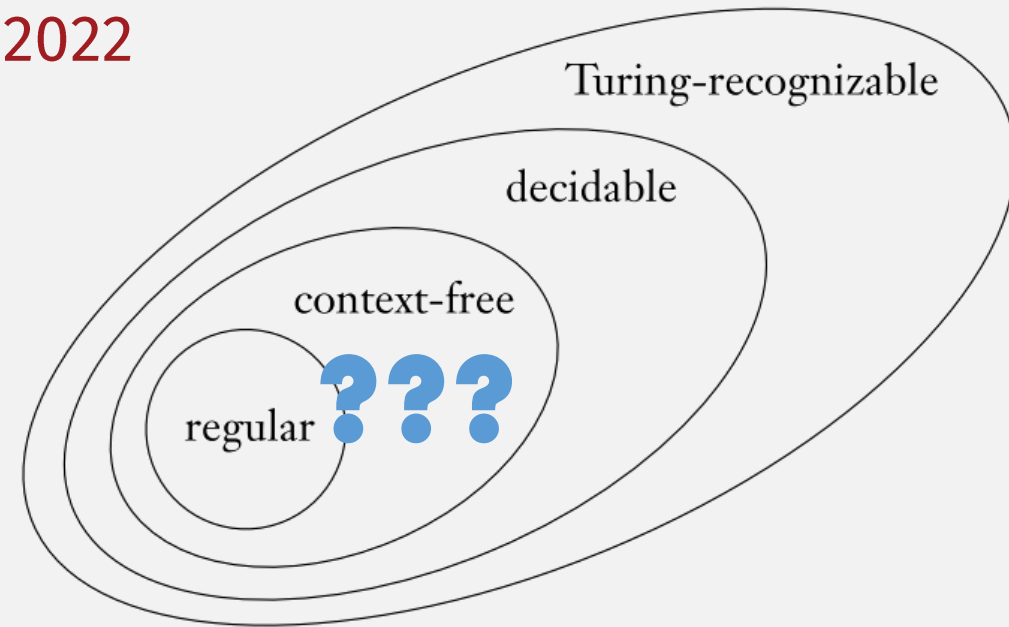


UMB CS 420

Non-Regular Languages

Tuesday, October 11, 2022



Announcements

- HW 3 in
 - ~~Due Sun 10/9 11:59pm EST~~
- HW 4 out
 - Due Sun 10/16 11:59pm EST

So Far: Regular or Not?

- Many ways to prove that a language is regular:

- Construct a **DFA** recognizing it
- Construct an **NFA** recognizing it
- Come up with a **regular expression** describing the language

M recognizes language A
if $A = \{w \mid M \text{ accepts } w\}$

- Regular Expression \Leftrightarrow NFA \Leftrightarrow DFA \Leftrightarrow Regular Language

- But **not all languages are regular!**

- Most programming language syntaxes are not regular
 - e.g., language of all python programs, or all HTML/XML pages, are not regular
- That means:
 - There's no DFA or NFA recognizing that language
 - It can't be described with a regular expression (a common mistake)!

Someone Who Did Not T

Regex match open tags except XHTML self-con

Asked 10 years, 10 months ago Active 1 month ago Viewed 2.9m times

I need to match all of these opening tags:

1553

```
<p>  
<a href="foo">
```

Trying to use regular expressions to describe the non-regular HTML language

But not these:

6572

You can't parse [X]HTML with regex. Because HTML can't be parse

4414

Regex is not a tool that can be used to correctly parse HTML. As I h
HTML-and-regex questions here so many times before, the use of r
allow you to consume HTML. Regular expressions are a tool that is
sophisticated to understand the constructs employed by HTML. HTM

regular language and hence cannot be parsed by regular expression

queries are not equipped to break down HTML into its meaningful pa
times but it is not getting to me. Even enhanced irregular regular exp
used by Perl are not up to the task of parsing HTML. You will never

HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The <center> cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he

comes. HTML-plus-regex will liquify the nerves of the sentient whilst you observe, our psyche withering in the onslaught of horror. Regēx-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the transgression of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML h

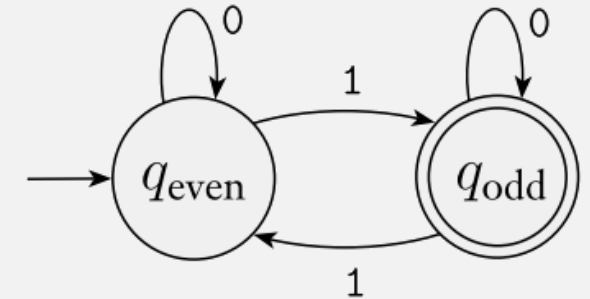
ummm ...

of dread torture and security holes *using regex as a tool to process HTML* establishes a breach *between this world* and the dread realm of corrupt entities (like SGML entities, but *more corrupt*) a mere glimpse of the world of regex parsers for HTML will instantly transport a programmer's consciousness into a world of ceaseless screaming, he comes, the pestilent slithy regex-infection will devour your HTML parser, application and existence for all time like Visual Basic only worse he comes he comes do not fight he comes, his unholy radiancé destroying all enlightenment, HTML tags *leaking from your eyes like liquid pain*, the song of regular expression parsing will extinguish the voices of mortal man from the sphere I can see it can you see if it is beautiful the final snuffing of the lies of Man ALL IS LOST ALL IS LOST the pony he comes he comes he comes the Fichor permeates all MY FACE MY FACE oh god no NO NOOOO NO stop the angles are not real ZALGO IS TONY THE PONY, HE COMES

Have you tried using an XML parser instead?

Flashback: Designing DFAs or NFAs

- Each state “stores” some information
 - E.g., q_{even} = “seen even # of 1s”, q_{odd} = “seen odd # of 1s”.
 - Finite states = finite amount of info (must decide in advance)
- This means DFAs can’t keep track of an arbitrary count!
 - would require infinite states



A Non-Regular Language

An arbitrary count

$$L = \{ 0^n 1^n \mid n \geq 0 \}$$

- A DFA recognizing L would require infinite states! (impossible)
 - States representing zero 0s, one 0, two 0s, ...
- This language represents the essence of many PLs, e.g., HTML!
 - To better see this replace:
 - “0” with “<tag>” or “(“
 - “1” with “</tag>” or “)”
- The problem is tracking the nestedness
 - Regular languages cannot count arbitrary nesting depths
 - E.g., `if { if { if { ... } } }`
 - So most programming language syntax is not regular!

Still, how do we
prove non-regularness?

Prove: Ghosts Do Not Exist



It's hard to prove that something is not true!

In some cases, it's possible, but typically requires complicated proof techniques!

So: proving a language is not regular...
is harder than proving a language is regular

A Lemma About Regular Languages

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Specifically, all regular languages satisfy these 3 conditions!

This lemma describes a property that all regular languages have.

Hint: This is an "If X then Y " statement

Note: this lemma only applies to known regular languages!

Can we use this to prove that language is regular?

NO (but we already know how to do that anyways)

(but maybe it **can** be used to prove that a language is **not regular**!)

Equivalence of Conditional Statements

- Yes or No? “If X then Y ” is equivalent to:
 - “If Y then X ” (converse)
 - No!
 - “If not X then not Y ” (inverse)
 - No!
 - “If not Y then not X ” (contrapositive)
 - Yes!

If-then statement

... then the language is **not** regular

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Equivalent (**contrapositive**):
If any of these are **not** true ...

Contrapositive:

“If X then Y ” is equivalent to “If **not** Y then **not** X ”

A Lemma About Regular Languages

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

All regular languages satisfy these three conditions!

Specifically, these conditions apply to strings in the language longer than length p

NOTE: **Lemma doesn't tell you an exact p !**
(just that there must exist "some" p)

The Pumping Lemma: Finite Lang

Conclusion: pumping lemma is only interesting for infinite langs!
(containing strings with repeatable parts)

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$

Lemma doesn't say what p is!
Just that there is "some length"

Possible p for finite langs?

How about:
 $\text{Length}(\text{longest string}) + 1$

So finite langs (specifically, all strings in the language "of length at least p ") must satisfy these conditions

strings in the language with at least length p ? **None!**

Therefore, all strings with length at least p satisfy the pumping lemma conditions! 😊

Example: a finite language {"ab", "cd"}

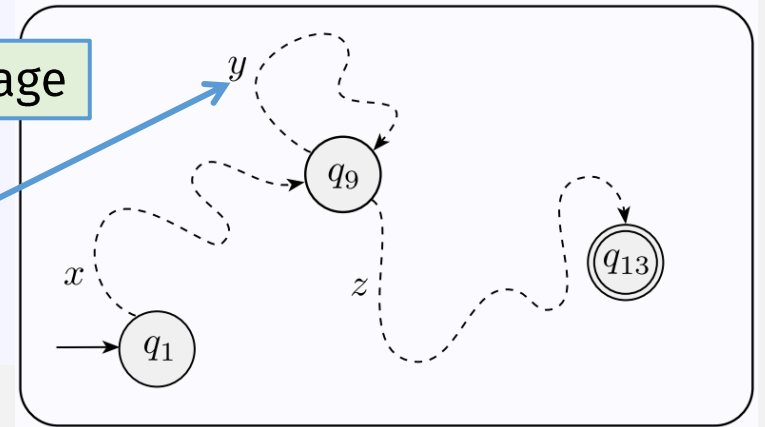
- All finite langs are regular (can easily construct DFA/NFA recognizing them)

The Pumping Lemma, a Closer Look

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$, ← “pumped” string still in language
2. $|y| > 0$, and
3. $|xy| \leq p$.

(“long enough”) strings of length p have a repeatable (“pumpable”) part



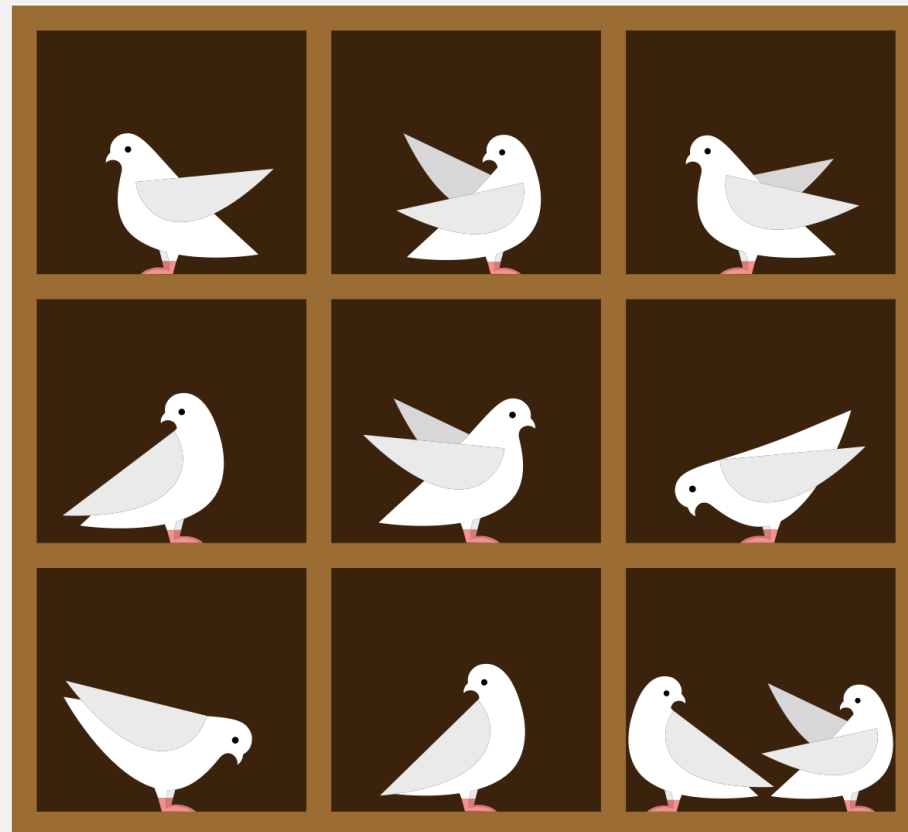
Strings that have a repeatable part can be split into:

- x = part before any repeating
- y = repeated (or “pumpable”) part
- z = part after any repeating

This makes sense because DFAs have finite states, so for “long enough” (i.e., \geq length p) inputs, some state must repeat

e.g., “long enough length” = $p = \# \text{ states} + 1$
(The Pigeonhole Principle)

The Pigeonhole Principle



If # birds > # holes,
then there must be > 1
bird in some hole

The Pumping Lemma, a Closer Look

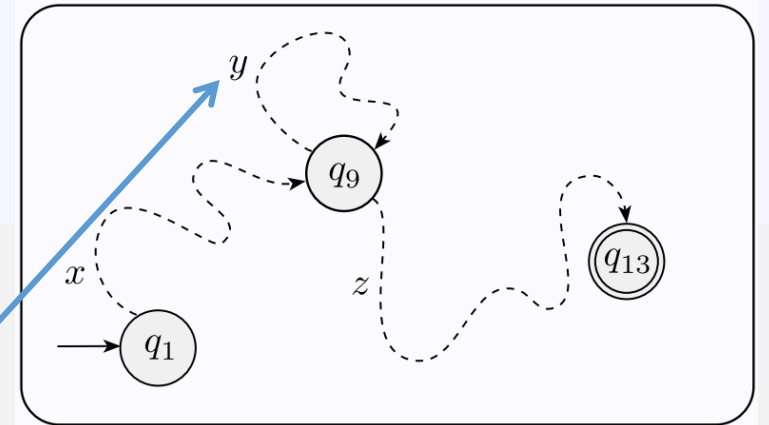
Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

So a substring that can repeat once, can also be repeated any number of times

In essence, the pumping lemma is a theorem about the structure of repeatable patterns in regular languages

Also, this is the only way for regular languages to repeat (Kleene star)



"long enough length" = $p = \# \text{ states} + 1$
(some state must repeat)

The Pumping Lemma: Infinite Languages

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,

2. $|y| > 0$, and

3. $|xy| \leq p$.

“pumpable” part of string

Note: “pumpable” part cannot be empty

E.g., “010” $\in A$, so pumping lemma says splittable into three parts xyz
- $x = 0, y = 1, z = 0$

Example: *infinite* language $A = \{“00”, “010”, “0110”, “01110”, \dots\}$

- It’s regular bc it has regular expression 01^*0

... and “pumping” (repeating) middle y part creates a string that is still in the language

- repeat once ($i = 1$): “010”,
- repeat twice ($i = 2$): “0110”,
- repeat three times ($i = 3$): “01110”

Summary: The Pumping Lemma ...

- ... states properties that are true for all regular languages
- ... specifically, properties about repetition in regular languages

IMPORTANT:

- The Pumping Lemma cannot prove that a language is regular!
- But ... we can use it to prove that a language is not regular

If-then statement

... then the language is **not** regular

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:


1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Equivalent (**contrapositive**):
If any of these are **not** true ...

Contrapositive:

“If X then Y ” is equivalent to “If **not** Y then **not** X ”

Kinds of Mathematical Proof

- Deductive Proof
 - Logically infer conclusion from known definitions and assumptions
- Proof by induction
 - Use to prove properties of recursive definitions or functions
- Proof by contradiction 
 - Proving the contrapositive

How To Do Proof By Contradiction

3 easy steps:

1. Assume the opposite of the statement to prove
2. Show that the assumption leads to a contradiction
3. Conclude that the original statement must be true

Pumping Lemma: Non-Regularity Example

Let B be the language $\{0^n 1^n \mid n \geq 0\}$. We use the pumping lemma to prove that B is not regular. The proof is by contradiction.

Want to prove: $0^n 1^n$ **is not** a regular language

Proof (by contradiction):

Now we must find a contradiction ...

- Assume: $0^n 1^n$ **is** a regular language
 - So it must satisfy the pumping lemma
 - I.e., all strings \geq length p are pumpable
- Counterexample = $0^p 1^p$

Pumping lemma → If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^i z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Reminder: Pumping lemma says:
all strings $0^n 1^n \geq$ length p **are splittable** into xyz where y is pumpable

So find string \geq length p that is **not splittable** into xyz where y is pumpable

Want to prove: $0^n 1^n$ **is not** a regular language

Possible Split: $y = \text{all } 0\text{s}$

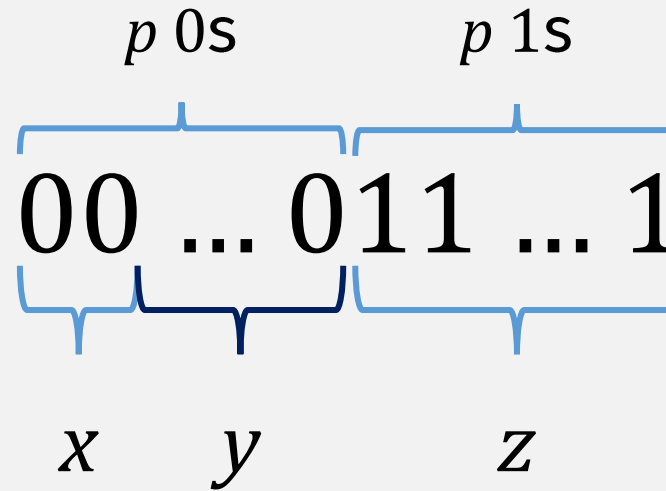
Proof (by contradiction):

- Assume: $0^n 1^n$ **is** a regular language

- So it must satisfy the pumping lemma
- I.e., all strings \geq length p are pumpable

- Counterexample = $0^p 1^p$

- Choose xyz split so y contains:
 - all 0s



- Pumping y : produces a string with more 0s than 1s

- Which is not in the language $0^n 1^n$
- So $0^p 1^p$ is not pumpable (according to pumping lemma)
- So $0^n 1^n$ is a not regular language (contrapositive)
- This is a **contradiction** of the assumption!

... then **not true**

Pumping lemma → If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^i z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Contrapositive: If **not true** ...

Reminder: Pumping lemma says:
all strings $0^n 1^n \geq$ length p are
splittable into xyz where y is pumpable

So find string \geq length p that is **not**
splittable into xyz where y is pumpable

BUT ... pumping lemma requires **only one** pumpable splitting

So the proof is not done!

Is there another way to split into xyz ?

contradiction

Want to prove: $0^n 1^n$ is **not** a regular language

Possible Split: $y = \text{all } 1\text{s}$

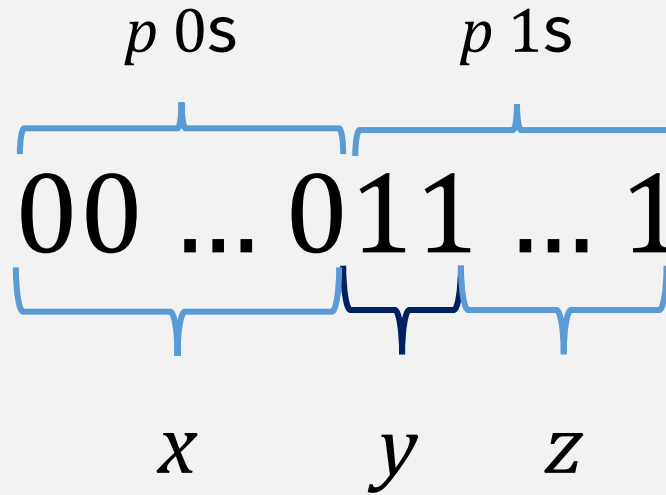
Proof (by contradiction):

- Assume: $0^n 1^n$ **is** a regular language

- So it must satisfy the pumping lemma
- i.e., all strings \geq length p are pumpable

- Counterexample = $0^p 1^p$

- Choose xyz split so y contains:
 - all 1s



- Is this string pumpable?
 - No!
 - By the same reasoning as in the previous slide

Is there another way to split into xyz ?

Want to prove: $0^n 1^n$ is **not** a regular language

Possible Split: $y = 0$ s and 1 s

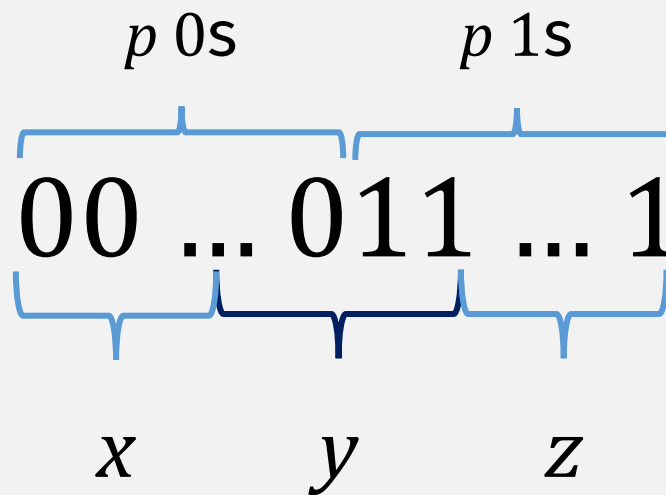
Proof (by contradiction):

- Assume: $0^n 1^n$ **is** a regular language

- So it must satisfy the pumping lemma
- I.e., all strings \geq length p are pumpable

- Counterexample = $0^p 1^p$

- Choose xyz split so y contains:
 - both 0s and 1s



Did we examine every possible splitting?

Yes! QED

- Is this string pumpable?

- No!
- Pumped string will have equal 0s and 1s
- But they will be in the wrong order: so there is still a **contradiction!**

But maybe we didn't have to ...

The Pumping Lemma: Condition 3

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.


The repeating part y ...
must be in the first p characters!

p 0s
 $\underbrace{00 \dots 0}_{y \text{ must be in here!}} 11 \dots 1$

The Pumping Lemma: Pumping Down

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.



Repeating part y must be non-empty ...
but can be repeated zero times!

Example: $L = \{0^i1^j \mid i > j\}$

Want to prove: $L = \{0^i 1^j \mid i > j\}$ **is not** a regular language

Pumping Down

Proof (by contradiction):

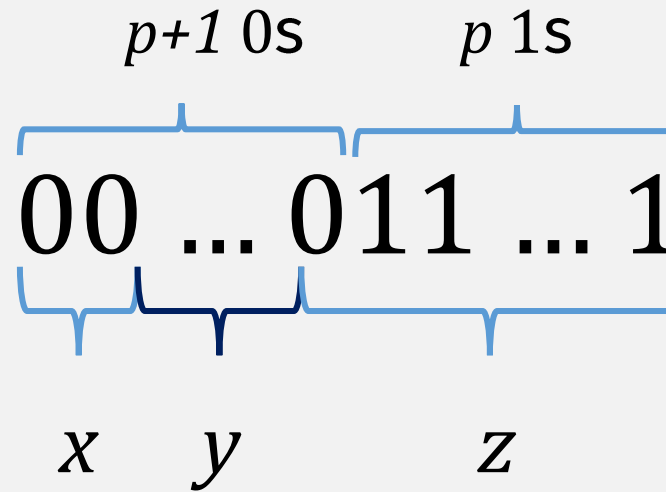
- Assume: L **is** a regular language

- So it must satisfy the pumping lemma
- I.e., all strings \geq length p are pumpable

- Counterexample = $0^{p+1} 1^p$

- Choose xyz split so y contains:

- all 0s
- (Only possibility, by condition 3)

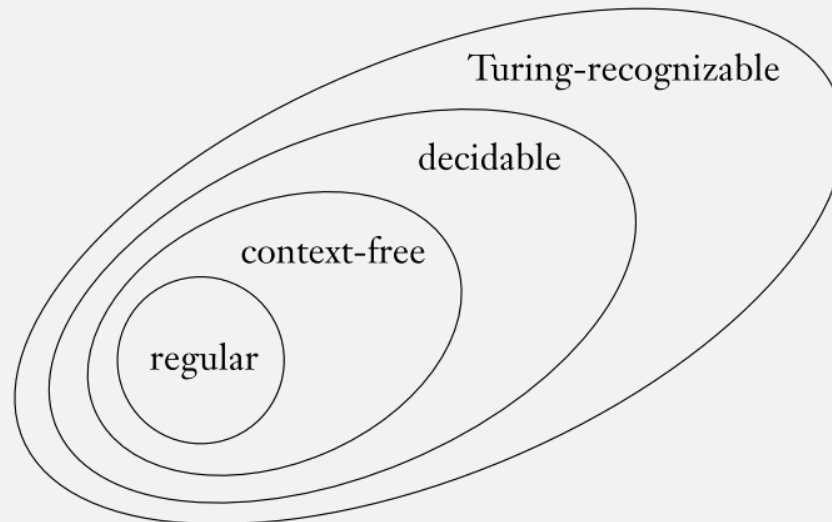


- Repeat y zero times (**pump down**): produces string with $\#$ 0s \leq $\#$ 1s
 - Which is not in the language $\{0^i 1^j \mid i > j\}$
 - So $\{0^i 1^j \mid i > j\}$ does not satisfy the pumping lemma
 - So it is a not regular language
 - This is a **contradiction** of the assumption!

contradiction

Next Time (and rest of the Semester)

- If a language is not regular, then what is it?
- There are many more classes of languages!



Check-in Quiz 10/11

On gradescope