Theory of Computation
Chapter 3

Guan-Shieng Huang

Mar. 24, 2003

Universal Turing Machines

- A Turing machine is a special hardware to do computation.
  A modern computer can load different programs and do the corresponding computational tasks.
  Can a Turing machine act as a universal computational device?

- Universal Turing Machines
  The input of a universal TM U is $M; x$, where M is the description of a TM, x is its input. We can imagine that U interprets M and executes M with the input x.
  Written as
  $$U(M; x) = M(x).$$

Halting Problem

Given the description of a TM M and its input x, will M halt on x?

$$H = \{M; x |\ M(x) \neq \nearrow\}.$$

(Note: A universal TM is implicitly assumed.)

Proposition 3.1

H is recursively enumerable (R.E.).

1. R.E. $\Rightarrow$ there is a TM D such that

$$D(M;x) = \begin{cases} \text{“yes”} & \text{if } M(x) \neq \nearrow \\ \nearrow & \text{otherwise.} \end{cases}$$

2. The universal TM U can serve this task. We only need to modify U s.t.

   when $M(x)$ halts, U terminates at “yes”.

Theorem 3.1

H is not recursive.

1. recursive $\Rightarrow$ there is a TM M_H such that

$$M_H(M;x) = \begin{cases} \text{“yes”} & \text{if } M(x) \neq \nearrow \\ \text{“no”} & \text{if } M(x) = \nearrow. \end{cases}$$

2. Proof By Contradiction.

   Suppose we have such a TM M_H. Construct a TM $D(x)$ as

   (a) On input x, D first simulates M_H on input $x;x$.

   (b) If M_H accepts $x;x$, D diverges (e.g. moves its cursor to the right of its string forever).

   (c) If M_H rejects $x;x$, D halts.

3. That is,

$$D(x) : \text{if } M_H(x;x) = \text{“yes” then} \nearrow \text{else “yes”}.$$

4. What is $D(D)$?

   (a) If $D(D) = \nearrow$:

   Step (b) $\Rightarrow M_H(D;D) =$ “yes” $\Rightarrow D(D) \neq \nearrow$.

   (b) If $D(D) \neq \nearrow$:

   Step (c) $\Rightarrow M_H(D;D) =$ “no” $\Rightarrow D(D) = \nearrow$.

There are countably-many TMs.

There are uncountably-many languages.

Hence, there exists a language that is not recursive.

Reduction

To show that Problem A is undecidable, we establish that if there were an algorithm for Problem A, then there would be an algorithm for HALTING H, which is absurd.

Proposition 3.2

The following languages are not recursive.

1. $L_a = \{M|\ M$ halts on all inputs$\}$.
2. $L_d = \{M; x; y|\ M(x) = y\}$.
3. $L_b = \{M; x|$ there is a y such that $M(x) = y\}$.
4. $L_c =$
   $\{M; x|$ the computation M on input x uses all states of $M\}$.

$L_a = \{M | \; M \text{ halts on all inputs}\}$.

Reduce HALTING to this problem.

Given $M; x$, we construct

$$M'(y) : M(x).$$

Hence M' halts on all inputs if and only if M halts on x.

$L_d = \{M; x; y|\ M(x) = y\}$.

Given $M; x$, we construct

$$M'(x') : \text{if } (M(x) \text{ halts}), \text{ then Output } \epsilon.$$

Hence $M'; x'; \epsilon \in L_d$ if and only if M halts on x.

$L_b = \{M; x|$ there is a y such that $M(x) = y\}$

The meaning of this problem is not clear.

- $M(x) = \{$"yes", "no", "halt", $\nearrow\}$.
- Does M halts on x?
- $\{M; x|\ M(x) = c\}$ for some constant string c.

Proposition 3.3

If L is recursive, then so is $\bar{L}$.

1. Let D be the TM that decides L:

$$D(x) = \begin{cases} \text{“yes”} & \text{if } x \in L \\ \text{“no”} & \text{if } x \notin L. \end{cases}$$

2. Construct D' such that

$$D'(x) = \begin{cases} \text{“yes”} & \text{if } D(x) = \text{“no”} \\ \text{“no”} & \text{if } D(x) = \text{“yes”}. \end{cases}$$

   Then D' decides $\bar{L}$.

Proposition 3.4

L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.

1. L is recursive $\Rightarrow$

$$D_L(x) = \begin{cases} \text{“yes”} & \text{if } x \in L \\ \text{“no”} & \text{if } x \notin L. \end{cases}$$

2. $\bar{L}$ is recursively enumerable

$$M_{\bar{L}}(x) = \begin{cases} \text{“yes”} & \text{if } x \in \bar{L} \text{ or } x \notin L \\ \nearrow & \text{if } x \notin \bar{L} \text{ or } x \in L. \end{cases}$$

3. L is recursively enumerable

$$M_L(x) = \begin{cases} \text{“yes”} & \text{if } x \in L \\ \nearrow & \text{if } x \notin L. \end{cases}$$

4. Given D_L, we construct M_l and $M_{\bar{L}}$ as follows.

$$\begin{aligned} M_L(x): \quad & \text{if } D_L(x) = \text{“yes” then “yes”} \\ & \text{else } \nearrow. \\ M_{\bar{L}}(x): \quad & \text{if } D_L(x) = \text{“no” then “yes”} \\ & \text{else } \nearrow. \end{aligned}$$

5. Given M_L and $M_{\bar{L}}$, we construct D_L as

$$D_L(x) = \begin{cases} \text{if } (M_L(x) = \text{“yes”}) \text{ then “yes”} \\ \text{if } (M_{\bar{L}}(x) = \text{“yes”}) \text{ then “no”}. \end{cases}$$

Enumerator

$$E(M) = \{x | (s, \triangleright, \epsilon) \stackrel{M*}{\rightarrow} (q, y \sqcup x \sqcup \epsilon) \text{ for some } q, y\}.$$

That is, $E(M)$ is the set of all strings x such that during M's operation on empty string, there is a time at which M's string ends with $\sqcup x \sqcup$.

Proposition 3.5

L is R.E. if and only if there is a machine M such that $L = E(M)$.

1. Suppose $L = E(M)$. We construct a TM M' that accepts L as follows.

   $M'(x)$: if x appears in the string of $M(\epsilon)$ then "yes"
   else $\nearrow$.

   Then $M'(x) =$"yes" iff $x \in E(M) = L$.

2. Suppose L is R.E. Then we have a TM M s.t.

   $$M(x) = \begin{cases} \text{“yes”} & \text{if } x \in L \\ \nearrow & \text{if } x \notin L. \end{cases}$$

   We need to construct a TM M' such that $E(M') = L$.

$M'(\epsilon)$ works as follows.

(a) For $i = 1, 2, 3, \ldots$, simulate M on the i first inputs, one after the other, and each for i steps.

(b) If at any point M would halt with "yes" on one of these i inputs, say x, then M' write $\sqcup x \sqcup$ at the end of its string before continuing.

Theorem 3.2: Rice's Theorem

Suppose that $\mathcal{C}$ is a proper, non-empty subset of the set of all R.E. languages. Then

"Given a TM M, is $L(M) \in \mathcal{C}$" is undecidable.

1. A TM is a string, and a string is a TM.
2. WLOG, we assume that $L \in \mathcal{C}$ & $\emptyset \notin \mathcal{C}$. We reduce HALTING to this problem. Given $M; x$, we construct

   $$M'(y) : \text{if } (M(x) \text{ halts}) \text{ then } M_L(y).$$

   Then $M; x \in H$ iff $L(M') = L$ (and $M; x \notin H$ iff $L(M') = \emptyset$). That is, $L(M') \in \mathcal{C}$ iff $M; x \in H$.

Recursive Inseparability

Two disjoint languages L_1 and L_2 are recursively inseparable if there is no recursive language R such that $L_1 \cap R = \emptyset$ and $L_2 \subset R$. (That is, $\bar{R}$ contains L_1 and R contains L_2.)

Theorem 3.3

Define $L_1 = \{M \mid M(M) = \text{“yes”}\}$ and $L_2 = \{M \mid M(M) = \text{“no”}\}$. Then L_1 and L_2 are recursively inseparable.

1. Suppose that recursive language R separates them. Thus, $R \cap L_1 = \emptyset$ and $L_2 \subset R$.

2. Consider the M_R that decides R. “What is $M_R(M_R)$”?

   (a) If $M_R(M_R)$=“yes”, then $M_R \in L_1$ and $M_R \notin R$, and then $M_R(M_R)$=“no”.

   (b) If $M_R(M_R)$=“no”, then $M_R \in L_2$ and $M_R \in R$, and then $M_R(M_R)$=“yes”.

Hence, this R is absurd.

Corollary

Let $L'_1 = \{M | \ M(\epsilon) = \text{"yes"}\}$ and $L'_2 = \{M | \ M(\epsilon) = \text{"no"}\}$. Then L_1 and L_2 are recursively inseparable.

1. We reduce L_1 and L_2 to L'_1 and L'_2. Given any M, we construct $M'(x)$ simply as $M(M)$. Hence,

   $$M(M)=\text{"yes" iff } M'(\epsilon)=\text{"yes"}$$

   and

   $$M(M)=\text{"no" iff } M'(\epsilon)=\text{"no"}.$$

2. If L'_1 and L'_2 are recursively separable, then so do L_1 and L_2.