**TURING MACHINES**

- TM can: 1) read & write on tape
  2) pointer (R/W head) moves L or R
  3) Tape is oo (// in 1 direction)
  4) Accept state, reject state: take effect immediately.

\[ B = \{ w \# w \mid w \in \{0,1\}^* \} \]

**TM:** 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})\)

- Input alphabet, cannot contain blank symbol, \(\#\)
- Tape alphabet \(\Gamma\)
- \(Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\)

**Config:**
- \(u \# q \# v\)
- \(u \# a \# q_i \# b \# v\)

- \(u \# q \# c \# v\)
  - if \(\delta(q, b) \rightarrow (q', c, L)\)
- \(u \# a \# c \# q_j \# v\)
  - if \(\delta(q, b) \rightarrow (q', c, R)\)

**Start Config:** \(q_0 \# w\)

**Defn:** TM accepts \(w\) if \(q_0 \# w \rightarrow c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_k \mid c_k\) is a accepting config. (ie accept or reject config)
\[ A = \{ 0^2^n \mid n \geq 0 \} \quad // \text{Given, a } w \text{ in unary notation, is it a power of 2?} \]

**Iter 1**

**Iter 2**

**Iter 3**

**Iter 4**

**Iter 5** \& accept

Input: \( w \).

Loop:

1. Go \( L \to R \), crossing out every 'other' 0
2. Case: if only 1 '0' in step 1 then ACCEPT
   Case: if > 1 and odd # of '0's in step 1 then REJECT
3. Go to L end
$L(M) = \{ w \# w | w \in \{0,1\}^* \}$

**N.B.: Missing transition → transition to $q_{reject}$**

$E = \{ \# x_1 \# x_2 \# \cdots \# x_l | \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j \}$

```
#    ···    #    ···    #    ···    \\
1st  2nd  3rd  4th  5th  6th  7th
```

```
loop $i = 1$ to $k-1$
  loop $j = i+1$ to $k$
    if $x_i \neq x_j$ no-op
    else REJECT
  end loop
end loop
```

- Mark # by # (new member of tape alphabet $\Gamma$)
- Allows you to unmark also simulating extra pointers
Defn: \( L(M) = \{ w \mid M \text{ accepts } w \} \)

Defn: \( L \) is **Turing-recognizable** if some TM recognizes \( L \)

Observation: A TM may accept, reject, or loop forever.

Defn: \( L \) is **Turing-decidable** if some TM that decides \( L \)

Defn: TM decides \( L \) if it (is guaranteed to) enter

- \( \text{accept} \) or \( \text{reject} \)
- on each \( w \in L \) and \( w \notin L \)

<table>
<thead>
<tr>
<th>( w \in L_{174} )</th>
<th>accept</th>
<th>reject</th>
<th>reject or keep looping</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w \notin L_{174} )</td>
<td>accept or keep looping</td>
<td>reject or keep looping</td>
<td></td>
</tr>
</tbody>
</table>
**Variants of TMs**

→ TM are robust

- N-TM, multi-tape TM, shift L/R by multiple positions,
  k-dim grid, 2-Dim infinite tape, ...

Exception: doing an number of actions in a single step

**Oracle**

1) Multitape TM.

**Theorem:** $k$-tape TM $\iff$ 1-tape TM

$$
\begin{align*}
   \delta : Q \times \Gamma^k &\rightarrow Q \times \Gamma^k \times \{L, R\}^k \\
   \delta(q_0, a_1, a_2, \ldots, a_k) &\rightarrow (q', b_1, b_2, \ldots, b_k, L/R, L/R, \ldots, L/R)
\end{align*}
$$

1. Initialize 1-tape TM:

2. Each transition: 2-passes L/R:
   1. Read the $k$ $a_i$'s
   2. Write the $k$ $b_i$'s, shift L/R each

1-step of $k$-tape TM $\iff$ 2$k$ steps of 1-tape TM

**Result:** $L$ is $T$-recognizable iff some $k$-tape TM recognizes it.
Theorem: \( \text{Non-DET. TM} \xrightarrow{} (D)-\text{TM} \)

Given a N-TM, simulate by D-TM

\[ S_N : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}) \]

Accept by N-TM = some branch (of the tree of possible computations allowed by the choice \( S_N \)) leads to Accept state

\[ b = |Q| \times |\Gamma| \times |\{L, R\}| \]

D-TM uses 3-tapes

INPUT TAPE

SIMULATION TAPE

ADDRESS TAPE

Key: Use BFS (not DFS) of tree // obj: to search for Accept state

Eg. Choice `2 6 4 7` config resulting after choices \( \# \) ed 2, 6, 4, 7 in first 4 steps.

Use ADDRESS TAPE as a counter - of node-labels- in BFS-traversal of tree
Th: \( L \) is \( \Sigma \)-Recognizable iff some \( \Sigma \)-TM recognizes it

Defn: \( \Sigma \)-TM is a decider iff \( \forall w \in L \), all branches of (for a language \( L \)) \((w \notin L) \land \text{tree halts}\)

Cor: \( L \) is decidable iff some \( \Sigma \)-TM decides it

---

\[ \text{ENUMERATER} = \text{TM} + \text{printer} \]

\[ L(E) = \{ w \mid w \text{ printed} \} \]

Theorem: Long \( L \) is \( \Sigma \)-recognizable \( \iff \) some \( E \) enumerated long \( L \)

\( L \) is arbitrary, cannot assume \( L \) is already "enumerated"

\( (\iff) \) i.e., show that, given \( E \), build a TM that given any \( w \), if \( w \in L \) accepts \( w \), if \( w \notin L \) rejects \( w \) or loops

M: On input \( w \):

1) Runs \( E \). Each string \( s \) printed, compare \( s \) with \( w \).
   - If \( s = w \), accept \( w \)
   - Else loop

(\( \implies \)) Given TM \( E \), build \( E \) for \( L \)

(\( \iff \)) Given \( w \), ignore it.

1) Loop for \( i = 1, 2, 3, \ldots \)
   - Run TM for \( i \) steps on inputs \( s_1, s_2, s_3, \ldots \)
   - If any input is accepted by TM, print it (i.e., that \( s \))
ALGORITHMS:

CHURCH-TURING THESIS

\[ x^2 - 8x + 15 = 0 \]
\[ x^2 - 8x + 13 = 0 \]

Given a polynomial \( p \), give an algorithm to tell if \( p \) has integral roots.

1900: Hilbert's 10th problem
1970: This is UNSOLVABLE

\[ D_1 = \{ p \mid p \text{ is a polynomial with an integral root} \} \]

\( D_1 \) is recognizable.

\( M_1: \) \( p \) is input

1) Evaluate \( p \) with \( x \) set to 0, -1, +1, -2, +2, ...  
   If \( p \) evaluates to 0, accept.

roots of \( p \) over 1 variable \( \in \left[ -k \frac{c_{\text{max}}}{c_n}, k \frac{c_{\text{max}}}{c_n} \right] \)

\( k = \# \text{ Terms in } p, \quad c_{\text{max}} = \text{coeff with highest mod value}, \quad c_n = \text{coeff of } x^n \)

\[ D = \{ p \mid p \text{ is a polynomial with an integral root} \} \]

\[ 27x^2y^2z^2 - 11x^4y^4z - \cdots = 0 \]
Build M (like M₁) that recognizes D.

D is T-recognizable (only)
D₁ is T-decidable

**Descriptions of TMs**

1) formal description (7-tuple) //assembly lang,
2) Implementation level (English prose describing TMs steps)
3) high-lvl (English prose, ignoring implementation model)

• How expressive are Lars (ie # sets of strings) i.e.
  describing problems?

Very. Strings can ENCODE 'anything', eg: polynomial,
graphs, grammar, automata,

\[ \langle 0 \rangle = \text{encoding of object } 0 \]

\[ A = \{ \langle G \rangle \mid G \text{ is connected undirected graph} \} \]

TM decider of A:

M: Input \( w = \langle G \rangle \):

1) Check for correctness of encoding
2) Mark some node
3) Loop:
   \( \forall x \text{ unmarked}, \text{mark } x \text{ if } x \text{ has edge to marked node} \)

   until: no \( x \text{ gets marked} \)
Implementation-level details

e.g. Step 0) Encodings. \( \langle G \rangle = (1, 2, 3)(1, 3, 2) \)