MATH 2301

* Non-regular languages

Q: How do we know that non-regular languages exist?

A1: Counting!

Say Σ = alphabet (finite)

Σ^* = all possible strings.

Σ^* has ∞ly many, but countably many elements.

This means that there is a procedure to sequentially number (1, 2, 3, ...) the elements of Σ^*, namely, the lexicographic order.

[There is a bijection $\Sigma^* \xleftrightarrow{1:1} \mathbb{N}$]

A language L is just any subset of Σ^*.

How many languages? Infinitely many, as many as there are subsets of Σ^*, or subsets of $\mathbb{N}$.

Theorem: There are uncountably many subsets of $\mathbb{N}$.

$\Rightarrow$ there is no way to number the subsets of $\mathbb{N}$ by 1, 2, 3, ... [you'll always miss something if you try.]

(Proof goes via an argument called Cantor's diagonalisation argument.)

$\Rightarrow$ There are uncountably many languages!

Q: How many are regular?

Observation: At most as many as there are regular expressions.

A regex is just a special string in the alphabet Σ, together with extra symbols: $*$, $|$, (,)

$$\underbrace{\text{Number of regexes}}_{\text{Countable.}} \le \underbrace{\text{Number of strings in } \Sigma \cup \{*, |, (,)\}}_{\text{Countable! Lexicographically orderable.}}$$

But $$\underbrace{\text{Number of regular languages}}_{\text{Countable.}} \le \text{Number of regexes.}$$

Since there are uncountably many languages, at least one (in fact, uncountably many) are non-regular.

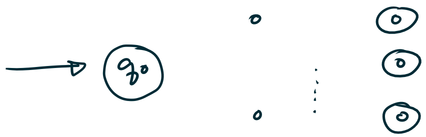
A2: The pumping lemma

We exploit a non-obvious feature that every regular language shares

If and only if

Recall: A language is regular iff there is a DFA that recognises it.

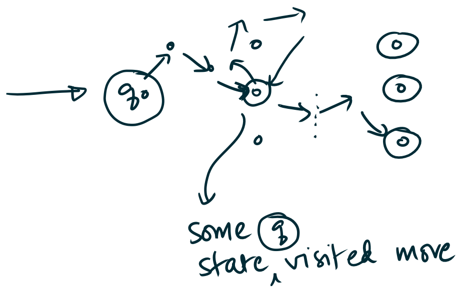
Suppose M is a DFA. Schematic below.



Suppose it has n states.

If w is any word of length k, then the computation of w through M passes through (k+1) states, and gives a path of length k through the DFA.

(Recall from graphs): If $|w| > n$, then it must repeat a state.



If w is long enough, there is a "loop" within the calculation path

$\Rightarrow$ $w = xyz$: x = portion before q
y = loop portion ← nonzero length.
z = portion after q

Suppose w has length $\geq n$ and w is accepted by M.
$w = xyz$ as before.

$\Rightarrow$ xz is also accepted by M.
& $xyyz$, $xyyyyyz$, etc are all accepted by M.

$\Rightarrow$ any pattern of the form $x y^* z$ is accepted by M.

$\Rightarrow$ If M accepts a string w of length $> n$ then it accepts all the strings fitting into the pattern $x y^* z$: there is a non-empty "y" portion that can be pumped.

Use this to detect non-regular languages: If there are long enough strings that can't be pumped, then the language is not regular.

** **Theorem** (Pumping lemma): Let L be a regular language. Then there exists some $n_L \in \mathbb{N}$, such that: if $w \in L$ and $|w| \geq n_L$, then:

n_L ← # states in a DFA recognising L

$w = xyz$ with

1) $|y| \geq 1$
2) $|xy| < n_L$
3) $xy^kz \in L$ for every $k \in \mathbb{N}$ (including $k=0$)!

Example: $\{0^k 1^k \mid k \in \mathbb{N}\} = L$

If L were regular, there would be a DFA M, such that $L = L(M)$.

Let n = # states in this hypothetical DFA.

Consider the string $0^{n+1} 1^{n+1} = w$.

0^{n+1} ← already longer than n.

Running w through M, we will encounter a repeated state even before we get to the 1s

$\Rightarrow w = xyz$

y ← involves only zeroes.

$\Rightarrow x = 0^a$

$y = 0^b \quad (b>0)$

$z = 0^c 1^{n+1}$

} previous argument $\Rightarrow xyyz$, $xyyyz$ etc are all in L.

$(a+b+c = n+1)$

$xyyz = 0^a 0^b 0^b 0^c 1^{n+1}$

$= 0^{n+1+b} 1^{n+1}$

→ DFA calculation $\Rightarrow$ in L

→ doesn't satisfy def!

Contradiction: