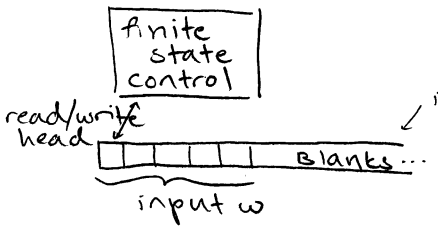
Turing machine



To "program" a T.M., specify the transitions:

Depending on: • State
• Symbol read by head

Effect: • New state
• Overwrite tape cell
• Move head left/right

Formally $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q - states $q_0 \in Q$ initial state

Σ - input alphabet Γ tape alphabet

$F \subseteq Q$ final states $B \in \Gamma$ blank symbol

Note: $\Sigma \subseteq \Gamma$ (since input w is on tape)

$B \in \Gamma \setminus \Sigma$ (rest of tape is initially Bs)

Q, Σ, Γ: finite sets

Deterministic Turing machine (DTM)

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

$\delta(q, a) = (p, b, L)$ means in state q, if reading symbol a, replace a by b, go to state p, and move head one cell to left

for convenience, we allow δ to be undefined for some (q, a), but if defined it is unique

Initially State - q_0

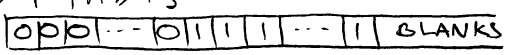
Tape - w followed by blanks

Head - leftmost cell on tape

Halting? Machine halts immediately if it enters a final state and "accepts"

"Rejects" - halt in non-final state (stuck)
- infinite loop
- head falls off left end of tape

Example $L = \{0^n 1^n \mid n \geq 1\}$

Input tape: 

Program idea: Match leftmost 0s and 1s

Step 0: Replace 0 by X, go right

Step 1: Move right till 1, replace by Y, go left

Step 2: Move left till X, then one cell right

Step 3: If cell has 0, then replace by X, go right, return to step 1
else if cell has Y then go to step 4

Step 4: Move right, skipping Ys, till a blank, and accept

note: can only read 0 or Y

ensures no extra symbols left over at end?

Machine M $Q = \{q_0, q_1, \ldots, q_4, f\}$

$F = \{f\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, X, Y, B\}$

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$, $\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_2, Z) = (q_2, Z, L)$ for $Z \in \{0, Y\}$

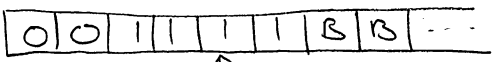
$\delta(q_2, X) = (q_3, X, R)$

$\delta(q_3, 0) = (q_1, X, R)$, $\delta(q_3, Y) = (q_4, Y, R)$

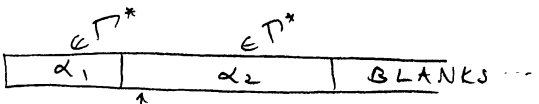
$\delta(q_4, Y) = (q_4, Y, R)$

$\delta(q_4, B) = (f, B, R)$

Describing execution



represented by $00\underset{\uparrow}{1}q11$



by $\alpha_1 q \alpha_2$

- omitting trailing blanks!
- both α_1, α_2 may be empty

Example $q_0 0011 \vdash X q_1 011 \vdash X0 q_1 11 \vdash X q_2 0Y1$
$\vdash q_2 X0Y1 \vdash X q_3 0Y1 \vdash XX q_1 Y1$
$\vdash XXY q_1 1 \vdash XX q_2 YY \vdash X q_2 XYY$
$\vdash X X q_3 YY \vdash XXY q_4 Y \vdash XXYY q_4$
$\vdash XXYYBf$

Language

$$L(M) = \{ w \mid q_0 w \overset{*}{\vdash} \alpha_1 p \alpha_2 , \ p \in F , \ \alpha_1, \alpha_2 \in \Gamma^* \}$$

Note: On any given input, a Turing machine can either accept, reject (crash), or run forever. The language $L(M)$ includes only those strings that are accepted.

Definition:
- A language is Turing recognizable, also known as recursively enumerable if it is the language of some Turing machine.
- A language is decidable, AKA recursive if it is the language of a Turing machine that halts on all inputs (does not run forever).

Turing machine examples

Example

$L = \{0^{2^n} \mid n \geq 0\}$ — all strings of 0s whose length is a power of 2

① High-level description

Repeatedly sweep through the input, crossing off every other zero in each pass. If during a pass, an odd number of 0s (more than 1) is seen, then reject. Otherwise accept.

② Implementation description (with more details of how the head moves and data is stored)

On input string w:

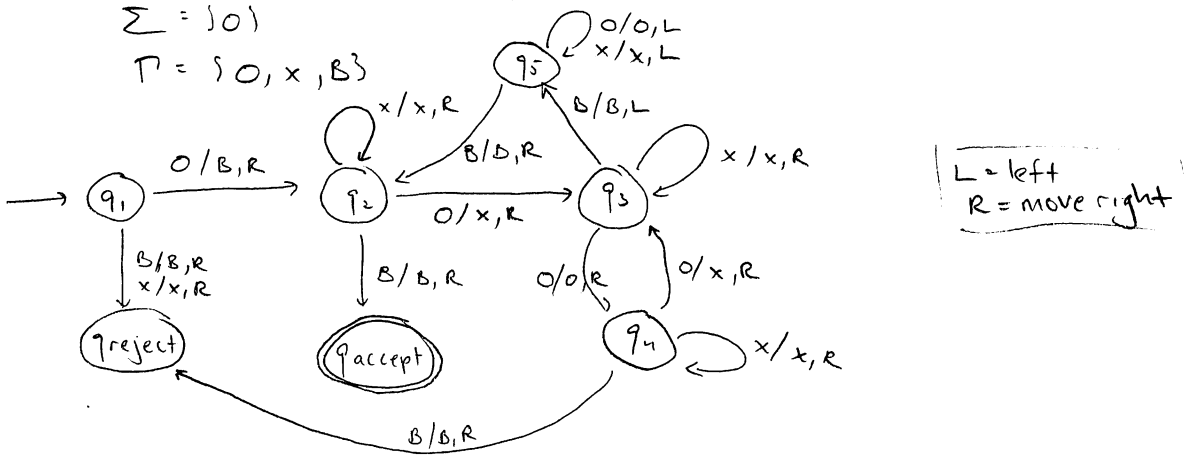
1. Sweep left to right across the tape, crossing off every other 0
2. If in stage 1 the tape contained a single 0, accept.
3. If in stage 1, tape contained an odd number of 0s greater than 1, then reject
4. Return the head to the left end of the tape.
5. Go to step 1.

③ Formal description (most detailed, usually too detailed)

$Q = (q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject})$

$\Sigma = \{0\}$

$\Gamma = \{0, x, B\}$



E.g., on input 00: $q_100 \vdash B q_2 0 \vdash B x q_3 B \vdash B q_5 x \vdash q_5 B x \vdash q_{accept} B x$

on input 0000: ...

Note: We start by writing a blank symbol B so the machine can find the left end of the tape without falling off!

Example $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

On input w:

1. Scan the string left to right, making sure string $\in L(a^* b^* c^*)$ reject if not
2. Return the head to the left end of the tape
3. Cross off an a and scan right to find a b. Shuttle between b's and c's, crossing off one of each until all b's are gone.
4. Restore crossed-off b's and repeat stage 3 if there is another a. If no more a's then check whether all c's are crossed off. If so, accept, else reject.

Example:

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

  $L = \{\# x_1 \# x_2 \# \cdots \# x_\ell \mid \text{each } x_j \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$

Turing machine programming tricks

① Multiple tracks

0	1	X	
1	0	B	...
A	C	*	

Γ ? $(0,1,A), (1,0,C), \ldots$

δ ? $\delta(q, (0,1,A)) = (p, (1,1,C), R)$

Example Checking off symbols

markers	✓	✓	B	B	✓	✓	...
input	0	0	0	1	1	1	...

- also often used for pointers

② Subroutines

Example: Shifting memory over $\alpha q_i \beta \mapsto \alpha B q_j \beta$, $\alpha, \beta \in \Gamma^*$ (useful for creating space)

Procedure call: $\delta(q_i, a) = ((p,a), [\dot{B}], R)$ $\forall a \in \Gamma$

- memorize return state, erased symbol a
- state p invokes procedure

Procedure p: ① Shift 1 cell to right $\delta((p,a), X) = ((p,X), a, R)$ $\forall a, X \in \Gamma, X \neq B$

② Till reach end of β $\delta((p,a), B) = (t, a, L)$ $\forall a \in \Gamma$

③ Return to calling point $\delta(t, a) = (t, a, L)$ $\forall a \neq [\dot{B}]$

④ Exit procedure $\delta(t, [\dot{B}]) = (q_j, B, R)$.

Useful for implementing counters:

$\# 0 \# \longrightarrow \# 1 \# \longrightarrow \# 10 \# \longrightarrow \# 11 \# \longrightarrow \# 100 \# \longrightarrow \ldots$

③ Storage in finite-state memory ...