Non-Regular Languages

Monday, February 14, 2022
Announcements

• HW 2 due yesterday

• HW 3 released, due Sun 2/20 11:59pm EST

• No class next Monday Feb 2/21
So Far: Regular or Not?

• Many ways to prove that a language is regular:
  • Construct a DFA or NFA (or GNFA) recognizing it
  • Come up with a regular expression describing the language
    • Regular Expression \(\Leftrightarrow\) NFA \(\Leftrightarrow\) DFA \(\Leftrightarrow\) Regular Language

• But not all languages are regular!
  • Most programming lang syntaxes, e.g., HTML / XML, are not regular
  • That means they can’t be represented with a regular expression (a common mistake)!
Someone Who Did Not Try

RegEx match open tags except XHTML self-closing

I need to match all of these opening tags:

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

But not these:

- `<div>`

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

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

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

• A DFA recognizing $L$ would require infinite states! (impossible)
  • States representing zero $\theta$s, one $\theta$, two $\theta$s, ...

• This language represents the essence of many PLs, e.g., HTML!
  • To better see this replace:
    • “$\theta$” 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!
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. $|xy| \leq p$.

This lemma describes a property that all regular languages have.

Note: this lemma cannot be used to prove that a language is a regular language! (but we already know how to do that anyways)
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 $p$
- Lemma doesn’t tell you an exact $p$! (just that there must exist “some” $p$)
The Pumping Lemma: Finite 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. \( |xy| \leq p \).

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

- Lemma doesn’t tell us what \( p \) is! Just that there is one.
- What could \( p \) be? How about: Length of longest string + 1
  - # 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, there is a number $p$ (the pumping length) where if $s$ is any string in $A$ divided into three pieces, $s = xyz$, satisfying:

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

Strings that have a **repeateable** part can be split into:

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

Strings of length $p =$ “long enough”: should have a **repeateable** (“pumpable”) part; where “pumped” string is still in the language.

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

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

So if possible to repeat once, then repeating any number of times is also possible.

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 $i ≥ 0$, $xy^iz ∈ A$,
2. $|y| > 0$, and
3. $|xy| ≤ p$.

*Note:* “pumpable” part of string 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: $x = 0$, $y = 1$, $z = 0$
  - Pumping (repeating) the middle part creates a string that is still in the language
  - E.g., 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
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!
Pumping Lemma: Proving Non-Regularity

If \( X \) then \( Y \)

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

Equivalent (contrapositive):

If any of these are **not** true ...

This is the essence of “proof by contradiction”

Contrapositive:

“If \( X \) then \( Y \)” is equivalent to “If **not** \( Y \) then **not** \( X \)”
Kinds of Mathematical Proof

- Proof by construction
  - Construct the object in question

- 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 $\{0^n 1^n | n \geq 0\}$. We use the pumping lemma to prove that $B$ is not regular. The proof is by contradiction.
Want to prove: $0^n1^n$ is not a regular language

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** = $0^p1^p$

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

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. $|xy| \leq p$. 

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

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

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 pumbable
- **Counterexample** = $0^p1^p$

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

$$p \quad 0s \quad \quad p \quad 1s \quad \quad 00 \ldots 011 \ldots 1$$

$$x \quad y \quad z$$

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

... then not true

Contrapositive: If not true...

Reminder: Pumping lemma says all strings $0^n1^n \geq \text{length } p \text{ are splittable into } xyz \text{ where } y \text{ is pumbable}

So find string $\geq \text{length } p \text{ that is not splittable into } xyz \text{ where } y \text{ is pumbable}

BUT ... pumping lemma requires only one pumbable 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

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

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 $xyz$ ?

\[
\begin{align*}
\text{Proof (by contradiction):} & \\
- \textbf{Assume}: & 0^n1^n \textbf{ is a regular language} & & & \text{So it must satisfy the pumping lemma} \\
& \text{So it must satisfy the pumping lemma} & & & \text{I.e., all strings length } p \text{ or longer are pumpable} \\
- \textbf{Counterexample}: & 0^p1^p & & & \text{Choose } xyz \text{ split so } y \text{ contains:} \\
& \text{Choose } xyz \text{ split so } y \text{ contains:} & & & \text{all 1s} \\
- \text{Is this string pumpable?} & & & & \text{No!} \\
& \text{No!} & & & \text{By the same reasoning as in the previous slide}
\end{align*}
\]
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 $p$ or longer are pumpable

• Counterexample = $0^p1^p$

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

$$y = \underbrace{00 \ldots 0} \ldots \underbrace{011 \ldots 1}$$

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

The 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 party 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

### 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 \( \leq 1s \)
  - 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!
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 2/14

On gradescope