

# The Pumping Lemma: limitations of regular languages

## Informatics 2A: Lecture 7

John Longley

School of Informatics  
University of Edinburgh  
jrl@inf.ed.ac.uk

6 October, 2015

# Non-regular languages

We have hinted before that not all languages are regular. E.g.

- The language  $\{a^n b^n \mid n \geq 0\}$ .
- The language of all *well-matched* sequences of brackets  $(, )$ .  
N.B. A sequence  $x$  is well-matched if it contains the same number of opening brackets '(' and closing brackets ')', and no initial subsequence  $y$  of  $x$  contains more ')'s than '('s.
- The language of all prefixes of well-matched sequences of brackets  $(, )$ . A string  $x$  is in this language if no initial subsequence  $y$  of  $x$  contains more ')'s than '('s.

But **how do we know** these languages aren't regular?

And can we come up with a **general technique** for proving the non-regularity of languages?

# The basic intuition: DFAs can't count!

Consider  $L = \{a^n b^n \mid n \geq 0\}$ . Just suppose, hypothetically, there were some DFA  $M$  with  $\mathcal{L}(M) = L$ .

Suppose furthermore that  $M$  had just processed  $a^n$ , and some continuation  $b^m$  was to follow.

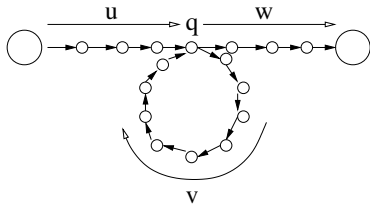
**Intuition:**  $M$  would need to have *counted* the number of  $a$ 's, in order to know how many  $b$ 's to require.

More precisely, let  $q_n$  denote the state of  $M$  after processing  $a^n$ . Then for any  $m \neq n$ , the states  $q_m, q_n$  must be different, since  $b^m$  takes us to an accepting state from  $q_m$ , but not from  $q_n$ .

In other words,  $M$  would need **infinitely many states**, one for each natural number. Contradiction!

## Loops in DFAs

Let  $M$  be a DFA with  $k$  states. Suppose, starting from any state of  $M$ , we process a string  $y$  of length  $|y| \geq k$ . We then pass through a sequence of  $|y| + 1$  states. So there must be some state  $q$  that's visited *twice or more*:



(Note that  $u$  and  $w$  might be  $\epsilon$ , but  $v$  definitely isn't.)

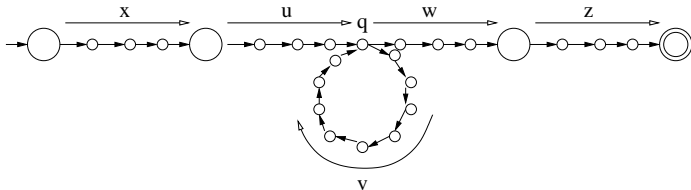
So any string  $y$  with  $|y| \geq k$  can be decomposed as  $uvw$ , where

- $u$  is the prefix of  $y$  that leads to the first visit of  $q$
- $v$  takes us once round the loop from  $q$  to  $q$ ,
- $w$  is whatever is left of  $y$  after  $uv$ .

## A general consequence

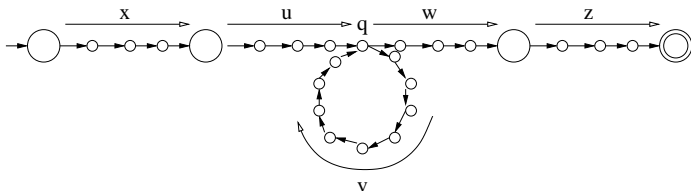
If  $L$  is *any* regular language, we can pick *some* corresponding DFA  $M$ , and it will have some number of states, say  $k$ .

Suppose we run  $M$  on a string  $xyz \in L$ , where  $|y| \geq k$ . There must be at least one state  $q$  visited twice in the course of processing  $y$ :



(There may be other 'revisited states' not indicated here.)

# The idea of 'pumping'



So  $y$  can be decomposed as  $uvw$ , where

- $xu$  takes  $M$  from the initial state to  $q$ ,
- $v \neq \epsilon$  takes  $M$  once round the loop from  $q$  to  $q$ ,
- $wz$  takes  $M$  from  $q$  to an accepting state.

But now  $M$  will be oblivious to whether, or how many times, we go round the  $v$ -loop!

So we can 'pump in' as many copies of the substring  $v$  as we like, knowing that we'll still end in an accepting state.

## The pumping lemma: official form

The pumping lemma basically summarizes what we've just said.

**Pumping Lemma.** Suppose  $L$  is a regular language. Then  $L$  has the following property.

*(P) There exists  $k \geq 0$  such that, for all strings  $x, y, z$  with  $xyz \in L$  and  $|y| \geq k$ , there exist strings  $u, v, w$  such that  $y = uvw$ ,  $v \neq \epsilon$ , and for every  $i \geq 0$  we have  $xuv^i wz \in L$ .*

## The pumping lemma: contrapositive form

Since we want to use the pumping lemma to show a language *isn't* regular, we usually apply it in the following equivalent but back-to-front form.

Suppose  $L$  is a language for which the following property holds:

*( $\neg P$ ) For all  $k \geq 0$ , there exist strings  $x, y, z$  with  $xyz \in L$  and  $|y| \geq k$  such that, for every decomposition of  $y$  as  $y = uvw$  where  $v \neq \epsilon$ , there is some  $i \geq 0$  for which  $xuv^i wz \notin L$ .*

Then  $L$  is not a regular language.

**N.B.** The pumping lemma can only be used to show a language *isn't* regular. Showing  $L$  satisfies (P) doesn't prove  $L$  is regular!

To show that a language *is* regular, give some DFA or NFA or regular expression that defines it.



## The pumping lemma: a user's guide

So to show some language  $L$  is not regular, it's enough to show that  $L$  satisfies  $(\neg P)$ .

Note that  $(\neg P)$  is quite a complex statement:  $\forall \dots \exists \dots \forall \dots \exists \dots$ .

We'll look at a simple example first, then offer some advice on the general pattern of argument.

## Example 1

Consider  $L = \{a^n b^n \mid n \geq 0\}$ .

We show that  $L$  satisfies  $(\neg P)$ .

Suppose  $k \geq 0$ . (We can't choose the value of  $k$ . The argument has to work for all numbers.)

Consider the strings  $x = \epsilon$ ,  $y = a^k$ ,  $z = b^k$ . Note that  $xyz \in L$  and  $|y| \geq k$  as required. (We make a cunning choice of  $x, y, z$ .)

Suppose now we're given a decomposition of  $y$  as  $uvw$  with  $v \neq \epsilon$ . (We can't choose the strings  $u, v, w$ . The argument has to work for all possibilities.)

Let  $i = 0$ . (We make a cunning choice of  $i$ .)

Then  $uv^i w = uw = a^l$  for some  $l < k$ . So  $xuv^i wz = a^l b^k \notin L$ .

Thus  $L$  satisfies  $(\neg P)$ , so  $L$  isn't regular.

## Use of pumping lemma: general pattern

On the previous slide, the full argument is in black, whereas the **parenthetical comments in blue** are for pedagogical purposes only.

The comments emphasise the care that is needed in dealing with the quantifiers in the property ( $\neg P$ ). In general:

- **You are not allowed to choose** the number  $k \geq 0$ . Your argument has to work for every possible value of  $k$ .
- **You have to choose** the strings  $x, y, z$ , which might depend on  $k$ . You must choose these to satisfy  $xyz \in L$  and  $|y| \geq k$ . Also,  $y$  should be chosen cunningly to **'disallow pumping'** ...
- **You are not allowed to choose** the strings  $u, v, w$ . Your argument has to work for every possible decomposition of  $y$  as  $uvw$  with  $v \neq \epsilon$ .
- **You have to choose** the number  $i (\neq 1)$  such that  $xuv^i wz \notin L$ . Here  $i$  might depend on all the previous data.

## Example 2

Consider  $L = \{a^{n^2} \mid n \geq 0\}$ .

We show that  $L$  satisfies  $(\neg P)$ :

Suppose  $k \geq 0$ .

Let  $x = a^{k^2-k}$ ,  $y = a^k$ ,  $z = \epsilon$ , so  $xyz = a^{k^2} \in L$ .

Given any splitting of  $y$  as  $uvw$  with  $v \neq \epsilon$ , we have  $1 \leq |v| \leq k$ .

So taking  $i = 2$ , we have  $xuv^2wz = a^n$  where  $k^2 + 1 \leq n \leq k^2 + k$ .

But there are no perfect squares between  $k^2$  and  $k^2 + 2k + 1$ .

So  $n$  isn't a perfect square. Thus  $xuv^2wz \notin L$ .

Thus  $L$  satisfies  $(\neg P)$ , so  $L$  isn't regular.

## Reading and prospectus

This concludes the part of the course on regular languages.

**Relevant reading:** Kozen chapters 11, 12.

Next time, we start on the next level up in the Chomsky hierarchy:  
**context-free languages.**