Harvard CS 121 and CSCI E-121

Lecture 16: Turing's Theorem

Harry Lewis

October 29, 2013

- Reading: Sipser §4.2, §5.1.

A Universal Turing machine

Theorem: There is a Turing machine U, such that when U is given $\langle M, w \rangle$ for any TM M and w, U produces the same result (accept/reject/loop) as running M on w.

Proof: Initially,

- First tape contains $\langle M \rangle$, including in particular its transition function δ_M.
- Second tape contains $\langle w \rangle$.
- Third tape contains $\langle q_{\text{start}} \rangle$.
- Simulate steps of M by multiple steps of U.

(Brief return to implementation description.)

$\Rightarrow$ Turing machines can be "programmed".

Consequences of the Existence of Universal Turing Machines

- **Corollary:** $A_{TM} = \{\langle M, w\rangle : M \text{ accepts } w\}$ is Turing-recognizable.
- **Corollary:** $HALT_{TM} = \{\langle M, w\rangle : M \text{ eventually halts on } w\}$ ("The Halting Problem") is Turing-recognizable.
- **Corollary:** "The Turing Machines that halt on some input are an r.e. set" (What does this mean?)
- **Q:** Are these sets decidable?
- **Q:** Are there undecidable languages?

How Can We Find an Undecidable Language?

- By the Church–Turing thesis, such a language has a membership problem that cannot be solved by any kind of algorithm

- We know such languages exist, by a counting argument.
  - Every recursive language is decided by a TM
  - There are only countably many TMs
  - There are uncountably many languages

$\therefore$ Most languages are not recursive (or even r.e.)

Is every Turing-recognizable set decidable?

This would be true if there were an algorithm to solve

The Acceptance Problem:

Given a TM M and an input w, does M accept input w?

Formally, $A_{TM} = \{\langle M, w\rangle : M \text{ accepts } w\}$.

Completeness of A_{TM}

Proposition: If A_{TM} is recursive, then every r.e. language is recursive.

"A_{TM} is the hardest r.e. language."

- A_{TM} is said to be *r.e.-complete*, that is, it is a problem

(a) that is r.e. and

(b) to which every r.e. problem is reducible

Proof:

A simplifying detail: every string represents some TM

- Let Σ be the alphabet over which TMs are represented (that is, $\langle M \rangle \in \Sigma^*$ for any TM M)
- Let $w \in \Sigma^*$
- if $w = \langle M \rangle$ for some TM M then w represents M
- Otherwise w represents some fixed TM M_0 (say the simplest possible TM).
- (Of course for every T.M. there are infinitely many equivalent T.M.s)

Thm: A_{TM} is not recursive

- Look at A_{TM} as a $\Sigma^* \times \Sigma^*$ table answering every question:

	w_0	w_1	w_2	w_3	
M_0	Y	N	N	Y	
M_1	Y	Y	N	N	(WLOG assume
M_2	N	N	N	N	every string w_i
M_3	Y	Y	Y	Y	encodes a TM M_i)

- Entry matching (M_i, w_j) is Y iff M_i accepts w_j
- If A_{TM} were recursive, then so would be the diagonal D and its complement.
  - $D = \{w_i : M_i \text{ accepts } w_i\}$.
  - $\overline{D} = \{w_i : M_i \text{ does not accept } w_i\}$.
- But $\overline{D}$ differs from every row, i.e. it differs from every r.e. language. $\Rightarrow\Leftarrow$.

Unfolding the Diagonalization

- Suppose for contradiction that A_{TM} were recursive.
- Then there is a TM M^* that decides $\overline{D} = \{\langle M \rangle : M \text{ does not accept } \langle M \rangle\}$:
  - $M^*(\langle N \rangle)$ runs the decider for A_{TM} on $\langle N, \langle N \rangle \rangle$ and does the opposite.
- Run M^* on its own description $\langle M^* \rangle$.
- Does it accept?

  M^* accepts $\langle M^* \rangle$

  $\Leftrightarrow \langle M^* \rangle \in \overline{D}$

  $\Leftrightarrow M^*$ does not accept $\langle M^* \rangle$.
- Contradiction!

Alan Mathison Turing (1912-1954)

24 years old when he published *On computable numbers . . .*

Some More Undecidable Problems About TMs

- The Halting Problem: Given M and w, does M halt on input w?

  Proof:

  Suppose $\text{HALT}_{\text{TM}} = \{\langle M, w\rangle : M \text{ halts on } w\}$ were decided by some TM H.

  Then we could use H to decide A_{TM} as follows.

  On input $\langle M, w\rangle$,

  - Modify M so that whenever it is about to go into q_{reject}, it instead goes into an infinite loop. Call the resulting TM M'.
  - Run $H(\langle M', w\rangle)$ and do the same.

  Note that M' halts on w iff M accepts w, so this is indeed a decider for A_{TM}. $\Rightarrow\Leftarrow$.

”Given pro program P, will P terminate?” is undecidable!

- Proof:

Undecidable Problems, Continued

- For a certain fixed M_0:

  Given w, does M_0 halt on input w?

Undecidable Problems, Continued

- For a certain fixed M_0:

  Given w, does M_0 halt on input w?

What about:

- For a fixed M_0 *and* a fixed w_0, does M_0 halt on input w_0?

Further Undecidable Problems

- Given M, does M halt on the empty string?

  Proof by reduction:

  Suppose M_1 decided $\{ \langle M \rangle : M \text{ halts on } \varepsilon \}$.
  Then M_1 could be used to decide HALT_{TM}:

  Given $\langle M, w \rangle$,

  Construct $\langle M_w \rangle$, where M_w is a TM that writes

  w on the empty tape and then runs M.

  Then run M_1 on input $\langle M_w \rangle$

  M_1 halts on $\langle M_w \rangle \Leftrightarrow M_w$ halts on $\varepsilon \Leftrightarrow M$ halts on w

  But HALT_{TM} is undecidable. $\Rightarrow\Leftarrow$

"Co-X"

- For any property X that a set might have, a set S is **co-X** iff $\overline{S}$ has property X.
- For example, a co-finite set of natural numbers is a set that is missing only a finite number of elements.
- A co-regular language is . . . ?
- A co-recursive language is . . . ?
- What about a co-CF language?
- We proved earlier today:
  - A language is recursive if and only if it is both r.e. and co-r.e.

Non-r.e. Languages

Theorem: The following languages are not r.e.:

- $\overline{\text{A}_{\text{TM}}} = \{\langle M, w\rangle : M \text{ does not accept } w\}$
- $\overline{\text{HALT}_{\text{TM}}} = \{\langle M, w\rangle : M \text{ does not halt on } w\}$
- $\overline{\text{HALT}^{\varepsilon}_{\text{TM}}} = \{\langle M\rangle : M \text{ does not halt on } \varepsilon\}$

Proof: If these languages were r.e., then A_{TM}, HALT_{TM}, and $\text{HALT}^{\varepsilon}_{\text{TM}}$ would be both r.e. and co-r.e. and hence recursive.

Formalizing the Notion of Reduction

- L_1 "reduces" to L_2 if we can use a "black box" for L_2 to build an algorithm for L_1.

- A function $f : \Sigma_1^* \to \Sigma_2^*$ is computable if there is a Turing machine that for every input $w \in \Sigma_1^*$, M halts with just $f(w)$ on its tape.

- A (mapping) reduction of $L_1 \subseteq \Sigma_1^*$ to $L_2 \subseteq \Sigma_2^*$ is a computable function
  $f : \Sigma_1^* \to \Sigma_2^*$ such that, for any $w \in \Sigma^*$,

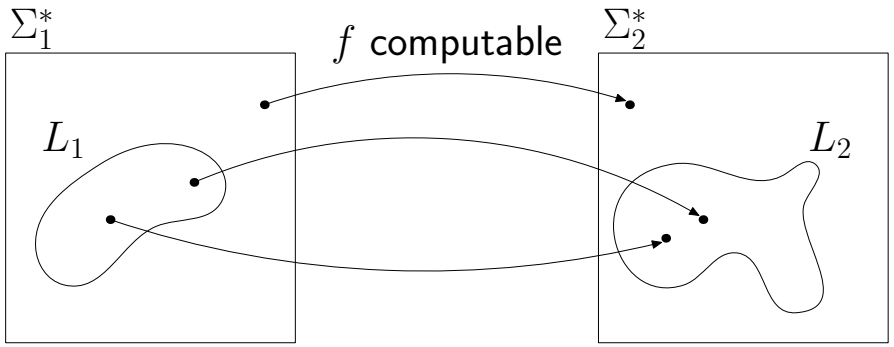
  $w \in L_1$ iff $f(w) \in L_2$

  We write $L_1 \leq_m L_2$.

Properties of Reducibility

Lemma: If $L_1 \leq_m L_2$, then

- if L_2 is decidable (resp., r.e.), then so is L_1;
- if L_1 is undecidable (resp., non-r.e.), then so is L_2.



Examples of Reductions from This Lecture

- For every Turing-recognizable L, $L \leq_m A_{TM}$.

- $A_{TM} \leq_m HALT_{TM}$.

- $HALT_{TM} \leq_m HALT^{\varepsilon}_{TM}$.