UCT

Classical vs. Constructivist

KLAUS SUTNER
CARNEGIE MELLON UNIVERSITY
SPRING 2022

> Show that a set is decidable
> iff
> its principal function is computable.

Left-to-right is easy: use the decision algorithm to test all $x \in \mathbb{N}$ and define f accordingly.

But right-to-left runs into a problem: you are given a program that computes the principal function $f : I \to \mathbb{N}$. Here $I \subseteq \mathbb{N}$ is some initial segment.

For the decision algorithm we essentially want check whether

$$f(i) = x \qquad \text{or} \qquad f(i) < x < f(i+1)$$

Disaster?

For $I = \mathbb{N}$ this works out fine, just compute $f(i)$ for $i = 0, 1, 2, \ldots$

But if I is finite, say, $I = \{0, 1, \ldots, n{-}1\}$ then the computation $f(n)$ diverges and we are sunk. Right?

Not at all: in this case $A = \text{rng}\, f$ is finite and we can simply do a finite table lookup.

You might object that checking whether A is finite, given an index for f, is undecidable.

Absolutely true, but it does not matter, not one bit.

Algorithm I: return Yes

Algorithm II: return No

One of those two algorithms works. Done.

Doing Too Much

The exact same situation arises here: depending on whether f is total, one method or another works.

It is undecidable which one is correct, but that does not matter: we know the decision algorithm exists. Done.

If you are a constructivist you will reject this argument.

Alas, constructivism as the default system for mathematics and/or CS is pretty much dead. It has very important applications in certain areas, but it is not the general system of reasoning used everywhere.

Robertson-Seymour

The decision algorithm for minor-closed classes of graphs is a perfect example for a non-constructive description: we know we can do things in quadratic time, but we have no idea how.

By "we know" I mean we have a proof in Zermelo-Fraenkel set theory. We do not have an algorithm to compute the obstruction set.

The Message

Trying to do things constructively whenever possible is absolutely the right method.

BUT: Always remember that our basic definitions to **not** require constructive solutions:

f is computable if there **exists** a Turing machine ...

No one says that you have to be able to construct the machine from assorted parameters.