Turing Machine

(A. Turing, 1936)

- basic model in Computability Theory and Complexity Theory
- control unit with a finite set of states
- number of unlimited memory tapes, each with a read/write head
- state transformation rules

Non-deterministic k-tape Turing machine

(k ≥ 1) consists of

- finite set of states Q
- finite sets of tape symbols T and input symbols I ⊆ T
- blank symbol b ∈ T
- start state and final state q₀, q_f ∈ Q
- transition relation

\[ \delta : Q \times T^k \rightarrow \mathcal{P}(Q \times (T \times \{L,S,R\})^k) \]

Meaning of \(\delta(q,a_1,\ldots,a_k)\):

If machine \(M\) is in state \(q\) and tape heads point at symbols \(a_1,\ldots,a_k\), and \((q',(a_1',d_1),\ldots,(a_k',d_k)) \in \delta(q,a_1,\ldots,a_k)\) then \(M\) can, in a single step,

- move to state \(q'\) and
- at each tape \(i = 1,\ldots,k\), replace \(a_i\) by \(a_i'\) and move the tape head to left \((d_i = L)\) or right \((d_i = R)\) or leave it where it was \((d_i = S)\)

\(M\) is deterministic if \(|\delta(q,a_1,\ldots,a_k)| \leq 1\) for all \(q \in Q, a_1,\ldots,a_k \in T\).
**Configuration** of TM $M$ is a $k$-tuple

$$(x_1y_1, \ldots, x_ky_k),$$

where $q \in Q$ and $x_i \in T^*$, and $y_i \in T^+$

(exclude leading blanks of $x_i$, and include the terminating blank of $y_i$ only when $y_i$ is empty)

Meaning: $M$ is in state $q$, tape $i$ contains $x_iy_i$, and its tape head points at first char of $y_i$

Single move according to $M$’s transition relation:

$$C_1 \vdash_M C_2$$

**Execution** of $M$:

Configuration $C_1$ leads to configuration $C_n$, if $n = 1$ or $C_1 \vdash_M C_2 \vdash_M \cdots \vdash_M C_n$ through some configurations $C_2, \ldots, C_{n-1}$

---

Turing machine $M$ accepts input $w \in I^*$, if

$$(q_0w, q_0b, \ldots, q_0b) \vdash_M^* (x_1q_1y_1, x_2q_2y_2, \ldots, x_kq_3y_k)$$

(with some tape contents $x_iy_i$, $i = 1, \ldots, k$)

The language accepted by $M$:

$$L(M) = \{w \in I^* \mid M \text{ accepts } w\}$$

---

**Computing a function** with a deterministic Turing machine $M$:

Define $f_M(u) = v$, if

$$(q_0u, q_0b, \ldots, q_0b) \vdash_M^* (x_1q_1y_1, x_2q_2y_2, \ldots, x_kq_ky_k)$$

(and final state $q_f$ does not appear earlier)

Execution may be non-terminating → function $f_M$ may be partial

A total function that is computed by some Turing machine is **Turing-computable**

A decision problem $L$ is **Turing-solvable**, if its membership function $L \rightarrow \{0, 1\}$ is Turing computable

---

"Church–Turing Thesis"

Turing-computability $\equiv$ algorithmic computability

Turing-solvability $\equiv$ algorithmic solvability

Not a provable theorem

Observation: All "realistic" models of computation can be used to simulate computations that can be expressed in other models
Example Recognizing binary palindromes

\[ Q = \{ q_0, \ldots, q_4 \}, T = \{ 0, 1, b \}, I = \{ 0, 1 \}, q_f = q_4 \]

Transition relation:

<table>
<thead>
<tr>
<th>( q )</th>
<th>( x )</th>
<th>( y )</th>
<th>( q_0 )</th>
<th>( b )</th>
<th>( q_1 )</th>
<th>( b )</th>
<th>( q_2 )</th>
<th>( b )</th>
<th>( q_3 )</th>
<th>( b )</th>
<th>( q_4 )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>b</td>
<td>b</td>
<td>q_4</td>
<td>0</td>
<td>b</td>
<td>q_1</td>
<td>0</td>
<td>b</td>
<td>q_2</td>
<td>0</td>
<td>b</td>
<td>q_3</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>q_1</td>
<td>0</td>
<td>b</td>
<td>q_1</td>
<td>0</td>
<td>b</td>
<td>q_2</td>
<td>0</td>
<td>b</td>
<td>q_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(b, S)</td>
<td>0</td>
<td>0</td>
<td>(b, S)</td>
<td>0</td>
<td>0</td>
<td>(b, S)</td>
</tr>
</tbody>
</table>

accept empty
go to copy state
copy tape 1 to 2
move head 2 left
compare tapes

Complexity of Turing Machine

**Time complexity** \( T(n) \): max \# of steps in an execution that accepts an input of length \( n \); undefined, if no input of length \( n \) is accepted

A nondeterministic TM may have alternative accepting computations; the shortest of those is taken:

\[ T_M(n) = \max_{w \in \{ 0, 1 \}^n} \min_{M} \{ k | M \text{ accepts } w \text{ in } k \text{ steps} \} \]

**Space complexity** \( S(n) \): max \# of tape cells used for accepting input of length \( n \)

Obs: \( S(n) \leq kT(n) \)

Time and space complexity of a TM for computing a function are defined similarly

Example For palindrome TM, \( T(n) = 3n + 4 \) and \( S(n) = 2n + 4 \)

---

Example TM for multiplying binary numbers

1. (Init) Receive input \( x \times y \) on Tape 1, say, \( 1001101 \times 101101 \)

   Copy \( x \) to Tape 2, writing equally many zeros on Tape 3 (*result*):

   \( 1001101 \times 101101 \)

2. (Addition)

   a. If Tape 1 contains 0, go to 3.

   b. Otherwise, compute the sum of Tapes 2 and 3 to Tape 3 with a right-to-left scan, and return tape heads back to right

   Situation after the first addition:

   \( 1001101 \times 101101 \)

   \( 1001101 \)

   \( 1001101 \)

   \( 0000000 \)
3. (Continue?) Move head of Tape 1 to right. If $b$ found there, finish. Otherwise move head of Tape 3 to right, write 0, and go to 2.

Situation after the first round:

1001101 * 101101
  1001101
  10011010

Situation at the end of computation:

1001101 * 101101b
  1001101
  110110001001

**Complexity?**

$S(n) \approx 5n$

$T(n)$:

Init: $n + 1$ steps

Additions: the $k$th one writes $x \cdot y_1 \cdots y_k$ on Tape 3, and returns tape head to right

Length of the value is $\leq n + k$

$\rightarrow \# \text{ of steps } \leq \sum_{k=1}^{n}(2(n + k + 1))$

Each 'Continue?' test takes two steps