# CSC165 winter 2013
## Mathematical expression

Danny Heap

heap@cs.toronto.edu

BA4270 (behind elevators)

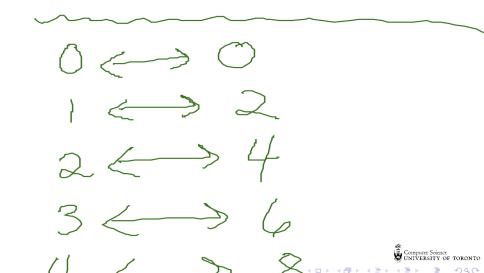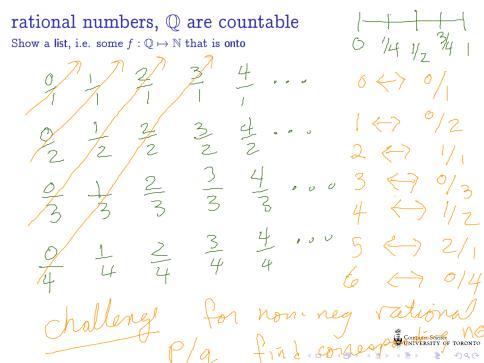http://www.cdf.toronto.edu/~heap/165/W13/

416-978-5899

Course notes, chapter 5

# Outline

infinities and functions

notes

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

# rational numbers, $\mathbb{Q}$ are countable

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

$\begin{array}{ccccc} \frac{0}{1} & \frac{1}{1} & \frac{2}{1} & \frac{3}{1} & \frac{4}{1} \cdots \\ \frac{0}{2} & \frac{1}{2} & \frac{2}{2} & \frac{3}{2} & \frac{4}{2} \cdots \\ \frac{0}{3} & \frac{1}{3} & \frac{2}{3} & \frac{3}{3} & \frac{4}{3} \cdots \\ \frac{0}{4} & \frac{1}{4} & \frac{2}{4} & \frac{3}{4} & \frac{4}{4} \cdots \end{array}$

0   1/4   1/2   3/4   1

$0 \longleftrightarrow 0/1$

$1 \longleftrightarrow 0/2$

$2 \longleftrightarrow 1/1$

$3 \longleftrightarrow 0/3$

$4 \longleftrightarrow 1/2$

$5 \longleftrightarrow 2/1$

$6 \longleftrightarrow 0/4$

challenge for non-neg rational $p/q$ find corresponding $n \in$

# 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 | $0.000000000000\cdots$ |
| 1 | $0.010101010101\cdots$ |
| 2 | $0.012012012012\cdots$ |
| 3 | $0.012301230123\cdots$ |
| $\vdots$ | $\vdots$ |

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 computer 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.

# Notes

def fn

retur