Formal Languages, Automata and Computation

Identifying Nonregular languages

Pumping Lemma

Carnegie Mellon University in Qatar

SUMMARY

- DFAs to Regular Expressions
- Minimizing DFA's
- Closure Properties
- Decision Properties

IDENTIFYING NONREGULAR LANGUAGES

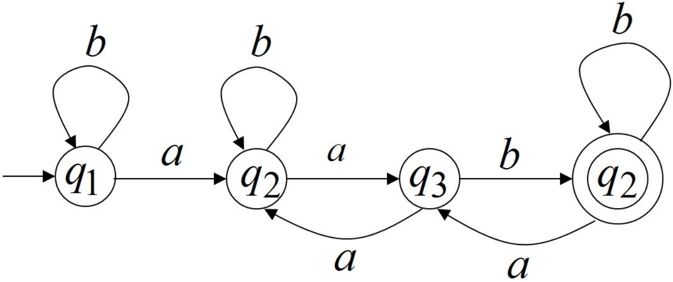
- Given language L how can we check if it is not a regular language ?
  - The answer is not obvious.
  - Not being able to design a DFA does not constitute a proof!

The Pigeonhole Principle

- If there are n pigeons and m holes and $n > m$, then at least one hole has > 1 pigeons.

- What do pigeons have to do with regular languages?

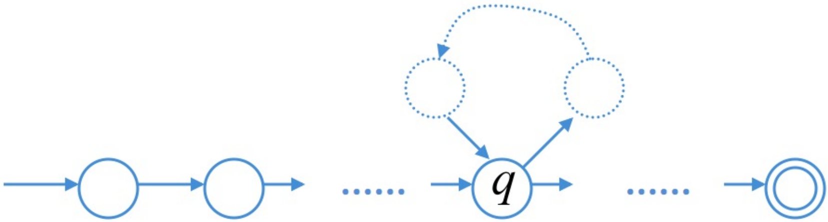
- Consider the DFA



- With strings *a*, *aa* or *aab*, no state is repeated
- With strings *aabb*, *bbaa*, *abbabb* or *abbbabbabb*, a state is repeated
- In fact, for any ω where $|\omega| \geq 4$, some state has to repeat? Why?

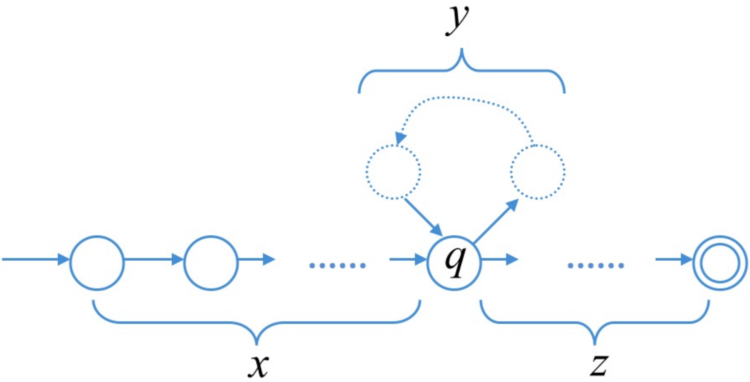
THE PIGEONHOLE PRINCIPLE

- When traversing the DFA with the string ω, if the number of transitions $\geq$ number of states, some state q has to repeat!
- Transitions are pigeons, states are holes.



Pumping a String

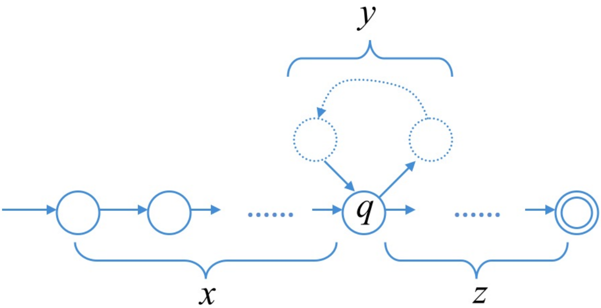
- Consider a string $\omega = xyz$



- $|y| \geq 1$
- $|xy| \leq m$ (m the number of states)

PUMPING A STRING

- Consider a string $\omega = xyz$



- If $\omega = xyz \in L$ that so are xy^iz for all $i \geq 0$
- The substring y can be pumped.
- So if a DFA accepts a sufficiently long string, then it accepts an infinite number of strings!

A NONREGULAR LANGUAGE

- Consider the language $L = \{a^n b^n | n \geq 0\}$
- Suppose L is regular and a DFA with p states accepts L
- Consider $\delta^*(q_0, a^i)$ for $i = 0, 1, 2, \ldots$
- Since there are infinite i's, but a finite number states, the Pigeonhole Principle tells us that there is some state q such that
    - $\delta^*(q_0, a^n) = q$ and $\delta^*(q_0, a^m) = q$, but $n \neq m$
    - Thus if M accepts $a^n b^n$ it must also accept $a^m b^n$, since in state q is does not "remember" if there were n or m a's.
- Thus M can not exist and L is not regular.

The Pumping Lemma

Lemma

Given an infinite regular language L

1. *There exists an integer m such that*
2. *for any string $\omega \in L$ with length $|\omega| \geq m$,*
3. *we can write $\omega = xyz$ with $|y| \geq 1$ and $|xy| \leq m$,*
4. *such that the strings xy^iz for $i = 0, 1, 2 \ldots$ are also in L*

Thus any sufficiently long string can be "pumped."

Proof Idea

We already have some hints.

The Pumping Lemma

Proof.

- If L is regular then M with p states recognizes L. Take a string $s = s_1 s_2 \cdots s_n \in L$ with $n \geq p$.
- Let $r_1 r_2 \cdots r_{n+1}$ be the sequence of $n+1 (\geq p+1)$ states M enters while processing s ($r_{i+1} = \delta(r_i, s_i)$)
- r_j and r_l (for some j and l ($j < l \leq p+1$) should be the same state (Pigeons!)
- Now let $x = s_1 \cdots s_{j-1}$, $y = s_j \cdots s_{l-1}$, and $z = s_l \cdots s_n$.
- x takes M from r_1 to r_j, y takes M from r_j to r_j, and z takes M from r_j to r_{n+1}, which is an accepting state. So M must also accept xy^iz for $i \geq 0$.
- We know $j \neq l$, so $|y| > 0$ and $l \leq p+1$ so $|xy| \leq p$

USING THE PUMPING LEMMA

- If a language violates the pumping lemma, then it can not be regular.
- Two Player Proof Strategy:
  - Opponent picks m
  - Given m, we pick ω in L such that $|\omega| \geq m$. We are free to choose ω as we please, as long as those conditions are satisfied.
  - Opponent picks $\omega = xyz$ - the decomposition subject to $|xy| \leq m$ and $|y| \geq 1$.
  - We try to pick an i such that $xy^iz \notin L$
  - If for all possible decompositions the opponent can pick, we can find an i, then L is not regular.

Using the Pumping Lemma

Consider $L = \{a^n b^n | n \geq 0\}$

1. Opponent picks m
2. We pick $\omega = a^m b^m$. Clearly $|\omega| \geq m$.
3. Since the first m symbols are all a's, the opponent is forced to pick $x = a^j$, $y = a^k$ and $z = a^l b^m$, with $j + k \leq m$ and $l \geq 0$ and $j + k + l = m$

$$\omega = \underbrace{a \cdots a}_{x}\underbrace{a \cdots a}_{y}\underbrace{a \ldots ab \cdots b}_{z}$$

4. We choose $i = 2$ which means $a^j a^k a^k a^l b^m = a^{m+k} b^m \in L$ but it can not be!
5. The opponent does not have any other way of partitioning ω, so L is not regular.

USING THE PUMPING LEMMA

Consider $L = \{\omega | n_a(\omega) < n_b(\omega)\}$

1. Opponent picks m
2. We pick $a^m b^{m+1}$. Clearly $|\omega| \geq m$.
3. Opponent is forced to pick $y = a^k$ for some $1 \leq k \leq m$
4. We pick $i = 2$ which means $a^{m+k} b^{m+1} \in L$ but it can not be!
5. The opponent does not have any other way of partitioning ω, so L is not regular.

USING THE PUMPING LEMMA

Consider $L = \{1^{n^2} | n \geq 0\}$

1. Opponent picks m
2. We pick $\omega = 1^{m^2}$. Clearly $|\omega| \geq m$.
3. Opponent chooses any partitioning of $\omega = xyz = 1^j 1^k 1^l$ with $1 \leq k \leq m$ and $j + k \leq m$
4. With $|xyz| = m^2$ and $i = 2$, $m^2 < |xyyz| \leq m^2 + m$.
   But $m^2 < m^2 + m < m^2 + 2m + 1 = (m+1)^2$
5. $|xyyz|$ lies between to perfect squares. So $xyyz \notin L$.
6. L can not be regular.

SUMMARY

- Symbols, Strings, Languages, Set of all Languages
- DFAs, Regular Languages, NFAs, Regular Expressions
- DFA $\Leftrightarrow$ REs
- Minimal DFAs
- Closure properties, Decision properties
- Nonregular Languages, Pumping Lemma

LET'S SEE IF WE CAN TIE THINGS TOGETHER

True or False?

1. If L_1 is not regular and L_2 is regular then $L = L_1 L_2 = \{xy : x \in L_1 \text{and } y \in L_2\}$ is not regular.
2. $L = \{a^i b^j a^k : i + k < 10 \text{ and } j > 10\}$ is not regular.
3. $L = \{w \in \{a, b\}^* : n_a(w) \times n_b(w) = 0 \bmod 2\}$ is regular.
4. $L = \{a^i b^j : i + j \geq 10\}$ is not regular.
5. $L = \{a^i b^j : i - j > 10\}$ is not regular.
6. $L = \{a^i a^j : i/j = 5\}$ is not regular.
7. If $L_1 \cap L_2$ is regular then L_1 and L_2 are regular.
8. If $L_1 \subseteq L_2$ and L_2 is regular, then L_1 must be regular.

Let's see if we can tie things together

True or False?

1. There are subsets of a regular language which are not regular.
2. If L_1 and L_2 are nonregular, then $L_1 \cup L_2$ must be nonregular.
3. If F is a finite language and L is some language, and $L - F$ is a regular language, then L must be a regular language.
4. $L = \{w \in \{a, b\}$: the number a's times the number of b's in w is greater than 1333$\}$ is not regular.
5. If the start state of a DFA has a self-loop, then the language accepted by that DFA is infinite.
6. The set of strings of 0's, 1's, and 2's with at least 100 of each of the three symbols is a regular language.
7. The union of a countable number of regular languages is regular.

Let's see if we can tie things together

True or False?

1. $L = \{uww^R v | u, v, w \in \{a, b\}^+\}$ is not regular.
2. If L is nonregular then $\overline{L}$ is nonregular.
3. If $L_1 \cap L_2$ is finite then L_1 and L_2 are regular.
4. The family of regular languages is closed under *nor* operation, $nor(L_1, L_2) = \{w \ : \ w \notin L_1 \ and \ w \notin L_2\}$
5. If L is a regular language, then so is $\{xy\colon x \in L$ and $y \notin L\}$
6. Let L be a regular language over $\Sigma = \{a, b, c\}$. Let us define $SINGLE(L) = \{w \in L : \text{all symbols in } w \text{ are the same}\}$. $SINGLE(L)$ is regular.

LET'S SEE IF WE CAN TIE THINGS TOGETHER

Let $\Sigma = \{a\}$ and let M be a *deterministic finite state acceptor* that accepts a regular language $L \subseteq \Sigma^*$.

A) Describe with very simple diagrams, possible structures of the state graph of M, if M has only a single final state. Show any relevant parameters that you feel are necessary.

B) Describe with a regular expression the language accepted by M, if M has a single final state. If necessary, use any parameters you showed in part *a)*.

C) Describe *mathematically* the language accepted by M, if *M has more than one final state*.

WHERE DO WE GO FROM HERE?