Undecidability and the Halting Problem

This material was presented in the second half of class on Thursday, January 28, 2016.

Throughout, points that were not covered in detail in class and that you are encouraged to think through and justify are marked by **"(WHY?)."**

If a set $S \subseteq \{0,1\}^*$ is recognized by a TM M, *i.e.*, $S = L(M)$, then we say that S is *decidable* or *computable*. One fundamental fact about the TM model is that there are sets that are not decidable. We first construct such a set, denoted UC, using a technique called *diagonalization* and then use the undecidability of UC to prove the undecidability of *the halting problem*.

Recall that every $x \in \{0,1\}^*$ encodes a TM, which we denote by M_x. Moreover, every TM is encoded by infinitely many binary strings. To see this, suppose that s encodes M and then consider all strings of the form $s\delta\sigma$, where δ is a binary string the semantics of which are "end of TM specification," and σ is an arbitrary string in $\{0,1\}^*$; there are infinitely many strings of this form, and they all encode M. Of course, many strings will encode TMs that are not well formed, and we follow the convention that an ill-formed TM recognizes the empty set. Let $\llcorner M \lrcorner$ be a binary string that encodes M.

The set UC is defined as follows: If $M_x(x) = 1$, *i.e.*, if M_x halts on input x and outputs 1, then $x \notin UC$. Otherwise, *i.e.*, if M_x does not halt on input x or if it halts and outputs something other than 1, then $x \in UC$.

Proposition 1 *UC is undecidable.*

Proof. Suppose that UC were the set accepted by M. By definition, $M(\llcorner M \lrcorner) = 1$ if and only if $\llcorner M \lrcorner \notin UC$, which is clearly a contradiction. ∎

Figure 1.7 of your textbook, a copy of which is attached at the end of these notes, shows why this proof technique is called "diagonalization." In this construction, UC is defined by "negating the diagonal set" and, in particular, is defined in a manner that precludes its being the set recognized by any TM. In the table of Figure 1.7, the rows are labeled by binary strings that encode TMs; remember that every TM labels infinitely many rows in the table. Every binary string except the empty string occurs once as a row label and once as a column label. (Observe that there is a typo in that figure in the book: The second row should be labeled "1," not "0." This has been corrected in the copy attached to these notes.) The columns of the table are strings that may or may not be in the language accepted by a particular TM. In the $(i,j)^{th}$ square of the table *before anything gets crossed out*, there is a 1 if M_i halts on input x_j and outputs 1, there is a 0 if M_i halts on input x_j and outputs 0, and there is a $*$ if M_i does not halt on input x_j or if it halts and outputs something other than 0 or 1. (Here M_i is the TM encoded by the string labeling row i, and x_j is the binary string labeling column j.)

The "cross outs" illustrate the process of going down the major diagonal of this table to define UC. A 1 is entered next to the crossed-out symbol in the $(i,i)^{th}$ square (*i.e.*, x_i is put into UC) if and only if the crossed-out symbol was a 0 or a $*$ (*i.e.*, if and only if M_i either does not halt on input x_i or halts and outputs something other than 1). UC cannot

be decidable, because it cannot be the set recognized by any TM: For every i, the TM of row i gives the wrong answer to the question "is x_i in UC?"

Those of you who have seen the proof that the reals are uncountable will have noticed by now that it is essentially the same as the proof that UC is undecidable.

It could be argued that the undecidability of UC is not particularly interesting, because membership in UC is not a naturally occurring computational problem. That argument does not apply to the halting problem. Let HALT be the set of pairs (α, x), where α and x are in $\{0,1\}^*$, such that M_x halts on input α. Membership in HALT is clearly a natural problem in the context of computer programming.

Proposition 2 *HALT is undecidable.*

Proof. We reduce UC to HALT. Note that this reduction need not be polynomial-time or many-to-one. **(WHY?)**

Suppose that HALT is decidable. That means that there is a Turing Machine A that halts and returns an output on every input (α, x), that $A(\alpha, x) = 1$ if M_x halts on input α, and that $A(\alpha, x) = 0$ if M_x does not halt on input α. Here is a specification for a machine that recognizes UC and uses A as a subroutine. We wish to decide whether x is in UC. If $A(x, x) = 0$, then output 1. Otherwise, let $z = M_x(x)$. If $z = 1$, then output 0; otherwise (i.e., if M_x halts on input x and outputs something other than 1), output 1. ∎

within any finite number of steps! This section gives a brief introduction to this fact and its ramifications. Though this material is not strictly necessary for the study of complexity, it forms the intellectual background for it.

The next theorem shows the existence of uncomputable functions. (In fact, it shows the existence of such functions whose range is {0, 1}, i.e. *languages*; such uncomputable functions with range {0, 1} are also known as *undecidable languages*.) The theorem's proof uses a technique called *diagonalization*, which is useful in complexity theory as well; see Chapter 3.

Theorem 1.10
There exists a function $\mathsf{UC} : \{0,1\}^* \to \{0,1\}$ *that is not computable by any TM.* ◇

PROOF: The function UC is defined as follows: For every $\alpha \in \{0,1\}^*$, if $M_\alpha(\alpha) = 1$, then $\mathsf{UC}(\alpha) = 0$; otherwise (if $M_\alpha(\alpha)$ outputs a different value or enters an infinite loop), $\mathsf{UC}(\alpha) = 1$.

Suppose for the sake of contradiction that UC is computable and hence there exists a TM M such that $M(\alpha) = \mathsf{UC}(\alpha)$ for every $\alpha \in \{0,1\}^*$. Then, in particular, $M(\llcorner M \lrcorner) = \mathsf{UC}(\llcorner M \lrcorner)$. But this is impossible: By the definition of UC,

$$\mathsf{UC}(\llcorner M \lrcorner) = 1 \Leftrightarrow M(\llcorner M \lrcorner) \neq 1$$

■

Figure 1.7 demonstrates why this proof technique is called *diagonalization*.

1.5.1 The Halting problem (first encounter with reductions)

The reader may well ask why should we care whether or not the function UC described above is computable—who would want to compute such a contrived function anyway?

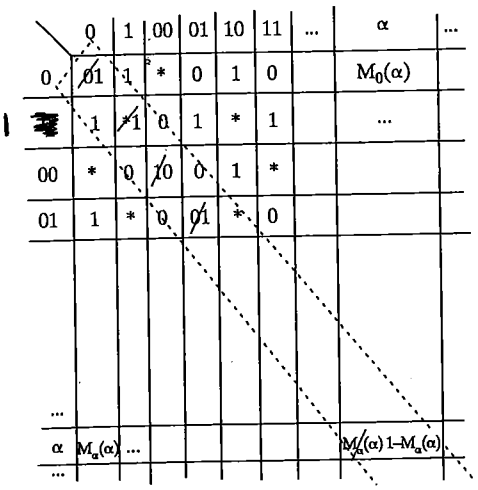

Figure 1.7. Suppose we order all strings in lexicographic order, and write in a table the value of $M_\alpha(x)$ for all strings α, x, where M_α denotes the TM represented by the string α and we use $*$ to denote the case that $M_\alpha(x)$ is not a value in {0, 1} or that M_α does not halt on input x. Then, function UC is defined by "negating" the diagonal of this table. Since the rows of the table represent *all* TMs, we conclude that UC cannot be computed by any TM.