Turing machine

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

- State
- Effect: New state
- Symbol read by head
- Overwrite tape cell
- Move head left/right

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

- $Q$: states
- $\Sigma$: input alphabet
- $\Gamma$: tape alphabet
- $F$: set of final states
- $B$: blank symbol

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

$Q, \Sigma, \Gamma$: finite sets

Deterministic Turing machine (DTM)

\[ \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \]

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

For convenience, we allow $\delta$ to be undefined for some $(q, a)$, but if defined it is unique.

- Initially $Q_0$
- Tape = $w$ followed by blanks
- Head = leftmost cell on tape

 Halts: Machine halts immediately if it enters a final state and accepts.
 Rejects: machine halts in non-final state (stuck)

Example $L = \{a^n b^n | n \geq 1\}$

Input tape

Program idea: Match leftmost Os and Is

- Step 0: Replace O 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 O, 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.

Machine $M$

- $Q = \{q_0, q_1, \ldots, q_9, f\}$
- $F = \{f\}$
- $\Sigma = \{0, 1, \text{blank}\}$
- $\Gamma = \{0, 1, X, Y, B\}$

Input:

Note: can only read $0$ or $Y$.

Ensures no extra symbols left over at end.

\[
\begin{align*}
\delta(q_0, 0) &= (q_1, 0, R) \\
\delta(q_0, 1) &= (q_1, 1, R) \\
\delta(q_1, 0) &= (q_2, 0, L) \\
\delta(q_1, 1) &= (q_2, 1, L) \\
\delta(q_2, 0) &= (q_3, 0, R) \\
\delta(q_2, 1) &= (q_2, y, R) \\
\delta(q_3, 0) &= (q_4, 0, R) \\
\delta(q_3, 1) &= (q_4, y, R) \\
\delta(q_4, B) &= (f, B, R)
\end{align*}
\]
Describing execution

represented by 0011q1
- omitting trailing blanks!
- both $a_1$, $x_2$ may be empty

Example

$\begin{align*}
q_0, 0011 & \rightarrow X_1, 011 \rightarrow X_1, 011 \rightarrow X_1 q_0 x_1 y_1 \\
q_1 & \rightarrow q_1 x_0 y_1 \rightarrow q_1 x_0 y_1 \rightarrow q_1 x_1 y_1 \\
q_2 & \rightarrow q_2 x_2 y_1 \rightarrow q_2 x_2 y_1 \rightarrow q_2 x_2 y_1 \\
q_3 & \rightarrow q_3 x_3 y_1 \rightarrow q_3 x_3 y_1 \rightarrow q_3 y_1 \\
q_4 & \rightarrow q_4 y_1 \rightarrow q_4 y_1 \\
q_5 & \rightarrow q_5 x_5 y_1 \\
q_6 & \rightarrow q_6 x_6 y_1 \\
q_7 & \rightarrow q_7 x_7 y_1 \\
q_8 & \rightarrow q_8 x_8 y_1 \\
q_9 & \rightarrow q_9 x_9 y_1 \\
q_{10} & \rightarrow q_{10} x_{10} y_1 \\
\end{align*}$

Language

$L(M) = \{ w \mid q_0 w \rightarrow x, p a_2, p \in F, a_1, a_2 \in \Sigma^* \}$

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

$L = \{0^n \mid n \geq 0\}$ — 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 is seen, then reject. Otherwise accept.

2. **Implementation description** (with more details of how the head moves and data is stored)
   On input string 0:
   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)

   $Q = \{q_0, q_1, q_5, q_9, q_{accept}, q_{reject}\}$
   $\Sigma = \{0\}$
   $\Gamma = \{0, 1, B\}$

   ![Diagram of the Turing machine](image)

   Example on input 00: $q_1 \to q_0 \to Bq_5 \to Bx_9x \to Bq_9x \to q_{accept}Bx$ + $\ldots$

   Example on input 0000: $\ldots$

   Note: We start by writing a blank symbol B so the machine can find the left end of the tape without falling off!
Example: \( L = \{ a^i b^j c^k \mid i \cdot j = k \text{ and } i, j, k > 1 \} \)

On input \( w \):
1. Scan the string left to right, making sure string \( \epsilon \) \((a^i b^j c^k)\)
2. Return the head to the left end of the tape
3. Cross off an \( \epsilon \) 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 \( \epsilon \). If no more \( \epsilon \) 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
\( L = \{ \# x_1 \# x_2 \# \ldots \# x_k \mid \text{each } x_j \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j \} \)
Turing machine programming tricks

1. Multiple tracks

<table>
<thead>
<tr>
<th>Σ</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

Σ = (0,1,A), (1,0,C), ...

δ ? δ(q,(0,1,A)) = (p,(1,1,C),R)

**Example**: Checking off symbols

<table>
<thead>
<tr>
<th>markers</th>
<th>✓</th>
<th>✓</th>
<th>B</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- also often used for pointers

2. Subroutines

**Example**: Shifting memory over \( \alpha q i \beta \rightarrow \alpha e q i \beta \), \( \alpha, \beta \in \Sigma^* \) (useful for creating space)

**Procedure** call: \( δ(q, a) = (p, a, [\varepsilon], R) \) \( \forall a \in \Sigma \)
- memorize return state, erased symbol \( a \)
- state \( p \) invokes procedure

**Procedure** \( p \):
1. Shift 1 cell to right \( δ((p, a), X) = (p, X, a, R) \)
2. Till reach end of \( \beta \) \( δ((p, a), B) = (t, a, L) \) \( \forall a \in \Sigma \)
3. Return to calling point \( δ(t, a) = (t, a, L) \) \( \forall a \in \Sigma \)
4. Exit procedure \( δ(t, [\varepsilon]) = (q, B, R) \)

Useful for implementing counters:

$\$: $0$ $→ $1$ $→ $10$ $→ $11$ $→ $100$ $→ $...

3. Storage in finite-state memory...