The Halting Problem for Turing Machines

Mark Greenstreet, CpSc 421, Term 1, 2006/07

- The Undecidability of A_{TM}
  - Diagonalizing Turing Machines
  - Turing Recongizable $>$ Turing Decidable
- Turing Unrecognizable Languages
  - How do we know if M is a decider?
  - The Halting Problem
  - Turing Unrecognizable Languages

Trying to Decide A_{TM}

- $A_{TM} = \{M\#w \mid \text{Turing machine } M \text{ accepts string } w\}$
  - A_{TM} is Turing recognizable:
    We constructed a Turing Machine, U that recognizes A_{TM} in the November 3 lecture.
  - U was not a decider – it would loop on input $M\#w$ if M loops on input w.
  - Can we make a Turing machine that decides A_{TM}?

    This machine must halt (either accept or reject) for all possible inputs.
- Assume that E is a TM that decides A_{TM}.
  We'll show that this leads to a contradiction on the next few slides.

A_{TM} Is Undecidable

- $A_{TM} = \{M\#w \mid M \text{ describes a TM that accepts string } w\}$
- Let D be a Turing machine that does not have $\#$ in its input alphabet. On input w, D does the following:
  - Appends $\#w$ onto its input tape to produce $w\#w$.
  - Runs E (the decider for A_{TM} as a "subroutine".
    - If E accepts $w\#w$, D rejects.
    - If E rejects $w\#w$, D accept.s.
- Now, run D with its own description as its input:
  - If E says that D accepts when run with D as input, then D rejects when run with D as input.
  - If E says that D rejects when run with D as input, then D accepts when run with D as input.
  - Either way, we have a contradiction.
- $\therefore E$ cannot exist.
  - There is no TM that decides A_{TM}.
  - A_{TM} is not Turing decidable.

Why is this Diagonalization?

- The set of all Turing machines is countable:
  - Turing Machines can be described by strings.
    - In the Nov. 3 lecture we described TMs using strings over the alphabet $\Sigma_{TM} = \{\texttt{0}, \texttt{1}, \texttt{(}, \texttt{,}, \texttt{)}\}$.
    - Not all strings are valid TM descriptions. Thus, $|TM| \leq |\Sigma_{TM}^*| = |\mathbb{N}|$.
  - For every $k \geq 3$ there is a valid TM with k states. Thus $|TM| \geq |\mathbb{N}|$.
  - We conclude that $|TM| = |\mathbb{N}|$.
- The set of all languages is uncountable.
  The set of all languages has size $2^{|\Sigma^*|} = 2^{|\mathbb{N}|}$.
- There are more languages than there are Turing machines.

  $\therefore$ There are languages that are neither Turing decidable nor recognizable.

Why is this Diagonalization?

- The set of all Turing machines is countable:
- The set of all languages is uncountable.

  The set of all languages has size $2^{|\Sigma^*|} = 2^{|\mathbb{N}|}$. For example, with $\Sigma = \{\mathtt{0},\mathtt{1}\}$ we have:

	ϵ	0	1	00	01	10	11	000	$\ldots$
L_0	R	R	R	R	R	R	R	R	$\ldots$
L_1	A	R	R	R	R	R	R	R	$\ldots$
L_2	R	A	R	R	R	R	R	R	$\ldots$
L_3	A	A	R	R	R	R	R	R	$\ldots$
L_4	R	R	A	R	R	R	R	R	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- There are more languages than there are Turing machines.

  $\therefore$ There are languages that are neither Turing decidable nor recognizable.

Constructing an Undecidable Language

- Consider the matrix where entry (i, j) is 1 iff Turing machine i accepts the string that encodes Turing machine j:

	M_0	M_1	M_2	$\dots$	M_{117}	M_{118}	M_{119}	$\dots$
M_0	∞	∞	∞	$\dots$	∞	∞	∞	$\dots$
M_1	A	A	A	$\dots$	A	A	A	$\dots$
M_2	R	R	R	$\dots$	R	R	R	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
M_{117}	A	∞	R	$\dots$	R	R	A	$\dots$
M_{118}	R	R	R	$\dots$	∞	∞	∞	$\dots$
M_{119}	R	A	∞	$\dots$	R	A	A	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- Let L_D be the language $\{M_i \mid \text{Turing machine } M_i \text{ rejects input } M_i\}$:

Constructing an Undecidable Language

- Consider the matrix where entry (i, j) is 1 iff Turing machine i accepts the string that encodes Turing machine j:

	M_0	M_1	M_2	$\ldots$	M_{117}	M_{118}	M_{119}	$\ldots$
M_0	$\underline{R}$	R	R	$\ldots$	R	R	R	$\ldots$
M_1	A	$\underline{A}$	A	$\ldots$	A	A	A	$\ldots$
M_2	R	R	$\underline{R}$	$\ldots$	R	R	R	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
M_{117}	A	∞	R	$\ldots$	$\underline{R}$	R	A	$\ldots$
M_{118}	R	R	R	$\ldots$	∞	$\underline{\infty}$	∞	$\ldots$
M_{119}	R	A	∞	$\ldots$	R	A	$\underline{A}$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- Let L_D be the language $\{M_i \mid$ Turing machine M_i rejects input $M_i\}$:

	M_0	M_1	M_2	$\ldots$	M_{117}	M_{118}	M_{119}	$\ldots$
L_D	A	R	A	$\ldots$	A	A	R	$\ldots$

Constructing an Undecidable Language

- Consider the matrix where entry (i, j) is 1 iff Turing machine i accepts the string that encodes Turing machine j:
- Let L_D be the language $\{M_i \mid \text{Turing machine } M_i \text{ rejects input } M_i\}$:

	M_0	M_1	M_2	$\ldots$	M_{117}	M_{118}	M_{119}	$\ldots$
L_D	A	R	A	$\ldots$	A	A	R	$\ldots$

- L_D is the language that we tried to construct D to decide.

Diagonalization and Halting

- A_{TM} is not Turing decidable (slide 3).
- A_{TM} is Turing recognizable (Nov. 3 lecture).
  - The set of Turing recognizable languages is strictly larger than the set of Turing decidable languages.
  - This is because a recognizer is allowed to loop: failure to halt means the recognizer rejects.
- $L_D = \{M \mid M\#M \in A_{TM}$ is not Turing recognizable (slide 5).
  - This is because the recognizer must halt whenever M loops when run with input M.
  - In fact, we could modify our machines to never use the $reject$ state — they could just loop to reject.
  - Then, recognizing L_D would mean determining that the machine will never halt.
  - Our argument that L_D is not Turing recognizable shows that this variant is not Turing recognizable.
- $\therefore HALT = \{M\#w \mid \text{Turing machine } M \text{ halts when run with input } w\}$ is Turing recognizable but not Turing decidable.
  - $\overline{HALT}$ is not even Turing recognizable.

Turing Co-Recognizable Languages

- The class of Turing decidable languages is closed under complement.
- The class of Turing recognizable languages is not closed under complement.
  - We say that a language, L, is Turing co-recognizable iff the complement of L is Turing recognizable.
  - For example, the language

    $LOOPS = \{M\#w \mid \text{Turing machine } M \text{ loops when run with input } w$ is Turing co-recognizable because it is the complement of $HALT$, a Turing recognizable language.

Relating Recognizability

- If a language is Turing recognizable and Turing co-recognizable, then it is Turing decidable.
  - Let L be a language that is both Turing recognizable and co-recognizable.
  - Because L is Turing recognizable, there is a Turing machine, M_L that for any $w \in L$ accepts w, and for any $w \notin L$ rejects or loops.
  - Because L is Turing co-recognizable, there is a Turing machine, M_{co-L} that for any $w \notin L$ rejects w, and for any $w \in L$ accepts or loops.
  - Now, we build a new TM, N that has two tapes, one for M_L and one for M_{co-L}. Each step of L takes a step for each of M_L and M_{co-L}. If either M_L or M_{co-L} accepts N accepts. Likewiese, if either rejects, N rejects. N is guaranteed to halt.
  - N is a TM that decides L.
  - $\therefore L$ is Turing decidable.

Why Allow Loopy Machines?

- Couldn't we just insist that we'll only consider TM's that halt on all inputs (i.e. deciders)?
- Problem 1:
  - We could do this, and our diagonalization would still work.
  - The obvious way to construct a TM for the diagonal (slide 3) produces a TM that loops. Language L_D remains undecidable.
- Problem 2: How do we know if a TM is a decider?
  - This is the question of whether or not a TM halts on all inputs, not just on one, specific input.
  - We say that a TM is total iff it halts on all inputs, and we write

$$TOTAL = \{M \mid M \text{ is a TM that halts on all inputs}\}$$

  - The language $TOTAL$ is neither Turing recognizable nor co-recognizable.
  - Thus, deciding whether or not a TM is a decider is even harder than the halting problem.

Reading List:

- Today: Sipser, 4.2 (midterm 2 cutoff)
- Nov. 8: Sipser, 5.1
- Nov. 10: Sipser, 5.1 (cont.)
- Nov. 13: Remembrance Day (no lecture)
- Nov. 15: Midterm 2
- Nov. 17: Sipser, 5.2
- Nov. 20: Sipser, 5.3