

CSC165 winter 2013

Mathematical expression

Danny Heap

heap@cs.toronto.edu

BA4270 (behind elevators)

<http://www.cdf.toronto.edu/~heap/165/W13/>

416-978-5899

Course notes, chapter 5



Outline

infinities and functions

some assignment 3 questions

notes



recall $f : \mathbb{N} \mapsto \{\text{even natural numbers}\}$
 $f(n) = 2n$ is **onto** and 1-1



rational numbers, \mathbb{Q} are countable

Show a **list**, i.e. some $f : \mathbb{N} \mapsto \mathbb{Q}$ that is **onto**



Cantor's example

To show that the set of infinite decimals in $[0, 1]$ was bigger than the natural numbers, Cantor showed that any so-called list of these numbers would always miss entries:

list position	decimal
0	0.000000000000...
1	0.010101010101...
2	0.012012012012...
3	0.012301230123...
\vdots	\vdots

No matter how you try to generate the list it will omit the number formed by taking '0.' and then traversing the diagonal and changing the digit by adding 1 (if it's not a 9), and subtracting 1 (if it's a 9).

This means that the real numbers (which contain $[0, 1]$) are a larger infinity than the natural numbers.



two specifications of a function

A precise, but infeasible, specification of a function is its behaviour on **every** input:

```
def f(n) :  
    if n == 0 : return 3  
    if n == 1 : return 4  
    if n == 2 : return 5  
    # ...  
    if n == "foo" : # throw a type error
```

Or you could write a **procedure** to compute its behaviour:

```
def f(n) :  
    return n + 3
```

There are more ways to do the former than the latter. So many more that they don't match up...!

how many python functions?

Every python function can be written in UTF-8, as a string of characters and whitespace out of 256 characters to define a function:

```
def f(n) :  
    return n + 3
```

Each string can be converted to a different number by treating each character as a digit in base 256. This gives us an onto function from \mathbb{N} to the set of python programs — there are countably many python functions.



diagonalization

Make a column of each of the countably many python functions. In each row, list the **behaviour** of whether that function halts or loops given another function as input:

Function f	H(f,f0)	H(f,f1)	H(f,f2)	H(f, f3)	H(f, f4)	H(f, f5)	H(f, f6)
f0	halts	halts	halts	halts	halts	halts	halts
f1	loops	loops	loops	loops	loops	loops	loops
f2	halts	loops	halts	loops	halts	loops	halts
f3	halts	loops	loops	halts	loops	loops	halts
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

If you toggle the diagonal — switch loops to halts and vice-versa — you will get the behaviour of a “function” that can’t possibly be on the list — `navel_gaze`. There are more (a larger infinity) of behaviours than python functions.

$$\lim_{n \rightarrow \infty} (5n^3 + 3n + 7) / (2n^3 + 4n + 8) = 5/2$$

What does that tell us about big-Oh, big-Omega?

another uncomputable function

```
def halt(f,i):
    def initialized(g,v):
        """ g initializes v on every possible input """
        ...code for initialized goes here...

    # Put some code here to scan the code for f and figure out
    # a variable name that doesn't appear, and store it in v

    def f_prime(x):
        # Ignore the argument x, call f with the fixed argument
        # (the one passed in to halt).
        f(i)
        exec("print " + v) #

    return not initialized(f_prime,v)
```



Notes