Turing Machines

There's more here than meets the eye!

I ❤ SPAM!

Features:

1. Read and write on the infinite tape.
2. Read/write head can move left and right.
3. Unique accept and reject states take immediate effect.
4. Deterministic.

I wonder if it's a relative of mine—It only has one eye!
A Turing Machine is a 7-tuple \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \) where \( Q, \Sigma, \Gamma \) are finite sets and:

1. \( Q \) is the set of states.
2. \( \Sigma \) is the input alphabet which may not contain the "\( \_ \)" blank symbol.
3. \( \Gamma \) is the tape alphabet. \( \Sigma \subseteq \Gamma \), \( \epsilon \in \Gamma \). \( \leftarrow \) deterministic
4. \( \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{ L, R \} \) is the transition function.
5. \( q_0 \) is the start state.
6. \( q_{\text{accept}} \) is the accept state.
7. \( q_{\text{reject}} \) is the reject state.
Computing with a TM

- The TM receives its input \( w \in \Sigma^* \) on the tape beginning at the left end of the tape. Rest of tape blank.

- The tape head begins on the left end of the tape.

- Computation proceeds according to \( S \)-function.

- Attempt to go off left edge of tape keeps head at left edge of tape. (Ok Sipser, we'll humor you!)

- On \( q_{accept} \), \( M \) halts and accepts.

- On \( q_{reject} \), \( M \) halts and rejects.

- \( M \) may run forever!
A Very Important Distinction...

Definition: A language $L$ is said to be recursively enumerable (r.e.) if there exists a TM that enters $\text{accept}$ $\forall w \in L$ and either enters $\text{reject}$ or runs forever $\forall w \notin L$.

aka "enumerable" or "semi-decidable"

Definition: A language $L$ is said to be recursive if there exists a TM that enters $\text{accept}$ $\forall w \in L$ and enters $\text{reject}$ $\forall w \notin L$.

aka "decidable"

Question: Are the recursively enumerable languages equal to the recursive languages?
$L = \{ a^i b^i c^i \mid i \geq 1 \}$ is recursive!

$\Sigma = \{ a, b, c \}$
input alphabet

$\Gamma = \{ \varepsilon, a, b, c, \hat{a}, \hat{b}, \hat{c} \}$
tape alphabet

Diagram:
- Start state: $q_{\text{start}}$, transitions: $a \rightarrow \hat{a}, R$
- $q_{\text{rewind}}$, transitions: $a \rightarrow a, L$
- $q_{\text{skip a's}}$, transitions: $b \rightarrow b, R$
- $q_{\text{skip b's}}$, transitions: $c \rightarrow \varepsilon, L$
How About...

$L_1 = \{ \text{ }0^p \mid \text{ } p \text{ is prime}\}$

$L_2 = \{ b_i \# b_{i+1} \mid b_j \text{ is the binary representation of the number } j \}$

$L_3 = \{ b_i \# b_j \# b_i.j \}$

Are all languages recursive?
Adding Features to Turing Machines

- Multiple Tracks

```
 spam!

 |  |  |  |  |
 |  |  |  |  |
 |  |  |  |  |
 |  |  |  |  |

↑

↑

spam!

|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
```

21" color display
Flatbed scanner
DVD drive
- Two-way infinite tape

- Multiple tapes with independent heads
• Nondeterminism

\[ \delta(q_i, a) = \{ (q_{j_1}, b, L), \ldots, (q_{j_k}, d, R) \} \]

Let \( r \) denote the max # of nondet. choices.

\[
\begin{align*}
\text{spam!uuu} & \quad \cdots \sim \text{input} \hspace{1cm} \text{tape} \\
\uparrow & \\
\text{spin!u} & \quad \cdots \sim \text{work} \hspace{1cm} \text{tape} \\
\uparrow & \\
13212u & \quad \cdots \sim \text{branch enumeration} \hspace{1cm} \text{tape}
\end{align*}
\]
First, let's find a language that's not even recursively enumerable!

- TM encodings

- $L_d$
TMs as computers of functions and number generators

Hey, this slide is empty!
Question: If $L$ is recursive, is $\overline{L}$ recursive?

Question: If $L$ is r.e., is $\overline{L}$ r.e.?

Question: If $L$ and $\overline{L}$ are r.e. what else can be inferred?
$L_{TM} = \{ <M, w> | \text{TM } M \text{ accepts string } w \}$

$L_{Halt} = \{ <M, w> | \text{TM } M \text{ halts on string } w \}$
$L\emptyset$ for DFA's, PDA's and TM's

$L\emptyset, \text{DFA} = \{ \langle D \rangle \mid D \text{ is a DFA and } L(D) = \emptyset \}$

$L\emptyset, \text{PDA} = \{ \langle P \rangle \mid P \text{ is a PDA and } L(P) = \emptyset \}$

$L\emptyset, \text{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}$

which of these are recursive? a.k.a. "decidable"
Rice's Theorem

Every nontrivial property of the languages accepted by TMs is undecidable.

How do we formally define "nontrivial property"?

Ex

$L_{\text{infinite}} = \{ <m> | L(m) \text{ is infinite} \}$

$P = \{ L_1, L_2, L_3, \ldots \}$

A set of all infinite languages accepted by TMs (all infinite r.e. languages)

Def A property of the r.e. languages is any set $P$ of r.e. languages.

Def Property $P$ is nontrivial if $P \neq \emptyset$ and $P \neq \text{all r.e. langs}$.
An Undecidable Tiling Problem

- Given finite number of different types of tiles.
- Given an infinite number of tiles of each type.
- No rotating of tiles allowed!

An Origin Tile must be placed here.
Claim: If we could solve the tiling problem then we could solve the halting problem! (therefore the tiling problem is undecidable!)

\( <M, w> \) --- input to halting problem

\( S \) --- spam

\( (A, S, T, S, Q_0, Q_{accept}, Q_{reject}) \)

Build the following tile types:

1. \( (q_0, s) \)

2. \( a \quad \forall \ a \in T \)

3. \( (p, o) \quad \delta(q_0, a) = (p, b, R) \)

4. \( (p, o) \quad \delta(q_0, a) = (p, b, L) \)
\[ \delta(q_0, s) = (q_1, t, R) \]

What goes here?

\[ a \in \Gamma \]

Origin tile!
State Minimization for TMs

Theorem: Every recursively enumerable language is accepted by a TM with only 4 states (5 with explicit "reject") but determining if 3 (4 I) or fewer states suffice is undecidable.

Are you serious about this?!
claim: Every recursively enumerable language is accepted by a TM with only 4 states. Amazing!

Assume that TM's don't move their tape heads...

[Diagram of Turing machine transitions:]
- $q_0 \xrightarrow{a \rightarrow b, s} q_5$
- $q_5 \xrightarrow{b \rightarrow f, s} q_{42}$
- $q_5 \xrightarrow{c \rightarrow d, s} q_8$
- $q_8 \xrightarrow{d \rightarrow g, s} q_{23}$
- $q_{23} \xrightarrow{s \rightarrow 2, s} q_{accept}$

read
write
stay put
Ok, but TM's can move their heads!
TMs as Generators

Generator  TMs

\[ 011\#110\#\ldots \]

↑  right-only

output tape

\[
\vdots \\
\vdots
\]

work tapes

- Generator halts if language is finite, runs forever if language is infinite.
- Prints out every word in the language in finite time.
Theorem 1:

$L$ is recursive iff
$L$ is generated in lexicographic order by some generating TM.
Theorem 2:

$L$ is recursively enumerable iff

$L$ is generated by some generating $\exists$ TM.

or "enumerated"

In no particular order!
The Recursion Theorem!
(How to construct a program that prints itself out, and why it's useful)

Budget

First, the ^Quote-O-Matic!

\[ \text{input: word } w \]
\[ \text{quote-O-matic chugs!} \]

\[ \text{output: code for a TM } \]
\[ M \text{ s.t. when run, } M \]
\[ 1. \text{ erases its tape} \]
\[ 2. \text{ stamps } w \text{ out on tape} \]
\[ 3. \text{ halts} \]
Now, the NEW top-of-the-line
Super Quote-and-Shift-O-Matic! (For serious TM enthusiasts ages 18 to 218)

Input: word w

Through the miracle of Turing Machine technology!

Output: The previous TM (erases tape, prints w on tape, 4 halts) followed immediately by w.

Wow! The Super Quote-and-Shift O-Matic! I want one now!!
A Self-Printing TM!

Objective: Construct a TM that, when run, ignores its input, prints its own TM code on the tape, and halts.
The Recursion Theorem

Let $T$ be a TM that takes $\langle M, w \rangle$ as input and computes some function on $\langle M, w \rangle$.

It is possible to construct a TM $R$ s.t. when $R$ is run on input $w$...

1. $R$ prints itself out on the tape and shifts $w$ over.
   ( $\langle R, w \rangle$ now on tape)

2. $R$ now computes/perform $T$ on $\langle R, w \rangle$.

Uh, um, excuse me Ran... are you feeling entirely normal? Did you sleep last night? Perhaps you need a nice relaxing sabbatical in Bora Bora.
why it's useful...

Example 1

Recall $L_{TM} = \{ \langle M, w \rangle \mid M$ accepts $w \}$

Assume $L_{TM}$ is decidable.
Let $M_{TM}$ be a TM for $L_{TM}$.
Recall...

\[ L_{\emptyset, \text{in}} = \{ <M> \mid \lambda(M) = \emptyset \} \]

\[ L_{42} = \{ <M> \mid \lambda(M) \text{ contains exactly 42 words} \} \]

The value "42" is key!
The **Doozy!**

*“Doozy” is a technical term.*

\[ MIN = \{ <M> | \text{There is no TM with a shorter description that accepts } L(M) \} \]

Is \( MIN \) recursive?

No!! It's not even r.e.!!

Assume \( MIN \) is r.e.

Let \( G \) be a generator for it.
• What is a fixed point of a function?

Theorem:

Let $f : E^* \rightarrow E^*$ map TM codes to TM codes and be computable. Then $\exists$ a TM $M$ s.t. $t(<m>) = <m'>$ where $M$ and $M'$ are equivalent.
Unrestricted Grammars

- Recall that
  \[ L = \{ 0^{2^i} \mid i \geq 1 \} \]
  terminal symbol is NOT context-free

- consider this weird unrestricted grammar ...

1. \[ S \rightarrow AC \cdot B \]
2. \[ C \rightarrow CC \cdot C \]
3. \[ CB \rightarrow DB \]
4. \[ CB \rightarrow E \]
5. \[ D \rightarrow DG \]
6. \[ AD \rightarrow AC \]
7. \[ E \rightarrow EO \]
8. \[ AE \rightarrow e \]

I like being in grammars! It's fun & there are lots of neat variables to meet!
One More thing...

Amazingly neat

... about Turing Machines

Claim: The languages generated by unrestricted grammars are exactly the recursively enumerable languages.

\[ S \rightarrow ABabcD \mid cDe \]

\[ Ba \rightarrow aaB \]

Claim 1: Every unrestricted grammar has a corresponding TM.

\[ \text{may not halt on strings not generated by grammar!} \]

Claim 2: Every TM has a corresponding unrestricted grammar.
Claim 1: Every unrestricted grammar has a corresponding TM
Claim 2: Every TM has a corresponding unrestricted grammar

Proof: \[ M = (Q, \Sigma, \Gamma, s, q_{\text{start}}, q_{\text{accept}}, q_{\text{reject}}) \]

- Input alphabet: \( \Sigma \neq \emptyset \)
- Tape alphabet: \( \Sigma \subseteq \Gamma \)