CSE 311: Foundations of Computing I

Section : Cardinality & Uncomputability Solutions

1. Cardinality

(a) You are a pirate. You begin in a square on a 2D grid which is infinite in all directions. In other words, wherever you are, you may move up, down, left, or right. Some single square on the infinite grid has treasure on it. Find a way to ensure you find the treasure in finitely many moves.

Solution:

Explore the square you are currently on. Explore the unexplored perimeter of the explored region until you find the treasure (your path will look a bit like a spiral).

(b) Prove that $\{3x \ : \ x \in \mathbb{N}\}$ is countable.

Solution:

We can enumerate the set as follows:

$$f(0) = 0$$
$$f(1) = 3$$
$$f(2) = 6$$
$$f(i) = 3i$$

Since every natural number appears on the left, and every number in S appears on the right, this enumeration spans both sets, so S is countable.

(c) Prove that the set of irrational numbers is uncountable.
Hint: Use the fact that the rationals are countable and that the reals are uncountable.

Solution:

We first prove that the union of two countable sets is countable. Consider two arbitrary countable sets C_1 and C_2. We can enumerate $C_1 \cup C_2$ by mapping even natural numbers to C_1 and odd natural numbers to C_2.
Now, assume that the set of irrationals is countable. Then the reals would be countable, since the reals are the union of the irrationals (countable by assumption) and the rationals (countable). However, we have already shown that the reals are uncountable, which is a contradiction. Therefore, our assumption that the set of irrationals is countable is false, and the irrationals must be uncountable.

(d) Prove that $\mathcal{P}(\mathbb{N})$ is uncountable.

Solution:

Assume for the sake of contradiction that $\mathcal{P}(\mathbb{N})$ is countable.
This means we can define an enumeration of elements S_i in $\mathcal{P}$.
Let s_i be the binary set representation of S_i in $\mathbb{N}$. For example, for the set $0, 1, 2$, the binary set representation would be $111000\ldots$
We then construct a new subset $X \subset \mathbb{N}$ such that $x[i] = \ s_i[i]$ (that is, $x[i]$ is 1 if $s_i[i]$ is 0, and $x[i]$ is 0 otherwise).

Note that X is not any of S_i, since it differs from S_i on the ith natural number. However, X still represents a valid subset of the natural numbers, which means our enumeration is incomplete, which is a contradiction. Since the above proof works for any listing of $\mathcal{P}(\mathbb{N})$, *no* listing can be created for $\mathcal{P}(\mathbb{N})$, and therefore $\mathcal{P}(\mathbb{N})$ is uncountable.

2. Uncomputability

(a) Let $\Sigma = \{0, 1\}$. Prove that the set of palindromes is decidable.

Solution:

We can implement the function that takes a string as input and reverses that string, using the recursive definition of string reverse given in class. So on input x we run that reversing program to create the string $y = x^R$. Then we compare x against y character by character and output yes iff we find that $x = y$.

(b) Prove that the set $\{(\texttt{CODE}(R), x, y) \ : \ R \text{ is a program and } R(x) \neq R(y)\}$ is undecidable where we write $R(x) = \uparrow$ if R runs forever.

Solution:

Let S be the set $\{(\texttt{CODE}(R), x, y) \ : \ R \text{ is a program and } R(x) \neq R(y)\}$. Assume for the sake of contradiction that S is decidable. Then there exists some program `Q(String input, String x, String y)` which returns `true` iff $(\texttt{CODE}(R), x, y) \in S$.

Let `P()` be some arbitrary program. We will show that we can use `Q` to determine if `P` halts.

We first write a program `I(String input)` that incorporates the code of `P`:

```
String I(String input) {
   if (input.equals("kittens")) {
       // Run forever
       while (true) {
       }
   } else {
       // Execute P
       <Code of P>
   }
}
```

Note that this program will always run forever when the input is “kittens” OR `P` runs forever, but will otherwise return whatever P returns.
Now, we can write `DOESHHALT()`:

```
boolean DOESHHALT() {
   return Q(CODE(I),"kittens","bunnies");
}
```

If `Q(CODE(I),"kittens","bunnies")` returns true, then `I("kittens")` $\neq$ `I("bunnies")`, so `P` does not run forever, so `P` halts.

If `Q(CODE(I),"kittens","bunnies")` returns `false`, then `I("kittens")` $=$ `I("bunnies")`, so `P` runs forever, so `P` does not halt.

Since `P` was arbitrary, we can construct a program using `Q()` like `DOESHHALT()` for *any* program, which allows us to decide the halting set. Since we can use `Q` to decide the halting set, but the halting set is undecidable, `Q` cannot exist.
Since `Q` was an arbitrary function that decides S, no function that decides S can exist, and therefore S is undecidable.