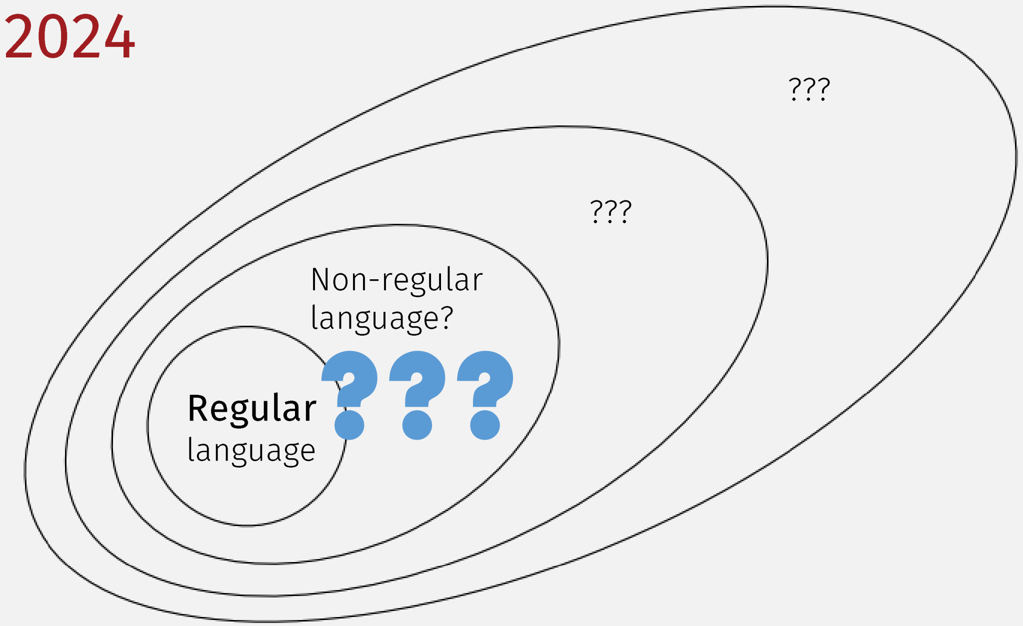
UMB CS 420

Non-Regular Languages

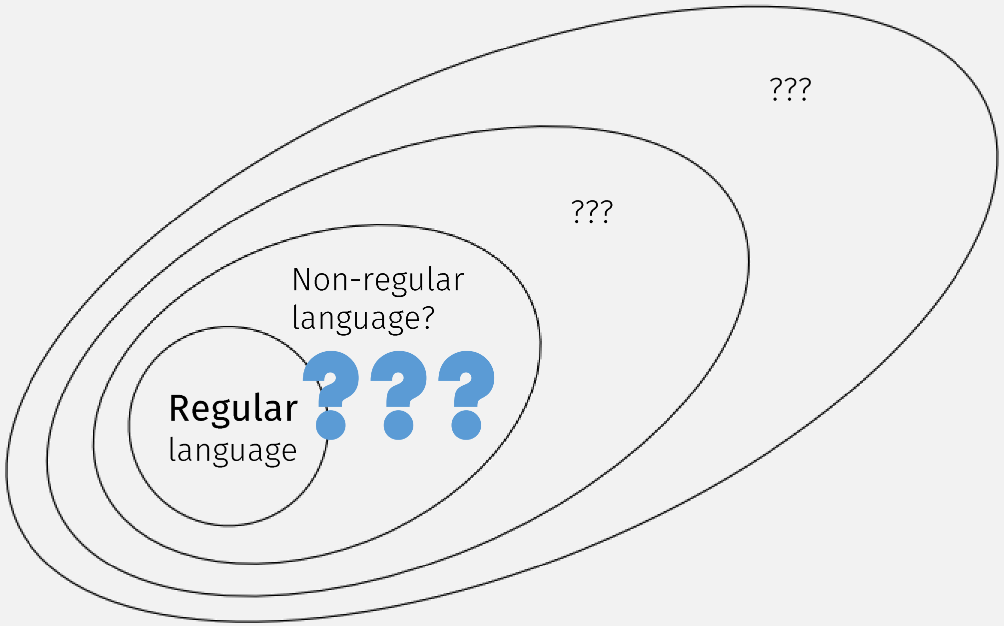
Wednesday March 6, 2024



Announcements

- HW 4 out
  - Due Mon 3/18 12pm EST (noon)
  - (After spring break)

- Problem 4, Part 2c Update:
  - Prove the statement for
    - 1 base case
    - 1 recursive case



A ***language*** is a set of strings.

So Far: Regular or Not?

M ***recognizes language*** A

if $A = \{w |\ M \text{ accepts } w\}$

- Many ways to **prove a <u>language is regular</u>:**
  - Construct a **DFA** recognizing it
  - Construct an **NFA** recognizing it
  - Create a **regular expression** describing the language

- Bc we proved: **Regular Expression ⇔ NFA ⇔ DFA ⇔ Regular Language**

- But ***<u>not</u>*** **all languages** are **regular!**
  - E.g., <u>**programming language syntaxes**</u> are <u>**not regular**</u>
    - **language of all Python programs,** or **all HTML/XML pages,** are <u>**not regular**</u>
  - That means:
    - There is ***<u>no</u>*** **DFA** or **NFA** that: **accepts valid Python programs** (and **rejects invalid ones**)
    - And, there is ***<u>no</u>*** **regular expression** that: **describes all valid Python** or **HTML programs** (a common mistake)**!**

Someone Who Did Not Pa

RegEx match open tags except XHTML self-con

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

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

1553

6572

I need to match all of these opening tags:

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

But not these:

4414

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

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 re

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 expressior

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

Someone who paid attention in 420 …

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-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession 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 has doomed humanity to an eternity 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 glimp*se of the world of rege**x parsers for HTML will ins**tantly transport a p*rogrammer's consciousness* into a w*orl*d of ceaseless screaming, he comes, the pestilent slithy regex-infection wil**l devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fi*ght h**e com̡e̶s, ̕h̵i**s un̨ho͞ly radiańcé destro҉ying all enli̍̈́̂̈́ghtenment, HTML tags **lea͠ki̧n͘g fr̶ǫm ̡yo͟ur eye͢s̸ ̛l̕ik͏e liq**uid pain, the song of re̸gular expression parsing will exti*nguish the voices of mor*tal man from the sp*here I can see it can you see ̲͚̖͔̙î̩́t̲͎̩̱͔́̋̀ it is beautiful t*he f`inal snuf`fing o*f the lie*s of Man ALL IS LOŚ͖̩͇̗̪̏̈́T ALL I**S LOST the pon̷y he comes he c̶̮omes he comes the ich**or permeates all MY FAC**E MY FACE ᵒh god n**o NO NOO̼**O**O N**Θ stop the an*̶͑̾̾̅ͫ͏̙̤g͇̫͛͆̾ͫ̑͆l͖͉̗̩̳̟̍ͫͥͨe̠̅s* `͎a̧͈͖r̽̾̈́͒͑e` n**ot rè̑ͧ̌aͨl̘̝̙̃ͤ͂̾̆ ZA̡͊͠͝LGΌ ISͮ̂҉̯͈͕̹̘̱ TO͇̹̺ͅƝ̴ȳ̳ TH̘Ë͖́̉ ͠P̯͍̭O̚N̐Y̡ H̸̡̪̯ͨ͊̽̅̾̎Ȩ̬̩̾͛ͪ̈́̀́͘ ̶̧̨̱̹̭̯ͧ̾ͬC̷̙̲̝͖ͭ̏ͥͮ͟Oͮ͏̮̪̝͍M̲̖͊̒ͪͩͬ̚̚͜Ȇ̴̟̟͙̞ͩ͌͝S̨̥̫͎̭ͯ̿̔̀ͅ**

ummm … this is getting a little weird

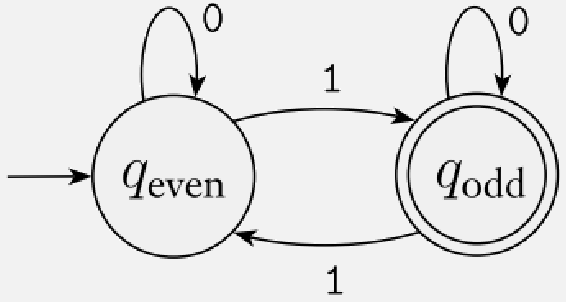
very weird …

?????

hmm … what's this?

Have you tried using an XML parser instead?

Flashback: Designing DFAs or NFAs



- Each state "remembers" information about input
    - E.g., q_{even} = "seen even # of 1s"
      q_{odd} = "seen odd # of 1s"
  - But <u>finite</u> states = <u>finite</u> amount of info storage (and must decide in advance)

- So <u>DFAs can't remember</u> an <u>arbitrary count</u>!
  - would require infinite states

A Non-Regular Language

An arbitrary count

L = { $\mathtt{0}^n\mathtt{1}^n$ | $n \geq 0$ }

- A DFA recognizing L would **require infinite states!** (impossible)
  - States representing zero `0`s seen, one `0` seen, two `0`s, …
- **This language is the same as many PLs,** e.g., **HTML!**
  - To better see this replace:
    - "`0`" with "`<tag>`" or "`(`"
    - "`1`" with "`</tag>`" or "`)`"
- The Problem: **remembering <u>nestedness</u>**
  - Need to count arbitrary nesting depths
    - E.g., `if { if { if { … } } }`
  - Thus: **most programming language syntax** is <u>**not regular**</u>!

So, how can we *<u>prove</u>* **non-regularness?**

Prove: Spider-Man does not exist ???

In general, **proving something not true** is **different** (and harder) **than proving it true**

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

We know how to: **prove** a **language** is **regular**
Can we: **prove** a **language** is **not regular?**

YES! but **requires a new proof technique!**

Step 1: **find** a **fact that is true for all regular languages ...**

A Fact (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. $|xy| \leq p$.

This is an "If X then Y" statement

Remember: To ***use*** an "If X then Y" statement,
1. *prove* X is true,
2. *conclude* that Y is true

Flashback: The Modus Ponens Inference Rule

If we know these statements are true …

- If P then Q

- P

Then we also know this statement is true …

- Q

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. $|xy| \leq p$.

... then we can *conclude* ...

Uh ... whatever this says ...

To *use* The **Pumping lemma** for a language A ...

... first *prove* that A is a regular language ...

Q: Can we *use* The **Pumping lemma** to **prove** that a **language** is **regular?**

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

NO (but we already know many other ways to do that!)

Equivalence of Conditional Statements

- Yes or No? "If X then Y" **is equivalent to:**
  - "If Y then X" (**converse**) Seen Previously
    - 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 $\boxed{\text{If } A \text{ is a regular language, then}}$ 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 <u>equivalent</u> to "If **not** *Y* then **not** *X*"

Logical Inference Rules

Modus Ponens

Premises (known facts)

- If P then Q
- P is true

Conclusion (new fact)

- Q is true

Modus Tollens (contrapositive)

Premises (known facts)

- If P then Q
- Q is *not* true

Step 1: **find** a **fact that is true for all regular languages** …

Step 2: where **the fact can be proven not true!**

Conclusion (new fact)

- P is *not* true

How to: **prove** a **language** is **not regular?**

Fact About Regular Languages: Details

Pumping lemma If A is a regular language, then $\boxed{\text{there is a number } p}$ (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. $|xy| \leq p$.

Conditions are on **strings in the language** with <u>**length**</u> $\geq$ ***p***

<u>**Any regular language**</u> **satisfies these three conditions!**

The exact value of p differs for every regular language

<u>NOTE</u>:
- **Lemma doesn't give an exact** p**!**
- Only that **there is *some* string length** p ...

The Pumping Lemma: Finite Lang

Conclusion: **pumping lemma is only interesting for infinite langs!** (which contain strings with **repeating parts**)

Lemma doesn't say what p is! Just that "there is a 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 $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

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

Possible p for finite langs?

How about:
p = LENGTH(longest string) + 1

 # strings in the language with length ≥ p? **None!**

Therefore, **all strings with length ≥ p satisfy the pumping lemma conditions!** ☺

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

- **All finite languages are regular!**
- (can easily construct DFA/NFA/Regular Expression recognizing them)

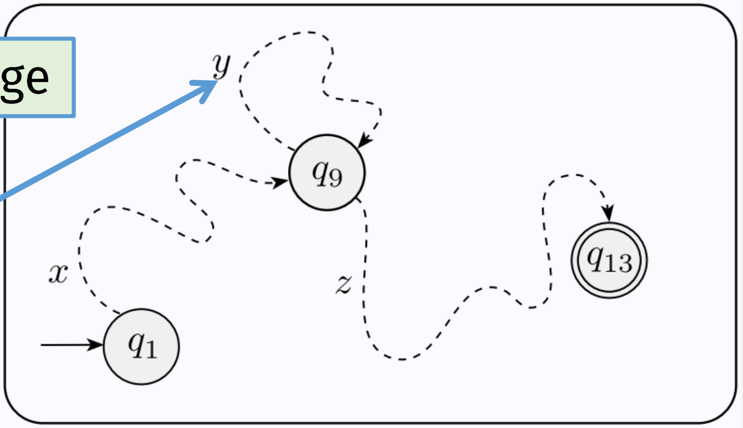
Langs With Strings With Repeatable Parts

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 $i \geq 0$, $x\boxed{y^i}z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

"pumped" string still in language

<u>repeatable</u> ("pumpable") part
(= repeatable state in DFA!)



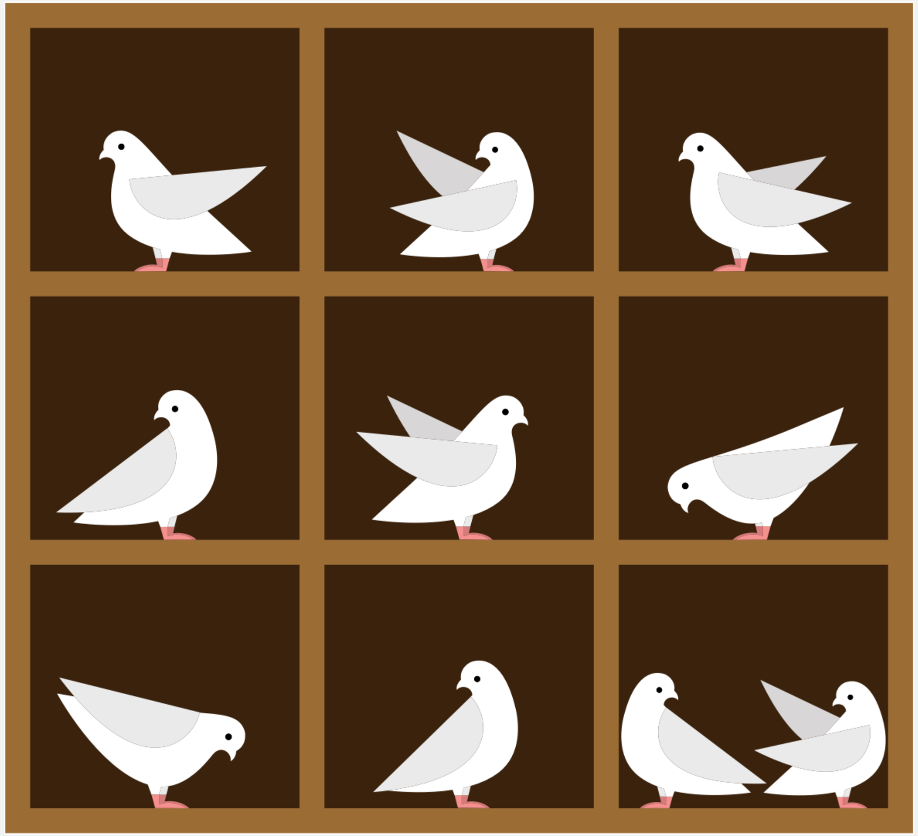
Strings that **have a <u>repeatable</u> part** can be **split** into 3 parts:

- x = part <u>before</u> any repeating
- y = <u>repeated</u> (or "pumpable") part
- z = part <u>after</u> any repeating

DFAs have finite states,
so for "long enough" (i.e., length ≥ p) inputs,
<u>some state must repeat!</u>

e.g., "long enough length" = p = **# states** +1 (The Pigeonhole Principle)

The Pigeonhole Principle



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

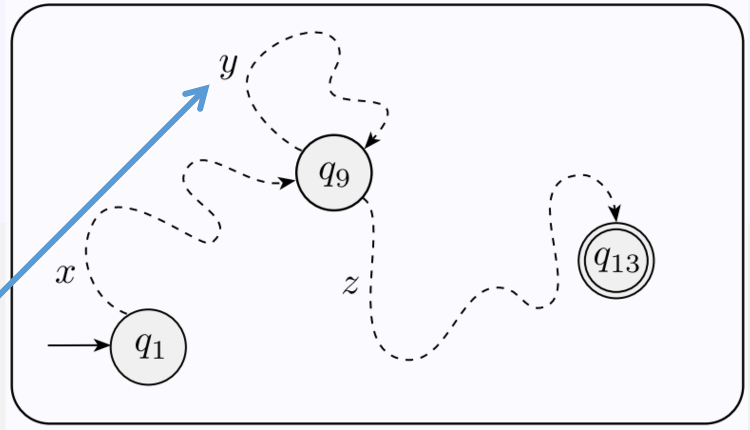
The Pumping Lemma, a Closer Look

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 $i \geq 0$, $x\boxed{y^i}z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

So a **substring** that *can* **repeat** <u>**once**</u>, *can also* be **repeated** <u>**any number of times**</u>

This is the <u>only</u> way for regular languages to have repeating patterns (Kleene star)



In essence, **the Pumping lemma is a theorem about <u>repeating patterns in regular languages</u>**

"**long enough length**" = p = **# states** +1
(some state must repeat)

In-class exercise: Infinite Languages

Split the string "010" into three parts xyz, e.g.

x = ??, y = ??, z = ??

so that <u>repeating</u> y part any number of times results in a new string still in A

Now do "0110":

x = ??, y = ??, z = ??

<u>Example</u>: *infinite* language A = {"00", $\boxed{\text{“010”}}$, $\boxed{\text{“0110”}}$, "01110", ...}

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" ∈ A, so pumping lemma says it's splittable into three parts xyz, e.g.
$x = 0, \quad y = 1, \quad z = 0$

Example: ***infinite*** **language** A = {"00", "010", "0110" , "01110", ...}

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

Pumping lemma summary:
"All infinite regular languages must have a star in its regular expression"!

... and "pumping" (repeating) middle y part creates a string that is still in the language
- repeat once ($i = 1$): "0**1**0",
- repeat twice ($i = 2$): "0**11**0",
- repeat three times ($i = 3$): "0**111**0"

Summary: The Pumping Lemma …

- … states properties that are true for all regular languages
- … specifically, properties about "long enough" strings in reg. langs
- In general, it describes repeating patterns in reg. langs

IMPORTANT:

- The **Pumping lemma** cannot prove that a language is **regular**!
- But … we can use it to prove that a language is **not regular**

Pumping lemma summary:
"All infinite regular languages must have a star in its regular expression"!

… by showing that the repeating pattern is not expressible with a star regular expression!

If-then statement

... then **the language is <u>not</u> regular**

Pumping lemma $\boxed{\text{If } A \text{ is a regular language, then}}$ 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 <u>not</u> true ...**

Contrapositive:
"If *X* then *Y*" is <u>equivalent</u> 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. <u>Assume:</u> the <u>opposite</u> **of** the **statement to prove**

2. <u>Show:</u> the **assumption leads to** a <u>contradiction</u>

3. <u>Conclude:</u> the **original statement must be true**

Pumping Lemma: Non-Regularity Example

This repetition pattern cannot be expressed with a star regular expression?

Let B be the language $\{0^n1^n | n \geq 0\}$. We use the pumping lemma to prove that B is not regular. The proof is by contradiction.

Pumping lemma summary:
"All infinite regular languages must have a star in its regular expression"!

... **by showing** that the repeating pattern is not expressible with a star regular expression!

Want to prove: 0^n1^n **is not** a regular language

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. $|xy| \leq p$.

Reminder: Pumping lemma says: all strings $0^n1^n \geq$ 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

Proof (by contradiction):

Now we must find a contradiction …

- Assume: 0^n1^n **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings $\geq$ length p are pumpable
- Counterexample = 0^p1^p

We must show that there is no possible way to split this string to satisfy the conditions of the pumping lemma!

Want to prove: 0^n1^n **is not** a regular language

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

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

Possible Split: y = all 0s

Reminder: Pumping lemma says: all strings $0^n1^n \geq$ 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

Proof (by contradiction):

Contradiction?

- Assume: 0^n1^n **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings $\geq$ length p are pumpable

Not yet!

- Counterexample = 0^p1^p

$$\overbrace{00\ldots0}^{p\ 0\text{s}}\overbrace{11\ldots1}^{p\ 1\text{s}}$$

$x \quad y \quad z$

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

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

So the proof is not done!

Is there another way to split into xyz ?

- Pumping y: produces a string with **more 0s than 1s**
  - ... not in the language 0^n1^n !
  - So 0^p1^p **is not pumpable?** (according to pumping lemma)
  - So 0^n1^n is a not regular language? (contrapositive)
  - This is a **contradiction** of the assumption?

Want to prove: 0^n1^n is **not** a regular language

Possible Split: y = all 1s

Proof (by contradiction):

- Assume: 0^n1^n **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings $\geq$ length p are pumpable
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - $\boxed{\text{all 1s}}$

$$\underbrace{\overbrace{00\ldots0}^{p\text{ 0s}}}_{x}\overbrace{\underbrace{11}_{y}\underbrace{\ldots1}_{z}}^{p\text{ 1s}}$$

Is there another way to split into xyz ?

- Is this string pumpable (repeating y produces string still in 0^n1^n)?
  - No!
  - By the same reasoning as in the previous slide

Want to prove: 0^n1^n is **not** a regular language

Possible Split: y = 0s and 1s

Proof (by contradiction):

- Assume: 0^n1^n **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings ≥ length p are pumpable
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - $\boxed{\text{both 0s and 1s}}$

$$\overbrace{00\ldots0}^{p\text{ 0s}}\overbrace{11\ldots1}^{p\text{ 1s}}$$

$$\underbrace{00\ldots}_{x}\underbrace{011}_{y}\underbrace{\ldots1}_{z}$$

Did we examine every possible splitting?

Yes! QED

- Is this string pumpable (repeating y produces string still in 0^n1^n)?
  - 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 did'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!

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

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 $\boxed{i \geq 0,}$ $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

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

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

Want to prove: $L = \{0^i1^j \mid i > j\}$ **is not** a regular language

Pumping Down

Proof (by contradiction):

contradiction

- Assume: L **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings ≥ length p are pumpable
- Counterexample = $0^{p+1}1^p$
- Choose xyz split so y contains:
  - all 0s
  - (Only possibility, by condition 3)

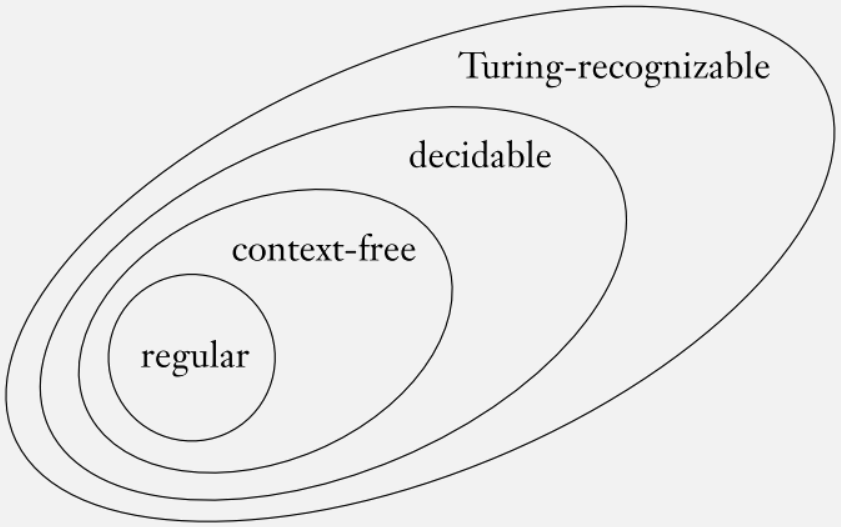
$$\overbrace{00\ldots0}^{p+1\text{ 0s}}\overbrace{11\ldots1}^{p\text{ 1s}}$$

$$\underbrace{00}_{x}\underbrace{\ldots}_{y}\underbrace{011\ldots1}_{z}$$

- Repeat y zero times (**pump down**): produces string with # 0s ≤ # 1s
  - … not in the language $\{0^i1^j \mid i > j\}$
  - So $\{0^i1^j \mid i > j\}$ does not satisfy the pumping lemma
  - So it is a not regular language
  - This is a **contradiction** of the assumption!

Next Time (and rest of the Semester)

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



Submit in-class work 3/6

On gradescope