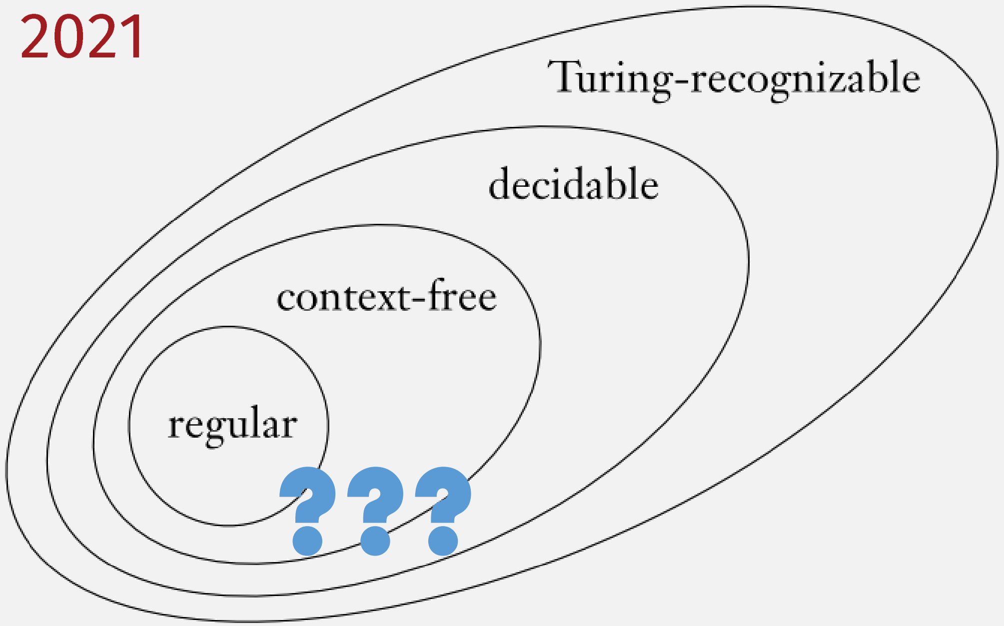
UMBCS622

Non-Regular Languages

Monday September 27, 2021



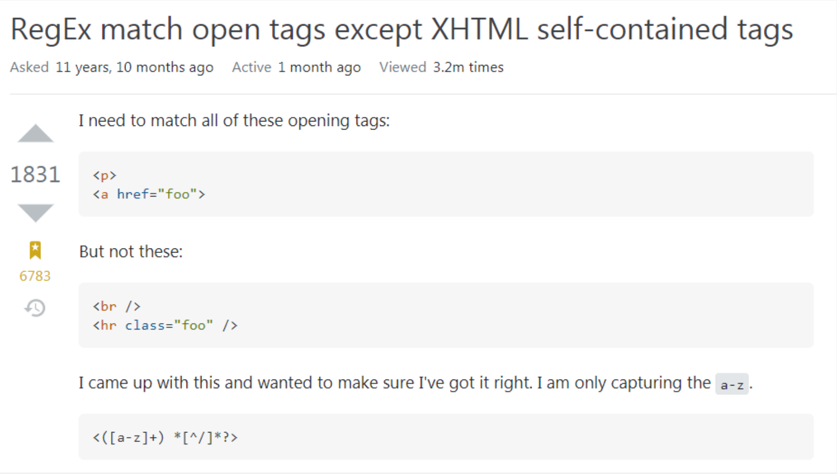
Announcements

- HW2 due yesterday
- HW3 released, due Sun 10/3 11:59pm EST
- First in-person class: next Monday 10/4
  - McCormack M01-0209

So Far: Regular or Not?

- Many ways to prove that a language is regular:
  - Construct a <u>DFA</u> or <u>NFA</u> (or GNFA)
  - Come up with a <u>regular expression</u> describing the language

- But how to show that a language is **not regular**?
  - E.g., HTML / XML is <u>not</u> a regular language
  - Can't be represented with a regular expression (common mistake)!



RegEx match open tags except XHTML self-contained tags

Asked 11 years, 10 months ago Active 1 month ago Viewed 3.2m times

1831

6783

I need to match all of these opening tags:

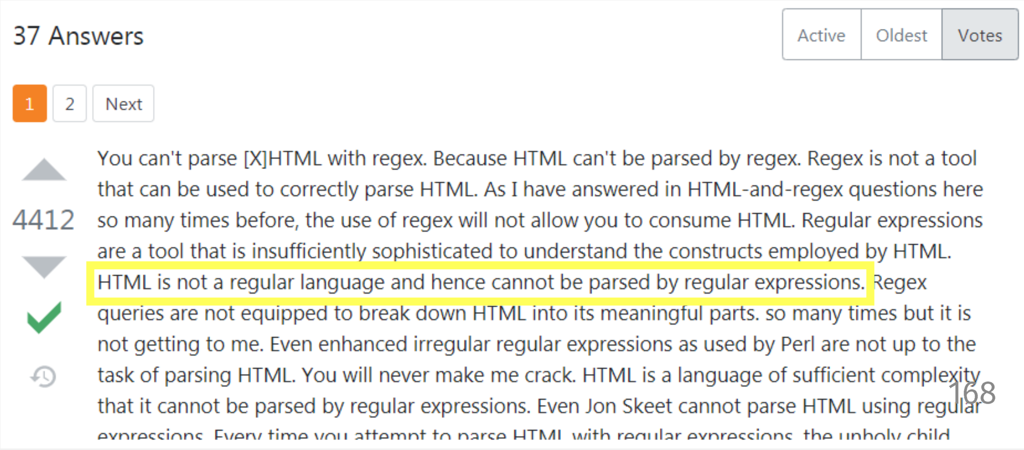
```
<p>
<a href="foo">
```

But not these:

```
<br />
<hr class="foo" />
```

I came up with this and wanted to make sure I've got it right. I am only capturing the `a-z`.

```
<([a-z]+) *[^/]*?>
```



37 Answers

Active Oldest Votes

1 2 Next

4412

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. 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

Flashback: Designing DFAs or NFAs

- Each state "stores" some information
    - E.g., q_0 = "seen zero **1**s", q_1 = "seen one **1**", q_2 = "seen two **1**s" etc.
  - Finite states = finite amount of info (decided in advance)

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

A Non-Regular Language

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

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

Still, how do we **prove non-regularness?**

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, strings in the language longer than length p satisfy the conditions

Lemma doesn't tell you an exact p!
(just that there exists "some" p)

The Pumping Lemma: Finite Langu

The pumping lemma is <u>only interesting for **infinite langs**</u>! (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. $|xy| \leq p$.

In finite langs, these are true for all strings "of length at least p" (for some p)

What's a possible p?
Length of longest string + 1

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

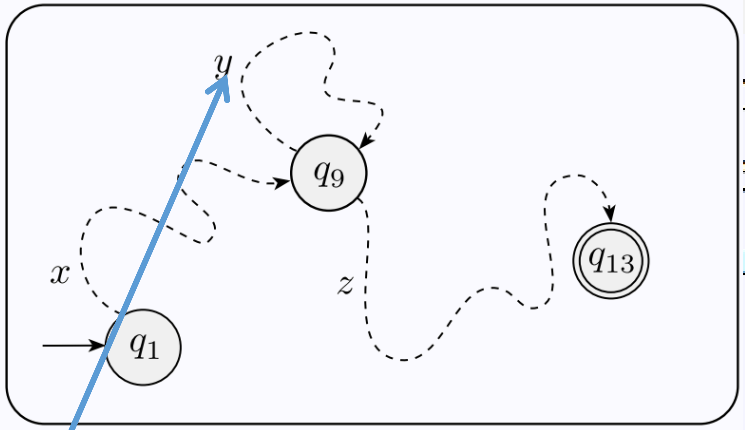
Therefore, <u>all</u> strings with length at least p satisfy the pumping lemma conditions! ☺

<u>Example</u>: a finite language {"ab", "cd"}

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

The Pumping Lemma, a Closer Look

Pumping lemma If A is a regular lang nber p (the pumping length) where if s is any string in en s may be divided into three pieces, $s = xyz$, satisfyin s:



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

"long enough" strings, should have a repeatable ("pumpable") part; "pumped" string is still in the language

Strings that have a repeatable part can be split into:

- x = the part before any repeating
- y = the repeated part
- z = the part after any repeating

This makes sense because DFAs have a finite number of states, so for "long enough" (i.e., some length p) inputs, some state must repeat

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

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

"pumpable" part cannot be empty

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

- Language is regular bc it's described by the regular expression 01*0
- Notice that the middle part is pumpable!
- E.g., "010" in the language can be split into three parts: x = 0, y = 1, z = 0
  - Any pumping (repeating) of the middle part creates a string that is still in the language
    - i = 1 -> "0**1**0", i = 2 -> "0**11**0", i = 3 -> "0**111**0"

Summary: The Pumping Lemma …

- … states properties that are true for all 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

Poll: Conditional Statements

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) ← Proof by contradiction
    - Yes!

Pumping Lemma: Proving Non-Regularity

… 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. $|xy| \leq p$.

If any of these are **not** true …

Contrapositive:
"If X then Y" is equivalent to "If **not** Y then **not** X"

Pumping Lemma: Non-Regularity Example

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

How To Do Proof By Contradiction

- Assume the opposite of the statement to prove
- Show that the assumption leads to a contradiction
- Conclude that the original statement must be true

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. $|xy| \leq p$.

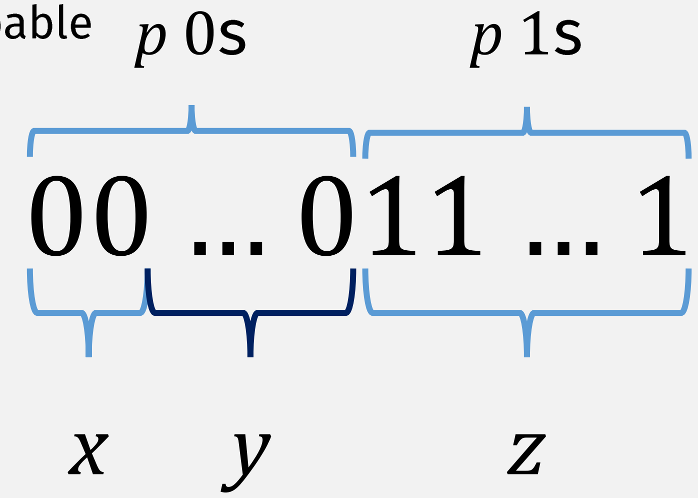
Contrapositive: If **not** true ...

Reminder: Pumping lemma says strings >= length p splittable into xyz where y is pumpable

Possible Split: y = all 0s

Proof (by contradiction):

- Assume: 0^n1^n **is** a regular language
  - So it must satisfy the pumping lemma
  - I.e., all strings length p or longer are pumpable
- Counterexample = 0^p1^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^n1^n
  - This means that 0^p1^p does not satisfy the pumping lemma
  - Which means that that 0^n1^n is a not regular language
  - This is a **contradiction** of the assumption!

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

So the proof is not done!

Is there another way to split into xyz ?

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

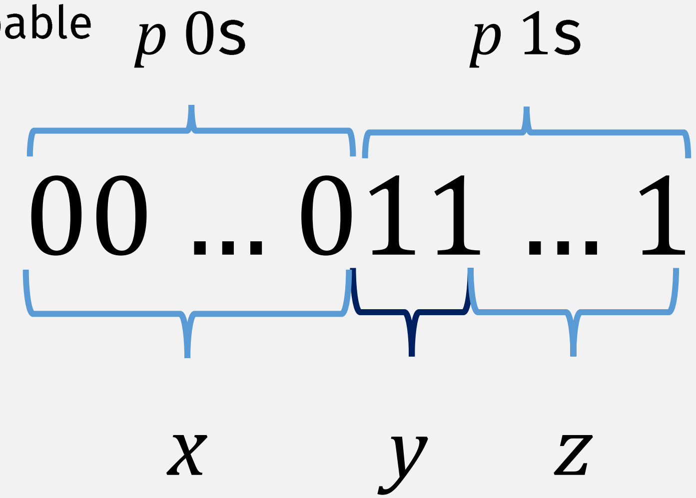
Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

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

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 or longer are pumpable
- Counterexample = 0^p1^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^n1^n is **not** a regular language

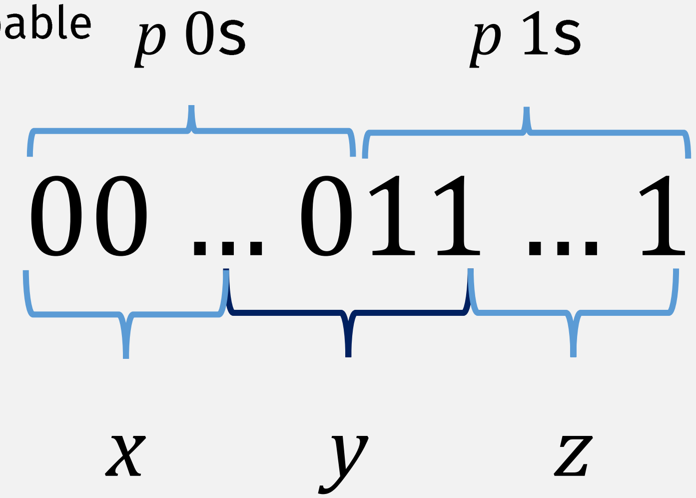
Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

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

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 or longer are pumpable
- Counterexample = 0^p1^p
- Choose xyz split so y contains:
  - both 0s and 1s



Did we examine every possible splitting?

Yes! QED

But maybe we did't have to …

- 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**!

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

Repeating part y ...
must be in the first p characters!

p 0s

00 ... 011 ... 1

y must be in here!

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!

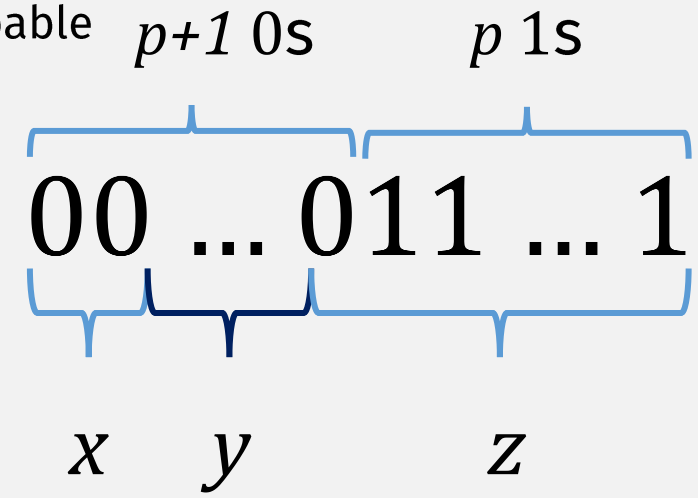
Example: $L = \{0^i1^j \mid i > j\}$

Want to prove: $L = \{0^i1^j \mid i > j\}$ **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 or longer are pumpable
- Counterexample = $0^{p+1}1^p$



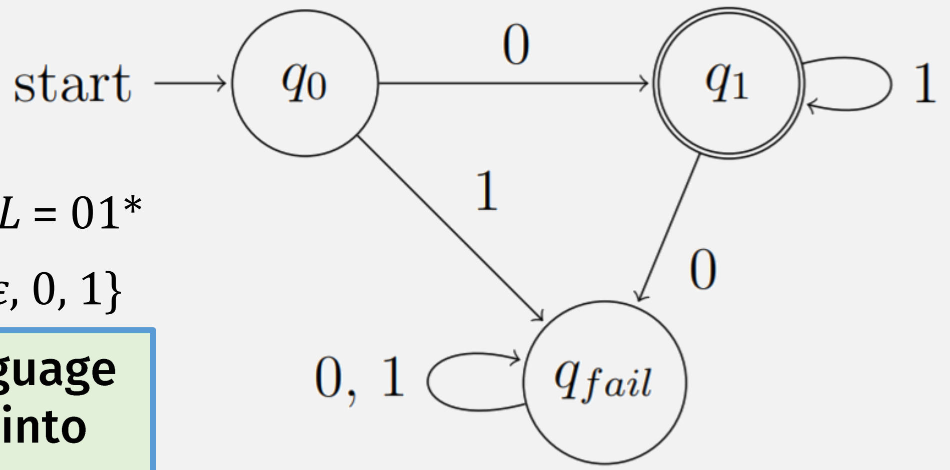
- Choose xyz split so y contains:
  - all 0s
  - (Only possibility, by condition 3)
- Repeat y zero times (**pump down**): produces string with 0s =< 1s
  - Which is not in the language $\{0^i1^j \mid i > j\}$
  - This means that $\{0^i1^j \mid i > j\}$ does not satisfy the pumping lemma
  - Which means that that it is a not regular language
  - This is a **contradiction** of the assumption!

Pumping Lemma Doesn't Always Work!

- What if you can't figure out a counterexample?

Another Way to Prove Regularity

- A set of strings S is "representative" of a language L if:
  - Every possible string $w \in \Sigma^*$ maps to a string s in S via REP where …
  - $\mathrm{REP}(w) = s$, if for every possible string z, $wz \in L$ iff $sz \in L$



$L = 01^*$

Representative set $S = \{\varepsilon, 0, 1\}$

Viewed this way **a language organizes all strings into distinct groups**

A language is regular if this number of groups is finite, i.e. it has a finite representative set!

For regular languages, strings in the "representative" set correspond to states in a DFA!

S contains one string that reaches each state

Then $\mathrm{REP}(w) = s$ if w reaches the same state that s represents

Then for any string z, $wz \in L$ iff $sz \in L$ because they started in the same state!

Another Way to Prove Non-Regularity

- A set of strings S is "representative" of a language L if:
  - Every possible string $w \in \Sigma^*$ maps to a string s in S via REP where ...
  - $\mathrm{REP}(w) = s$, if for every possible string z, $wz \in L$ iff $sz \in L$

$L = \{ 0^n1^n \mid n >= 0 \}$

- There must be a $\mathrm{REP}(0^k)$ every k ...
  - Because for every two strings 0^k and 0^m ...
  - ... there's some z that completes it such that $0^kz \in L$ but 0^mz is not
  - E.g., let $z = 1^k$, then $0^k1^k \in \mathrm{L}$ but 0^m1^k is not in L

The representative set is infinite!

So the language is not regular!

Check-in Quiz 9/27

On gradescope