MFCS

Classifying Functions

Klaus Sutner
Carnegie Mellon University
Fall 2022

Recall: Calculus

It is a standard exercise to classify real functions in calculus according to various properties they may or may not have:

- bounded
- monotonic
- periodic
- continuous
- differentiable

The last two properties depend heavily on the reals, the first three are slightly more general†.

We are here interested in basic properties that apply to all possible functions.

†It's a good exercise to figure out for which domains/codomains they make sense.

Applying a function to arguments usually destroys information: given the result, we cannot reproduce the input. A mild example is

$$f : \mathbb{R} \to \mathbb{R} \qquad x \mapsto x^2$$

where the fibers can have size 2. Much worse is multiplication:

$$* : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \qquad x, y \mapsto x * y$$

Here the fibers can be arbitrarily large and even infinite.

Definition

A function $f : A \to B$ is injective, reversible or one-one if $f(x) = f(y)$ implies $x = y$.

There is a slightly strange asymmetry in our definition of a function: for $f : A \to B$ we require

$$\forall\, x \in A\, \exists\, y \in B\, (f(x) = y)$$

but we do **not** require the opposite:

$$\forall\, y \in B\, \exists\, x \in A\, (f(x) = y)$$

Definition

A function $f : A \to B$ is surjective or onto if its range is the whole codomain: rng f = cod f.

Definition

A function that is both injective and surjective is a bijection.

Endofunctions that are bijections are often called permutations, in particular on a finite set.

If $f : A \to B$ is a bijection, then we can actually think of the inverse map as a plain function

$$f^{-1} : B \to A \qquad f^{-1}(b) = a \Leftrightarrow f(a) = b$$

Bijections are hugely important in combinatorics and group theory, and we will use them to define a notion of size of an arbitrary set (cardinality).

A Modest Proposal

Injectivity is a deep-seated property of a function, but surjectivity is not: we can simply redefine the codomain to be the range.

Here is a proposal by Herr Prof. Dr. Wurzelbrunft:

> We can improve the definition of a function: all functions are required to be surjective.

He thinks this is not a big deal, since we can just shrink the codomain to the range of the original function.

How about it? Wouldn't Wurzelbrunft-functions make life much easier?

Doom and Disaster

Wurzelbrunft is out of his mind.

Consider calculus: figuring out the range of a function can be very difficult. Try the real function

$$f(x) = \sum (-1)^i \, \frac{x^{2i+1}}{(2i+1)!}$$

without using results from calculus.

Even worse, there are computable functions $f : \mathbb{N} \to \mathbb{N}$ where the range is not computable.

It might even be an open problem what the range of a pretty simple function is (think about prime twins).

Some people will tell you that a definition is just an abbreviation, a shortcut that allows us to contract a possibly complicated and long expression into a short one.

x is a foofoo $:\Leftrightarrow$ blahblahblahblah x blahblahblahblahblah

Consequently, there is no such thing as a wrong definition.

In the sense of mathematical logic this is correct. But it's utter nonsense in the RealWorld$^{\text{TM}}$. A definition is supposed to encapsulate an idea, a concept–done right, it will structure and advance the argument. Done wrong, it will just produce cognitive clutter and will stand in the way.

Ask Cauchy about continuity.

Understanding Definitions

Recall our old checklist?

- intuitive meaning
- intuitive meaning
- intuitive meaning
- formal meaning
- examples
- counterexamples
- basic results
- links to other concepts

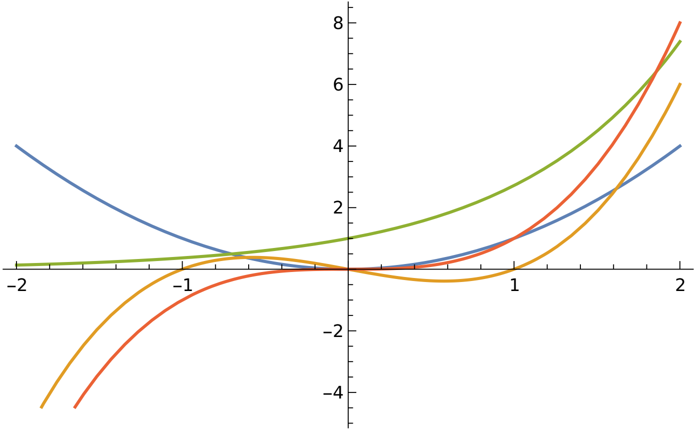
If several of these items are difficult to handle, maybe the definition is wrong and needs to be changed.

For the usual suspects from calculus functions, it may be straightforward to determine their properties, and it may require a bit of knowledge in analysis.

$x \mapsto x^2$	not surjective, not injective
$x \mapsto x^3 - x$	surjective, not injective
$x \mapsto e^x$	not surjective, injective
$x \mapsto x^3$	surjective, injective

Polynomials are easy, but exponentiation, logarithms, trig functions are already quite messy (and it took a long time to develop a good understanding of these functions).

Picture



Life becomes much more interesting in the discrete realm, in part because the highly developed machinery from analysis does not apply†. Here are some functions on the naturals.

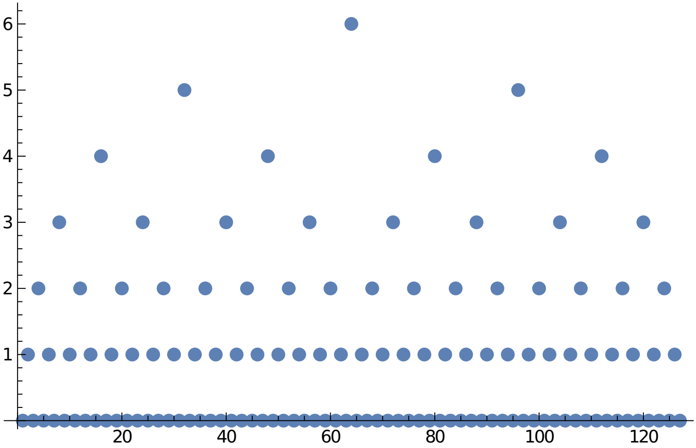
$$\begin{aligned} \nu_2(0) &= \infty \\ \nu_2(x) &= \max\big(k \mid 2^k \text{ divides } x\big) \\ |x| &= \text{number of binary digits in } x \end{aligned}$$

ν_2 is called the 2-adic valuation or 2-adic order and plays a big role in number theory.

$\nu_2(x)$ simply counts the trailing 0s in the binary expansion of x.

So $x/2^{\nu_2(x)}$ is odd.

†Not my idea, von Neumann pointed this out.



6
5
4
3
2
1
20
40
60
80
100
120

Now define a strange function $f : \mathbb{N} \to \mathbb{N}$ by

$$f(x) = 2^\ell(2k+1) - 1$$

$$k = \nu_2(x+1)$$

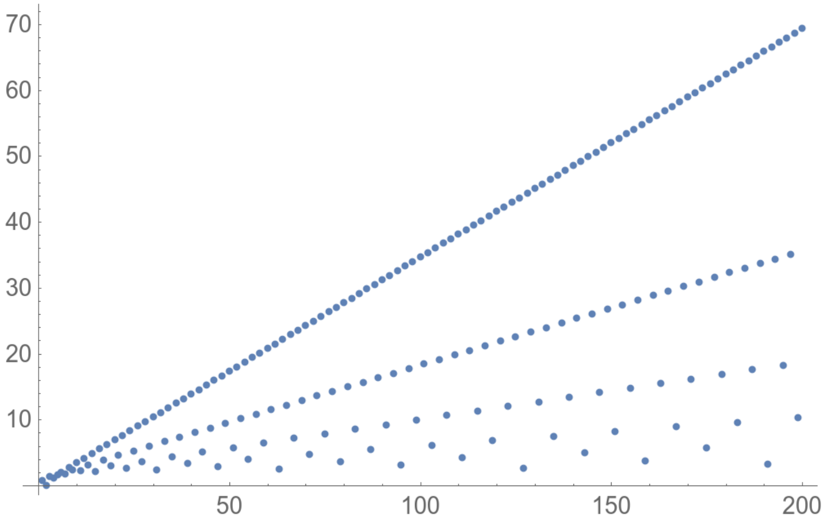
$$\ell = \big((x+1)/2^k - 1\big)/2$$

Intuition? No clue.

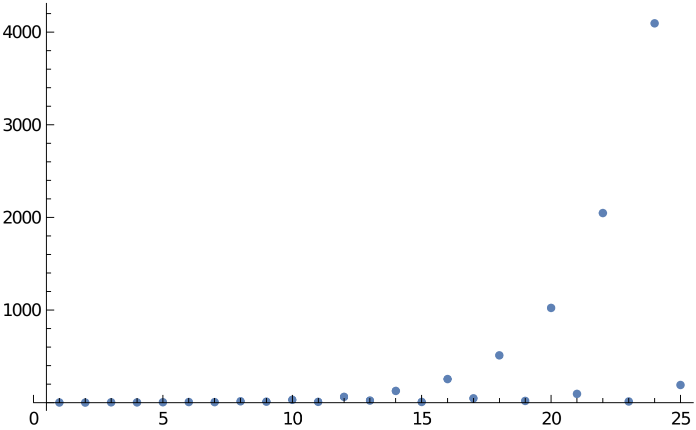
Formal definition? Mildly messy, not too horrible.

Properties? No clue.

Log-Plot

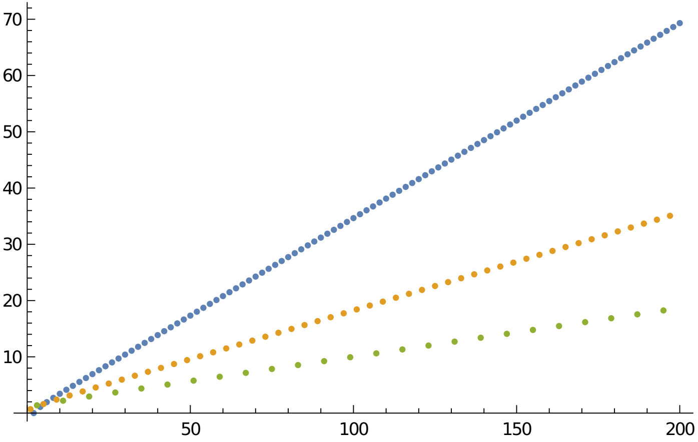


Why the log-plot? f grows exponentially and an ordinary plot is quite useless.



What are the "lines" in the picture?

k is critical, so it is natural to filter out the arguments for which $k = 0, 1, 2, \ldots$



FAQ

Why is f injective?

Suppose $f(x) = f(x')$. Then

$$2^{\ell}\big(2k+1\big) = 2^{\ell'}\big(2k'+1\big)$$

so that $k = k'$ and $\ell = \ell'$. Why

But $\big((x+1)/2^k - 1\big)/2 = \big((x'+1)/2^k - 1\big)/2$ and $x = x'$.

So the trick is to split x into two parts k and ℓ and combine those in a reversible fashion.

FAQ

Why is f surjective?

Fix $z \in \mathbb{N}$, we need some $x \in \mathbb{N}$ so that

$$2^\ell(2k+1) = z+1$$

where $k = \nu_2(x+1)$ and $\ell = \big((x+1)/2^k - 1\big)/2$

Note that ℓ and k are uniquely determined by z.

To match k consider $x+1 = 2^k(2r+1)$.

Then $\ell = ((2r+1)-1)/2 = r$ and we are done.

Four Cases

$x \mapsto \lvert x\rvert^2$	not surjective, not injective
$x \mapsto \nu_2(x)$	surjective, not injective
$x \mapsto x + \lvert x\rvert$	not surjective, injective
$x \mapsto f(x)$	surjective, injective

For item 2 we have to adjust the definition of ν_2 slightly: $\nu_2(0) = 0$.

Exercise

Make sure you understand the proofs of all these assertions.

Last Question

Recall the definition of f:

$$f(x) = 2^{\ell}(2k+1) - 1$$

$$k = \nu_2(x+1)$$
$$\ell = \big((x+1)/2^k - 1\big)/2$$

What would happen if we switch k and ℓ in the first line?

Functions as Input

One usually thinks of functions that map simple objects to other simple objects: numbers to numbers, pairs of numbers to numbers, lists of numbers to numbers, lists to lists, matrices to numbers, . . .

That's fine, but there is another important class of functions, ones that take other functions as input.

These are sometimes called higher-order functions or functionals or operators. They are conceptually a little more difficult to deal with, but they are extremely useful, nothing moves without them.

This idea is quite familiar from calculus:

- Differentiation of real functions

$$\partial_x : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

- Integration of real functions

$$\int : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$$

Strictly speaking, this is not quite right. Why?

More relevant to us are the following examples.

- Composition

$$\circ : (B \to C) \times (A \to B) \to (A \to C)$$

- Currying

$$\mathsf{curry} : (A \times B \to C) \to (A \to (B \to C))$$

- Map

$$\mathsf{map} : (A \to B) \times \mathsf{List}(A) \to \mathsf{List}(B)$$

- Iteration

$$\mathsf{iterate} : (A \to A) \times \mathbb{N} \to (A \to A)$$

- Orbits

$$\mathsf{orb} : (A \to A) \times A \to A^{\mathbb{N}}$$

Having a small arsenal of carefully curated higher order functions lying around is hugely useful, both conceptually and computationally.

This takes a bit of getting-used-to, but pays off in the end. Using abstraction is better than to wallow in minute details.

Computational Example: Fold

Here is a well-known example of a higher order function:

$$\mathsf{fold} : (A \times B \to A) \times A \times \mathsf{List}(B) \to A$$

Suppose we have $f : A \times B \to A$ and a starting element $e \in A$. By recursion, we can define

$$\mathsf{fold}(f, e, .) : \mathsf{List}(B) \to A$$

$$\mathsf{fold}(f, e, \mathsf{nil}) = e$$
$$\mathsf{fold}(f, e, L :: b) = f(\,\mathsf{fold}(f, e, L), b\,)$$

$L :: b$ means append b to L. For example,

$$\mathsf{fold}(f, e, (a, b, c, d)) = f(f(f(f(e, a), b), c), d)$$

Who Cares?

We can now implement at a surprising number of list operations:

$f(a,b)$	e	fold$(f, e, .)$
$a+1$	0	length
prep(a,b)	nil	reverse
app$(a, g(b))$	nil	map g
$a \vee [b=c]$	false	search for c
$a + [b=c]$	0	count c

Here $[\alpha = \beta]$ follows Knuth's convention: it is 1 (or True) if indeed $\alpha = \beta$, and 0 (or False) otherwise.