Formalizing Problems & Inputs

- A problem for us would always mean computing a fn. f whose input & output are finite boolean strings.

multivalued boolean function $f: \{0,1\}^* \to \{0,1\}^*$.

- This captures almost all math. fns. as we can express the objects "integer x", "graph G", "vector v", "matrix A", etc. in bits!

In CS 0/1 refers to OFF/ON or False/True.

- Usually, problems have meaningful boolean/decision versions, i.e. $g: \{0,1\}^* \to \{0,1\}$.

e.g. $+$ has the decision version $(+, i)$ which asks for the i-th bit in the sum $(a+b)$.

- A decision problem f has an associated language $L_f := \{ x \in \{0,1\}^* \mid f(x) = 1 \}$.

  eg. $L_{(+,i)} = \{ (a,b) \in \{0,1\}^* \mid$ i-th bit in $(a+b)$ is $1 \}$.

- The problems "computing a boolean f" & "deciding a language L_f" are the same!

Formalizing Machines

- A Turing machine (TM) M is described by a tuple (Γ, Q, δ) & two tapes.
  (finite control; infinite memory)

  Where,

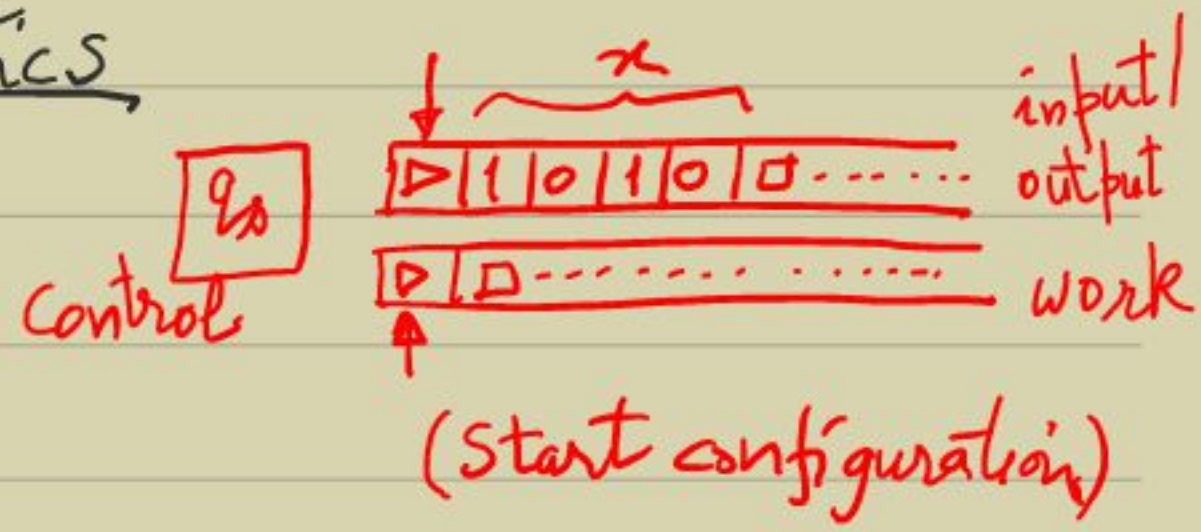
  • Γ is the alphabet: It contains $\triangleright$ (start), $\square$ (blank) & other symbols (eg. 0,1).

  • Q is the set of states: It contains the states M can "be in". Q has at least q_s & q_f.

• δ is the transition fn.:

$$\delta: Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{S, L, R\}^2.$$

TM Semantics



- There are two tapes: input/output + work.
- Respectively, two heads.

- Say, the machine is at state $= q$, input cell value $= i$ & work cell value $= w$.

Then, $\delta(q, i, w) = (q', i', w', \varepsilon_1, \varepsilon_2)$ means:

In the next step (or configuration), state $= q'$, input cell value $= i'$, work $= w'$, input head moves via ε_1 (Stay, Left or Right), work head " " ε_2.

(No head moves to the left of $\triangleright$.)

- The computation "starts" with the start-configuration $(q_s, \triangleright, \triangleright)$ & "stops" (or halts) at the state q_f.

- The number of steps in the middle is the <u>time taken by M on x</u> to halt.

- The number of work-tape cells used is the <u>space taken by M on x</u>.

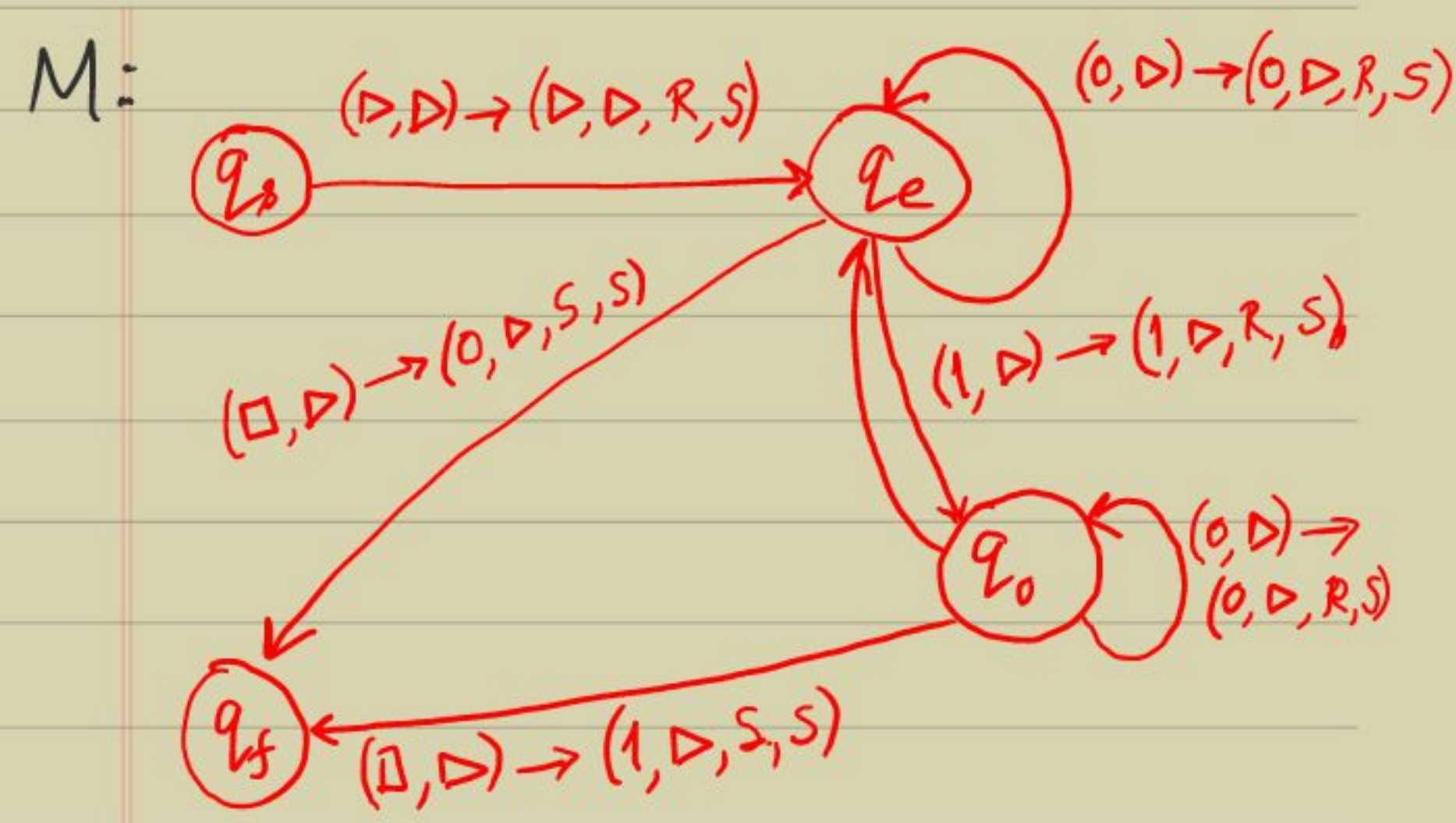
We disallow any writing on the first tape, except the output.

- Eg. <u>A TM to compute parity (x)</u>.

We want $x_1 \oplus x_2 \oplus \cdots$. Roughly, the TM has steps:

(1) Read a bit of x.
(2) If it's 1 flip the state,
(3) continue till input-cell has $\square$.
(4) write 0 or 1, as the output, based on the state.

M:



Flowchart?

– This is called a <u>state-diagram of the TM M</u>.

It describes δ; with the <u>nodes</u> being the states $Q = \{q_s, q_f, q_0, q_e\}$, & the <u>edges</u> specify a transition.

In this eg. the output bit is at the output-head in the end of the computation!

– The size of the state-diagram $\approx$ size of a C-program simulating this algorithm!

Time complexity (of a TM)

Definition: Let $f: \{0,1\}^* \to \{0,1\}^*$ & M be a TM computing f. Let $T: \mathbb{N} \to \mathbb{N}$ be a fn.

We say that "M computes f in $T(n)$ time" if $\forall x \in \{0,1\}^*$, M on input x halts after $\leq T(|x|)$ steps & outputs $f(x)$.

- Similarly, define M computes f in $S(n)$ space

Asymptotic notation

- Let f, g be fns. from $\mathbb{N}$ to $\mathbb{R}_{\geq 0}$.

- **Big-Oh**: $f = O(g)$, if $\exists$ constants c, n_0 s.t. $\forall n \geq n_0, f(n) \leq c \cdot g(n)$.
- **Big-Omega**: $f = \Omega(g)$, if $g = O(f)$.
- **Theta**: $f = \Theta(g)$, if $f = O(g)$ & $f = \Omega(g)$.
- **small-oh**: $f = o(g)$, if $\lim_{n \to \infty} f(n)/g(n) = 0$.
- **small-omega**: $f = \omega(g)$, if $g = o(f)$.
- **Oh-tilde**: $f = \tilde{O}(g)$, if $f = g \cdot (\log g)^{O(1)}$.

- Eg. $100(n+n^2) = O(n^2)$, $\Theta(n^2)$.

" $= o(n^{2.001})$, $\omega(n^2/\log n)$.

$n\log n = O(n^{1.01})$, $o(n^{1+\varepsilon})$.

▷ $n^c = o(2^n)$, for all constant c.

- Some remarks:

(1) The input/output-tape is only for writing the output string (preceded by the input string).

For all the other stuff use the work-tape.

(2) Though you can search a name in a directory in #steps = "$\log_2 N$", it does not mean that the problem has time complexity $O(\log n)$!

Because, here you are assuming N to be fixed & the directory already indexed. (Random-access required.)