Non-Regular Languages (1.4)

COT 4210 Discrete Structures II
Summer 2025
Department of Computer Science
Dr. Steinberg

Non-Regular Languages

We need to know our limitations!

A Language to Consider

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

Is L regular?

A Language to Consider

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

Is L regular?

NO!

L has to count the number of 0's and 1's. Plus that number is arbitrary.

A Language to Consider

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

Is L regular?

Do you remember what the F in FSM, DFA, and NFA stand for?

A Language to Consider

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

Is L regular?

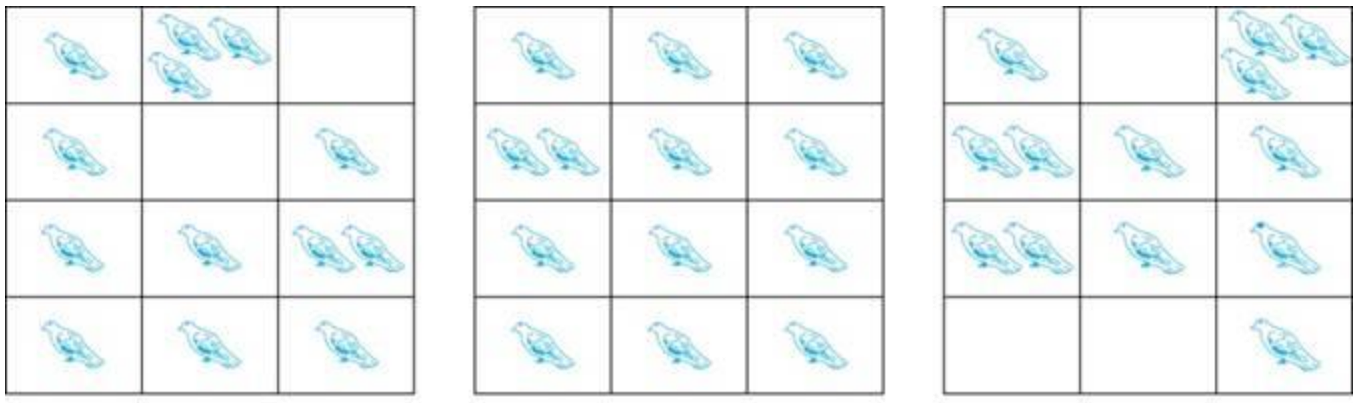
FINITE!!!!

Pigeonhole Principle

Time for a flashback because this does play a role!

The Pigeonhole Principle

- Suppose if a flock of 20 pigeons roosts in a set of 19 pigeonholes, one of the pigeonholes must have more than 1 pigeon.
- **Pigeonhole Principle Theorem**: If k is a positive integer and $k + 1$ objects are placed into k boxes, then at least one box contains two or more objects.
- **Proof**: Lets use a proof by contraposition. Suppose none of the k boxes has more than one object. Then the total number of objects would be at most k. This contradicts the statement that we have $k + 1$ objects.



(a) (b) (c)

Image from rosen, discrete math and its applications eighth edition , mcgraw hill, 2019

The Pigeonhole Principle

- Suppose if a flock of 20 pigeons roosts in a set of 19 pigeonholes, one of the pigeonholes must have more than 1 pigeon.
- **Pigeonhole Principle Theorem**: If k is a positive integer and $k + 1$ objects are placed into k boxes, then at least one box contains two or more objects.
- **Proof**: Lets use … n one object. Then the total n … at we have $k + 1$ objects.

Don't worry you will understand where this is going!

(a) (b) (c)

Image from rosen, discrete math and its applications eighth edition , mcgraw hill, 2019

Counting and DFAs

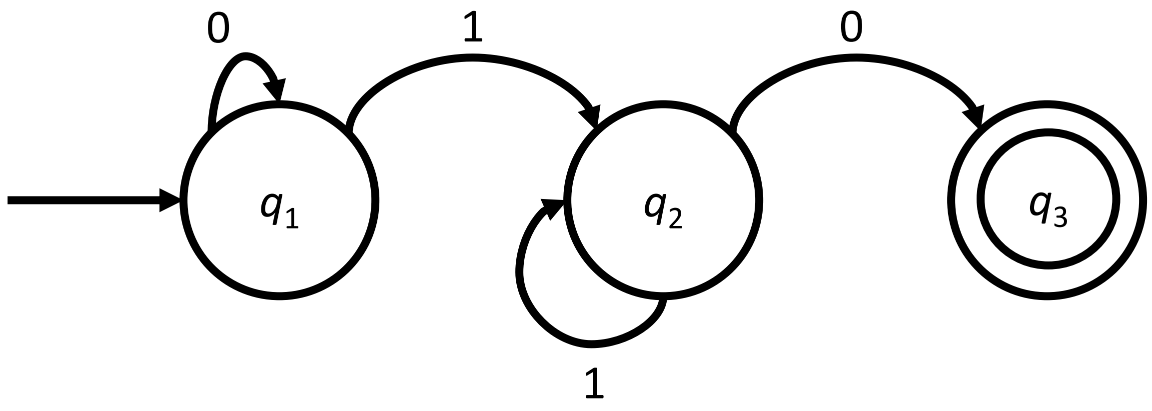
This is where we are going!

Pigeonholes and DFAs

Now consider a DFA, and consider a string we are accepting

- Say that **the string has as many symbols as the DFA has states**
- What does that mean we can say, with complete certainty?

110



Pigeonholes and DFAs

Now consider a DFA, and consider a string we are accepting

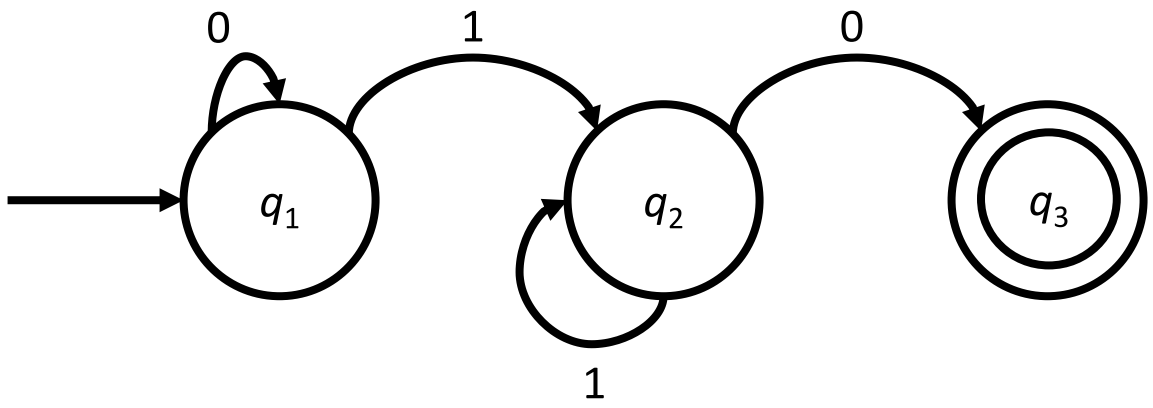
- Say that **the string has as many symbols as the DFA has states**

- What does that mean we can say, with complete certainty?

We cycled at least once

- But that means…

110



Pigeonholes and DFAs

Now consider a DFA, and consider a string we are accepting

- Say that **the string has as many symbols as the DFA has states**
- What does that mean we can say, with complete certainty?

We cycled at least once

But that means…

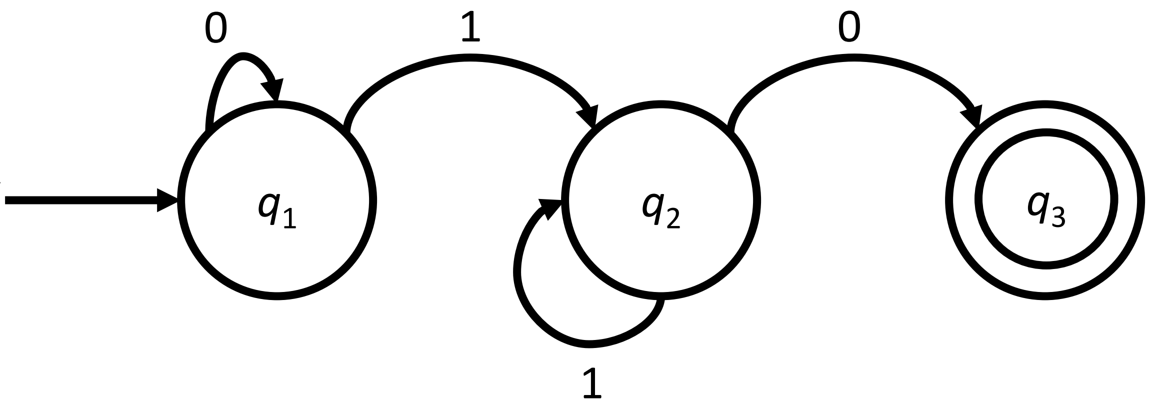
We can run that same cycle indefinitely

10

1**1**0

1**11**0

1**1111111111111111111111111**0



The Pumping Lemma

The Pumping Lemma for Regular Languages

If A is a regular language, then there is a number p – the **pumping length** – so that if s is a string in A with length of at least p, then $s = xyz$ so that:

- xy^iz is a string in A for all $i \geq 0$,
- $|y| > 0$, and
- $|xy| \leq p$

Notes:

- y^i just means "y concatenated to itself i times"
- $|s|$ means the length of a string s
- x and z can be empty, but $\boldsymbol{y}$ **can't be** – this is the whole point of the lemma
- We call it a lemma because all it's good for is showing that some languages **aren't** regular

Proof Idea: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in $\boldsymbol{A}$ for all $\boldsymbol{i \geq 0}$**, with $|\boldsymbol{y}| > \mathbf{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Proof Idea: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in $\boldsymbol{A}$ for all $\boldsymbol{i \geq 0}$**, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

We already showed the important parts of this.

- We go around a cycle – that is, we hit at least one state at least twice
- x is the part of the string **before** the cycle, y is the **cyclic** part, and z is the part **after** the cycle

Proof Idea: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

We already showed the important parts of this.

- We go around a cycle – that is, we hit at least one state at least twice
- x is the part of the string **before** the cycle, y is the **cyclic** part, and z is the part **after** the cycle

All we're saying is:

1. We can go around the cycle **as many times as we want**, since it's a cycle

Proof Idea: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in $\boldsymbol{A}$ for all $\boldsymbol{i \geq 0}$**, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

We already showed the important parts of this.

- We go around a cycle – that is, we hit at least one state at least twice
- x is the part of the string **before** the cycle, y is the **cyclic** part, and z is the part **after** the cycle

All we're saying is:

1. We can go around the cycle **as many times as we want**, since it's a cycle
2. The before and after parts can be empty, but **the cyclic part can't be empty** or we don't have enough states

Proof Idea: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

We already showed the important parts of this.

- We go around a cycle – that is, we hit at least one state at least twice
- x is the part of the string **before** the cycle, y is the **cyclic** part, and z is the part **after** the cycle

All we're saying is:

1. We can go around the cycle **as many times as we want**, since it's a cycle
2. The before and after parts can be empty, but **the cyclic part can't be empty** or we don't have enough states
3. We have to hit some state twice **by the time we hit a number of symbols equal to the number of states**

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^i z}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Let $s = s_1 s_2 \ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1 r_2 \ldots r_{n+1}$ be the sequence of states that M enters while computing S.

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Let $s = s_1 s_2 \ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1 r_2 \ldots r_{n+1}$ be the sequence of states that M enters while computing S. Observe that:

- The state sequence has length $n + 1$, which is at least $p + 1$
- Within the first $p + 1$ states in the sequence, two <u>different points in the sequence</u> have to be the same state, by the pigeonhole principle
- Call the first one r_j and the second one r_k

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Let $s = s_1 s_2 \ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1 r_2 \ldots r_{n+1}$ be the sequence of states that M enters while computing S. Observe that:

- The state sequence has length $n + 1$, which is at least $p + 1$
- Within the first $p + 1$ states in the sequence, two different points in the sequence have to be the same state, by the pigeonhole principle
- Call the first one r_j and the second one r_k

Now let $x = s_1 \ldots s_{j-1}$, $y = s_j \ldots s_{k-1}$, and $z = s_k \ldots s_n$.

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Let $s = s_1 s_2 \ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1 r_2 \ldots r_{n+1}$ be the sequence of states that M enters while computing S. Observe that:

- The state sequence has length $n + 1$, which is at least $p + 1$
- Within the first $p + 1$ states in the sequence, two <u>different points in the sequence</u> have to be the same state, by the pigeonhole principle
- Call the first one r_j and the second one r_k

Now let $x = s_1 \ldots s_{j-1}$, $y = s_j \ldots s_{k-1}$, and $z = s_k \ldots s_n$.

Observe that:

- x takes M from r_1 to r_j
- y takes M from r_j to r_k
- z takes M from r_k to r_{n+1}
- But $\boldsymbol{r_j}$ **and** $\boldsymbol{r_k}$ **are the same state!**
- **Therefore,** $\boldsymbol{M}$ **must accept** $\boldsymbol{xy^iz}$ **for all** $\boldsymbol{i \geq 0}$**.**

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $\boldsymbol{s = xyz}$ so that $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $|\boldsymbol{y}| > \boldsymbol{0}$ and $|\boldsymbol{xy}| \leq \boldsymbol{p}$.

Let $s = s_1 s_2 \ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1 r_2 \ldots r_{n+1}$ be the sequence of states that M enters while computing S. Observe that:

- The state sequence has length $n + 1$, which is at least $p + 1$
- Within the first $p + 1$ states in the sequence, two different points in the sequence have to be the same state, by the pigeonhole principle
- Call the first one r_j and the second one r_k

Now let $x = s_1 \ldots s_{j-1}$, $y = s_j \ldots s_{k-1}$, and $z = s_k \ldots s_n$.

Observe that:

- x takes M from r_1 to r_j
- y takes M from r_j to r_k
- z takes M from r_k to r_{n+1}
- **But** $\boldsymbol{r_j}$ **and** $\boldsymbol{r_k}$ **are the same state!**
- **Therefore,** $\boldsymbol{M}$ **must accept** $\boldsymbol{xy^iz}$ **for all** $\boldsymbol{i \geq 0}$**.**

Observe that:

- Since $j \neq k, |y| > 0$

Proof: The Pumping Lemma for Regular Languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing language A, and let $p = |Q|$.

- Consider $s \in A$ so that $|s| = n$, with $n \geq p$.
- Show that $s = xyz$ so that **xy^iz is a string in A for all $i \geq 0$**

Let $s = s_1 s_2 \dots s_n$ be

Let $r_1 r_2 \dots r_{n+1}$ be t
enters while computing S. Observe that:

- The state sequence has length $n + 1$, which is at least $p + 1$
- Within the first $p + 1$ states in the sequence, two different points in the sequence have to be the same state, by the pigeonhole principle
- Call the first one r_j and the second one r_k

We have shown that all three conditions of the pumping lemma hold. □

Now let $x = s_1 \dots s_{j-1}$, $y = s_j \dots s_{k-1}$, and $z = s_k \dots s_n$.

Observe that:

- x takes M from r_1 to r_j

$_{-1}$

ame state!

- **Therefore, M must accept xy^iz for all $i \geq 0$.**

Observe that:

- Since $j \neq k, |y| > 0$

Finally, observe that:

- Since $k \leq p + 1, |xy| \leq p$

Using the Pumping Lemma

The pumping lemma is basically only good for proofs by contradiction. Three steps:

1. **Set Up the Pump**
   - **Assume** a language *A* is regular
   - Observe that, therefore, by the pumping lemma, there is a *p* so that any string *s* in *A*, of length *p* or greater, can be cut into *xyz* and **pumped**
     - You don't need to know what *p* is – only that it exists!
2. **Break the Pump**
   - Find a string *s* in it, of length *p* or greater, that **can't** be pumped
   - Demonstrate that no matter how you cut it into *xyz*, it **still** can't be pumped
     - Remember **all** parts of the pumping lemma here – part 3 can be more useful than you'd think
3. **Clean Up the Mess**
   - Observe that since string *s* in *A*, of length *p* or greater, can't be pumped; and *A* is regular; we have a **contradiction** with the pumping lemma
   - Conclude that ***A* is not regular**

Examples of Non-Regular Languages

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $n > 0$ such that $\forall w \in L$, $|w| \geq n$, w can be decomposed $w = xyz$ with $|xy| \leq n$ and $|y| > 0$ and $w_i = xy^i z \in L$.

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $n > 0$ such that $\forall w \in L$, $|w| \geq n$, w can be decomposed $w = xyz$ with $|xy| \leq n$ and $|y| > 0$ and $w_i = xy^i z \in L$.
3. Choose our string w. $w = a^n b^n$ ($w = xyz$). Since $|xy| \leq n \Rightarrow y$ contains only a's. $y = a^k$ with $k \geq 1$
   Let $x = a^u$

$$w = a^u \ a^k \ a^{n-u-k} b^n$$

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $n > 0$ such that $\forall w \in L$, $|w| \geq n$, w can be decomposed $w = xyz$ with $|xy| \leq n$ and $|y| > 0$ and $w_i = xy^iz \in L$.
3. Choose our string w. $w = a^n b^n$ ($w = xyz$). Since $|xy| \leq n \Rightarrow y$ contains only a's. $y = a^k$ with $k \geq 1$ Let $x = a^u$

$$w = \underbrace{a^u}_{x} \underbrace{a^k}_{y} \underbrace{a^{n-u-k} b^n}_{z}$$

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $n > 0$ such that $\forall w \in L$, $|w| \geq n$, w can be decomposed $w = xyz$ with $|xy| \leq n$ and $|y| > 0$ and $w_i = xy^iz \in L$.
3. Choose our string w. $w = a^n b^n$ ($w = xyz$). Since $|xy| \leq n \Rightarrow y$ contains only a's. $y = a^k$ with $k \geq 1$ Let $x = a^u$

$$w = \underbrace{a^u}_{x}\ \underbrace{a^k}_{y}\ \underbrace{a^{n-u-k}b^n}_{z}$$

$$w_i = a^u\left(a^k\right)^i a^{n-u-k}b^n = a^{n-k}\left(a^k\right)^i b^n$$

Example 1: Show that $L = \{a^n b^n \mid n > 0\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $n > 0$ such that $\forall w \in L$, $|w| \geq n$, w can be decomposed $w = xyz$ with $|xy| \leq n$ and $|y| > 0$ and $w_i = xy^i z \in L$.
3. Choose our string w. $w = a^n b^n$ ($w = xyz$). Since $|xy| \leq n \Rightarrow y$ contains only a's. $y = a^k$ with $k \geq 1$ Let $x = a^u$

$$w = \underbrace{a^u}_{x} \underbrace{a^k}_{y} \underbrace{a^{n-u-k} b^n}_{z}$$

$$w_i = a^u \left(a^k\right)^i a^{n-u-k} b^n = a^{n-k} \left(a^k\right)^i b^n$$

4. Choose i. $i = 0$ $w_0 = a^{n-k} \left(a^k\right)^0 b^n = a^{n-k} b^n$

$$a^{n-k} b^n \notin L \Rightarrow \text{Contradicts our ASSUMPTION!}$$

$$\Rightarrow L \text{ is NOT regular}$$

Example 2: Show that $L = \{ww^R \mid w \in \{a, b\}^*\}$ is not regular.

Example 2: Show that $L = \{ww^R \mid w \in \{a, b\}^*\}$ is not regular.

1. Assume by contradiction that L is regular.

Example 2: Show that $L = \{ww^R \mid w \in \{a, b\}^*\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $m > 0$ such that $\forall w \in L$, $|w| \geq m$, w can be decomposed $w = xyz$ with $|xy| \leq m$ and $|y| > 0$ and $w_i = xy^iz \in L$.

Example 2: Show that $L = \left\{ww^R \mid w \in \{a,b\}^*\right\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $m > 0$ such that $\forall w \in L$, $|w| \geq m$, w can be decomposed $w = xyz$ with $|xy| \leq m$ and $|y| > 0$ and $w_i = xy^iz \in L$.
3. Choose our string w. $w = a^m b^m b^m a^m$ ($w = xyz$). Since $|xy| \leq m$ $|y| \geq 1 \Rightarrow y$ contains only a's.
   $y = a^k$ with $k \geq 1$

Example 2: Show that $L = \{ww^R \mid w \in \{a, b\}^*\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $m > 0$ such that $\forall w \in L$, $|w| \geq m$, w can be decomposed $w = xyz$ with $|xy| \leq m$ and $|y| > 0$ and $w_i = xy^iz \in L$.
3. Choose our string w. $w = a^m b^m b^m a^m$ ($w = xyz$). Since $|xy| \leq m$ $|y| \geq 1 \Rightarrow y$ contains only a's. $y = a^k$ with $k \geq 1$

$$w = \underbrace{a^{m-k}}_{x}\ \underbrace{a^k}_{y}\ \underbrace{b^m b^m a^m}_{z}$$

$$w_i = a^{m-k}\left(a^k\right)^i b^m b^m a^m \in L$$

Example 2: Show that $L = \{ww^R \mid w \in \{a,b\}^*\}$ is not regular.

1. Assume by contradiction that L is regular.
2. Pumping Lemma states there exists $m > 0$ such that $\forall w \in L$, $|w| \geq m$, w can be decomposed $w = xyz$ with $|xy| \leq m$ and $|y| > 0$ and $w_i = xy^iz \in L$.
3. Choose our string w. $w = a^m b^m b^m a^m$ ($w = xyz$). Since $|xy| \leq m$ $|y| \geq 1 \Rightarrow y$ contains only a's.
   $y = a^k$ with $k \geq 1$

$$w = \underbrace{a^{m-k}}_{x}\,\underbrace{a^k}_{y}\,\underbrace{b^m b^m a^m}_{z}$$

$$w_i = a^{m-k}\left(a^k\right)^i b^m b^m a^m \in L$$

4. Choose i. $i = 0$ $w_0 = a^{m-k}\left(a^k\right)^0 b^m b^m a^m = a^{m-k}\ b^m b^m a^m$

$$a^{m-k}\ b^m b^m a^m \notin L \Rightarrow \text{Contradicts our ASSUMPTION!}$$
$$\Rightarrow L \text{ is NOT regular}$$

More Examples on the Board

Categorizing Languages

- We have shown that there are plenty of languages we can't process using the tools we use for regular languages
- That does *not* mean we can't process them
  - Any language we can think of an algorithm to recognize can be recognized
  - It just can't be done with a DFA
- We consider regular languages the simplest class of languages worth putting serious thought into
- We have other tools for processing more complex classes of languages
- Over the next few weeks, we will walk our way up this *hierarchy of languages*

Acknowledgement

Some Notes and content come from Dr. Gerber, Dr. Hughes, and Mr. Guha's COT4210 class and the Sipser Textbook, *Introduction to the Theory of Computation*, 3rd ed., 2013