- Design a DFA or Turing machine that decides a given language L.
  $\forall x \in L$, machine accepts x.
  $\forall x \notin L$, machine rejects x.
- Prove a language is not regular
- Find equivalence classes of a given language.
- Closure properties of Regular languages and Turing-decidable, Turing-acceptable languages.
- cross-product construction
  given DFA M_A deciding A, construct $M_{\bar{A}}$ deciding $\bar{A}$
- using standard Turing machines to simulate other machines.

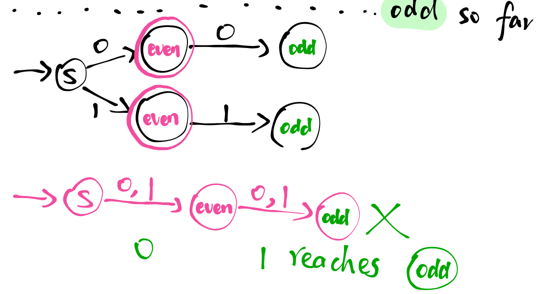
mimic the design of #(1,x) is even

① Analysis : one occurrence of 00 or 11 is counted as 1.

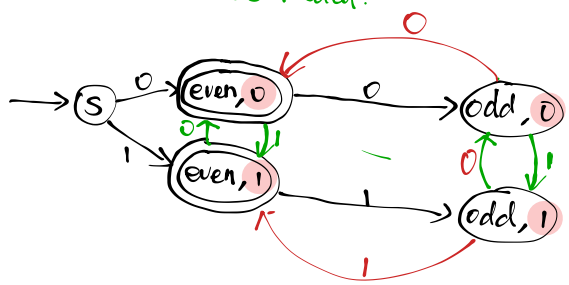
② start with what we want to accept and assign meanings to states.

the # of immediate bit repetition is even so far

the . . . . odd so far



③ Finish the missing transitions, and maybe further refine the meanings of states, and add more states as needed.



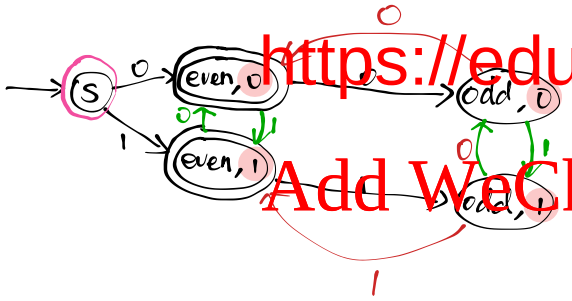
since one occurrence of 00 or 11 is counted 1 for #IBR(X), the last digit read so far affect which state to it go to while reading the next 0 (or 1).

④ Mark the final states.

Always check the edge case λ:

if $\lambda \in L$, start state $\in F$

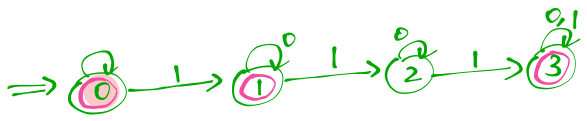
if $\lambda \notin L$, start state $\notin F$



Independent requirements.

$A \cap \bar{B}$

$M_{\bar{B}} \equiv$



$S = (S_A, S_{\bar{B}})$, $F = \{(q_A, q_{\bar{B}}) \mid q_A \in F_A \text{ and } q_{\bar{B}} \in F_{\bar{B}}\}$

$\delta((q_A, q_{\bar{B}}), a) = (\delta_A(q_A, a), \delta_{\bar{B}}(q_{\bar{B}}, a))$

*This is another problem from spring19 exam2 thus the page numbering is off for this problem

KEY

Proof: For each $n \in Z^+$ (positive integer), the first extension of the string 110^n10 is 0^{n-1}, since $A^{(1)} \supseteq \{0^{n-1} \mid n \in Z^+\}$, $|A^{(1)}| = \infty$. By ordinal extensions theorem, A is not regular.

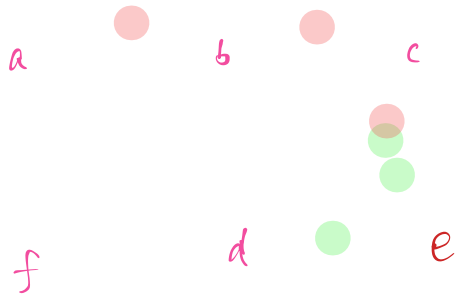
① You don't need to get all strings in $A^{(1)}$. Showing it contains an infinite subset suffices.

② How to get $A^{(1)}$?

In principle, check each string $x \in \Sigma^*$, and analyze how to make it in A by appending 0 to it. (make $xy \in A$ and y is the shortest such string).

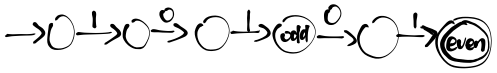
If you just want to get a subset of $A^{(1)}$, then analyze strings with a pattern. eg. 110^n10 in terms of n, or some strings x such that y is in terms of n

③ If $A^{(1)}$ does not work, try $A^{(2)}$.

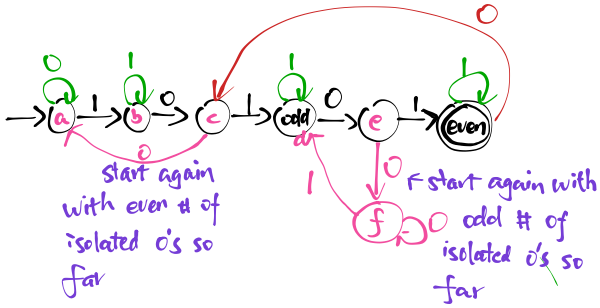


① Analysis:
Count the # of substrings 101 allowing overlapping 1's.

② start with what we want to accept, try assign meanings to states



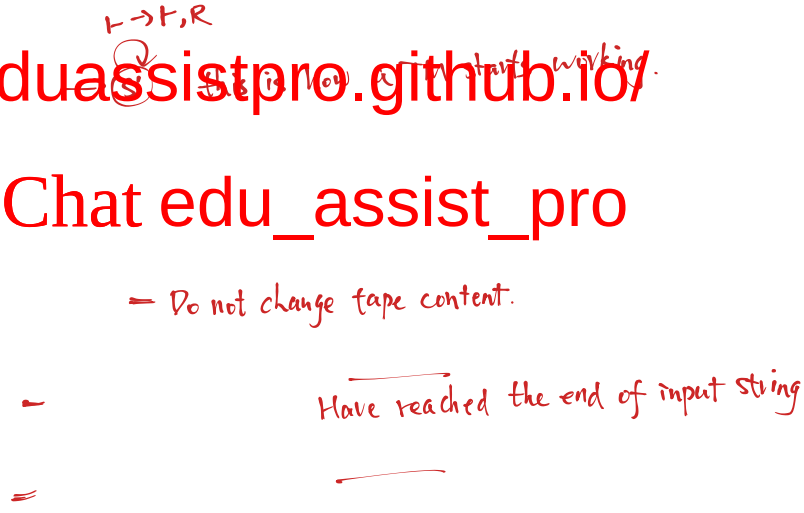
③ Complete the missing transitions, add more states as needed, refine meaning of states if needed.



④ Assign final states.
check λ always.
F = {a, b, c, even}

We can merge state b and even to minimize the DFA

Even though a TM can simulate a DFA, if you are asked to give a Turing Machine, then you need to draw the state diagram following the semantics/rules of a TM.



$\#(1,X) \bmod 3 = r \bmod 3$

$\#(1,X) \bmod 3 = \#(1,y) \bmod 3$

strings in different equivalence class are not A-equivalent

extend by one more symbol

$x \equiv_A y \iff \forall z \in \Sigma^* \ (xz \in A \iff yz \in A)$

always consider Σ^*, $\emptyset$ for disprove/counterexamples.

when A is $\emptyset$, $A \cap B$ being regular does not give any information about B.

A common (wrong) proof:

Proof by contradiction:

Assume A is Regular and $A \cap B$ is regular, and B is non-regular.

Using the closure property $A \cap B$ should be non-regular, which contradicts the assumption that $A \cap B$ is regular.

This proof used the closure property in a wrong way. Closure properties of regular languages can only be applied on regular languages, but B is non-regular.

$A \in Reg$ and $B \in Reg \Rightarrow A \cap B$ is Reg.

$A \in Reg$ and $B \in Non\text{-}Reg \Rightarrow$ we don't know about $A \cap B$.