CS 6505: Computability & Algorithms
Lecture 1, January 11, 2010

Computation and Functions

Computation is the evaluation of *functions*. A function has a *domain* and a *range*. Suppose a function f has a domain D and a range R. Then we say that

$$f : D \to R, x \in D, f(x) \in R$$

Another way of saying this is that f *maps* elements of D into elements of R

Examples of functions

1. Suppose that $D = \{0, 1, 2, \ldots\}$ and $R = 0, 1$. Then

$$f(x) = \begin{cases} 1 & \text{if x is prime} \\ 0 & \text{otherwise} \end{cases}$$

2. $D = \{$ dog, cat, spider, caterpillar, $\ldots \}$, $R = \{$ 0, 1, 2, $\ldots \}$
   $f(x) =$ x's number of legs

3. $D = \{$ set of all dates $\}$, $R = \{0, 1\}$

$$f(x) = \begin{cases} 1 & \text{if humans exist or existed on date x} \\ 0 & \text{otherwise} \end{cases}$$

Computability

What functions can be computed? By what? Using what? How efficiently? If a function cannot be computed, then why not? What is a computer program, really?

A *computer program* is a function that maps an input to to an output. Not all functions have valid computer programs.

Sample Problems

We can think of some common math problems, such as "Add x and y", "Find the Least Common Multiple (LCM) of x and y", or even much more difficult ones. Computer programs can be written to solve these math problems.

To formalize all such problems, we have a finite alphabet Σ and the set of strings over Σ. A *language* is a subset of such strings. We can view languages as a function:

$$L \equiv f_L : \Sigma^* \rightarrow \{0, 1\}$$

$$f_L(x) = \begin{cases} 1 & \text{if x is prime} \\ 0 & \text{otherwise} \end{cases}$$

A language L is *computable* if we can compute whether x belongs to L for any string x. Are all languages computable? Can we enumerate (i.e., list) all languages?

We have:

Σ: finite alphabet (e.g., { a, b, c, ... })

Σ^* : set of strings over Σ

The set of strings over Σ is infinite, but it is countable!

Another way of saying this is:

$$\exists g : \Sigma^* \rightarrow N$$

Now, if Σ = { 0, 1 }, then $g(x)$ is just the number represented by x in binary. If Σ has k elements, then $g(x)$ is k-ary (e.g., $k = 10$ means that the alphabet is decimal, $k = 16$ means that the alphabet is hexadecimal, etc.)

Now note that the set of all languages is uncountable! The set of all languages is different from the set of all strings - it is the set of all possible subsets of strings.

Let's prove that the set of all languages is uncountable. We'll proceed using proof by contradiction. Suppose this is not true. This means there exists an ordering L_1, L_2, L_3, ... of languages.

Now consider the following matrix of strings (x_1, x_2, ...) and languages (L_1, L_2, ...). The entry for language L_i and string x_j is 1 if $x_j \in L_i$ and is 0 if $x_j \in L_i$.

	x1	x2	x3	x4	...
L1	1	0	0	0	...
L2	0	0	1	1	...
L3	1	1	1	1	...
L4	0	1	0	0	...
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	

Now we will define a new language L that does not match any of the languages in the matrix. For language L, define $x_j \in L$ iff $x_j \notin L_j$. This

means that we have a new language L that is different from every other language with respect to at least one string. Then the set of all languages cannot be enumerated using the list presented in the matrix. So the set of all languages is uncountable.

Relevant Readings

- Sipser, Chapters 3 & 4