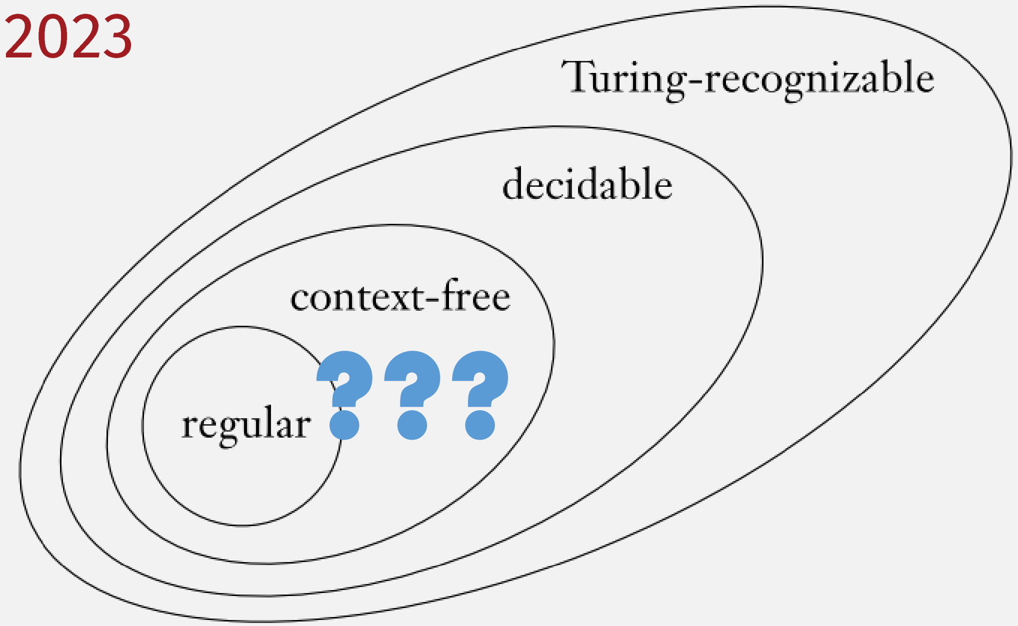
UMB CS 420

Non-Regular Languages

Wednesday, March 1, 2023



Announcements

- HW 4 out
  - Due Sun 3/5 11:59pm EST

- Submitted hw must correctly assign pages to each problem
  - Incorrectly assigned problems are marked zero
  - We will re-grade one time, if re-grade request is submitted

Quiz Preview

1. Are all languages (sets of strings) regular languages?
2. Can the Pumping Lemma be used to prove that a language is regular?
3. What can be the pumping lemma be used for?

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

So Far: Regular or Not?

M ***recognizes language*** A

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

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

- Regular Expression ⇔ NFA ⇔ DFA ⇔ Regular Language

- But *not* all languages are regular!
  - E.g., programming language syntaxes are **not regular**
    - language of all python programs, or all HTML/XML pages, are not regular
  - That means:
    - There is *no* DFA or NFA recognizing those languages
    - And they *cannot* 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

Trying to use regular expressions to describe the <u>non-regular</u> 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 parsec
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

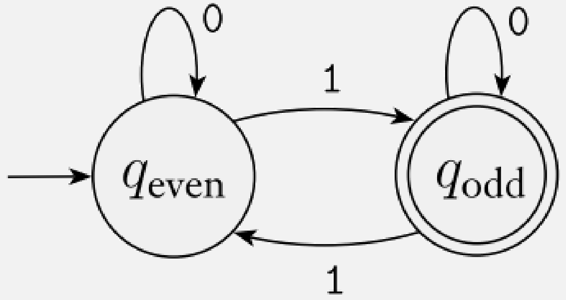
HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing 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 pars... anity to an eternity of dread torture and security holes *using*... HTML establishes a breach *between this world*... rrupt entities (like SGML entities, but *more corrupt*) *a mere glimpse* of the world of rege**x parsers for HTML will ins**tantly transport a p*rogrammer's consciousness* into a world of ceaseless screaming, he comes~~, the pestilent s~~lithy 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 comes, hi**s unholy radiancé destroying all *enlightenment, HTML tags* ***leaking from your eyes like liq**uid* pain, the song of re~~gular expression parsing~~ will exti*nguish the voices of mor**tal man from the sp**here I can see it can you see it it is beautiful t*he f`inal snuf`fing *o**f the lies of Man ALL IS LOST ALL IS L**OST the pony he comes he c~~omes he comes~~ **the** ichor permeates all MY FACE MY FACE ᵒh god **no NO NOOOO N**Θ stop *the an*gles `are` not real **ZALGO IS** TONY TH**E PONY HE COMES**

ummm … getting weird

hmm … what's this?

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"
  - But finite states = finite amount of info storage (and must decide in advance)

- So DFAs can't keep track of an arbitrary count!
  - would require infinite states

A Non-Regular Language

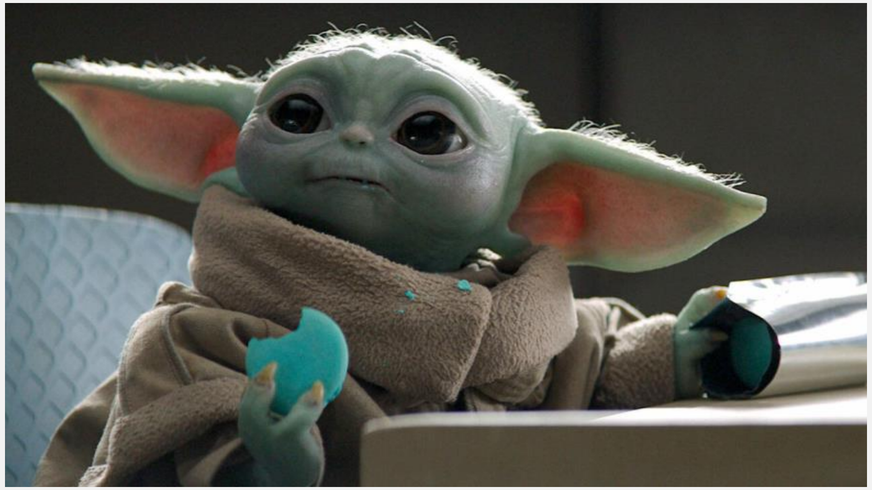
An arbitrary count

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

- A DFA recognizing L would require infinite states! (impossible)
  - States representing zero `0`s seen, one `0` seen, two `0`s, …
- This language 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 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: Aliens Do Not Exist



In general, **proving something <u>not true</u> is different** (and often harder) **than proving it <u>true</u>**

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

So: **We know how to prove a language is regular**
But **can we prove a language is not regular**?

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

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. $|xy| \leq p$.

Note: To *use* an "If X then Y" statement, first *prove* X is true, then *conclude* that Y is true

This is an "If X then Y" statement

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* ...

Strings in A satisfy these 3 conditions!
(whatever they are ...)

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

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

Q: Can we *use* The **Pumping Lemma** to prove that a 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**)
    - <u>Yes!</u>

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 <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

Conclusion (new fact)

- P is *not* true

Lemma About Regular Languages: Details

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. $|xy| \leq p$.

Any regular language satisfies these three conditions!

Specifically, these are conditions on strings in the language with length $\geq$ ***p***

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

The exact value of p depends on the regular language we're looking at!

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 a p

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

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)

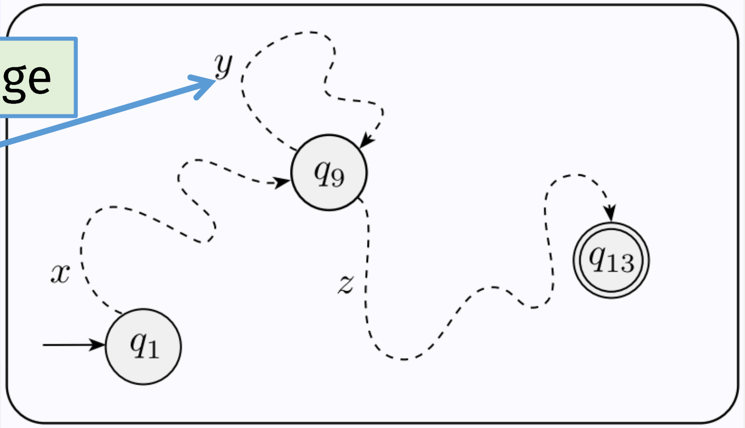
The Pumping Lemma, a Closer Look

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. $|xy| \leq p$.

"pumped" string still in language

<u>repeatable</u> ("pumpable") part



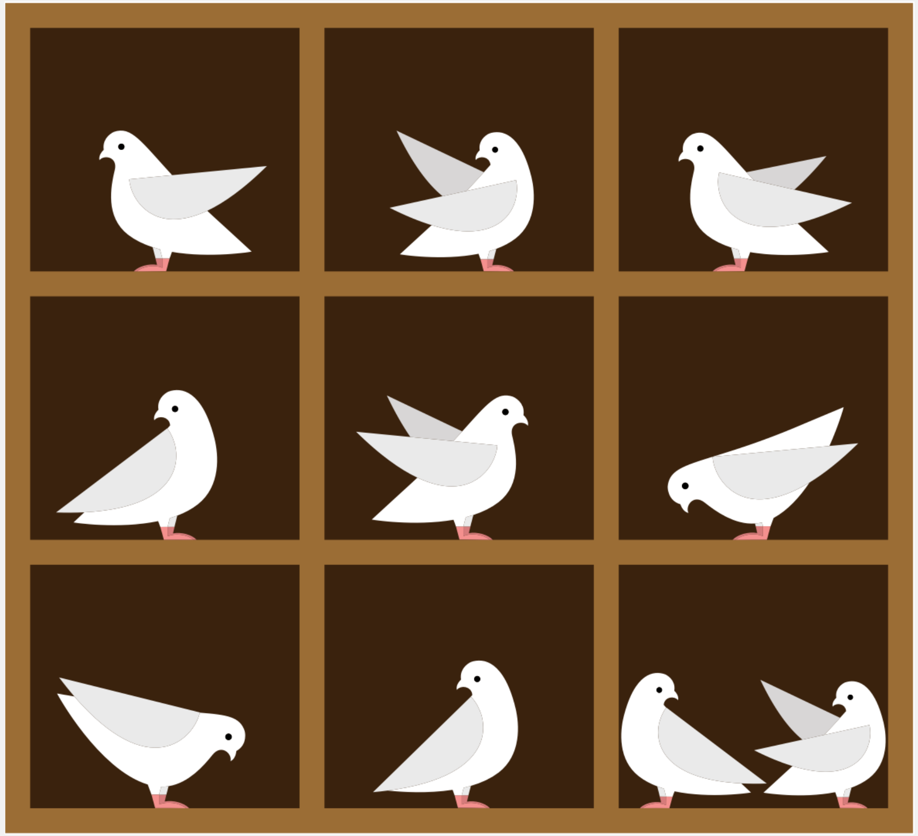
Strings that have a <u>repeatable</u> part can be split into:

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

This makes sense because 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



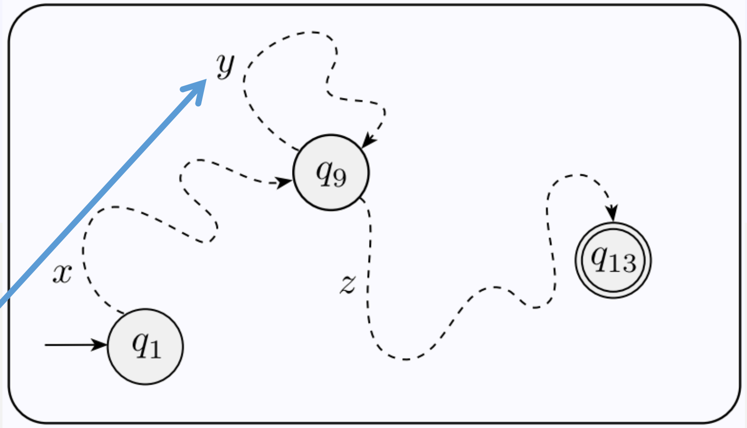
If # birds > # holes, then there must be > 1 bird in some hole

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>



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

Also, **this is the <u>only</u> way for regular languages to repeat** (Kleene star)

"**long enough length**" = p = **# 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" ∈ 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

... 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**

If-then statement

... then **the language is <u>not</u> 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 <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. 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 $\{0^n1^n | n \geq 0\}$. 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

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
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - all 0s

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

x y z

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 ≥ length p are pumpable
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - all 1s

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

- 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^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 $\geq$ length p are pumpable
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - 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?
  - 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!

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

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

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!

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

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

Pumping Down

Proof (by contradiction):

- 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)

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

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

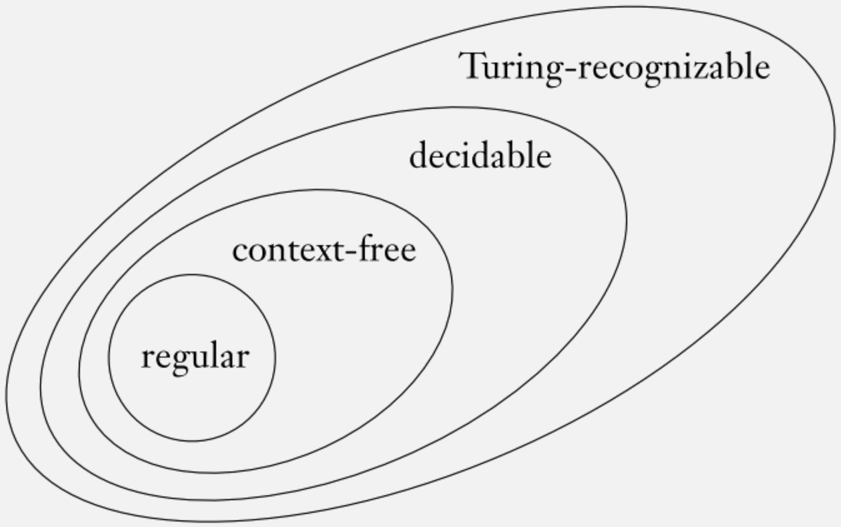
- Repeat y zero times (**pump down**): produces string with # 0s ≤ # 1s
  - … not in the language $\{0^i1^j \mid i > j\}$
  - So $\{0^i1^j \mid i > j\}$ does not satisfy the pumping lemma
  - So it is a not regular language
  - This is a **contradiction** of the assumption!

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 3/1

On gradescope