CSC165 fall 2014

Mathematical expression

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cs.toronto.edu/~heap/165/F14/
416-978-5899

Course notes, chapter 5

Outline

infinities and functions

Induction

notes

annotated slides

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

countable is listable:

A set is **countable** if, and only if, it can be described as a list:

$n \in \mathbb{N}$	$\longrightarrow$	$f(n)$
0	$\longrightarrow$	0
1	$\longrightarrow$	2
2	$\longrightarrow$	4
3	$\longrightarrow$	6
4	$\longrightarrow$	8
$\vdots$	$\vdots$	$\vdots$

Correspondence to $\mathbb{N}$ is built in to a list — each item has a position, corresponding to some element of **N**

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 (to make representations unique, no infinite strings of 9s are allowed in the list):

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 less than 5), and subtracting 1 (if it's 5 or greater).

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.

principle of simple induction

Suppose $P(n)$ is a predicate of the natural numbers. If P...

- starts out true, i.e. $P(0)$, and
- the truth of P transfers from each number to the next, i.e. $\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)$, then

... we believe P is true for all natural numbers, i.e. $\forall n \in \mathbb{N}, P(n)$.

nearly the principle of simple induction:

For which table is $\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)$ false?

n	$P(n)$	n	$P(n)$	n	$P(n)$	n	$P(n)$
0	True	0	False	0	True	0	False
1	True	1	False	1	True	1	False
2	True	2	False	2	False	2	True
3	True	3	False	3	True	3	True
4	True	4	False	4	True	4	True
5	True	5	False	5	True	5	True
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

tweak simple induction

The **fourth** table on the previous slides suggests a small modification

Suppose $P(n)$ is a predicate of the natural numbers. If P...

- starts out true, i.e. $P(k)$, some $k \in \mathbb{N}$,
- the truth of P transfers from each number, starting at k, to the next, i.e. $\forall n \in \mathbb{N}, n \geq k \Rightarrow (P(n) \Rightarrow P(n+1))$, then

... we believe P is true for all natural numbers greater than or equal to k, i.e. $\forall n \in \mathbb{N}, n \geq k \Rightarrow P(n)$.

illustrative example

$P(n): 3^n \geq n^3$

Write out the inductive hypothesis (IH) first, and try to construct an argument that gets us from $P(n)$ to $P(n+1)$ (inductive step):

example continued...

$P(n) : 3^n \geq n^3$

Take notice of which case(s) $P(n)$ is true for, but are **not** covered by the inductive step. These are **base cases,** and must be proved **without** induction.

simple induction principle...

We end up with:

$$[P(3) \wedge (\forall n \in \mathbb{N}, n \geq 3 \Rightarrow [P(n) \Rightarrow P(n+1)])]$$
$$\Rightarrow [\forall n \in \mathbb{N}, n \geq 3 \Rightarrow P(n)]$$

That's what induction gets us. $P(0), P(1)$, and $P(2)$ are verified separately.

another example

$P(n) : \sum_{i=0}^{i=n} 2^i \leq 2^{n+1}$

another example continued...

$P(n) : \sum_{i=0}^{i=n} 2^i \leq 2^{n+1}$

Notes

annotated slides

- monday's annotated slides
- wednesday's annotated slides
- friday's annotated slides