Today: A closer look at TMs

- Some basic subroutines
- The precise model doesn’t matter b/c simulation
- TMs are code
- TMs can run code (universal TMs)

Recall:

- Scan right until at a cell w/ $ \\
  while w≠$ or 0 \\
  i ← i+1 \\

- Shift everything right.

until 1 \\
noop \\

- Extra TM syntax:
- Ready tape, writes to tape
- Changes state, moves L→R
- keep a binary counter
  - idea: find the least significant bit
  - flip it &
    - if 1, move to "carry" state.
    - or if carrying, move on.

- "assume" that I can say things like
  the TM will find the first 0
  \[ \Rightarrow \text{failsafe: on } \square \]
  move to no 0 found state.

\[ \times \]

Many variations on TMs:
- m-th class: defined TM to have 1 tape, infinite to the right.
  \[ \square \text{ what if you move } \leftarrow \text{ from } \square ? \]
  crash!
  But... don't worry: always first shift everything \( \rightarrow \)
  so doesn't really matter.

- Some people define TMs to have multiple tapes.
  One workaround: expand \( \Gamma \) to have "columns"
one workaround: expand $\Gamma$ to have "columns" of symbols

- \[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\rightarrow
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

- \[
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
\end{array}
\rightarrow
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\end{array}
\]

- \[
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
\end{array}
\rightarrow
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{array}
\]

- \[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
\end{array}
\rightarrow
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

To simulate multiple tapes:
for each tape, scan for \( A \) for that tape & execute 1 step

\[\begin{array}{cccc}
0 & 1 & 1 & 0 \\
\end{array}\]
\[\begin{array}{cccc}
0 & 1 & 0 & 1 \\
\end{array}\]

What if I need to write more to first tape?
Shift everything else \( \rightarrow \)

A lot of this is bookkeeping on how your data is being stored

-----

Aside: \( L = \{ \omega \mid \omega \text{ is a regular } \} \) is not regular.
but it is context-free. --- exercise

Q: is \( L = \{ \omega \mid \omega \text{ is a TM}\} \) decidable by a TM?
- understand what "\( \omega \text{ is a TM}\)" means
- if a TM can tell if "\( \omega \text{ is a TM}\)"
- if a TM can tell if \( w \) is a TM, can we execute the TM described by \( w \)?

Universal Turing Machines

Suppose given

\[
M = (Q, \Gamma, \text{start}, \text{acc}, \text{rej}, \delta)
\]

create encoding over some other alphabet

\[\{0, 1, [], \cdot, \#, 1/2, \square\}\]

make some arbitrary encoding decisions

\[
\begin{align*}
\langle \text{start} \rangle & = 111 \quad \text{log 101 bits} \\
\langle \text{acc} \rangle & = 001 \\
\langle \text{rej} \rangle & = 000 \\
\langle \text{q} \rangle & = \langle p \rangle \cdot \langle a \rangle \cdot \langle b \rangle \cdot \langle q \rangle \cdot \langle s \rangle \\
\langle s \rangle & = \text{concat of all transitions.}
\end{align*}
\]

\[
\langle M \rangle = [\langle \text{rej} \rangle \cdot \langle \text{start} \rangle, [\langle s \rangle]]
\]

this allows us to sketch a TM that takes \( \langle M \rangle \cdot \langle w \rangle \) and then \( \text{run } M \text{ on input } w \).
UTM w/ 3 tapes
- input tape (won't modify it)
- state tape remembers current state of M
- work tape all computations here.

- Initialization:
  - copy \( \langle w \rangle \) to work tape.
  - mark first symbol of \( \langle w \rangle \):
    \[ \theta \rightarrow \theta, 1 \rightarrow 1 \]
  - copy \( \langle \text{start} \rangle \) to state tape

- Loop:
  - find marked char on work tape.
  - gives us \( \langle a \rangle \)
  - scan input tape for \( [[\langle p \rangle \cdot \langle a \rangle \cdot \langle b \rangle \cdot \langle q \rangle \cdot \langle a \rangle]] \)
  - change \( \langle a \rangle \) to \( \langle b \rangle \) on work tape
  - change \( \langle p \rangle \) to \( \langle q \rangle \) on state tape
  - mark the beginning of the reading of next symbol
  - if state tape is \( \langle \text{acc} \rangle \) or \( \langle \text{ rej} \rangle \)
    accept / reject accordingly.

---

TM for simulating assembly + RAM?

given: Encoding of set of assembly instructions.
We are all Universal Turing Machines.

Suppose ETM does not reject. Then ETM accepts (M').

Suppose ETM rejects (M'). Then ETM accepts (M).

Thus ETM accepts (M).

We keep track of what the result is by counting on another tape.

April 14, 2010

00:00

I am trying to figure out what is going on.
\[
\text{SELF HALT} = \{ \langle M \rangle \mid M \text{ halts on } \langle M \rangle \}
\]

Suppose \( \exists \text{ TM } M_{sh} \) deciding \( \text{SELF HALT} \).

\[
M_{sh} \rightarrow \overline{M_{sh}} \text{ by changing all transitions to acc to a new loop state}
\]

… all transitions to rej to acc.

\[\overline{M_{sh}} \text{ accepts } \langle M \rangle \iff \overline{M_{sh}} \text{ rejects } \langle M \rangle \iff M \text{ does not halt on } \langle M \rangle\]

Plug in \( \overline{M_{sh}} \)

\[\overline{M_{sh}} \text{ accepts } \langle \overline{M_{sh}} \rangle \iff \overline{M_{sh}} \text{ does not halt on } \langle \overline{M_{sh}} \rangle.
\]

Contradiction: \( \neg \exists \text{ TM deciding } \text{SELF HALT!} \).