The Limits of Regular Languages

22 September 2022
What’s a computer?
What’s a computer?

A finite-state machine!
Computing devices have internal workings that can be in one of finitely many possible configurations.

Each state in a DFA corresponds to some possible configuration of the internal workings.

If your computer has 8 GB of RAM and 200 GB of hard drive space, that's a total of 208 GB of memory, which is $1\,786\,706\,395\,136$ bits. There are “only” $2^{1391569403904}$ possible configurations of the memory in this computer.

You could, in principle, build a DFA representing this computer, where there’s one symbol per type of input the computer can receive.
Regular languages correspond to problems that can be solved with finite memory.

At each point in time, we only need to store one of finitely many pieces of information.
Nonregular languages correspond to problems that cannot be solved with finite memory.

Since every computer ever built has finite memory, nonregular languages correspond to problems that cannot be solved by physical computers!
Is every language regular?
To prove a language is regular, we can construct a DFA, NFA, or regular expression to recognize it.
To prove a language is not regular, we need to show that it’s impossible to construct a DFA, NFA, or regular expression for it.

It’s equivalent to just choose one, e.g., you can’t build a DFA for it.

But proving a negative is hard!
Arguing a language is not regular
A simple language

Let $\Sigma = \{a, b\}$ and consider this language:

$L = \{a^n b^n \mid n \in \mathbb{N}_0\}$

$L$ is the language of all strings of $n$ $a$s followed by $n$ $b$s:

$\{\varepsilon, ab, aabb, aaabbb, \ldots\}$

Is this language regular?
A simple language

Let $\Sigma = \{a, b\}$ and consider this language:

\[ L = \{a^n b^n \mid n \in \mathbb{N}_0\} \]

Could we write a regular expression for it?
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$L = \{a^n b^n \mid n \in \mathbb{N}_0\}$

Could we write a regular expression for it?

$a^n b^n$

Easy!
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Could we write a regular expression for it?

$${a^n b^n}$$

Easy!

Regular expressions can’t have variables!
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Could we write a regular expression for it?

$a^* b^*$

Maybe?
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Could we write a regular expression for it?

$$a^* b^*$$

Maybe?

Doesn’t require the same number of $a$s and $b$s.
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$$ L = \{a^n b^n | n \in \mathbb{N}_0\} $$

Could we write a regular expression for it?

$$ \varepsilon \cup ab \cup a^2 b^2 \cup a^3 b^3 \cup \cdots $$

Please? 😢
A simple language: Regular expression

Let $\Sigma = \{a, b\}$ and consider this language:

$L = \{a^n b^n | n \in \mathbb{N}_0\}$

Could we write a regular expression for it?

$\varepsilon \cup ab \cup a^2b^2 \cup a^3b^3 \cup \cdots$

Please? 😢

If you can’t finish writing it, it’s not a complete regular expression!
A simple language: Another attempt
A simple language: Another attempt

\[
\begin{align*}
&\text{a} & \rightarrow & q_a & \rightarrow & q_{aa} \\
&b & \rightarrow & q_{ab} & & q_{aab} \\
&b & \rightarrow & q_{aab} & & q_{aabb}
\end{align*}
\]
A simple language: Another attempt
This language is not regular!

Intuitive explanation:

Imagine a finite automaton to accept this language.

When any DFA for L is run on any two of the strings $\varepsilon$, a, aa, aaa, aaaa, etc., the DFA must end in different states.

Suppose $a^n$ and $a^m$ end up in the same state, where $m \neq n$.

Then $a^n b^n$ and $a^m b^n$ will end up in the same state.

The DFA will either accept a string not in the language or reject a string in the language, which it shouldn't be able to do.

*We can’t place all these strings into different states; there are only finitely many states!*
The intuition for this proof is helpful to think about.

However, actually writing one of these proofs becomes difficult for more complicated languages.

Instead, we’ll take the idea of the number of states required for a DFA to recognize a language and develop a powerful proof framework.
To prove $B$ is non-regular, we entertain the possibility that $B$ is regular: Imagine there’s a DFA $M$ that recognizes $B$.

We can try to “break” that DFA by finding a string that’s not in $B$ but is accepted by $M$.

If we show that we can break all possible DFAs this way, then that means we’ve shown there’s no DFA that recognizes $B$ – and $B$ must not be regular.
An important observation
Recall from CMPU 145 the *Pigeonhole Principle*:

If you place $n$ items (pigeons) in $m$ boxes (pigeonholes), where $n > m$, then at least one of the boxes must have more than one item.
Visiting multiple states

Let $M$ be a DFA with $p$ states.

Any string $s$ accepted by $M$ that’s at least $p$ characters long must visit some state twice within the first $p$ characters.

  Number of states visited is equal to $p + 1$.
  By the Pigeonhole Principle, some state is duplicated.

The substring of $s$ between those revisited states can be removed, duplicated, tripled, etc. without changing the fact that $M$ accepts $s$. 
Informally

Let $L$ be a regular language.

If we have a string $s \in L$ that is “sufficiently long”, then we can split the string into three pieces and “pump” the middle:

We can write $s = xyz$ such that $xy^0z, xy^1z, xy^2z, \ldots$ are all in $L$. 
Here’s how we find the string that breaks $M$.

Imagine Alice and Bob are playing a strange game:

Bob proposes a DFA $M$ for the language.

Alice gives Bob a “test” string $s$ that $M$ is supposed to accept.

But then she uses the information that Bob reveals to concoct another string that breaks $M$. 
An example game dialogue
**Alice:** The language $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

**Bob:** Yes, it is!

**Alice:** Oh really? Then show me a DFA that recognizes it.

**Bob:** Here’s one:
**Alice**: How many states ($p$) does it have?

**Bob**: $p = 2$

**Alice**: Does your automaton accept the string $s = a^p b^p$?

**Bob checks it:**

**Bob**: Of course!
**Alice**: Does this run use a state twice while reading in the first half of the string?

**Bob**: Yes.
**Alice:** What are the strings that it reads up to the first visit, between the first and second visits, and after the second visit?

**Bob:** $x = \varepsilon$, $y = a$, $z = abb$
Alice: So, does your automaton accept this string?

\[ s = xy^2z = \varepsilon aabb = aabb \]

Bob: Let’s try it…

Bob: Oh no! 😞 It does, but that’s not in B. I lose!
A more general game dialogue
We want to show that Bob will lose no matter what automaton he designs for the language $B$.

Let’s write a version of it that removes Bob’s specific choices.
Alice: The language $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

Bob: Yes, it is! Here, I can show you an automaton that –

Alice: I don’t need to see it. Just count how many states it has.

Bob: It has –

Alice: Don’t tell me. Just call it $p$.

Bob: Okay.

Alice: Does it accept the string $s = a^p b^p$?

Bob: Yes.
Alice: On reading the first $p$ symbols of $s$, your automaton goes through $(p + 1)$ configurations – the starting configuration plus one for each symbol.

Since your automaton has only $p$ states, by the Pigeonhole Principle, it must visit some state at least twice.

Bob: Yes.

Alice: Could you find the strings that it reads
    — up to the first visit,
    — between the first and second visits, and
    — after the second visit?

Bob: Yes, they’re –

Alice: I don’t need to know. Just call them $x$, $y$, and $z$.

Bob: Okay.
Alice: Does your automaton also accept $xy^2z$?

Bob: Yes.

Alice: But $y$ consists of only $a$s, so $xy^2z$ has more $a$s than $b$s. That means it isn’t in the language $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$.

Bob: 😱
Proof strategy: The Pumping Lemma
The *Pumping Lemma* is a formal statement of Alice’s strategy for defeating Bob.
The Pumping Lemma for Regular Languages

For every regular language $L,$

there exists a positive integer $p$ such that

for every string $s \in L$ such that $|s| \geq p,$

there exist strings $x, y,$ and $z$ such that

$s = xyz$

$|xy| \leq p$

$y \neq \varepsilon$

$xy^iz \in L$ for all $i \in \mathbb{N}_0$
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that

$s = xyz$

$|xy| \leq p$

$y \neq \varepsilon$

$xy^iz \in L$ for all $i \in \mathbb{N}_0$
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that

$$s = xyz$$

$$|xy| \leq p$$

$$y \neq \varepsilon$$

$$xy^iz \in L \text{ for all } i \in \mathbb{N}_0$$

Strings longer than the pumping length must have a special property
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that

$$s = xyz$$

$$|xy| \leq p$$

$$y \neq \varepsilon$$

$$xy^iz \in L \text{ for all } i \in \mathbb{N}_0$$
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x, y, \text{ and } z$ such that

\[
s = xyz \quad \text{s can be broken into three pieces,}
\]

\[
|xy| \leq p
\]

\[
y \neq \varepsilon
\]

\[
xy^iz \in L \text{ for all } i \in \mathbb{N}_0
\]
The Pumping Lemma for Regular Languages

For every regular language $L$,

there exists a positive integer $p$ such that

for every string $s \in L$ such that $|s| \geq p$,

there exist strings $x$, $y$, and $z$ such that

$s = xyz$

$s$ can be broken into three pieces,

$|xy| \leq p$

where the first two pieces occur at the start of the string,

$y \neq \varepsilon$

$xy^iz \in L$ for all $i \in \mathbb{N}_0$
The Pumping Lemma for Regular Languages

For every regular language \( L \),

there exists a positive integer \( p \) such that

for every string \( s \in L \) such that \( |s| \geq p \),

there exist strings \( x, y, \) and \( z \) such that

\[
\begin{align*}
  s &= xyz & \text{s can be broken into three pieces,} \\
  |xy| &\leq p & \text{where the first two pieces occur at the start of the string,} \\
  y &\neq \varepsilon & \text{the middle part isn’t empty, and} \\
  xy^iz &\in L \text{ for all } i \in \mathbb{N}_0
\end{align*}
\]
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that:

$s = xyz$

$s$ can be broken into three pieces,

$|xy| \leq p$

where the first two pieces occur at the start of the string,

$y \neq \varepsilon$

the middle part isn’t empty, and

$xy^iz \in L$

the middle piece can be repeated zero or more times.
Rationale for requirements in the Pumping Lemma

\( y \neq \varepsilon \)

Because \( y \) labels the loop, it has to consist of at least one symbol.

\( |xy| \leq p \)

Because \( xy \) is what you get when you take the loop once.

\( xy^iz \in L \) for all \( i \in \mathbb{N}_0 \)

Because \( y \) can be \textit{pumped} zero or more times.
The Pumping Lemma gets its name because the repeated string is “pumped” to get more.

Because of the nature of finite automata, we can’t control the number of times it is pumped.

So, a regular language with strings of length $\geq p$ is always infinite!

A (sad) pump
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w$ contains $00$ as a substring$\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 0 0 1 0
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 0 0 1 0
1 0 0 1 0
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be "pumped".
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains 00 as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

```
1 0 0 1 0
```
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 0 0 1 1 0
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

```
1 0 0 1 1 1 0
```
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 0 0
Let $\Sigma = \{0, 1\}$ and $L = \{w \in \Sigma^* \mid w$ contains $00$ as a substring\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

The first piece is just the empty string! This is perfectly fine.
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains 00 as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 0 0
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w$ contains $00$ as a substring$\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

1 1 0 0
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be "pumped".
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w$ contains 00 as a substring$\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”. 
Let $\Sigma = \{0, 1\}$
and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.
Let $\Sigma = \{0, 1\}$ and $L = \{w \in \Sigma^* \mid w$ contains 00 as a substring$\}$.

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.
Let $\Sigma = \{0, 1\}$ and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

```
1 1 0 0 0 0 1
1 1 0 0 0 1 0 0 1
```
Let $\Sigma = \{0, 1\}$ and $L = \{w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring}\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

```
1 1 0 0 0 0 1 0 0 1 0 0 1
```
Let $\Sigma = \{0, 1\}$
and $L = \{\varepsilon, 0, 1, 00, 01, 10, 11\}$
Let $\Sigma = \{0, 1\}$
and $L = \{\epsilon, 0, 1, 00, 01, 10, 11\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.
Let $\Sigma = \{0, 1\}$
and $L = \{\epsilon, 0, 1, 00, 01, 10, 11\}$

Any string of length three or greater can be split into three pieces, the second of which can be “pumped”.

*The Pumping Lemma holds for finite languages because the pumping length can be longer than the longest string!*
THEOREM $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.
THEOREM  $B = \{a^nb^n \mid n \in \mathbb{N}_0\}$ is not regular.

PROOF  By contradiction; assume $B$ is regular.
THEOREM  $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

PROOF  By contradiction; assume $B$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma.
THEOREM  \( B = \{a^n b^n \mid n \in \mathbb{N}_0\} \) is not regular.

PROOF  By contradiction; assume \( B \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Consider the string
\[ s = a^p b^p. \]
THEOREM  $B = \{a^n b^n \mid n \in \mathbb{N}_0 \}$ is not regular.

PROOF  By contradiction; assume $B$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Consider the string $s = a^p b^p$. Then $|s| = 2p \geq p$ and $s \in B$, so we can break this string into $s = xyz$, where $|xy| \leq p$ and $y \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, the string $xy^i z \in B$. 

THEOREM  $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

PROOF  By contradiction; assume $B$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Consider the string $s = a^p b^p$. Then $|s| = 2p \geq p$ and $s \in B$, so we can break this string into $s = xyz$, where $|xy| \leq p$ and $y \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, the string $xy^i z \in B$.

Because $|xy| \leq p$ and $|y| > 0$, the string $y$ has to consist only of $a$s.
THEOREM  \( B = \{a^n b^n \mid n \in \mathbb{N}_0\} \) is not regular.

PROOF  By contradiction; assume \( B \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Consider the string \( s = a^p b^p \). Then \(|s| = 2p \geq p\) and \( s \in B \), so we can break this string into \( s = xyz \), where \(|xy| \leq p\) and \( y \neq \epsilon \), and for any \( i \in \mathbb{N}_0 \), the string \( xy^i z \in B \).

Because \(|xy| \leq p\) and \(|y| > 0\), the string \( y \) has to consist only of \( a \)s.

So, no matter what segment of the string \( xy \) covers, pumping to the string \( xy^2 z \) adds to the number of \( a \)s, hence there are more \( a \)s than \( b \)s.
THEOREM  $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

PROOF  By contradiction; assume $B$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Consider the string $s = a^p b^p$. Then $|s| = 2p \geq p$ and $s \in B$, so we can break this string into $s = xyz$, where $|xy| \leq p$ and $y \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, the string $xy^i z \in B$.

Because $|xy| \leq p$ and $|y| > 0$, the string $y$ has to consist only of $a$s.

So, no matter what segment of the string $xy$ covers, pumping to the string $xy^2 z$ adds to the number of $a$s, hence there are more $a$s than $b$s.

There is no way to segment $s$ into $xyz$ that can’t be pumped to produce a string that isn’t in the language.
THEOREM  $B = \{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

PROOF  By contradiction; assume $B$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Consider the string $s = a^p b^p$. Then $|s| = 2p \geq p$ and $s \in B$, so we can break this string into $s = xyz$, where $|xy| \leq p$ and $y \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, the string $xy^i z \in B$.

Because $|xy| \leq p$ and $|y| > 0$, the string $y$ has to consist only of $a$s.

So, no matter what segment of the string $xy$ covers, pumping to the string $xy^2 z$ adds to the number of $a$s, hence there are more $a$s than $b$s.

There is no way to segment $s$ into $xyz$ that can’t be pumped to produce a string that isn’t in the language.

Contradiction! Therefore, $B$ is not regular. ■
Non-regular languages

The Pumping Lemma describes a property common to all regular languages.

Any language $L$ that does not have this property cannot be regular.
The Pumping Lemma “‘game’”

You can think of a Pumping Lemma proof as a game between *you* and an *adversary*.

*You win* by finding a contradiction of the Pumping Lemma for the given language.

*The adversary wins* if they can make a choice for which the Pumping Lemma succeeds.

The game goes as follows:

*The adversary* chooses a pumping length $p$.

*You* choose a string $s$ with $|s| \geq p$ and $s \in L$.

*The adversary* break it into $x$, $y$, and $z$ such that $|xy| \leq p$ and $y \neq \varepsilon$.

*You* choose an $i$ such that $xyz^i \notin L$. (If you can’t, you lose!)
Gameplay Magazine described the rules as “punishingly intricate”.
The Pumping Lemma Game

Adversary

You
The Pumping Lemma Game

**Adversary**

1 Maliciously choose pumping length \( p \)

**You**
# The Pumping Lemma Game

<table>
<thead>
<tr>
<th><strong>Adversary</strong></th>
<th><strong>You</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Maliciously choose pumping length $p$</td>
<td>2 Cleverly choose a string $s \in L$, $</td>
</tr>
</tbody>
</table>
The Pumping Lemma Game

**Adversary**

1. Maliciously choose pumping length \( p \)

2. Maliciously split \( w = xyz \) with \( y \neq \epsilon \) and \( |xy| \leq p \)

**You**

2. Cleverly choose a string \( s \in L, |s| \geq p \)
# The Pumping Lemma Game

<table>
<thead>
<tr>
<th><strong>Adversary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> Maliciously choose pumping length ( p )</td>
</tr>
<tr>
<td><strong>3</strong> Maliciously split ( w = xyz ) with ( y \neq \varepsilon ) and (</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>You</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2</strong> Cleverly choose a string ( s \in L,</td>
</tr>
<tr>
<td><strong>4</strong> Cleverly choose ( i ) such that ( xy^iz \not\in L )</td>
</tr>
</tbody>
</table>
The Pumping Lemma Game

<table>
<thead>
<tr>
<th><strong>Adversary</strong></th>
<th><strong>You</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Maliciously choose pumping length $p$</td>
<td>2 Cleverly choose a string $s \in L$, $</td>
</tr>
<tr>
<td>3 Maliciously split $w = xyz$ with $y \neq \varepsilon$ and $</td>
<td>xy</td>
</tr>
<tr>
<td>5 I’ll get you next time, Gadget! Next time!</td>
<td></td>
</tr>
</tbody>
</table>
What other languages can we prove are non-regular?
The **equality problem** is defined as follows: Given two strings, $x$ and $y$, decide whether $x = y$.

Let $\Sigma = \{0, 1, ?\}$.

We can encode the equality problem as a string of the form $x?y$.

“Is 001 equal to 110?” would be encoded as 001?110

“Is 11 equal to 11?” would be encoded as 11?11

“Is 110 equal to 110?” would be encoded as 110?110

Let $\text{EQUAL} = \{w?w \mid w \in \{0, 1\}^*\}$

Is $\text{EQUAL}$ a regular language?
The Pumping Lemma for Regular Languages

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ such that $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that

\[ s = xyz \]

$s$ can be broken into three pieces,

\[ |xy| \leq p \]

where the first two pieces occur at the start of the string,

\[ y \neq \varepsilon \]

the middle part isn’t empty, and

\[ xy^iz \in L \]

the middle piece can be repeated zero or more times.
Using the Pumping Lemma

\[ \text{EQUAL} = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

\[ EQUAL = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

\[ EQUAL = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

$EQUAL = \{w?w \mid w \in \{0, 1\}^*\}$
Using the Pumping Lemma

\[ \text{EQUAL} = \{ w?w \mid w \in \{0, 1\}^* \} \]
Using the Pumping Lemma

$EQUAL = \{w?w \mid w \in \{0, 1\}^*\}$
Using the Pumping Lemma

\[ \text{EQUAL} = \{ w ? w \mid w \in \{0, 1\}^* \} \]
Using the Pumping Lemma

\[ EQUAL = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

\[ EQUAL = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

\[ \text{EQUAL} = \{ w?w \mid w \in \{0, 1\}* \} \]
Using the Pumping Lemma

\[ \text{EQUAL} = \{w?w \mid w \in \{0, 1\}^*\} \]
Using the Pumping Lemma

\[ EQUAL = \{ w \mid w \in \{0, 1\}^* \} \]
What’s going on?

The Pumping Lemma says that for “sufficiently long” strings, we should be able to pump some part of the string.

We can’t pump any part containing the ? because we can’t duplicate it or remove it.

We can’t pump just one part of the string because then the strings on opposite sides of the ? wouldn’t match.

Can we formally show that \textit{EQUAL} is not regular?
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular.

\begin{quote}
For every regular language $L$,
\begin{itemize}
  \item there exists a positive integer $n$ such that
  \item for every string $s \in L$ with $|s| \geq p$,
  \item there exist strings $x$, $y$, and $z$ such that
  \item $s = xyz$
  \item $|xy| \leq p$
  \item $y \neq \varepsilon$
  \item $xyz \in L$ for all $i \in \mathbb{N}_0$
\end{itemize}
\end{quote}
THEOREM  *EQUAL* is not regular.

PROOF  By contradiction; assume that *EQUAL* is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma.

---

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ with $|s| \geq p$, there exist strings $x, y, z$ such that

- $s = xyz$
- $|xy| \leq p$
- $y \neq \varepsilon$
- $xyz^i \in L$ for all $i \in \mathbb{N}_0$
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma.

\begin{center}
\begin{tikzpicture}
\node at (0,0) {For every regular language \( L \),
\begin{itemize}
\item there exists a positive integer \( p \) such that
\item for every string \( s \in L \) with \( |s| \geq p \),
\item there exist strings \( x, y, \) and \( z \) such that
\item \( s = xyz \)
\item \( |xy| \leq p \)
\item \( y \neq \varepsilon \)
\item \( xyz \in L \) for all \( i \in \mathbb{N}_0 \)
\end{itemize}
\end{tikzpicture}
\end{center}
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma.

For every regular language $L$,
there exists a positive integer $n$ such that

\textbf{for every string} $s \in L$ with $|s| \geq p$,
there exist strings $x, y, \text{ and } z$ such that

$s = xyz$
$|xy| \leq p$
y \neq \epsilon$

$xyz^i \in L$ for all $i \in \mathbb{N}_0$
THEOREM  \textit{EQUAL} is not regular.

PROOF By contradiction; assume that \textit{EQUAL} is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma.

\begin{quote}
\textit{The hardest part of most Pumping Lemma proofs is choosing a string that we should be able to pump but cannot.}
\end{quote}

For every regular language $L$, there exists a positive integer $n$ such that for every string $s \in L$ with $|s| \geq p$, there exist strings $x, y,$ and $z$ such that

$s = xyz$

$|xy| \leq p$

$y \neq \epsilon$

$xyz^i \in L$ for all $i \in \mathbb{N}_0$
THEOREM  \( \textit{EQUAL} \) is not regular.

PROOF  By contradiction; assume that \( \textit{EQUAL} \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = \theta^p\theta^p \).

\[
\text{For every regular language } L, \\
\quad \textbf{there exists} \text{ a positive integer } n \text{ such that} \\
\quad \textbf{for every} \text{ string } s \in L \text{ with } |s| \geq p, \\
\quad \quad \textbf{there exist} \text{ strings } x, y, \text{ and } z \text{ such that} \\
\quad \quad \quad s = xyz \\
\quad \quad \quad |xy| \leq p \\
\quad \quad \quad y \neq \varepsilon \\
\quad \quad \quad xy^iz \in L \text{ for all } i \in \mathbb{N}_0
\]
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = \emptyset^p \emptyset^p \).

\begin{center}
\begin{tcolorbox}
For every regular language \( L \),
there exists a positive integer \( p \) such that
\begin{itemize}
  \item [for every string] \( s \in L \) with \( |s| \geq p \),
  \item [there exist strings] \( x, y, \) and \( z \) such that
  \begin{align*}
    s &= xyz \\
    |xy| &\leq p \\
    y &\neq \epsilon \\
    xyz^i &\in L \text{ for all } i \in \mathbb{N}_0
  \end{align*}
\end{itemize}
\end{tcolorbox}
\end{center}
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = \emptyset^p \emptyset^p \). Then \( s \in \textit{EQUAL} \) and \(|s| = 2p + 1 \geq p\).

For every regular language \( L \),
there exists a positive integer \( p \) such that
\textbf{for every string} \( s \in L \) \textbf{with} \(|s| \geq p\),
\textbf{there exist strings} \( x, y, \) \textbf{and} \( z \) \textbf{such that}
\( s = xyz \)
\(|xy| \leq p \)
\( y \neq \varepsilon \)
\( xyz \in L \) for all \( i \in \mathbb{N}_0 \)
THEOREM  $EQUAL$ is not regular.

PROOF  By contradiction; assume that $EQUAL$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Let $s = \theta^p \theta^p$. Then $s \in EQUAL$ and $|s| = 2p + 1 \geq p$.

For every regular language $L$, there exists a positive integer $p$ such that for every string $s \in L$ with $|s| \geq p$, there exist strings $x$, $y$, and $z$ such that $s = xyz$, $|xy| \leq p$, $y \neq \epsilon$, $xy^iz \in L$ for all $i \in \mathbb{N}_0$. 
THEOREM  \( EQUAL \) is not regular.

PROOF  By contradiction; assume that \( EQUAL \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = 0^p?0^p \). Then \( s \in EQUAL \) and \( |s| = 2p + 1 \geq p \).

For every regular language \( L \),
there exists a positive integer \( p \) such that
for every string \( s \in L \) with \( |s| \geq p \),

\[ \text{there exist strings } x, y, \text{ and } z \text{ such that} \]
\[ s = xyz \]
\[ |xy| \leq p \]
\[ y \neq \varepsilon \]
\[ xy^iz \in L \text{ for all } i \in \mathbb{N}_0 \]
THEOREM \( \text{EQUAL} \) is not regular.

PROOF By contradiction; assume that \( \text{EQUAL} \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = \theta^p\theta^p \). Then \( s \in \text{EQUAL} \) and \( |s| = 2p + 1 \geq p \). Thus by the Pumping Lemma, we can write \( s = xyz \) such that \( |xy| \leq p \) and \( y \neq \varepsilon \) and for any \( i \in \mathbb{N}_0 \), \( xy^iz \in \text{EQUAL} \).

For every regular language \( L \), there exists a positive integer \( p \) such that for every string \( s \in L \) with \( |s| \geq p \), there exist strings \( x, y, \) and \( z \) such that
\[
\begin{align*}
&\quad \text{there exist strings } x, y, \text{ and } z \text{ such that} \\
&\quad \quad s = xyz \\
&\quad \quad |xy| \leq p \\
&\quad \quad y \neq \varepsilon \\
&\quad \quad xy^iz \in L \text{ for all } i \in \mathbb{N}_0
\end{align*}
\]
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Let $s = 0^p?0^p$. Then $s \in \textit{EQUAL}$ and $|s| = 2p + 1 \geq p$. Thus by the Pumping Lemma, we can write $s = xyz$ such that $|xy| \leq p$ and $y \neq \varepsilon$ and for any $i \in \mathbb{N}_0$, $xy^iz \in \textit{EQUAL}$.

\begin{center}
\begin{tabular}{|c|c|}
\hline
For every regular language $L$, \\
there exists a positive integer $p$ such that \\
for every string $s \in L$ with $|s| \geq p$, \\
there exist strings $x$, $y$, and $z$ such that \\
$s = xyz$ \\
$|xy| \leq p$ \\
y $\neq \varepsilon$ \\
$xy^iz \in L$ for all $i \in \mathbb{N}_0$ \\
\hline
\end{tabular}
\end{center}
THEOREM  $EQUAL$ is not regular.

PROOF  By contradiction; assume that $EQUAL$ is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Let $s = \emptyset^p?\emptyset^p$. Then $s \in EQUAL$ and $|s| = 2p + 1 \geq p$. Thus by the Pumping Lemma, we can write $s = xyz$ such that $|xy| \leq p$ and $y \neq \varepsilon$ and for any $i \in \mathbb{N}_0$, $xy^iz \in EQUAL$.

At this point, we have some string that we should be able to split into pieces and pump. The rest of the proof shows that no matter what choice we made, the middle can't be pumped.

For every regular language $L$,

there exists a positive integer $p$ such that

for every string $s \in L$ with $|s| \geq p$,

there exist strings $x, y,$ and $z$ such that

$s = xyz$

$|xy| \leq p$

$y \neq \varepsilon$

$xy^iz \in L$ for all $i \in \mathbb{N}_0$
THEOREM  \textit{EQUAL} is not regular.

PROOF  By contradiction; assume that \textit{EQUAL} is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Let $s = 0^p?0^p$. Then $s \in \textit{EQUAL}$ and $|s| = 2p + 1 \geq p$. Thus by the Pumping Lemma, we can write $s = xyz$ such that $|xy| \leq p$ and $y \neq \varepsilon$ and for any $i \in \mathbb{N}_0$, $xy^iz \in \textit{EQUAL}$. The string $y$ must consist only of 0s before the ? or it would violate that $|xy| \leq p$.

\begin{center}
\begin{tabular}{|l|}
\hline
\textit{For every} regular language $L$, \\
\textit{there exists} a positive integer $p$ such that \\
\textit{for every} string $s \in L$ with $|s| \geq p$, \\
\textit{there exist} strings $x$, $y$, and $z$ such that \\
$s = xyz$ \\
$|xy| \leq p$ \\
$y \neq \varepsilon$ \\
$xy^iz \in L$ for all $i \in \mathbb{N}_0$ \\
\hline
\end{tabular}
\end{center}
THEOREM  *EQUAL* is not regular.

PROOF  By contradiction; assume that *EQUAL* is regular. Let $p$ be the pumping length guaranteed by the Pumping Lemma. Let $s = \theta^p?\theta^p$. Then $s \in \text{EQUAL}$ and $|s| = 2p + 1 \geq p$. Thus by the Pumping Lemma, we can write $s = xyz$ such that $|xy| \leq p$ and $y \neq \epsilon$ and for any $i \in \mathbb{N}_0$, $xy^iz \in \text{EQUAL}$. The string $y$ must consist only of $\theta$s before the $?$ or it would violate that $|xy| \leq p$. Therefore, $xy^0z = \theta^m?\theta^p$, where $m < p$, and is not in *EQUAL*.
**THEOREM**  \( EQUAL \) is not regular.

**PROOF** By contradiction; assume that \( EQUAL \) is regular. Let \( p \) be the pumping length guaranteed by the Pumping Lemma. Let \( s = 0^p?0^p \). Then \( s \in EQUAL \) and \(|s| = 2p + 1 \geq p\). Thus by the Pumping Lemma, we can write \( s = xyz \) such that \(|xy| \leq p\) and \( y \neq \varepsilon \) and for any \( i \in \mathbb{N}_0 \), \( xy^iz \in EQUAL \). The string \( y \) must consist only of \( 0s \) before the \( ? \) or it would violate that \(|xy| \leq p\). Therefore, \( xy^0z = 0^m?0^p \), where \( m < p \), and is not in \( EQUAL \). This contradicts the Pumping Lemma, so our assumption was wrong. Thus \( EQUAL \) is not regular.  

\[
\begin{align*}
\text{For every regular language } L, \\
\text{there exists a positive integer } p \text{ such that} \\
\text{for every string } s \in L \text{ with } |s| \geq p, \\
\text{there exist strings } x, y, \text{ and } z \text{ such that} \\
s = xyz \\
|x| \leq p \\
y \neq \varepsilon \\
xy^iz \in L \text{ for all } i \in \mathbb{N}_0
\end{align*}
\]
Critical point

It’s necessary to show there is *no segmentation* of the chosen string that won’t lead to a contradiction.

This means considering *every possible* mapping of $xy$ onto the first $n$ symbols in the chosen string.

We chose our string to make this easy, since every possible segmentation consists of $a$s only.

Pumping therefore disrupts the equivalence of the number of $a$s and $b$s.
Critical point

We only need to show that there’s one string in the language for which the Pumping Lemma doesn’t work.

For some strings in $L$, it may work perfectly well!
The Pumping Lemma mascot, the Pumping Llama

by Kimberly Do
Example

Consider the alphabet $\Sigma = \{0, 1\}$ and the language

$$BALANCE = \{w \mid w \text{ has an equal number of 1s and 0s}\}$$

E.g.,

$$01 \in BALANCE$$
$$110010 \in BALANCE$$
$$11011 \not\in BALANCE$$

Is $BALANCE$ a regular language?
An incorrect proof

THEOREM \( BALANCE \) is regular.

PROOF We show that \( BALANCE \) satisfies the condition of the Pumping Lemma. Let \( p = 2 \) and consider any string \( s \in BALANCE \) such that \( |s| \geq 2 \). Then we can write \( s = xyz \) such that \( x = z = \varepsilon \) and \( y = s \), so \( y \neq \varepsilon \). Then for any natural number \( i \), \( xy^i z = s^i \), which has the same number of 0s and 1s. Since \( BALANCE \) passes the conditions of the Pumping Lemma, \( BALANCE \) is regular.
An incorrect proof

THEOREM  BALANCE is regular.  

PROOF  We show that BALANCE satisfies the condition of the Pumping Lemma. Let \( p = 2 \) and consider any string \( s \in \text{BALANCE} \) such that \( |s| \geq 2 \). Then we can write \( s = xyz \) such that \( x = z = \epsilon \) and \( y = s \), so \( y \neq \epsilon \). Then for any natural number \( i \), \( xy^iz = s^i \), which has the same number of 0s and 1s. Since BALANCE passes the conditions of the Pumping Lemma, BALANCE is regular.

What? No!
For every regular language \( L \), there exists a positive integer \( p \) such that for every string \( s \in L \) with \(|s| \geq p\), there exist strings \( x, y, \) and \( z \) such that

\[
s = xyz
\]

\[
|xy| \leq p
\]

\[
y \neq \varepsilon
\]

\[
xy^iz \in L \text{ for all } i \geq 0
\]
Caution with the Pumping Lemma

The Pumping Lemma describes a *necessary* condition of regular languages.

If $L$ is regular, $L$ passes the conditions of the Pumping Lemma.

The Pumping Lemma is not a *sufficient* condition to be a regular language.

If $L$ is *not* regular, it still might pass the conditions of the Pumping Lemma!
An incorrect proof

THEOREM  $BALANCE$ is regular.

PROOF  We show that $BALANCE$ satisfies the condition of the Pumping Lemma. Let $p = 2$ and consider any string $s \in BALANCE$ such that $|s| \geq 2$. Then we can write $s = xyz$ such that $x = z = \epsilon$ and $y = w$, so $y \neq \epsilon$. Then for any natural number $i$, $xy^iz = s^i$, which has the same number of 0s and 1s. Since $BALANCE$ passes the conditions of the Pumping Lemma, $BALANCE$ is regular.
An incorrect proof

THEOREM  BALANCE is regular.

PROOF  We show that BALANCE satisfies the condition of the Pumping Lemma. Let $p = 2$ and consider any string $s \in BALANCE$ such that $|s| \geq 2$. Then we can write $s = xyz$ such that $x = z = \varepsilon$ and $y = w$, so $y \neq \varepsilon$. Then for any natural number $i$, $xy^iz = s^i$, which has the same number of 0s and 1s. Since BALANCE passes the conditions of the Pumping Lemma, BALANCE is regular.

You don’t get to choose $p$!
An incorrect proof

**THEOREM**  *BALANCE* is regular.

**PROOF**  We show that *BALANCE* satisfies the condition of the Pumping Lemma. Let $p = 2$ and consider any string $s \in \text{BALANCE}$ such that $|s| \geq 2$. Then we can write $s = xyz$ such that $x = z = \epsilon$ and $y = w$, so $y \neq \epsilon$. Then for any natural number $i$, $xy^iz = s^i$, which has the same number of $0$s and $1$s. Since *BALANCE* passes the conditions of the Pumping Lemma, *BALANCE* is regular.

You don’t get to choose $p$!

But you do need to choose a specific string…
An incorrect proof

THEOREM  \( BALANCE \) is regular.

PROOF  We show that \( BALANCE \) satisfies the condition of the Pumping Lemma. Let \( p = 2 \) and consider any string \( s \in BALANCE \) such that \( |s| \geq 2 \). Then we can write \( s = xyz \) such that \( x = z = \varepsilon \) and \( y = w \), so \( y \neq \varepsilon \). Then for any natural number \( i \), \( xy^iz = s^i \), which has the same number of 0s and 1s. Since \( BALANCE \) passes the conditions of the Pumping Lemma, \( BALANCE \) is regular.

You don’t get to choose \( p \! \)!

But you do need to choose a specific string…

And a specific value for \( i \).
Example

CLAIM The language $L = \{w \mid w \text{ has an equal number of 1s and 0s}\}$ is not regular.

Given $p$, we choose the string $(01)^p$.

We need to show splitting this string into $xyz$ where $xy^iz$ is in $L$ is impossible…

But it is possible!

If $x = \epsilon$, $y = 01$, and $z = (01)^{p-1}$, $xy^iz$ is in $L$ for every value of $i$

Are we out of luck?
When using the Pumping Lemma:

*If your string does not succeed, try another!*
Let’s try $1p0p$.

Again, we need to show splitting this string into $xyz$ where $xy^iz$ is in $L$ is impossible…
Let’s try $1^p0^p$.

Again, we need to show splitting this string into $xyz$ where $xy^iz$ is in $L$ is impossible…

**But it is possible!**

If $x$ and $z$ are $\varepsilon$ and $y$ is $1^n0^n$, then $xy^iz$ always has an equal number of 0s and 1s
Let’s try $1^p0^p$.

Again, we need to show splitting this string into $xyz$ where $xy^iz$ is in $L$ is impossible…

**But it is possible!**

If $x$ and $z$ are $\varepsilon$ and $y$ is $1^n0^n$, then $xy^iz$ always has an equal number of $0$s and $1$s

**Are we still in trouble?**
Not this time…

The Pumping Lemma says that our string has to be divided so that $|xy| \leq p$ and $|y| > 0$

If $|xy| \leq p$, then $y$ must consist only of 1s, so $xyyz \not\in L$.

*Contradiction! We win!*
Remember

You only need to find one string for which the Pumping Lemma does not hold to prove a language is not regular.

But you must show that for any decomposition of that string into xyz the Pumping Lemma holds.

This sometimes means considering several different cases.
Proof strategy: Closure properties
Recall: Closure properties

Certain operations on regular languages are guaranteed to produce regular languages.

These *closure properties* can be used to prove a language is regular – or that it’s non-regular.
We’ve seen the regular languages are closed under the *regular operations*, i.e., those used to construct regular expressions:

- **Union**: $L_1 \cup L_2$
- **Concatenation**: $L_1 L_2$
- **Star-closure**: $L_1^*$

And that they’re also closed under these set operations:

- **Complementation**: $\overline{L_1}$
- **Intersection**: $L_1 \cap L_2$
- **Difference**: $L_1 - L_2$
To show that a language $L$ is non-regular using closure properties, we do a proof by contradiction:

Assume $L$ is regular.

Combine $L$ with known regular languages using operations the regular languages are closed under.

If you produce a known non-regular language, then the assumption was wrong and $L$ is non-regular.
CLAIM  $L = \{w \mid w \text{ in } \{a,b\}^* \mid n_a(w) = n_b(w)\}$ is non-
regular.

PROOF SKETCH  Assume $L$ is regular. We know $a^*b^*$ is 
a regular language because we can write it as a 
regular expression. Because the regular languages 
are closed under intersection,

$$L \cap a^*b^* = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

must be regular. However, $\{a^n b^n \mid n \in \mathbb{N}_0\}$ is easily 
proved non-regular using the Pumping Lemma. 
(As we did!)

Therefore, $L$ must be non-regular.
What the closure theorem for union does *not* say

Closure theorem for union says: If $L_1$ and $L_2$ are regular, then $L = L_1 \cup L_2$ is regular.

What happens if (for example) $L$ is regular? Does that mean that $L_1$ and $L_2$ are also?
What the closure theorem for union does *not* say

Closure theorem for union says: If $L_1$ and $L_2$ are regular, then $L = L_1 \cup L_2$ is regular.

What happens if (for example) $L$ is regular? Does that mean that $L_1$ and $L_2$ are also?

Maybe.
Example

We know $a^+$ is regular.

Consider two cases for $L_1$ and $L_2$:

$$a^+ = \{a^n \mid n > 0 \text{ and } n \text{ is prime}\} \cup \{a^n \mid n > 0 \text{ and } n \text{ is not prime}\}$$

$$a^+ = L_1 \cup L_2$$
Example

We know $a^+$ is regular.

Consider two cases for $L_1$ and $L_2$:

$$a^+ = \{a^n \mid n > 0 \text{ and } n \text{ is prime}\} \cup \{a^n \mid n > 0 \text{ and } n \text{ is not prime}\}$$

$$a^+ = L_1 \cup L_2$$

*Neither $L_1$ nor $L_2$ is regular!*

$$a^+ = \{a^n \mid n > 0 \text{ and } n \text{ is even}\} \cup \{a^n \mid n > 0 \text{ and } n \text{ is odd}\}$$

$$a^+ = L_1 \cup L_2$$
Example

We know $a^+$ is regular.

Consider two cases for $L_1$ and $L_2$:

\[
a^+ = \{a^n \mid n > 0 \text{ and } n \text{ is prime}\} \cup \{a^n \mid n > 0 \text{ and } n \text{ is not prime}\}
\]
\[
a^+ = L_1 \cup L_2
\]

*Neither $L_1$ nor $L_2$ is regular!*

\[
a^+ = \{a^n \mid n > 0 \text{ and } n \text{ is even}\} \cup \{a^n \mid n > 0 \text{ and } n \text{ is odd}\}
\]
\[
a^+ = L_1 \cup L_2
\]

*Both $L_1$ and $L_2$ are regular!*
Where are we?
We’ve ended up where we are by trying to answer the question “what problems can you solve with a computer?”

We defined a computer to be a DFA, which means that the problems we can solve are precisely the regular languages.

We’ve discovered several equivalent ways to think about regular languages (DFAs, NFAs, and regular expressions).
We now have a powerful intuition for these languages:

DFAs are finite-memory computers, and regular languages correspond to the problems solvable with finite memory.
Using the Pumping Lemma, we’ve shown that there are languages that are not regular!

Does that mean these languages aren’t computable?
Next

What does computation look like with unbounded memory?

What problems can you solve with unbounded-memory computers?
Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame
Nancy Ide, Vassar College
Keith Schwarz, Stanford University
Michael Sipser, *Introduction to the Theory of Computation*