# The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

## *Solution Manual*
## *Chapter 5, Additional Problem 5.2*

Classify each of the following sets and its complement as recursive, nonrecursive r.e., or non-r.e.

1. We can certainly produce the decimals of $\pi$, one after the other; since the decimals are enumerable, we can definitely semi-decide the set at hand: just proceed with the enumeration of the decimals of $\pi$, watching runs of consecutive 7s as we do so. As soon as a run of $x$ 7s is found, stop and answer "yes." But we can do better, although not constructively: this set is in fact decidable. Either $\pi$ has a longest consecutive run of 7s or it does not. (An example of an irrational number that does not have a longest run of consecutive 7s is 0.7077077707777077777 0 . . . .) If it does not, our set is just $\mathbb{N}$, which is certainly decidable! If it does, then let $n_0$ be the length of that longest consecutive run; but then our set is just $\{0, 1, 2, \ldots, n_0\}$, which is certainly decidable (just ask "is $x$ larger than $n_0$?). We do not know which of the two cases is the correct one, but one of them is and thus there exists an algorithm that correctly decides membership in our set. Which of the two algorithms is the correct one will require further insights in number theory, but we can state that the problem is decidable.
2. The difference between this case and the previous one is that we cannot assume that every value of $x$ below a certain limit (or every value at all) belongs to the set. If $\pi$ does not have a longest run of consecutive 7s and thus has an infinite number of runs of different lengths, it need not have runs of every possible length; thus deciding membership in our set would not be trivial. On the other hand, if $\pi$ does have a longest run of consecutive 7s, say of length $n_0$, then our set is definitely decidable, since we could build a table showing, for each $i \leq n_0$, whether $\pi$ has a run of consecutive 7s of length $i$. Again, we do not have enough results in number theory to be able to build such a table today, but such a table exists, and hence a decision algorithm exists.

So we can say this: if $\pi$ has a longest run of consecutive 7s, then the problem is decidable; otherwise, the problem may or may not be decidable. Thus, at present, all we can say is that our set is r.e.; we lack information to say anything about its recursiveness.

3. In words, this is the set of all functions that can always produce the double of whatever output they produce; in particular, any such function has infinite domain or is totally undefined or only returns 0. Since this property is strictly an I/O property, Rice's theorem tells us that neither the set nor its complement is recursive. In fact, neither one is r.e., as we proceed to prove with reductions from $\overline{K}$.

   To prove that $S$ is not r.e., we define the function

$$\theta(x, y) = \mathrm{Succ}\big(\mathrm{Zero}(\phi_{\mathrm{univ}}(x, x))\big)$$

   By our usual arguments, there exists $s(x)$ such that $\phi_{s(x)}(y) = \theta(x, y)$. Now, if $x \in \overline{K}$, then $\phi_{s(x)}(y)$ is totally undefined and thus $s(x) \in S$—because the predicate holds vacuously. On the other hand, if $x \in K$, then $\phi_{s(x)}(y)$ is the constant function Succ(Zero($y$)), always returning 1 for any argument, and thus $s(x) \notin S$, as desired (because there is no way to double the 1).

   Proving that $\overline{S}$ is not r.e. is tricky; we define the function $\theta$ as follows

$$\theta(x, y) = \begin{cases} 1 & step(x, x, y) = 0 \\ \Pi_1(y) & \text{otherwise} \end{cases}$$

   By our usual arguments, there exists $s(x)$ such that $\phi_{s(x)}(y) = \theta(x, y)$. Now, if $x \in \overline{K}$, then $\phi_{s(x)}(y) = 1$, thus $s(x) \in \overline{S}$, since the 1 cannot be doubled. On the other hand, if $x \in K$, then $\phi_{s(x)}(y)$ is 1 for $y < y_0$ ($y_0$ is some unknown constant) and $\Pi_1(y)$ for all $y \geq y_0$, so that the range of $\phi_{s(x)}$ is all of $\mathbb{N}$ (because the unpairing ensures that, no matter what the constant $y_0$, all natural integers will be encountered again: for any integer $i$, there exists an integer $y > y_0$, for instance $y = \langle i, y_0 \rangle$, such that $\Pi_1(y) = i$), so that $s(x) \in S$, as desired.

   Another possibility is to let

$$\theta(x, y) = \begin{cases} 1 & y = 0 \\ \mathrm{Succ}(y) + \mathrm{Zero}(\phi_{\mathrm{univ}}(x, x)) & \text{otherwise} \end{cases}$$

   then use the s-m-n theorem on $\theta$ to obtain $\phi_{s(x)}(y) = \theta(x, y)$. Note that, if $x \in \overline{K}$, then $\phi_{s(x)}(y)$ is only defined for $y = 0$ and has value 1, which cannot be doubled, so that $s(x) \in \overline{S}$, as desired. On the other hand, if $x \in K$, then $\phi_{s(x)}(y) = y + 1$ and so can double any value it generates, so that $s(x) \in S$.

4. This is a non-r.e. set. Informally, the fact that a machine accepts an infinite language cannot be verified, since we can only verify a finite number of acceptances, so that we cannot even semi-decide the set.
   We reduce the known non-r.e. set $\{x \mid M_x$ always halts$\}$ to our set. What we want to do is to take $M_x$ and transform it into a new Turing machine $M'_x$ such that $M'_x$ accepts an infinite language if and only if $M_x$ always halts. The problem is those machines $M_x$ that do not always halt, yet accept an infinite language; otherwise, the rest is trivial. An easy solution is the following: $M'_x$, on input $z$, will run $M_x$ on all strings less than $z$, one after the other; after running all of them and $z$ itself, the machine erases the tape and prints a 1. Now, if $M_x$ always halts, $M'_x$ will accept $\Sigma^*$, an infinite language. On the other hand, if $M_x$ does not always halt, then there is some smallest string $y$ on which $M_x$ does not halt and we can see that $M'_x$ will halt on all strings less than $y$ and fail to halt on $y$ and on all strings larger than $y$—in other words, $M'_x$ will only halt on a finite number of strings. Thus our reduction works correctly: $M_x$ always halts if and only if $M'_x$ accepts an infinite language.
5. Intuition about this set is harder to come. Given an enumerating machine M, how do we know whether the language that it enumerates is equal to its reverse? We can try running the enumerator (through dovetailing) and keeping track of strings whose reverse has not yet been generated, but we can never be sure that we have seen all strings in the language generated by M, and so can never decide to accept M. This line of thought seems to indicate that the set is non-r.e.
   We reduce the set $\{x \mid M_x$ always halts$\}$ to our set. Given $M_x$, we produce a new machine $M'_x$ which does the following: when given input $z$, it runs $M_x$ on all strings from $\epsilon$ up to $y$, then erases the tape and outputs a 1. As in our previous reduction, $M'_x$ will accept $\Sigma^*$ (a language equal to its reverse) if $M_x$ halts under all inputs. However, if $M_x$ does not always halt, let $y$ be the smallest string under which it does not halt; then $M'_x$ accepts all strings smaller than $y$ and enters an infinite loop if presented with any string equal to or larger than $y$. Now we want this finite language accepted by $M'_x$ in the second case not to be equal to its reverse in order to have a valid reduction; when could the language of all strings in lexicographic order from $\epsilon$ up to (but not including) some $y$ be equal to its reverse? Because of the lexicographic order, only when the string $y$ is one character longer than its predecessor in the order—that's because the first string of

length $n$ generated in lexicographic order has as reverse the last string of length $n$ generated in lexicographic order. So we need only modify our machine $M_x'$ a little bit: when presented with a string $y$ that is the first string in lexicographic order, it should accept it without running $M_x$ on it. Now $M_x'$ actually accepts an infinite language no matter what (at least one string for each length $n$, the first of that length in lexicographic order), but only accepts a language equal to its reverse if it accepts all strings. Thus our reduction is complete.