THE HALTING PROBLEM IS UNDECIDABLE

Now we are ready to prove Theorem 4.9, the undecidability of the language

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}.$$

PROOF We assume that A_{TM} is decidable and obtain a contradiction. Suppose that H is a decider for A_{TM}. On input $\langle M, w\rangle$, where M is a TM and w is a string, H halts and accepts if M accepts w. Furthermore, H halts and rejects if M fails to accept w. In other words, we assume that H is a TM, where

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Now we construct a new Turing machine D with H as a subroutine. This new TM calls H to determine what M does when the input to M is its own description $\langle M\rangle$. Once D has determined this information, it does the opposite. That is, it rejects if M accepts and accepts if M does not accept. The following is a description of D.

D = "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

Don't be confused by the idea of running a machine on its own description! That is similar to running a program with itself as input, something that does occasionally occur in practice. For example, a compiler is a program that translates other programs. A compiler for the language Pascal may itself be written in Pascal, so running that program on itself would make sense. In summary,

$$D(\langle M\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \textit{reject} & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

What happens when we run D with its own description $\langle D\rangle$ as input? In that case we get

$$D(\langle D\rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

No matter what D does, it is forced to do the opposite, which is obviously a contradiction. Thus neither TM D nor TM H can exist.

Let's review the steps of this proof. Assume that a TM H decides A_{TM}. Then use H to build a TM D that when given input $\langle M\rangle$ accepts exactly when M does not accept input $\langle M\rangle$. Finally, run D on itself. The machines take the following actions, with the last line being the contradiction.

- H accepts $\langle M, w\rangle$ exactly when M accepts w.
- D rejects $\langle M\rangle$ exactly when M accepts $\langle M\rangle$.
- D rejects $\langle D\rangle$ exactly when D accepts $\langle D\rangle$.

Where is the diagonalization in the proof of Theorem 4.9? It becomes apparent when you examine tables of behavior for TMs H and D. In these tables we list all TMs down the rows, M_1, M_2, ... and all their descriptions across the columns, $\langle M_1\rangle$, $\langle M_2\rangle$, ... The entries tell whether the machine in a given row accepts the input in a given column. The entry is *accept* if the machine accepts the input but is blank if it rejects or loops on that input. We made up the entries in the following figure to illustrate the idea.

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	...
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					...
M_4	*accept*	*accept*			
⋮			⋮		

FIGURE 4.4
Entry i, j is *accept* if M_i accepts $\langle M_j\rangle$

In the following figure the entries are the results of running H on inputs corresponding to Figure 4.4. So if M_3 does not accept input $\langle M_2\rangle$, the entry for row M_3 and column $\langle M_2\rangle$ is *reject* because H rejects input $\langle M_3, \langle M_2\rangle\rangle$.

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	...
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	...
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
⋮			⋮		

FIGURE 4.5
Entry i, j is the value of H on input $\langle M_i, \langle M_j\rangle\rangle$

In the following figure, we added D to Figure 4.5. By our assumption, H is a TM and so is D. Therefore it must occur on the list $M_1, M_2, \ldots$ of all TMs. Note that D computes the opposite of the diagonal entries. The contradiction occurs at the point of the question mark where the entry must be the opposite of itself.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$		$\vdots$			$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$		$\vdots$					$\ddots$

FIGURE 4.6
If D is in the figure, a contradiction occurs at "?"

A TURING-UNRECOGNIZABLE LANGUAGE

In the preceding section we demonstrated a language, namely, A_{TM}, that is undecidable. Now we demonstrate a language that isn't even Turing-recognizable. Note that A_{TM} will not suffice for this purpose because we showed that A_{TM} is Turing-recognizable on page 160. The following theorem shows that, if both a language and its complement are Turing-recognizable, the language is decidable. Hence, for any undecidable language, either it or its complement is not Turing-recognizable. Recall that the complement of a language is the language consisting of all strings that are not in the language. We say that a language is ***co-Turing-recognizable*** if it is the complement of a Turing-recognizable language.

THEOREM 4.16

A language is decidable if and only if it is both Turing-recognizable and co-Turing-recognizable.

In other words, a language is decidable if and only if both it and its complement are Turing-recognizable.

PROOF We have two directions to prove. First, if A is decidable, we can easily see that both A and its complement $\overline{A}$ are Turing-recognizable. Any decidable language is Turing-recognizable, and the complement of a decidable language also is decidable.

For the other direction, if both A and $\overline{A}$ are Turing-recognizable, we let M_1 be the recognizer for A and M_2 be the recognizer for $\overline{A}$. The following Turing machine M is a decider for A.