Midterm - Tuesday, Feb 8
Quiz 7 - today
HW 4 - due Saturday
HW 5 - due Monday 11:00 am
Review Session - Monday 7pm
Example: Turing machine that enumerates the set \( \{ n \mid n \text{ even} \} = \{ 0, 2, 4, 6, \ldots \} \)

\( \Sigma = \{ 1 \} \quad \Gamma = \{ 1, \# \} \) - Starts on blank tape

Turing machine: \((\Gamma, \Sigma, Q, q_0, \delta, Q_{\text{halt}}, \text{Out})\)

\( Q_{\text{halt}} = \emptyset \)

In my example: \( \text{Out} = q_0 \)

\[ \begin{array}{c}
q_0 \\
\# \# \# \# \# \\
\uparrow \\
q_1 \\
\# \# 1 \# \\
\uparrow \\
q_1 \\
\# \# \# \# \\
\uparrow \\
q_2 \\
\# 1 \# \\
\uparrow \\
q_2 \\
\# \# \# \# \\
\uparrow \\
q_0 \\
\# 1 \# \\
\uparrow \\
q_0 \\
\# \# \# \# \\
\uparrow \\
\scriptsize{\text{output}} \\
\scriptsize{\text{halt}}
\end{array} \]
3 type of machines

\[ Q_{\text{halt}} = \{ \text{acc}, \text{ rej} \} \] - (semi)decides a relation

\[ Q_{\text{halt}} = \{ \text{gout} \} \] - computes or partial computes a function

\[ Q_{\text{halt}} = \emptyset \] - enumerates a set or a relation.
Training machine that copies a string

Start \[ \# w \# \ 
End with \[ \# w \# w \#

\[ \# 0100 \# \ \rightarrow \ 
\# 0100 \# 0100 \#
\]

\[ \Sigma = \{ 0, 1 \} \ 
\Gamma = \{ 0, 1, \#, \}' \] \#

\underline{Strategy}

\[ g_0: \ At \ a \ symbol \ to \ be \ copied \ next, \]
\[ (\text{If } \# , \ \text{done} - \ \text{Add prime} \)
\[ + \ \text{remember } 0 \text{ or } 1 \ (i) + \ \text{move right} \]
\[ g_{0,1}, g_{0,2} \} \ \text{remember } i = 0 \text{ or } i = 1 \ (\text{to be copied}) \]
\[ g_{1,1}, g_{1,2} \} + \text{scanning rightward to 2nd } \#
\[ + \ \text{copy the symbol } i \]
\[ g_8, g_{8,2} - \text{scan leftward back to } 0 \text{ or } 1 \]
\[ g_{8,1}, g_{8,2} - \text{change back to } 0 \text{ or } 1 \]
\[ g_4, g_5 - \text{handle the next } 0 \text{ or } 1 \] (Similarly to \( g_0 \))
Theorem: the function \( \omega \mapsto \omega \omega \) is Turing computable.

Theorem \( \exists \ \omega \# \omega : \omega \in \Sigma^* \) is Turing decidable

\( \exists \ \omega \omega : \omega \in \Sigma^* \) " " " " "

(Non-negative)

Integers - binary notation - permit leading 0's

Theorem: \( n \mapsto n + 1 \) is Turing computable

\( 1100 \mapsto 1101 \quad 12 + 1 = 13 \)

\( 1011 \mapsto 1100 \quad 11 + 1 = 12 \)

Scan to end (right end) -

Change 1's to 0's

Change 1st encountered 0 to 1

Then change nothing else. \( \Rightarrow \) Scan left if \( \# \)

\( \# 10000 \)
Theorem: \( n, m \mapsto n \cdot m \) is Turing computable

\[
\begin{align*}
\text{Loop} & \quad \text{if } m = 0, \text{ output the left number } n \\
& \quad \text{Subtract } 1 \text{ from } n \\
& \quad \text{Add } 1 \text{ to } n \\
\text{End Loop}
\end{align*}
\]

Theorem: \( n, m \mapsto n \cdot m \) is Turing computable

Overall idea:

\[
\begin{align*}
0 \# n \# n \# m \# \\
\text{(same size as } n)
\end{align*}
\]

where the answer will be

\[
\begin{align*}
\text{Loop} & \quad \text{If } m = 0, \text{ output } N \\
& \quad \text{Set } n' = n \text{ (copy)} \\
& \quad \text{Add } n' \text{ to } P \\
\text{End of loop}
\end{align*}
\]
So far

if-then / conditional branching

loops

basic copying, arithmetic operations

Still needed Arbitrary memory access

e.g. Arrays of value v

Accessing data by its address

w

v

w

$ w \# v \$ w', \# v\$

Value v is stored at location w

(w is the variable named v)

From this: Church-Turing thesis.