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

• 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
So Far: Regular or Not?

- 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 $\Leftrightarrow$ 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 those languages
    - And they can’t be described 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:

- 1553
- 6572

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 have answered many HTML-and-regex questions here so many times before, the use of regex queries are not equipped to break down HTML into its meaningful parts but it is not getting to me. Even enhanced irregular regular expressions used by Perl are not up to the task of parsing HTML. You will never

Have you tried using an XML parser instead?

You can see it can't see it is beautiful. The final sniffling of the lies of Man ALL IS LOST. LOST is the bony comes he comes he comes the Corpera permeates all MY FACE MY FACE of god NOOOOO NO stop the angles are not real.

ZALGO IS TONG THE PONY, HE COMES
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)

- So 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 “\(<\text{tag}>\)” or “\(“\)
    - “\( 1 \)” with “\(</\text{tag}>\)” or “\”)”

- The problem is tracking the **nestedness**
  - Regular languages cannot count arbitrary nesting depths
    - E.g., \( \text{if } \{ \text{if } \{ \text{if } \{ \ldots \} \} \} \}
  - So most programming language syntax is not regular!
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
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.

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)

Hint: This is an “If $X$ then $Y$” statement (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!
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 ...

If $X$ then $Y$ is equivalent 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. $|xy| \leq p$.

All regular languages satisfy these three conditions!

Specifically, these conditions apply to strings in the language of length $\geq p$.

**Note:**
- Lemma doesn’t give an exact $p$!
- Just that there is some string length $p$ ...
- ... those strings must obey the 3 conditions
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. $|xy| \leq p$.

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

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., length $\geq p$) inputs, some state must repeat.

---

Note: "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. $|xy| \leq p$.

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

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

Also, this is the only 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. $|xy| \leq p$.

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

- It's regular because it has a 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
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
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.
Proof (by contradiction):

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

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 \(0^n1^n \geq p\) are **splittable** into \(xyz\) where \(y\) is pumpable

So find string \(\geq p\) that is not **splittable** into \(xyz\) where \(y\) is pumpable
Want to prove: \(0^n1^n\) is not a regular language

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

Proof (by contradiction):

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

2. Counterexample = \(0^p1^p\)

3. Choose \(xyz\) split so \(y\) contains:
   - all 0s

4. 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: $y = \text{all 1s}$

Proof (by contradiction):

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

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

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

  $p$ 0s  $p$ 1s

  00 ... 011 ... 1

  $x \quad y \quad z$

- Is this string pumpable?
  - No!
  - By the same reasoning as in the previous slide
Want to prove: $0^n 1^n$ is not a regular language

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

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

\[ x \quad y \quad z \]
\[ 00 \ldots 011 \ldots 1 \]

• 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!

Did we examine every possible splitting? Yes! QED

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

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 \( \geq \) 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**
  - ... not in the language \( \{0^i1^j \mid i > j\} \)
  - \( \{0^i1^j \mid i > j\} \) does not satisfy the pumping lemma
  - So it is a not regular language
  - This is a **contradiction** of the assumption!
\textbf{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