Session 13

Pumping Lemma

How can we tell whether a language is regular?

- Kleen's theorem: if we have a RE describing L or a FA accepting L, then L is regular
- But, what if we have a language described by some other means:
  - $L = \{0^n 1^n \in \Sigma^* \mid n > 0\}$
  - Is this language regular?
- Strategy: Find a property that
  - All regular languages must have
  - It is easy to verify
  - If a language does not have that property, then it is not regular!

We already know some properties…

- Myhill and Nerode Theorem: L is a regular language if and only if the set of equivalence classes is finite
- How easy is to verify this property for a given language?
  - The language of strings ending in 10?
  - The language of all palindromes?
  - An arbitrary language?

Another property

- If $L \subseteq \Sigma^*$ and for some n, there are n strings in Σ^*, any two of which are distinguishable with respect to L.
- Then, every FA recognizing L must have at least n states.

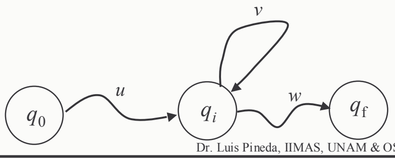
Another property: The loop!

- Let $M = (Q, \Sigma, q_0, A, \delta)$ be a FA with n states and x be a string of length at least n
  - $x = a_1 a_2 \ldots a_n y$
- Then, the sequence of $n + 1$ states:
  - $q_0 = \delta^*(q_0, \Lambda)$
  - $q_1 = \delta^*(q_0, a_1)$
  - $q_2 = \delta^*(q_0, a_1 a_2)$
  - ….
  - $q_n = \delta^*(q_0, a_1 a_2 \ldots a_n)$

  Contains a loop!

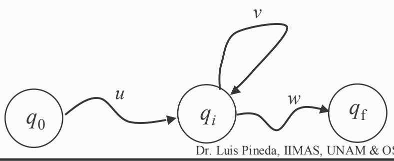
The form of a string with a loop!

- If $x \in L$ is large enough, the form of x is uvw



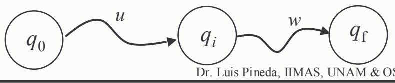
A loop is a loop!

- The loop can be repeated many times: $uv^m w \in L$
- The loop corresponds to the closure of $v^m \in L$
- The loop can occur 0 or more times!



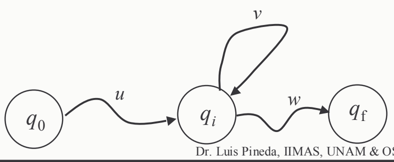
Pumping up the loop!

- The loop occurs 0 times: $uv^0 w \in L$



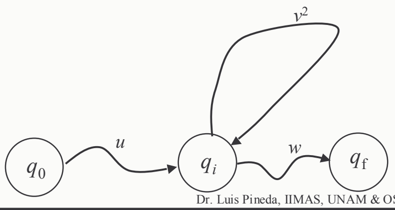
A loop is a loop!

- The loop occurs 1 time: $uv^1 w = uvw \in L$



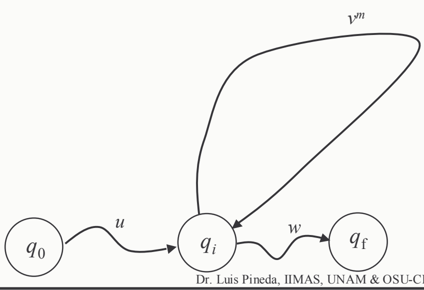
Pumping up the loop!

- The loop occurs 2 times: $uv^2 w \in L$



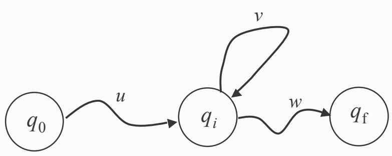
Pumping up the loop!

- The loop occurs m times: $uv^m w \in L$



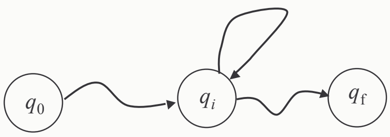
The loop and δ^*

- $\delta^*(q_i, v) = q_i$
- $\delta^*(q_i, v^m) = q_i$ for every $m \geq 0$ and
- $\delta^*(q_0, uv^m w) = q_f$ for every $m \geq 0$



The conditions for the loop

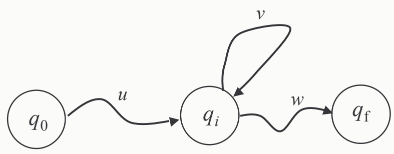
- Suppose $q_i = q_{i+p,}$ where $0 \le i \le p \le n$ where n is the number of states:
  - $\delta^*(q_0,\ a_1a_2\ldots a_i) = q_i$
  - $\delta^*(q_i,\ a_{i+1}a_{i+2}\ldots a_{i+p}) = q_i$
  - $\delta^*(q_i,\ a_{i+p+1}a_{i+p+2}\ldots a_ny) = q_f \in A$



  - Under what conditions is there a loop necessarily?

Simplifying notation!

- $u = a_1a_2\ldots a_i$
- $v = a_{i+1}a_{i+2}\ldots a_{i+p}$
- $w = a_{i+p+1}a_{i+p+2}\ldots a_ny$



Making sure there is a loop!

- We only know that FA has n states so:

here!

$$a_1a_2\ldots a_ia_{i+1}a_{i+2}\ldots a_{i+p}a_{i+p+1}a_{i+p+2}\ldots a_ny$$

- After reading n symbols, we need at least $n + 1$ states (we start in q_0)
- To make sure there is a loop: $n \ge i + p$

Making sure there is a loop!

- We only know that FA has n states so:

Minimum value for n

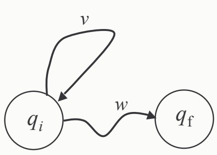
$$a_1a_2\ldots a_ia_{i+1}a_{i+2}\ldots a_{i+p}a_{i+p+1}a_{i+p+2}\ldots a_ny$$

- To make sure there is a loop $n \ge i + p$
- p must be at least 1 (otherwise there is no loop!)

Making sure there is a loop!

- u can be empty (but we cannot know):

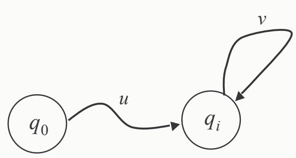
$$\Lambda a_{i+1}a_{i+2}\ldots a_{i+p}a_{i+p+1}a_{i+p+2}\ldots a_ny$$



Making sure there is a loop!

- w can be empty (but we cannot know):

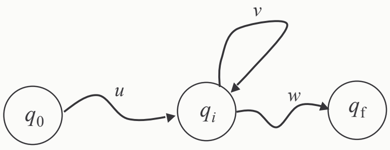
$$a_1a_2\ldots a_ia_{i+1}a_{i+2}\ldots a_{i+p}\Lambda$$



Making sure there is a loop!

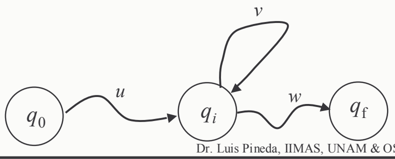
- v has to be at least 1 (although we cannot know):

$a_1 a_2 \ldots a_i a_{i+1} a_{i+p+1} a_{i+p+2} \ldots a_n y$



The Pumping Lemma

- Suppose L is a RL recognized by a FA with n states. For any $x \in L$ with $|x| \geq n$, x may be written as $x = uvw$ for some strings u, v and w satisfying:
  - $|uv| \leq n$: There is an initial segment including the loop
  - $|v| > 0$: The length of the loop is at least one
  - For any $m \geq 0$, $uv^m w \in L$: The loop can be pumped up!

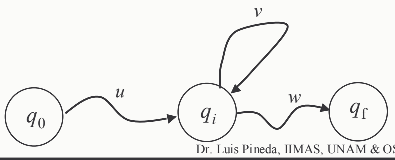


What if *L* is not regular?

- If we don't know whether L is regular:
  - We don't know whether there is a FA that accepts L
  - We don't now how many states such FA would have (in case L turned out to be regular!)

The Pumping Lemma

- Suppose L is a RL. Then there is an integer n so that for any $x \in L$ with $|x| \geq n$, there are strings u, v and w so that:
  - $x = uvw$
  - $|uv| \leq n$: There is an initial segment including the loop
  - $|v| > 0$: The length of the loop is at least one
  - For any $m \geq 0$, $uv^m w \in L$: The loop can be pumped up!



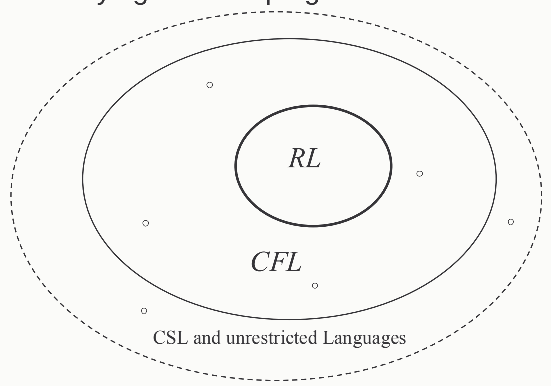
Belonging to a class!

- Necessary and sufficient conditions
  - Necessary: every member of the class has the property
  - Sufficient: having the property is enough for belonging to the class
- What kind of a property is for a language to satisfy the conditions of the pumping lemma?

Use of pumping lemma

- Myhill and Nerode Theorem: Necessary and sufficient condition for a language to be regular
- Pumping Lemma: only a necessary condition: there are CFL that satisfy the conditions of the pumping lemma for RL!
- The pumping lemma can be used to show that a language is not regular (by contradiction), but it cannot be used to show that a language is regular!

Satisfying the Pumping Lemma for *RL*



Using the pumping lemma

- If we have a language described by some other means:
  - $L = \{0^n 1^n \in \Sigma^* \mid n > 0\}$
  - Is this language regular?
- Strategy:
  - Assume that the pumping lemma holds
  - If a contradiction follows from this assumption the language is not regular

Using the pumping lemma

- The pumping game:
  - Player 1 provides a language L
  - Player 2 provides n
  - Player 1 provides a string x in L such that $|x| \geq n$
  - Player 2 divides x into uvw such that $|uv| \leq n$ and $|v| > 0$
  - Player 1 wins by finding a $m \geq 0$ such that $uv^m w \notin L$

$L = \{0^n 1^n \in \Sigma^* \mid n > 0\}$ is not regular

- Prove that $L = \{0^n 1^n \in \Sigma^* \mid n > 0\}$ is not regular
- Player one provides a language L
- Player two provides n
  - The parameter n is ok.
- Player one provides a string x in L such that $|x| \geq n$
  - The string $x = 0^n 1^n$ is ok! $|x| = 2n \geq n$
- Player two partitions x into uvw such that $|uv| \leq n$ and $|v| > 0$:
  - $u = 0^i$ and $v = 0^j$, such that $i \geq 0,\ j > 0,\ i + j = n$ so $|uv| = |0^i 0^j| = |0^n| \leq n$ and $|v| = j > 0$
- Player one wins by finding a $m \geq 0$ such that $uv^m w \notin L$
  - $m = 0$: $uv^m w = 0^i 1^n \notin L$ as $i < n$

L_{pr} = The prime numbers

- Player one provides a language L
  - $L_{pr} = \{x \in \Sigma^* \mid$ length of x is a prime number$\}$
- Player two provides n
  - The parameter n such that $p \geq n + 2$ for some p (i.e. there will always be a prime number larger than n)
- Player one provides a string x in L such that $|x| \geq n$
  - The string $x = 1^p$ is ok! $|x| = p \geq n + 2 \geq n$
- Player two partitions x into uvw such that $|uv| \leq n$ and $|v| > 0$:
  - Let $|v| = k$ for $k > 0$
  - Then $|uw| = p - k$,
  - we assume that $|uv| \leq n$

L_{pr} = The prime numbers

- Player one wins by finding a $m \geq 0$ such that $uv^m w \notin L$
  - Let $m = p - k$
  - If L_{pr} is regular then $uv^{p-k}w$ must be in L_{pr}
    - $|v| = k$ for $k > 0$
    - $|uw| = p - k$
  - $|uv^{p-k}w| = |uw| + (p - k)|v|$
    $= p - k + (p - k)k$
    $= (p - k) + (p - k)k$
    $= (p - k)(1 + k)$
  - $|uv^{p-k}w|$ has two factors, and cannot be a prime unless one of the factors is 1

L_{pr} = The prime numbers

- Player one wins by finding a $m \geq 0$ such that $uv^m w \notin L$
  - Let's make sure that both factors are different from 1:
    - $(1 + k) > 1$ since $k > 0$
    - $k \leq n$ since $k = |v| \leq |uv| \leq n$ (original assumption)
    - $(p - k) > 1$ since $p \geq n + 2$ and $k \leq n$ and $p \geq 2$
- Consequently,
  - $|uv^{p-k}w| = (p - k)(1 + k)$ where
    - $(p - k) > 1$
    - $(1 + k) > 1$
  - $uv^{p-k}w \notin L$