a contradiction, it’s not enough to show that just one of the three options isn’t possible, we have to show that all 3 options aren’t possible.”

Let s = ab*m*a*m*, which is definitely greater than *m* length.

*y* could be 3 different forms:

1. y could just be the letter a
2. y could just be some b’s
3. y could be an a followed by some b’s