Turing machine
Simple theoretical machine
Can do anything a real computer can do!

Detour: “Turing test”
A computer is “intelligent” if human investigator can’t tell if she’s talking to a human or a computer
Turing machine

Simple theoretical machine
Can do anything a real computer can do!

Turing machine structure
Infinite tape
At each step
• Must move left/right on tape
• Can change state
• Can change tape content
When reaches accept or reject state, terminates and outputs "accept" or "reject"
Can loop forever

Review of machines

• Finite state automaton (Regular languages)

• Push down automaton (Context free languages)

• Turing machine (beyond CFLs)

A Turing Machine for $B=\{w\#w \mid w \in \{0,1\}^*\}$
Assume the string is written on the tape and you start at the beginning of the string. What can we do?
Strategy:
Find left-most 0-or-1 character in first word
If match left-most character in second word, X out both chars
Else reject
If no characters left, accept

How do we move this with single actions:
move-by-one and write?

Turing machine: the formal definition

7 tuple: \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\)
- \(Q\) is set of states
- \(\Sigma\) is input alphabet
- \(\Gamma\) is the tape alphabet; \(\text{blank} \in \Gamma\) and \(\Sigma \subseteq \Gamma\)
- \(\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) transition function
- Start, accept, and reject state: \(q_0, q_{\text{accept}}, q_{\text{reject}}\)

The transition function

Given state \(q\) and symbol \(a\) at present location on tape,
change to state \(r\), change symbol on tape to \(b\), move Left or Right

Change in: (state, tape content, head location)
- called “configuration”

Example:
Start at \(q_2\). Current position underlined.
Step 0: \(q_2 \sim 0 0 1 1 \# \sim 0 0 1 1 \# \sim 0 0 1 \# \sim\)
Step 1: \(q_3 \sim 0 0 0 1 \# \sim 0 0 1 \# \sim 0 0 1 \# \sim\)
Step 2: \(q_4 \sim 0 1 0 1 \# \sim 0 1 0 1 \# \sim\)
Step 3: \(q_4 \sim 0 1 0 1 \# \sim\)
The transition function

Example:
Start at $q_2$. Current position underlined.
Step 0: $q_2 \sim 0 \ 0 \ # \ 1 \ \sim \sim 0 \ 0 \ # \ 1 \ \sim$
Step 1: $q_2 \sim 0 \ 0 \ # \ 1 \ \sim \uparrow \sim 0 \ 0 \ # \ 1 \ \sim$
Step 2: $q_? \ T$Lape: ? ? ?
Step 3: $q_? \ T$Lape: ? ? ?

Strategy: $B=\{w#w \mid w \in \{0,1\}^*\}$
Find left-most 0-or-1 character in first word
If match left-most character in second word, X out both chars
Else reject
If no characters left, accept

Strategy: $B=\{w#w \mid w \in \{0,1\}^*\}$
Define TM state sequence for each big-picture algorithmic step
Given character $s$ in left word
1. Move to right word
2. Check if first available symbol in right word == $s$
3. If match, keep going; else reject
Strategy: \( B = \{ w \# w \mid w \in \{0,1\}^* \} \)

1. Move to right word
2. Check if first available symbol in right word == \( s \)
3. If match, keep going; else reject
Strategy: \[ B = \{ w \# w \mid w \in \{0,1\}^* \} \]

Find left-most 0-or-1 character in first word
If match left-most character in second word, X out both chars
Else reject
If no characters left, accept

Analysis: We will always get an answer (accept or reject), because problem gets smaller after each step

"Turing recognizable" vs. "Decidable"

L(M) – "language recognized by M" is set of strings M accepts
Language is Turing recognizable if some Turing machine recognizes it
• Also called "recursively enumerable"
Machine that halts on all inputs is a decider. A decider that recognizes language L is said to decide language L
Language is Turing decidable, or just decidable, if some Turing machine decides it

Example non-halting machine

Turing machine structure

Infinite tape
At each step
• Must move left/right on tape
• Can change state
• Can change tape content
When reaches accept or reject state, terminates and outputs “accept” or “reject”
Can loop forever
Turing Machine for $C = \{0^{2^n} \mid n \geq 0\}$

*Recursive division by 2*

Sweep left to right across tape, cross off every-other 0

If

- Exactly one 0: accept
- Odd number of 0s: reject
- Even number of 0s, return to front

---

Alternating 0s in action:

TM M2 “decides” language C

If you land on a location and want to cross it out, but it is a ~, you crossed out an even number of 0s – do another loop!

If you land on a location and want to skip over it, but it is a ~, you crossed out an odd number of 0s – reject!

---

Language $D = \{a^i b^j c^k \mid k = ixj \text{ and } i, j, k > 0\}$

Multiplication on a Turing Machine!

Consider $2 \times 3 = 6$

---

Transducers: generating language

So far our machines accept/reject input

Transduction: Computers transform from input to output

- New TM: given $i$ a’s and $j$ b’s on tape, print out $ixj$ c’s
Transducer: Write $c^k$, $k=ixj$, given $i$ a’s, $j$ b’s,

Scan string to confirm form is $a^*b^+$
• if so: go back to front; if not: reject
$X$ out first $a$, for each $b$, $Y$ off that $b$ and add $c$ to the end
Restore crossed out $b$’s, repeat $b$—$c$ loop for next $a$
• If all $a$’s gone, accept

“Transducer” in action:

Symbol $X$ is an $a$ that is removed
Symbol $Y$ is a $b$ temporarily out of service as you go through all the other $b$’s

TM 4: Element distinctiveness

Given a list of strings over $\{0,1\}$, separated by $\#$, accept if all strings are different:

Example: 01101#1011#00010

TM 4 solution

1. Place mark on top of left-most symbol. If it is blank: accept; if it is $\#$: continue, otherwise: reject
2. Scan right to next $\#$ and place mark on it. If none encountered and reach blank: accept
3. Zig-zag to compare strings to right of each marked $\#$
4. Move right-most marked $\#$ to the right. If no more $\#$: move left-most $\#$ to its right and the right-most $\#$ to the right of the new first marked $\#$. If no $\#$ available for second marked $\#$: accept
5. Go to step 3
TM 4 solution: alternate description

1. Mark left-most un-removed word as wordA; if none available, accept
2. Move to right until reach new un-removed word (if reach blank, loop to step 1)
3. Mark new word as wordB
4. If wordA=wordB, reject; else temporarily remove wordB and continue
5. Loop to step 2

“Distinctiveness checker” in action:

Decidability

How do we know decidable?
• Simplify problem at each step toward goal
• Can prove formally – number of remaining symbols at each step

Showing language is Turing recognizable but not decidable is harder