In-Class Examples

Decidable versus Enumerable

Recall that a language $L$ is **decidable** if there exists a program $P$ such that for any string $w$, $P(w)$ halts and

$$P(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{if } w \notin L \end{cases}$$

A language $L$ is **enumerable** if there exists a program $P$ that prints out all the strings in $L$. In other words, $P$ must only print out strings that belong to $L$, and for any string $w \in L$, $P$ must run a finite amount of time before printing out $w$.

- **Are all decidable languages enumerable?**

  **Solution:** Yes. We can enumerate a decidable language $L$ by going through all the strings in $\Sigma^*$ in lexicographic order and skipping the ones that are not in $L$.

- **Are all enumerable languages decidable?**

  **Solution:** No. The halting set $K$ is enumerable but not decidable. To enumerate $K$ we can loop from $n = 1$ to $\infty$ and at each iteration of the loop print out all the programs of size at most $n$ that halt in at most $n$ steps.

The Digits of $\pi$

To print out all the digits of $\pi$ would take forever. Therefore, any program that prints out all the digits of $\pi$ must not halt. Let $F(n)$ be the maximum number of digits of $\pi$ that an $n$-character Java program can print out and still halt. Show that $F(n)$ is not computable.

**Solution:** We will show that if $F(n)$ was computable then the halting set would be decidable. Given an arbitrary program $P$, define $Q_P$ as follows.

**Procedure** $Q_P$:

1. Do:
   
   (a) Print out the next digit of $\pi$.
   
   (b) Simulate $P(P)$ for one more step. If $P(P)$ halts, then halt.

Let $k = F(length(Q_P))$. Suppose we run $Q_P$ for $k$ iterations of the loop. If $Q_P$ halts then $P$ must halt. If $Q_P$ does not halt in $k$ iterations then it must never halt, because otherwise it would print out more than $k$ digits of $\pi$. Thus by running $Q_P$ for $k$ iterations we can determine whether $P$ halts, contradicting the fact that the halting set is undecidable.

Group Problems

The Halting Set Revisited

In lecture we defined the halting set $K = \{ P \mid P$ is a program and $P(P)$ halts$\}$, and proved that $K$ is undecidable. Suppose we instead considered programs that take no input. Show that the set $K' = \{ P \mid P$ is a program and $P$ halts $\}$ is also undecidable.
**Solution:** Given an arbitrary program $P$, let $Q_P$ be the program that takes no input and simply calls $P(P)$. So $Q_P$ halts if and only if $P(P)$ halts, or equivalently $Q_P \in K'$ if and only if $P \in K$. Thus if $K'$ was decidable then $K$ would also be decidable, which contradicts what we proved in lecture.

**Resolving Berry’s Paradox**

Suppose you are given a program called $\text{meaning}$ that takes as input a string with less than 1000 characters and outputs the integer denoted by that string. For example, $\text{meaning} (“two plus two”) \text{ would output 4. If s is a string with } \geq 1000 \text{ characters, } \text{meaning}(s) \text{ prints an error message. Assume that garbage sentences are mapped to zero, so for example } \text{meaning} (“fdajsfdsalk”) \text{ would output 0. Finally, note that the } \text{meaning} \text{ function is computable because there are only a limited number of valid inputs, so if nothing else it could be implemented using a huge table.}

(a) Describe how you would write a script that computes the smallest non-negative integer $n$ such that for all strings $s$ with $< 1000$ characters, $\text{meaning}(s) \neq n$.

**Solution:** Enumerate all strings $s$ with $< 1000$ characters, and for each one run $\text{meaning}(s)$. Record all the values output by $\text{meaning}(s)$ in a list. Then return the smallest non-negative integer that isn’t on the list.

(b) Let $m = \text{meaning} (“the smallest non-negative integer that can’t be described using less than 1000 characters”)$. Argue that $m \neq n$ (where $n$ is the value output by the script from part (a)).

**Solution:** By definition, the value $n$ returned by the script did not appear on a list that contained $m$.

(c) Berry’s Paradox is the fact that the phrase “the smallest non-negative integer that can’t be described using less than 1000 characters” either denotes a unique integer or does not, but either interpretation leads to a contradiction. Explain why this is not such a paradox after all.

**Solution:** Berry’s Paradox is only a paradox if you assume that $m$ and $n$ must be the same integer, but this assumption is always false.