Lecture 5

Last Lecture Recap (Pumping Lemma)

L is regular. long string $w \in L$

xz xyz $xyyz$ $xyyyz$ $xyyyyz$ ... $\in L$.

$\Uparrow$

L.

0^i 1^j where i > j (Pump Down)

$L := \{ w \in 0^i 1^j \mid i > j \}$

$\exists p > 0$ such that $|w| > p$.

$s = 0^{p+1} 1^p \in L$

Assume L is regular. Then by P.L.

$s = \underbrace{\boxed{0 \cdots\cdots}}_{p+1} 0 \, \underbrace{1 \cdots\cdots 1}_{p}$

xy z

$xz \in L$.

$xyyz$

$xyyyz$

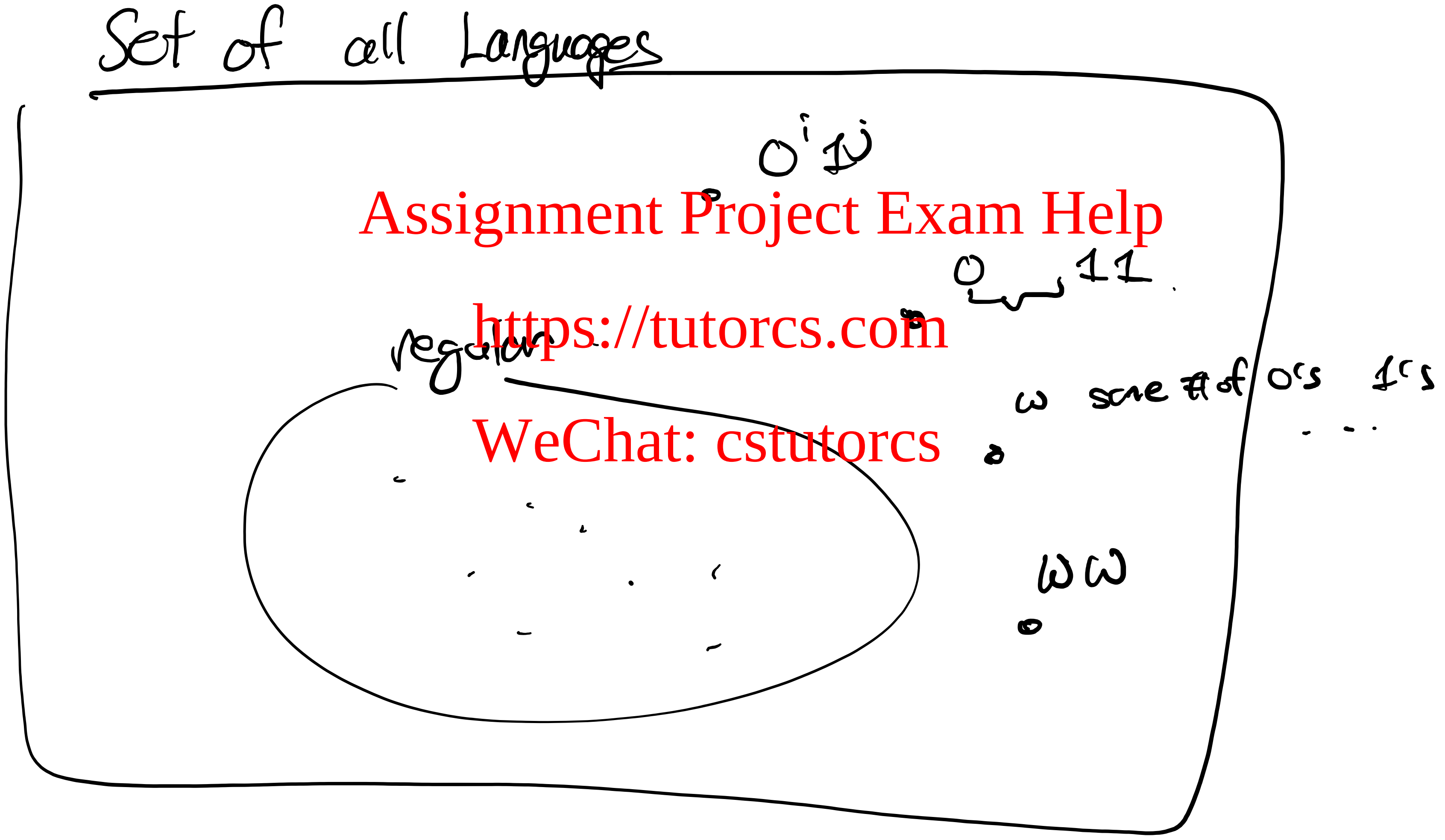
$\vdots$

contradiction

b/c.

$xz \notin L$.

Picture Now



Regular Language Recap

- DFA , NFA (equivalent)
- Languages accepted by these machines (DFA) is called regular.
  Can show that
- L is not regular via Pumping Lemma
  ↳ there are lots of them.

Why Regular Language is not so powerful ?

Motivation : Mathematically define Computation

Result : $\boxed{\text{Failure}}$ (DFA / NFA)

Cannot even do simple tasks.

Turing Machine

Main Restriction of DFA/NFA

Finite # of states

⇒ only remember few things (finitely many things) about your input.

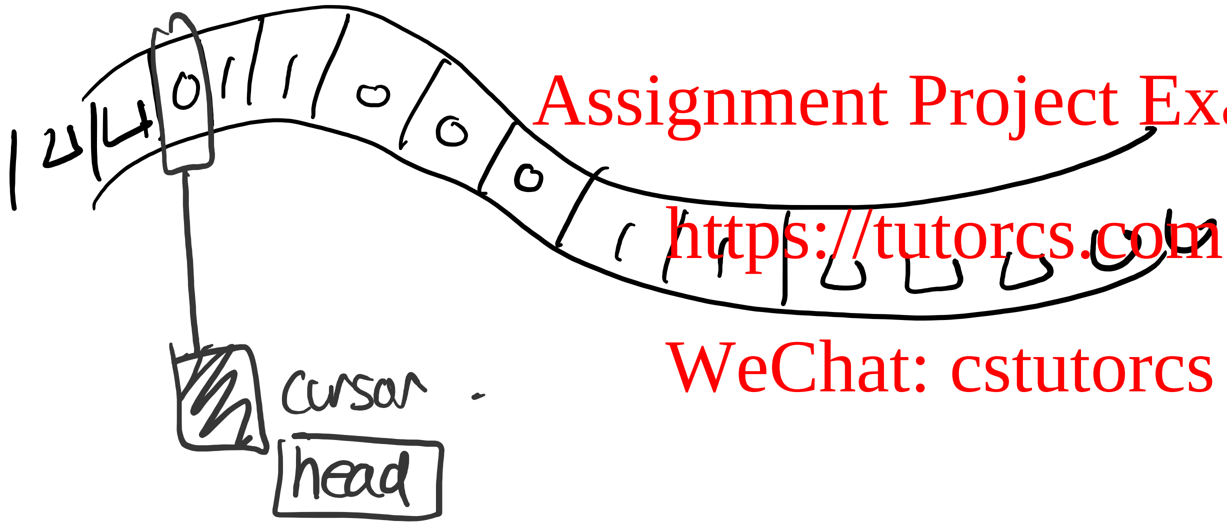
→ could have been avoided if the machine could "write" in some sense.

But only moves right?

Historical Context : Define Computation

Alan Turing 1920s. ! Turing Machine → Define Computation.

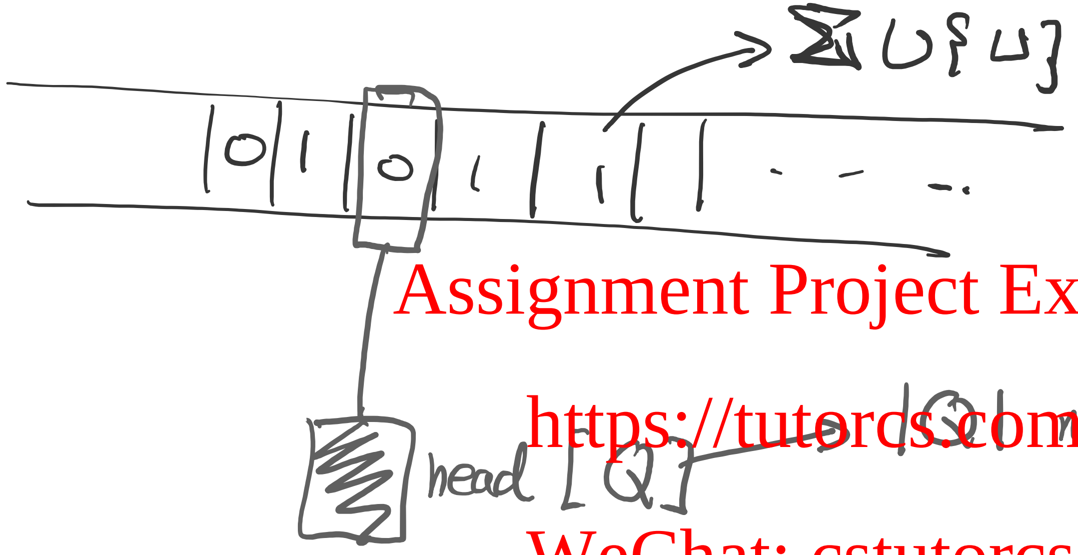
A tape (with input)



Head

- access to reading the content of the tape
- **write** contents to the tape.
- move left / right / stay still.

Transition Function (δ).



$$\delta : \underset{\substack{\text{input}\\\text{alphabet}}}{\Sigma} \times \underset{\substack{\text{current}\\\text{state}}}{Q} \longrightarrow \underset{\substack{\text{next}\\\text{state}}}{Q} \times \underbrace{\{L, R, S\}}_{\substack{\text{movement}\\\text{of the}\\\text{head}}} \times \underbrace{\Sigma}_{\substack{\text{output}\\\text{alphabet}}}$$

Accept / Reject

- $q_{accept} \in Q$.

If the machine transitions into q_{accept} then the machine halts & accepts.

- $q_{reject} \in Q$.

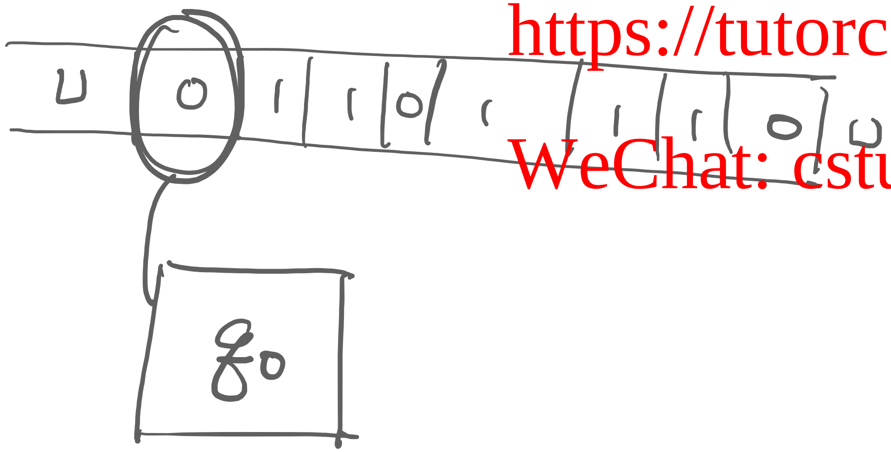
〃 q_{reject}, 〃

rejects.

Final Definition of T.M.

$(\Sigma, Q, q_{accept}, q_{reject}, \underline{\underline{q_0}}, \delta)$ → transition function.

↳ initial state



Example T.M.

$L(M) :=$ the set of inputs accepted by T.M. M.

$L = \{ w\#w \mid w \in \{0,1\}^* \}$

EXAMPLE 3.9

The following is a formal description of $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$, the Turing machine that we informally described (page 167) for deciding the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- $Q = \{q_1, \ldots, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0,1,\#\}$, and $\Gamma = \{0,1,\#,x,\sqcup\}$.
- We describe δ with a state diagram (see the following figure).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}, respectively.

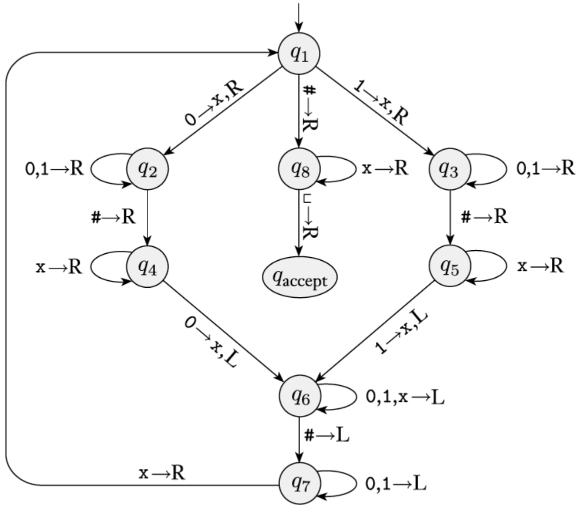


FIGURE 3.10
State diagram for Turing machine M_1

In class exercise (0^n 1^n)

① Either you can count

② Modify the previous T.M.

Language of T.M. M

the set of inputs accepted by the T.M. M.

$L(M)$.

Turing Decidable

L is Turing Decidable if $\exists$ T.M. M.

which accepts w if $w \in L$. $L(M) = L$.

rejects w if $w \notin L$.

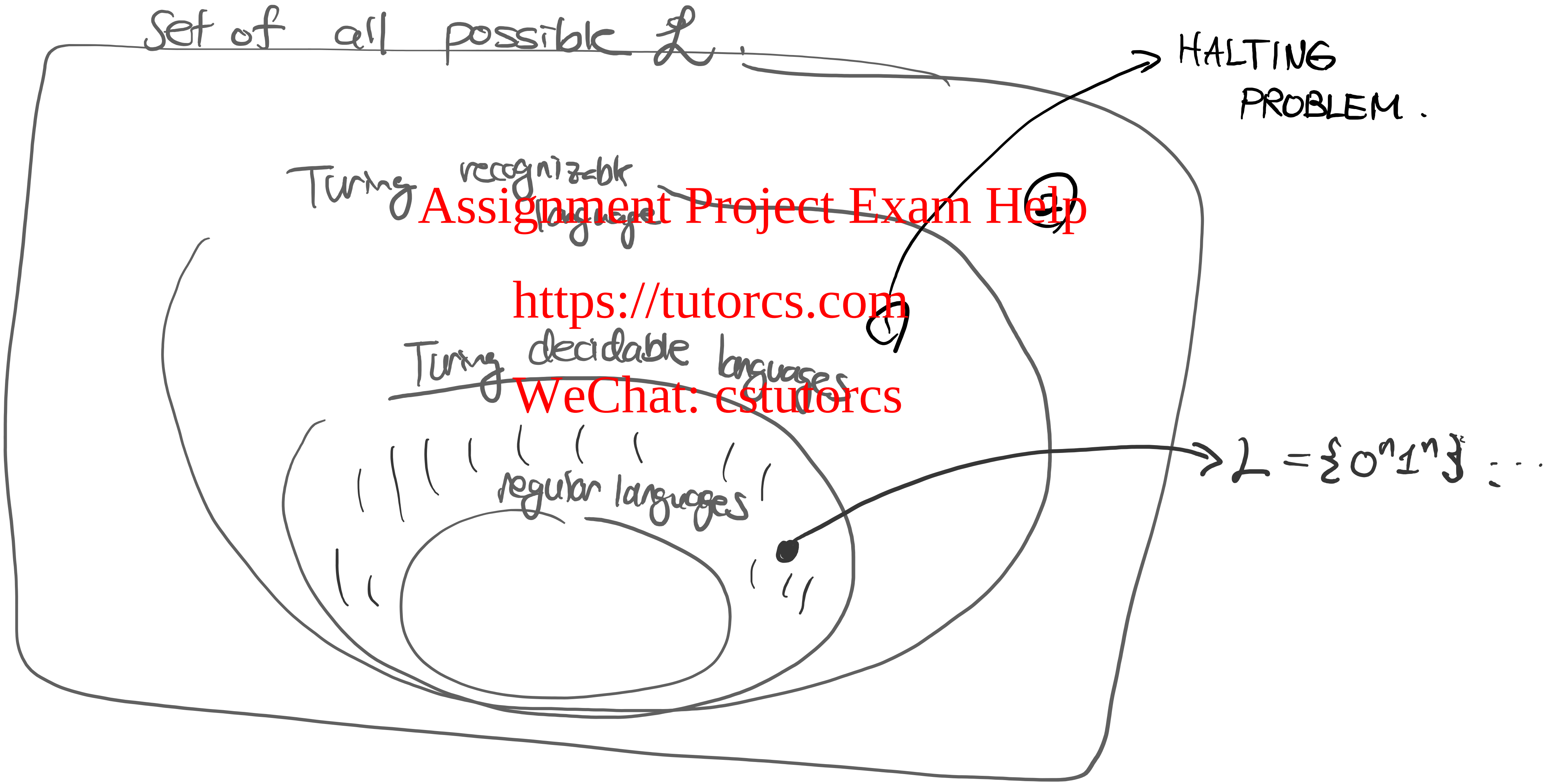
Turing Recognizable

L is Turing Recognizable if

$w \in L$ M accepts w.

$w \notin L$. reject / infinite loop.

What’s the difference ?



Essentially All computation tasks

- C.T. Thesis : Every computational models are equivalent to Turing Machine!!