Languages that are not regular

9/24

A language that is not regular (and the proof):

$L = \{ \underbrace{a^n b^n}_{} \mid n \in \mathbb{Z} \} = \{ \lambda, ab, aabb, aaabbb, \ldots \}$

a's followed by b's

\# of a's = \# of b's

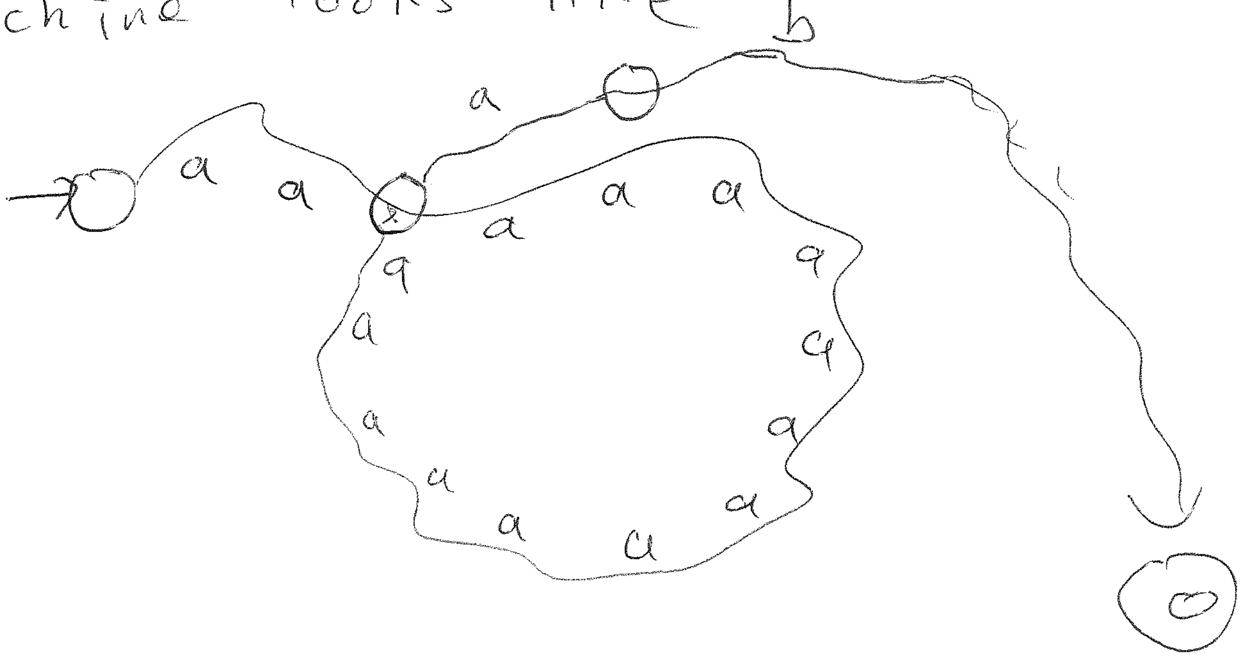
Why can't we create a DFA that accepts this language?

Suppose we could: this DFA would have a finite number of states, say k states (imagine k = 2,736,493)

Consider passing $w = a^{k+1} b^{k+1}$ through this DFA.

When I read this string w with my DFA, I must pass through some state twice (or more) ~~b/c~~ before getting to the end of the a's, because there are more a's than states.

My path to accepting w in the machine looks like



~~Let's say the loop has m states~~

Since we pass through the same state twice, there is a loop. Let's say this loop has m states (e.g. m=2,249,368)

Then the DFA will also accept the string that goes through the loop one extra time.

In other words, the DFA will accept ~~$a^{n+m} b^n$, which~~ $a^{k+l+m} b^{k+l}$, which is not supposed to be in L.

So there can't be a DFA that accepts precisely L.

Another language that is not regular:

L = { strings of "(" and ")" that match correctly - i.e. # of "(" = # of ")", and when reading left to right, # of "(" ≥ # of ")" at any point }

<u>Proof that L is not regular:</u>

Suppose it is regular. We know that

K = { strings matching R.E. $(^*)^*$ } is regular.

$L \cap K = \{ (^n)^n \mid n \in \mathbb{Z} \}$.

If L were regular, then $L \cap K$ would be regular (since intersection of reg. lang. is regular). We know $L \cap K$ is not regular, so L can't be.

We just proved this

Most proofs that languages are not regular will be "by reduction" - showing that if it is regular, we could apply a construction that maintains the regular property to get something known to be not regular.

Another direct proof:

Σ fixed
(e.g. $\Sigma = \{a,b,c\}$)

$$L = \{ w \underbrace{w^R}_{\text{reverse of } w} \mid w \in \Sigma^* \}$$

So L = language of even-length palindromes

Assume L can be recognized by some DFA M. M has some number of states which we call k. ~~Hence~~ M accepts some string of the form ww^R when $|w| > k$.

In reading w, M will go through a loop, ~~of size~~ ~~Sup~~ since $|w| >$ # states.

Suppose this loop has size ~~m~~ n.

Now do this loop twice ....

uhh - we have to be careful since the result could be accidentally a palindrome (e.g. if w were all a's)

# of states in loop
"
transitions in loop
"
length of string processed by loop

Suppose

$$w = ab\underbrace{bb}_{\text{loop part}}bbba$$

doing the loop twice results in

$$\underbrace{ab\underbrace{bb}\underbrace{bb}}_{\text{palindrome w/ this as first half}}bbba$$

Note: we get to choose w, but have no control over where the loop is.

---

ba

$$w = aabb\underbrace{ba}|abbbaa$$

does this work?

Maybe something simpler:

~~w = aa.... ab|b~~

w = a.... ab|ba.... a

would work?