# 11. Turing Machines

**DFAs**, **NFAs**, and **regular expressions** appear in many instances.

- pattern matching (text editors, IDEs)
- finite control (watches, circuits, microwaves)
- time-critical applications (agent-based computing)

So do **CFGs** and **PDAs**.

- compiles for all programming languages
- document markup (XML, LaTeX, HTML are all CFGs?)
- natural language processing and modeling

## 11.1 Basic Notions

Input is stored on an indefinite-length tape; following the input are blanks.

We use an automaton with two stacks to represent Turing Machines.

- read, write, head

### Turing Machines vs. Automata

- TMs can scan the tape forward and backward
- TM's tape is infinite (left end exists, right side doesn't)
- TMs can write on the tape

DEF: A Turing Machine is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.

- $Q$ → set of states
- $\Sigma$ → input alphabet $(\underbrace{\_\_\_}_{space} \notin \epsilon)$
- $\Gamma$ → tape alphabet $(\Sigma \geq \Gamma)$
- $q_0$ → start state

- $q_{accept}$ → accept state
- $q_{reject}$ → reject state

$q \in Q$,
$q_{accept} \neq$
$q_{reject}$

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

current state x symbol in current cell → new state x symbol to write in current cell x head movement

- can't move left from leftmost cell (will stay put)
- on entering $q_{accept}$ or $q_{reject}$, TM halts (terminates))
- possibilities on an input $w$:
  - halts (reach $q_{accept}$ or $q_{reject}$ eventually)
  - runs forever (never reach accept or reject state, implicitly reject)

**DEF**: The language recognized by TM $M$ is $L(M) = \{w : M$ halts in state $q_{accept}$ when started on $w\}$.

**DEF**: A language is Turing-recognizable if some TM recognizes it.

## 11.2 Equality Problem

### Example

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

We know this language is not context-free?

$0110011\#0110011$

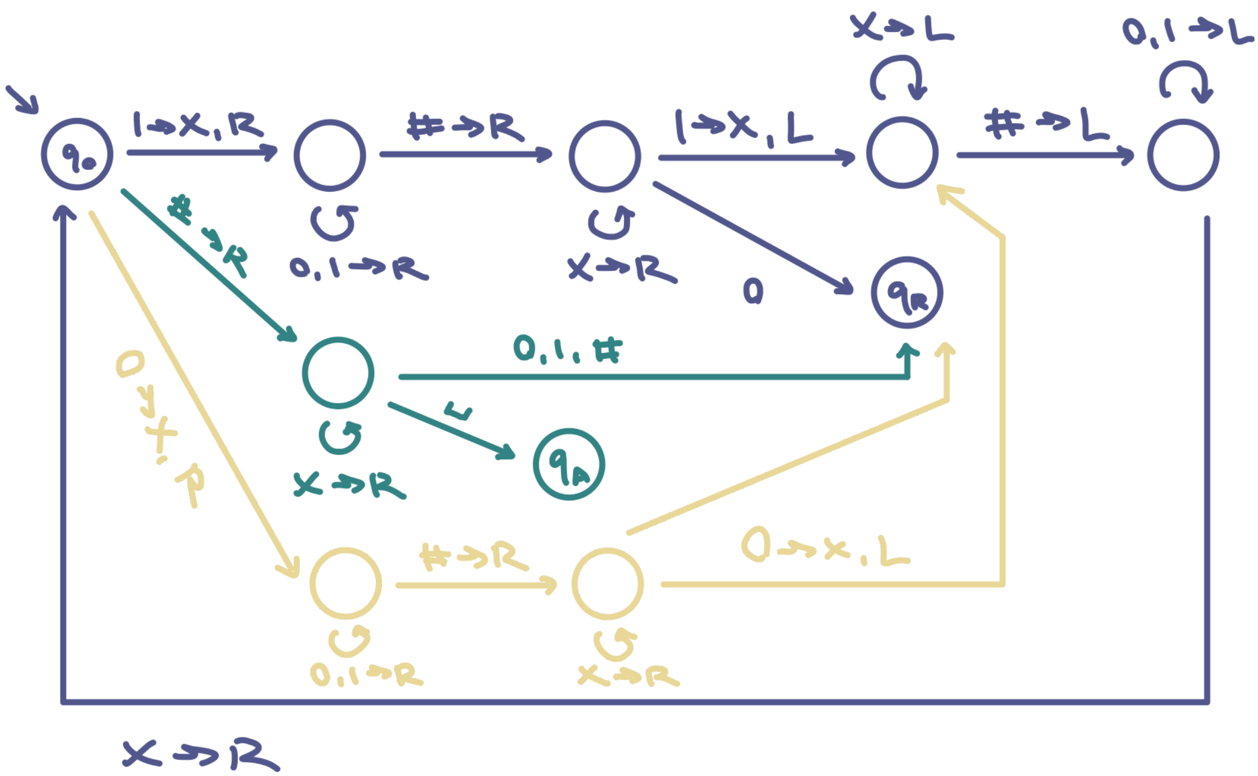
$\text{X}110011\#\text{X}110011$

$\text{XX}10011\#\text{XX}10011$

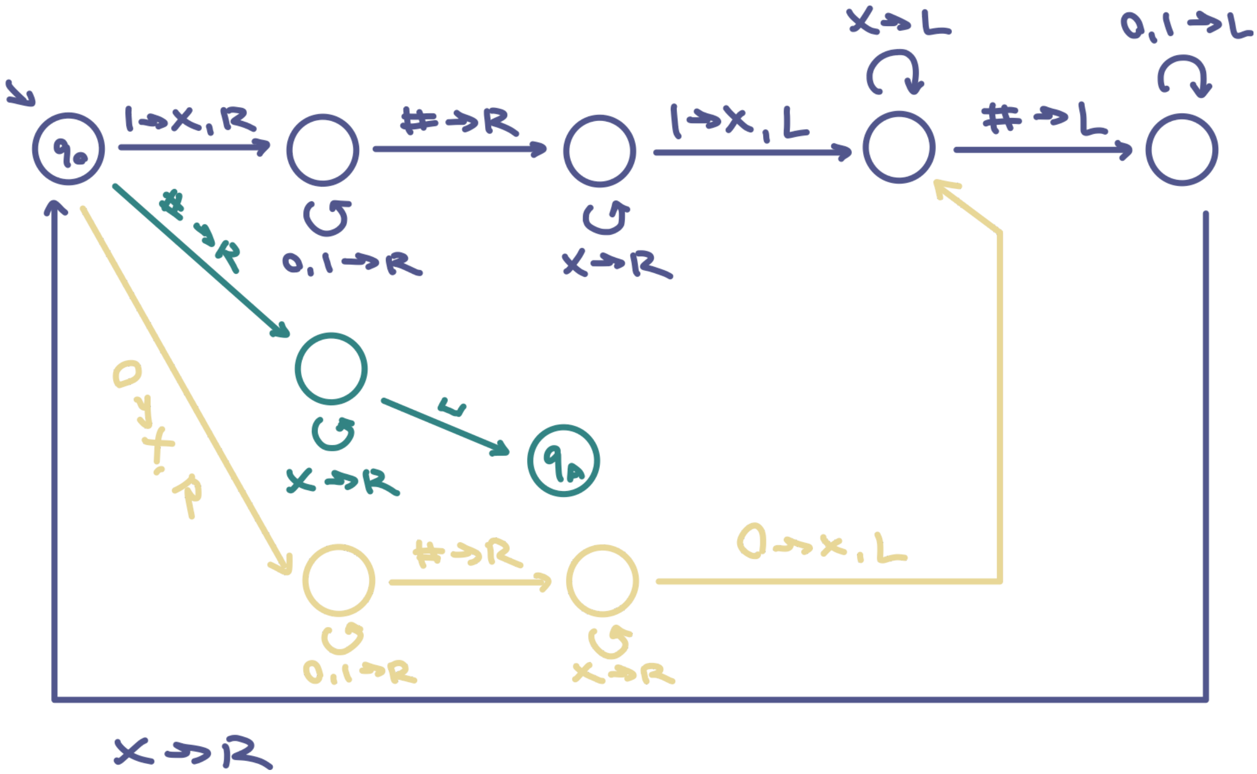
...

$\text{XXXXXXX}\#\text{XXXXXXX}$

- $Q = \{q_0, q_{accept}, q_{reject}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma = \{0, 1, \#, \underbrace{\_\_}_{space}, \text{X}\}$




We simplify the diagram; **missing transitions go to $q_{reject}$**.



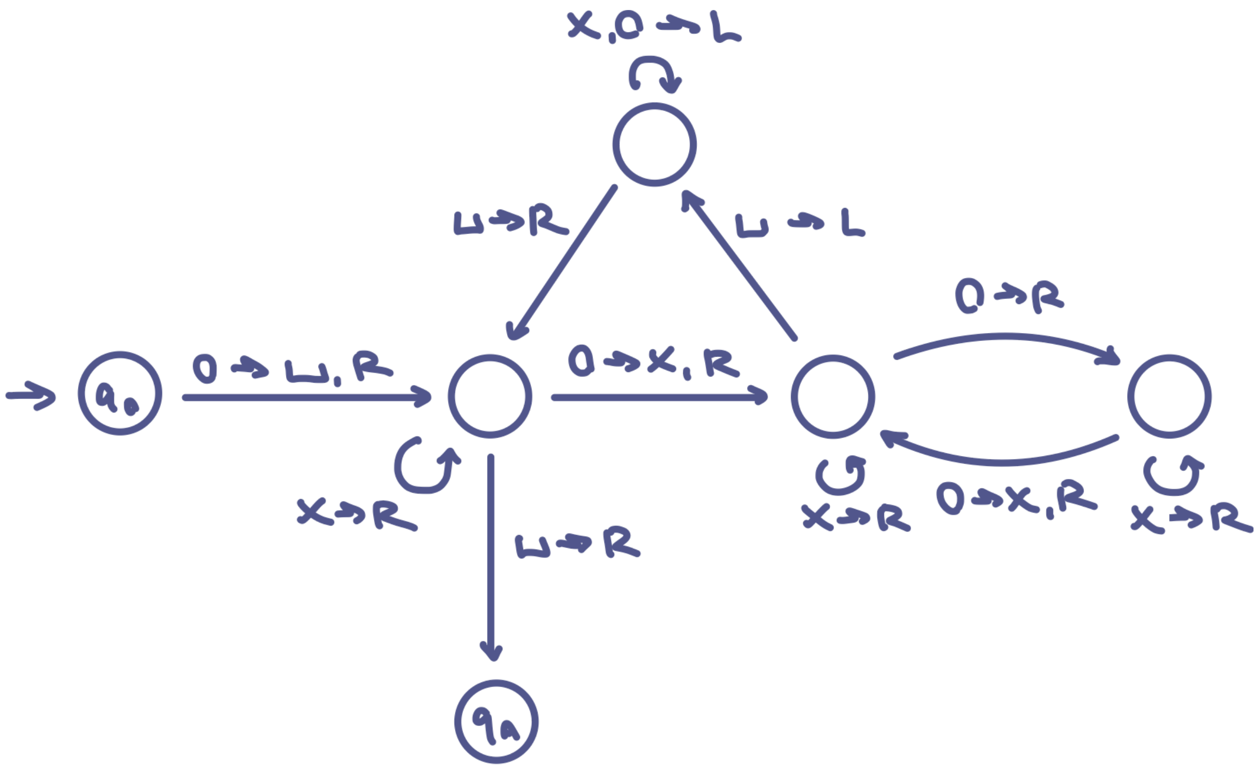

## 11.3 Powers of 2

### Example

$L = \{0^{2^n} : n \geq 0\}$

Erase every other 0 (recursively).

- $\Sigma = \{0\}$
- $\Gamma = \{0, \underset{space}{\text{—}}, \text{X}\}$




## 11.4 Divisibility

### Example

$L = \{a^n b^{kn} : n, k \geq 1\}$

TM Sketch

1. Dot start cell.

   ($a \to \dot{a}, b \to \dot{b}$)

2. Scan input to ensure it has form $a^+ b^+$. Reject otherwise.

3. Return head to left end.

4. Shuffle between $a$'s and $b$'s , crossing off one of each. If $b$'s are gone and $a$'s remain, reject.

   (Cross off $a, \dot{a}, b, \dot{b}$)

5. Restore crossed-off $a$'s . If no $b$'s, accept. Else, go to Step 4.

   (Restore crossed-off $a, \dot{a}$)

# 11.5 Distinctness

## Example

$L = \{\#w_1\#w_2\#w_3...\#w_k : k \geq 0, w_{1:k} \in \{0,1\}^*\}$

TM Sketch

1. If current symbol $\notin \{\underbrace{\text{—}}_{space}, \#\}$, reject. If $\underbrace{\text{—}}_{space}$, accept. If $\#$, dot it ($\# \to \dot{\#}$).
2. Scan right for next $\#$ and dot it. If none found, accept.
3. Compare the two dotted strings by shuttling (?). If equal, reject.
4. Move rightmost dot to next $\#$ and go to Step 3. If no $\#$ found, remove rightmost dot.
5. Put head immediately after the last dotted string (possibly on a blank). Go to Step 1.