Turing Machine (TM)

Ex1 \( \{ o^n 1^n 2^n \mid n \geq 1 \} \)

Ex2 \( \{ ww \mid w \in \{0,1\}^* \} \)

Ex3 \( \{ 0^n^2 \mid n \geq 0 \} \)

tricks - can copy, shift, ... (by marking)
- compose functions (subroutine)
- do loops
- keep multiple vars
etc.

Def Let \( f : \Sigma^* \to \Sigma^* \cup \{\text{undef}\} \).

A TM \( M \) computes \( f \) iff

\[ \forall x \in \Sigma^*, \; q_0 \stackrel{x}{\xrightarrow{*}} q \text{ acc } f(x) \; \text{ if } f(x) \neq \text{ undef} \]

\( M \) does not accept \( x \) else

Let \( f : \mathbb{N} \to \mathbb{N} \cup \{\text{undef}\} \)

\( M \) computes \( f \) iff

\[ \forall n \in \mathbb{N}, \; q_0 \stackrel{n^0}{\xrightarrow{*}} q \text{ acc } f(n) \; \text{ if } f(n) \neq \text{ undef} \]

\( M \) does not accept else

Let \( f : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \cup \{\text{undef}\} \)

\( n, \; n' \; x \to f(n, n') \)
Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be such that $f(n, n) = n$.

Similar $g_0, 0^n \# 0^n \xrightarrow{\sim} g_{\text{are } O(n^2)}$.

**Example**

- $f(n) = 2n$
- $f(m, n) = mn$
- $f(n) = 2^n$
- $\lceil \log_2 n \rceil$
- Fibonacci $\#$

**Extensions**

- Multitape TM

```
<table>
<thead>
<tr>
<th>tape 1</th>
<th>X_{i1}</th>
<th>X_{i2}</th>
<th>X_{i3}</th>
<th>...</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tape k</td>
<td>X_{k1}</td>
<td>X_{k2}</td>
<td>X_{k3}</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

**e.g.**

- $\{0^n, 1^n, 2^n \mid n \geq 13\}$

Can be simulated by single-tape TM

```
| X_{i1} | X_{i2} | ... | X_{k1} | X_{k2} | ... | X_{k3} | ... |
```

- Nondeterministic TM
can be simulated by det. TM

idea - try all possible execution paths simultaneously
(exponential slow down)

- Random Access Machine (RAM)
  memory as an array + finite # registers

\[
\begin{array}{c|c}
A(0) & R_1 \\
A(i) & R_2 \\
A(j) & R_3 \\
\vdots & \\
\end{array}
\]

Instruction set:
\[R_i := c\]
\[R_i := R_i + R_j\]
\[R_i := A(R_j)\]
\[A(R_i) := R_j\]
\[\text{goto line } l\]
\[\text{if } R_i = 0 \text{ goto line } l\]

can be simulated by TM

\[
\begin{array}{cccccccc}
\$ & 0 & \# & A(0) & \# & 1 & \# & A(i) & \#
\end{array}
\]

\[
\begin{array}{c}
\$ \\
R_i \\
\vdots
\end{array}
\]
Church-Turing Thesis

Any lang./function can be solved by
Some systematic procedure, i.e. "algorithm"
iff it can be accepted/computed by a TM.

Rmk: "thesis" can't be proved mathematically
(think of as "def'n"/"axiom")

Caveat: assume no bounds on time & space

A Universal TM

A TM that simulates all TMs

"Stored-program computer"

Let $\langle M \rangle$ denote code for M

```
input tape
< M >  #  x  1
```

tape for simulation

Current
```
< 92 > 1
```
An Undecidable Problem: Halting

\[ L_{\text{halt}} = \{ \langle M \rangle \# x \mid \text{TM } M \text{ halts on input string } x \} \]

(it is r.e.)

**Thm (Turing '36)** \( L_{\text{halt}} \) is not recursive, i.e. undecidable.

**Pf:** By contradiction.

Suppose you claim to have a program \( M_0 \) that solves \( L_{\text{halt}} \).

I'll construct a counterexample: \( \langle M_{\text{bad}} \rangle \# w_{\text{bad}} \)

\( M_{\text{bad}} \) is the following program:

\[
\begin{align*}
\text{On input } \langle M \rangle, & \\
\text{run } M_0 \text{ on } \langle M \rangle \# \langle M \rangle & \\
\text{if } M_0 \text{ returns yes,} & \\
\text{then go to infinite loop} & \\
\text{else halt.} & \\
\end{align*}
\]

\[ w_{\text{bad}} = \langle M_{\text{bad}} \rangle \]

**Case 1.** \( M_0 \) outputs yes on \( \langle M_{\text{bad}} \rangle \# \langle M_{\text{bad}} \rangle \)

\( M_{\text{bad}} \) on input \( \langle M_{\text{bad}} \rangle \) goes to infinite loop

So \( M_0 \) is wrong!
So Mo is wrong!

Case 2. Mo outputs no on \(\langle M_{bad} \rangle \neq \langle M_{bad} \rangle\)

Mo outputs no on \(\langle M_{bad} \rangle\) halts

So Mo is wrong!