Lecture 6

COT4210 DISCRETE STRUCTURES

DR. MATTHEW B. GERBER

6/7/2016

PORTIONS FROM SIPSER, *INTRODUCTION TO THE THEORY OF COMPUTATION*, 3RD ED., 2013

A Language to Consider

$$B = \{ 0^n 1^n \mid n \geq 0 \}$$

- Is *B* regular?

A Language to Consider

$$B = \{ 0^n 1^n \mid n \geq 0 \}$$

- Is *B* regular?
- No
- B has to count the number of zeroes – and that number is arbitrary
- What does the **F** in FSM, DFA and NFA stand for?

Pigeons and Pigeonholes

Here we see nine pigeons in nine pigeonholes

- If we put ten pigeons in these nine pigeonholes, we can say one thing for certain:
- *There is at least one pigeonhole with more than one pigeon*

We call the generalization of this idea the *pigeonhole principle*:

If *n* items are put into *m* containers with *n* > *m*, at least one container contains more than one item

- It's technically not an axiom, but like induction, it's so basic that we don't really call it a theorem

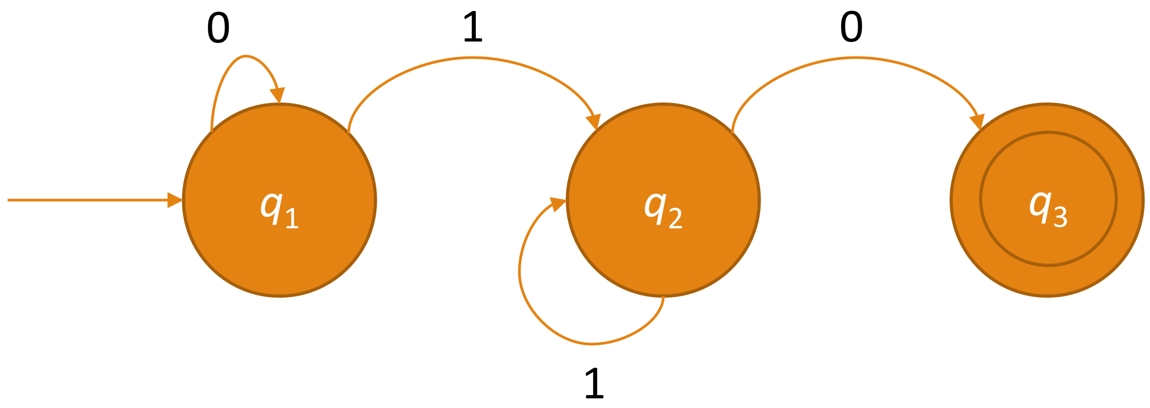
Image: Wikimedia Commons, "Too Many Pigeons", McKay from BenFrantzDale

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

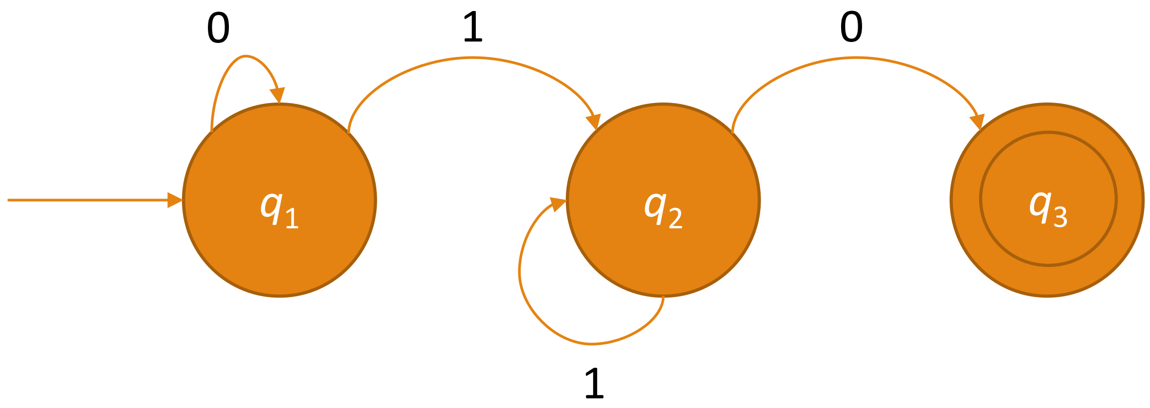
Now consider a DFA, and consider a string we are accepting

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

110

We cycled at least once

But that means…



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?

10

1**1**0

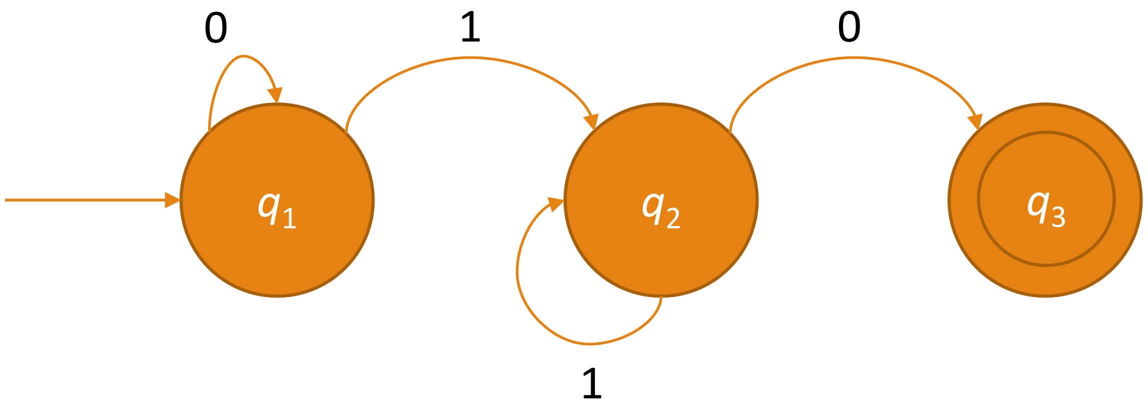
1**11**0

1**11111111111111111111111111**0

We cycled at least once

But that means…

We can run that same cycle indefinitely



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 ***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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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 $\boldsymbol{xy^iz}$ **is a string in** $\boldsymbol{A}$ **for all** $\boldsymbol{i \geq 0}$, with $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

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

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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

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* ≥ 0**, with **|*y*| > 0** and **|*xy*| ≤ *p***.

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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
- 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 ***s* = *xyz*** so that ***xy^i^z* is a string in *A* for all *i* ≥ 0**, with **|*y*| > 0** and **|*xy*| ≤ *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
- But **r_j and r_k are the same state!**
- **Therefore, M must accept xy^iz for all $i \geq 0$.**

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 $i \geq 0$**, with **$|y| > 0$** and **$|xy| \leq p$**.

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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
- But **r_j and r_k are the same state!**
- **Therefore, M must accept xy^iz for all $i \geq 0$.**

Observe that:

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

Finally, observe that:

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

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 $\boldsymbol{|y| > 0}$ and $\boldsymbol{|xy| \leq p}$.

Let $s = s_1s_2\ldots s_n$ be a string accepted by M, with $n \geq p$

Let $r_1r_2\ldots r_{n+1}$ 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
- **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 $j \neq k$, $|y| > 0$

Finally, observe that:

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

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

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**

Some Non-Regular Languages

(Board Work: 1.73, 1.74, 1.75, 1.76, 1.77)

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
  - Obviously, 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*

Next Time:
Context-Free Languages