Lecture 9: Proving non-regularity

17 February 2009

Reminder: The first midterm Exam, takes place on Tuesday, 24 of February 7-9pm room in 151 Loomis. Be there. Please check for conflicts NOW. If you have one, send a note to Sariel or Madhu explaining the nature of the conflict and including your schedule.

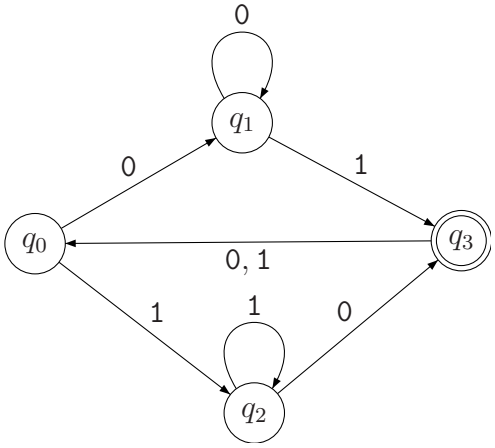
In this lecture, we will see how to prove that a language is **not** regular.

We will see two methods for showing that a language is not regular. The "pumping lemma" shows that certain key "seed" languages are not regular. From these seed languages, we can show that many similar languages are also not regular, using closure properties.

1 State and regularity

1.1 How to tell states apart

You are given the following DFA M, but we do not know what is its initial state (it was made in India, and the initial state indicator was broken during the shipment to the US). You want to figure out what is the initial state of this DFA.



You can do any of the following operations:

(i) Reset the DFA to its (unknown) initial position.

(ii) Feed input into the DFA.

The rule of the game is that when the DFA is in a final state, you would know it.

So, the question is how to decide in the above DFA what is the initial state?

Here is one possible solution.

1. Check if M is in already in a final state. If so q_3 is the initial state.

2. Otherwise, feed 0 to M. If M is now in a final state, then q_2 is the initial state.

3. Reset M, and feed it 1. If it accepts, then q_1 is the initial state.

4. Reset M, and feed it 01. If it accepts, then q_0 is the initial state.

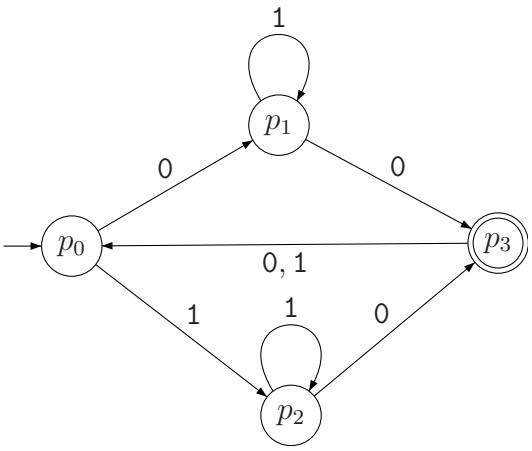
Definition 1.1 For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, $p \in Q$ and $x \in \Sigma^*$, let $M(p, x)$ be true if setting the DFA to be in the state p, and then reading the input x causes M to arrive to an accepting state. Formally, $M(p, x)$ is true if and only if $\delta(p, x) \in F$, and false otherwise.

The moral of this story. So, we can differentiate between two states p and q of a DFA M, by finding strings x and y, such that $M(p, x)$ accepts, but $M(q.y)$ rejects, or vice versa. If x le.

Definition 1.2 Two states p and q of a DFA M ***disagree*** with each other, if there exists a string x, such that $M(p, x) \neq M(q, x)$ (that is, $M(p, x)$ accepts but $M(q, x)$ rejects, or vice versa).

Example 1.3 Note, that two states might be different, but yet the agree on all possible strings x. For example, consider the the DFA on the right.

Clearly, p_0 and p_1 disagree (for example on 0). But notice that p_1 and p_2 agree on all possible strings.



Lemma 1.4 *Let M be a DFA and let $q_1, \ldots q_n$ be a list of states of M, such that any pair of them disagree. Then, M must have at least n states.*

Proof: For $i \neq j$, since q_i and q_j disagree with each other, they can not possibly be the same state of M, since if they were the same state then they would agree with each other on all possible strings. We conclude that $q_1, \ldots q_n$ are all different states of M; namely, M has at least n different states. ■

1.1.1 A Motivating Example

Consider the language $L = \left\{ \mathtt{a}^n \mathtt{b}^n \;\middle|\; n \geq 0 \right\}$. Intuitively, L can not be regular, because we have to remember how many a's we have seen before reading the b's, and this can not be done with a finite number of states.

Claim 1.5 *The language $L = \left\{ \mathtt{a}^n \mathtt{b}^n \;\middle|\; n \geq 0 \right\}$ is not regular.*

Proof: Suppose that L were regular. Then L is accepted by some DFA

$$M = (Q, \Sigma, \delta, q_0, F).$$

Let q_i denote the state M is in, after reading the string $\mathtt{a}^i$, for $i = 0, 1, 2, \ldots, \infty$. We claim that q_i disagrees with q_j if $i \neq j$. Indeed, observe that $M(q_i, \mathtt{b}^i)$ accepts but $M(q_j, \mathtt{b}^i)$ rejects, since $\mathtt{a}^i\mathtt{b}^i \in L$ and $\mathtt{a}^i\mathtt{b}^j \notin L$. As such, by Lemma 1.4, M has an infinite number of state, which is impossible. ■

2 Irregularity via differentiation

Definition 2.1 Two strings $x, y \in \Sigma^*$ are ***distinguishable*** by $L \subseteq \Sigma^*$, if there exists a word $w \in \Sigma^*$, such that *exactly one* of the strings xw and yw is in L.

Lemma 2.2 *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a given DFA, and let $x \in \Sigma^*$ and $y \in \Sigma^*$ be two strings distinguishable by $L(M)$. Then $q_x \neq q_y$, where $q_x = \delta(q_0, x)$ (i.e., the state M is in after reading x) and $q_y = \delta(q_0, y)$ is the state that M is in after reading y.*

Proof: Indeed, let w be the string causing x and y to be distinguished by $L(M)$, and assume that $xw \in L(M)$ and $xy \notin L(M)$ (the other case is symmetric). Clearly, if $q_x = q_y$, then $M(q_0, xw) = M(q_x, w) = M(q_y, w) = M(q_0, yw)$, but it is given to us that $M(q_0, xw) \neq M(q_0, yw)$ since exactly one of the words xw and yw is in $L(M)$. ■

Lemma 2.3 *Let L be a language, and let $W = \{w_1, w_2, w_3, \ldots\}$ be an infinite set of strings, such that every pair of them is distinguishable by L. Then L is not a regular language.*

Proof: Assume for the sake of contradiction, that L is regular, and let M be a DFA for $M = (Q, \Sigma, \delta, q_0, F)$. Let us set $q_i = \delta(q_0, w_i)$. For $i \neq j$, w_i and w_j are distinguishable by L, and this implies by Lemma 2.2, that $q_i \neq q_j$. This implies that M has an infinite number of states, which is of course impossible. ■

2.1 Examples

2.1.1 Example

Lemma 2.4 *The language*

$$L = \left\{ \mathtt{1}^k y \;\middle|\; y \in \{\mathtt{0}, \mathtt{1}\}^*, \text{ and } y \text{ contains at most } k \text{ ones} \right\}$$

is not regular.

Proof: Let $w_i = \mathtt{1}^i$, for $i \geq 0$. Observe that for $j > i$ we have that $w_i\mathtt{0}\mathtt{1}^j = \mathtt{1}^i\mathtt{0}\mathtt{1}^j \notin L$ but $w_j\mathtt{0}\mathtt{1}^j = \mathtt{1}^j\mathtt{0}\mathtt{1}^j \in L$. As such, w_i and w_j are distinguishable by L, for any $i \neq j$. We conclude, by Lemma 2.3, that L is not regular. ■

2.1.2 Example: ww is not regular

Claim 2.5 *For $\Sigma = \{0, 1\}$, the language $L = \left\{ ww \;\middle|\; w \in \Sigma^* \right\}$ is not regular.*

Proof: Set $w_i = 0^i$. And observe that, for $j > i$, we have that

$$0^i \underbrace{10^j 1}_{x_j} = w_i 1 w_j 1 \notin L \quad \text{but} \quad w_j 1 w_j 1 = 0^j \underbrace{10^j 1}_{x_j} \in L$$

but this implies that w_i and w_j are distinguishable by L, using the string $x_j = 10^j 1$. As such, by Lemma 2.3, we have that L is not regular. ■

3 The Pumping Lemma

3.1 Proof by repetition of states

We next prove Claim 1.5 by a slightly different argument.

Claim. *The language $L = \left\{ a^n b^n \;\middle|\; n \geq 0 \right\}$ is not regular.*

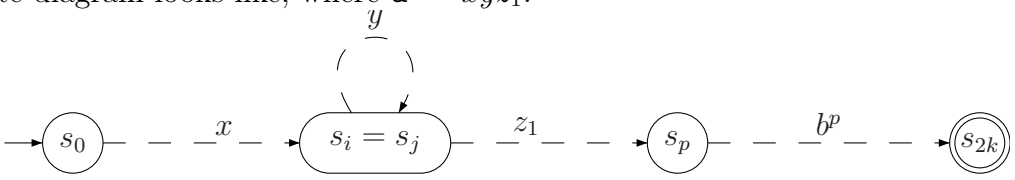
Proof: Suppose that L were regular. Then L is accepted by some DFA

$$M = (Q, \Sigma, \delta, q_0, F).$$

Suppose that M has p states.

Consider the string $a^p b^p$. It is accepted using a sequence of states $s_0 s_1 \ldots s_{2p}$. Right after we read the last a, the machine is in state s_p.

In the sub-sequence $s_0 s_1 \ldots s_p$, there are $p + 1$ states. Since L has only p distinct states, this means that two states in the sequence are the same (by the pigeonhole principle). Let us call the pair of repeated states q_i and q_j, $i < j$. This means that the path through M's state diagram looks like, where $a^p = xyz_1$.



But this DFA will accept all strings of the form $xy^j z_1 b^p$, for $j \geq 0$. Indeed, for $j = 0$, this is just the string $xz_1 b^p$, which this DFA accepts, but it is not in the language, since it has less as than bs. That is, if $|y| = m$, the DFA accepts all strings of the form $a^{p-m+jm} b^m$, for any $j \geq 0$. For any value of j other than 1, such strings are not in L.

So our DFA M accepts some strings that are not in L. This is a contradiction, because L was supposed to accept L. Therefore, we must have been wrong in our assumption that L was regular. ■

3.2 The pumping lemma

The pumping lemma generalizes the above argument into a standard template, which we can prove once and then quickly apply to many languages.

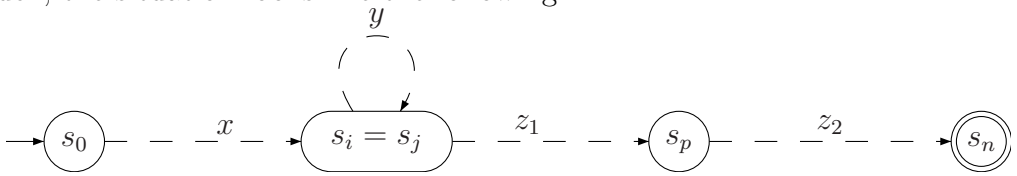
Theorem 3.1 (Pumping Lemma.) *Let L be a regular language. Then there exists an integer p (the "pumping length") such that for any string $w \in L$ with $|w| \geq p$, w can be written as xyz with the following properties:*

- $|xy| \leq p$.
- $|y| \geq 1$ *(i.e. y is not the empty string).*
- $xy^kz \in L$ *for every $k \geq 0$.*

Proof: The proof is written out in full detail in Sipser, here we just outline it.

Let M be a DFA accepting L, and let p be the number of states of M. Let $w = c_1c_2 \dots c_n$ be a string of length $n \geq p$, and let the accepting state sequence (i.e., trace) for w be $s_0s_1 \dots s_n$.

There must be a repeat within the sequence from s_0 to s_p, since M has only p states, and as such, the situation looks like the following.



So if we set $z = z_1z_2$, we now have x, y, and z satisfying the conditions of the lemma.

- $|xy| \leq p$ because repeat is within first $p + 1$ states
- $|y| \geq 1$ because i and j are distinct
- $xy^kz \in L$ for every $k \geq 0$ because a loop in the state diagram can be repeated as many or as few times as you want.

  Formally, for any k, the word xy^iz goes through the following sequence of states:

$$s_0 \xrightarrow{\mathtt{x}} \overbrace{s_i \xrightarrow{\mathtt{y}} s_i \xrightarrow{\mathtt{y}} \cdots \xrightarrow{\mathtt{y}}}^{k \text{ times}} s_i = s_j \xrightarrow{\mathtt{z}} s_n,$$

  and s_n is an accepting state. Namely, M accepts xy^kz, and as such $xy^kz \in L$.

This completes the proof of the theorem. ∎

Notice that we do not know exactly where the repeat occurs, so we have very little control over the length of x and z_1.

3.3 Using the PL to show non-regularity

If L is regular, then it satisfies the pumping lemma (PL). Therefore, intuitively, if L does not satisfy the pumping lemma, L cannot be regular.

3.3.1 Restating the Pumping Lemma via the contrapositive

We want to restate the pumping lemma in the contrapositive. Now, it is **not** true that if L satisfies the conditions of the PM, then L must be regular. Reminder from CS 173: contrapositive of if-then statement is equivalent, converse is not.

What does it mean to **not** satisfy the Pumping Lemma? Write out PL compactly:

$$\begin{array}{c}\text{L is}\\ \text{regular.}\end{array} \implies \left(\exists p \; \forall w \in L \quad |w| \geq p \Rightarrow \left(\exists x,y,z \text{ s.t. } \begin{array}{l} w = xyz, \\ |xy| \leq p, \\ |y| \geq 1, \end{array} \text{ and } \forall i \quad xy^i z \in L. \right) \right).$$

Now, we know that if A implies B, then $\overline{B}$ implies $\overline{A}$ (contraposition), as such the Pumping Lemma, can be restated as

$$\overline{\left(\exists p \; \forall w \in L \quad |w| \geq p \Rightarrow \left(\exists x,y,z \; \begin{array}{l} w = xyz, \\ |xy| \leq p, \\ |y| \geq 1, \end{array} \text{ and } \forall i \quad xy^i z \in L. \right) \right)} \implies \overline{L \text{ is regular.}}$$

Now, the logical statement $A \Rightarrow B$ is equivalent to $\overline{A} \vee B = \overline{A \wedge \overline{B}}$. As such $\overline{A \Rightarrow B} = A \wedge \overline{B}$. In addition, negation flips quantifies, as such, the above is equivalent to

$$\left(\forall p \; \exists w \in L \quad |w| \geq p \text{ and } \overline{\left(\exists x,y,z \; \begin{array}{l} w = xyz, \\ |xy| \leq p, \\ |y| \geq 1, \end{array} \text{ and } \forall i \quad xy^i z \in L. \right)} \right) \implies \begin{array}{c}\text{L is}\\ \text{not regular.}\end{array}$$

Since, $\overline{A \wedge \overline{B}} = A \Rightarrow B$ we have that $\overline{A \wedge B} = \big(A \Rightarrow \overline{B}\big)$. Thus, we have

$$\left(\forall p \; \exists w \in L \quad |w| \geq p \text{ and } \left(\forall x,y,z \; \begin{array}{l} w = xyz, \\ |xy| \leq p, \\ |y| \geq 1, \end{array} \implies \overline{\forall i \quad xy^i z \in L.} \right) \right) \implies \begin{array}{c}\text{L is}\\ \text{not regular.}\end{array}$$

Which is equivalent to

$$\left(\forall p \; \exists w \in L \quad |w| \geq p \text{ and } \left(\forall x,y,z \; \begin{array}{l} w = xyz, \\ |xy| \leq p, \\ |y| \geq 1, \end{array} \implies \exists i \quad xy^i z \notin L. \right) \right) \implies \begin{array}{c}\text{L is}\\ \text{not regular.}\end{array}$$

The translation into words is the contrapositive of the Pumping Lemma (stated in Theorem 3.2 below).

3.3.2 The contrapositive of the Pumping Lemma

Theorem 3.2 (Pumping Lemma restated.) *Consider a language L. If for any integer $p \geq 0$ there exists a word $w \in L$, such that $|w| \geq p$, and for any breakup of w into three strings x, y, z, such that:*

- $w = xyz$,
- $|xy| \leq p$,
- $|y| \geq 1$,

implies that there exists an i such that $xy^i z \notin L$, then the language L is not regular.

3.3.3 Proving that a language is not regular

Let us assume that we want to show that a language L is *not* regular.

Such a proof is done by contradiction. To prove L is not regular, we assume it is regular. This gives us a specific (but unknown) pumping length p. We then show that L satisfies the rest of the contrapositive version of the pumping lemma, so it can not be regular.

So the proof outline looks like:

- Suppose L is regular. Let p be its pumping length.
- Consider $w =$ [formula for a specific class of strings]
- By the Pumping Lemma, we know there exist x, y, z such that $w = xyz$, $|xy| \leq p$, and $|y| \geq 1$.
- Consider $i =$ [some specific value, almost always 0 or 2]
- xy^iz is not in L. [explain why it can't be]

Notice that our adversary picks p. We get to pick w whose length depends on p. But then our adversary gets to pick the specific division of w into x, y, and z.

3.4 Examples

3.4.1 The language $L = \mathtt{a}^n\mathtt{b}^n$ is not regular

Claim 3.3 *The language $L = \mathtt{a}^n\mathtt{b}^n$ is not regular.*

Proof: For any $p \geq 0$, consider the word $w = \mathtt{a}^p\mathtt{b}^p$, and consider any breakup of w into three parts, such that $w = xyz$ $|y| \geq 1$, and $|xy| \leq p$. Clearly, xy is a prefix of w made out of only $\mathtt{a}$s. As such, the word $xyyz$ has more $\mathtt{a}$s in it than $\mathtt{b}$s, and as such, it is not in L.

But then, by the Pumping Lemma (Theorem 3.2), L is not regular. ■

3.4.2 The language $\{ww\}$ is not regular

Claim 3.4 *The language $L = \left\{ ww \;\middle|\; w \in \Sigma^* \right\}$ is not regular.*

Proof: For any $p \geq 0$, consider the word $w = \mathtt{0}^p\mathtt{1}\mathtt{0}^p\mathtt{1}$, and consider any breakup of w into three parts, such that $w = xyz$ $|y| \geq 1$, and $|xy| \leq p$. Clearly, xy is a prefix of w made out of only $\mathtt{0}$s. As such, the word $xyyz$ has more $\mathtt{0}$s in its first part than the second part. As such, $xyyz$ is not in L.

But then, by the Pumping Lemma (Theorem 3.2), L is not regular. ■

Consider the word w used in the above claim:

- It is concrete, made of specific characters, no variables left in it.
- These strings are a subset of L, chosen to exemplify what is not regular about L.
- Its length depends on p.

- The 1 in the middle serves as a barrier to separate the two groups of 0's. (Think about why the proof would fail if it was not there.)
- The 1 at the end of w does not matter to the proof, but we nee it so that $w \in L$.

3.5 A note on finite languages

A language L is ***finite*** if has a bounded number of words in it. Clearly, a finite language is regular (since you can always write a finite regular expression that matches all the words in the language).

It is natural to ask why we can not apply the pumping lemma Theorem 3.1 to L? The reason is because we can always choose the threshold p to be larger than the length of the longest word in L. Now, there is no word in L with length larger than p in L. As such, the claim of the Pumping Lemma holds trivially for a finite language, but no word can be pumped - and as such L stays finite. So the pumping lemma makes sense even for finite languages!

4 Irregularity via closure properties

If we know certain seed languages are not regular, then we can use closure properties to show other languages are not regular.

We remind the reader that *homomorphism* is a mapping $h : \Sigma_1 \to \Sigma_2^*$ (namely, every letter of Σ_1 is mapped to a string over Σ_2). We showed that if a language L over Σ_1 is regular, then the language $h(L)$ is regular. We referred to this property as *closure of regular languages under homomorphism.*

Claim 4.1 *The language* $L' = \{\mathtt{0}^n\mathtt{1}^n \mid n \geq 0\}$ *is not regular.*

Proof: Assume for the sake of contradiction that L' is regular. Let h be the homomorphism that maps 0 to a and 1 to b. Then $h(L')$ must be regular (closure under homomorphism). But $h(L')$ is the language

$$L = \left\{ \mathtt{a}^n\mathtt{b}^n \;\middle|\; n \geq 0 \right\}, \tag{1}$$

which is not regular by Claim 1.5. A contradiction. As such, L' is not regular. ■

We remind the reader that regular languages are also closed under intersection.

Claim 4.2 *The language* $L_2 = \left\{ w \in \{\mathtt{a},\mathtt{b}\}^* \;\middle|\; w \text{ has an equal \# of } \mathtt{a}\text{'s and } \mathtt{b}\text{'s} \right\}$ *is not regular.*

Proof: Suppose L_2 were regular. Consider $L_2 \cap \mathtt{a}^*\mathtt{b}^*$. This must be regular because L_2 and $\mathtt{a}^*\mathtt{b}^*$ are both regular and regular languages are closed under intersection. But $L_2 \cap \mathtt{a}^*\mathtt{b}^*$ is just the language L from Eq. (1), which is not regular (by Claim 1.5). ■

Claim 4.3 *The language* $L_3 = \left\{ \mathtt{a}^n\mathtt{b}^n \;\middle|\; n \geq 1 \right\}$ *is not regular.*

Proof: Assume for the sake of contradiction that L_3 is regular. Consider $L_3 \cup \{\epsilon\}$. This must be regular because L_3 and $\{\epsilon\}$ are both regular and regular languages are closed under union. But $L_3 \cup \{\epsilon\}$ is just L from Eq. (1), which is not regular (by Claim 1.5).

A contradiction. As such, L_3 is not regular. ■

4.1 Being careful in using closure arguments

Most closure properties must be applied in the correct direction: We show (or assume) that all inputs to the operation are regular, therefore the output of the operation must be regular.

For example, consider (again) the language $L_B = \{0^n 1^n \mid n \geq 0\}$, which is not regular.

Since L_B is not regular, $\overline{L_B}$ is also not regular. If $\overline{L_B}$ were regular, then L_B would also have to be regular because regular languages are closed under set complement. However, many similar lines of reasoning do not work for other closure properties.

For example, L_B and $\overline{L_B}$ are both non-regular, but their union is regular. Similarly, suppose that L_k is the set of all strings of length $\leq k$. Then $L_B \cap L_k$ is regular, even though L_B is not regular.

If you are not absolutely sure of what you are doing, always use closure properties in the forward direction. That is, establish that L and L' are regular, then conclude that L OP L' must be regular.

Also, be sure to apply only closure properties that we know to be true. In particular, regular languages are **not** closed under the subset and superset relations. Indeed, consider $L_1 = \{001, 00\}$, which is regular. But L_1 is a subset of L_B, which is not regular. Similarly, $L_2 = (0+1)^*$ is regular. And it is a superset of L (from Eq. (1) in the proof of Claim 4.1)). But you can not deduce that L is therefore regular. We know it is not.

So regular languages can be subsets of non-regular ones and vice versa.