Computing with infinite objects

Martín Escardó
University of Birmingham, UK

Midlands Graduate School (MGS) 2012, April, Birmingham, UK

Topics touched in these four lectures

1. Infinite searchable sets.
2. Searchable ordinals and co-induction.
3. Continuity and uniform continuity, topology.
4. Haskell, Agda, Gödel's system T, Martin-Löf type theory, constructive mathematics, classical higher-type computability theory.
5. Computing with real numbers represented as infinite sequences of digits.

Brief sketch of the lectures

1. Start with an illustrative example of a computation with infinite objects.

   We will develop a program in a terminating functional language.

   Gödel's system T (can be regarded as a fragment of Haskell, OCaml, . . .).

   It will decide whether infinitely many cases hold or find a counter-example.

2. Then we will study what is behind this, from various points of view.
   (Constructive mathematics, computability theory, topology, . . .)

3. And we will develop more examples,
   in various languages of increasing strength.

Our version of system T

1. Simply typed lambda calculus.
2. Function types, finite product types, finite sum types.
3. Base types for binary numbers and natural numbers.
4. The type of binary numbers has constants 0 and 1, and definition by cases.
5. The type of natural numbers has 0, successor, and definition by induction.
6. This is strongly normalizing. All computations terminate.

See the typed lambda calculus course for syntactical details.
In this course we will work informally but rigorously.

Some terminology and notation

0. Let $2 = \{0, 1\}$ be the type of binary numbers.

1. Let $2^{\mathbb{N}}$ be the function type $(\mathbb{N} \to 2)$, called the Cantor space.

2. Think of a function $x \in 2^{\mathbb{N}}$ as a sequence $x_0, x_1, x_2, x_3, \ldots, x_n, \ldots$

3. In particular, we are interested in the set of decreasing binary sequences.

4. They are of the form $1^n 0^\omega$ and 1^ω. (Warning: constructive trap.)

5. The count of 1's gives you an extended natural number.

6. Call this set $\mathbb{N}_\infty$.

A sample computation with infinite objects

A root of a function $p\colon 2^{\mathbb{N}} \to 2$ is any $x \in 2^{\mathbb{N}}$ such that $p(x) = 0$.

Theorem (Specification of the computation to be performed).

In system T, it is possible to

1. decide whether any given such p has a root in the set $\mathbb{N}_\infty$, and
2. in this case construct such a root.

A sample computation with infinite objects

A root of a function $p\colon 2^{\mathbb{N}} \to 2$ is any $x \in 2^{\mathbb{N}}$ such that $p(x) = 0$.

Theorem. In system T, it is possible to write programs to

1. decide whether any given such p has a root in the set $\mathbb{N}_\infty$, and
2. in this case construct such a root.

The point is that the set $\mathbb{N}_\infty$ is infinite.

So the decision involves infinitely many cases.

Can we really check all of them with a program, in any language at all?

A sample computation with infinite objects

A root of a function $p\colon 2^{\mathbb{N}} \to 2$ is any $x \in 2^{\mathbb{N}}$ such that $p(x) = 0$.

Theorem. In system T, it is possible to decide whether any given such p has a root in the set $\mathbb{N}_\infty$, and in this case construct such a root.

Remarks.

1. I could have formulated this as a theorem in the constructive system HA^ω, whose term language is system T.

2. Also, this can be formulated as a theorem in Martin-Löf type theory, and I have written down a proof in Agda notation.

3. It can also be formulated as a theorem in Bishop mathematics, and I have done this in a paper.

A sample computation with infinite objects

A root of a function $p\colon 2^{\mathbb{N}} \to 2$ is any $x \in 2^{\mathbb{N}}$ such that $p(x) = 0$.

Theorem. In system T, it is possible to decide whether or not any given such p has a root in the set $\mathbb{N}_\infty$, and if so construct such a root.

This formulation of the theorem is correct but misleading:

$\boxed{\text{The proof constructs } x \text{ such that } p \text{ has a root if and only if } x \text{ is a root.}}$

To perform the decision of the theorem, one then checks whether $p(x) = 0$.

In this case x is a root, which is already constructed.

Otherwise there is no root.

A computation with infinite objects

Given $p\colon 2^{\mathbb{N}} \to 2$, construct the binary sequence

$$\begin{array}{rcl} y_0 & = & p(0^\omega) \\ y_1 & = & p(10^\omega) \\ y_2 & = & p(110^\omega) \\ y_3 & = & p(1110^\omega) \\ & \vdots & \\ y_n & = & p(1^n 0^\omega) \\ & \vdots & \end{array}$$

A computation with infinite objects

Given $p\colon 2^{\mathbb{N}} \to 2$, construct a sequence

$$y_n \quad = \quad p(1^n 0^\omega) \quad = \quad p(\lambda i.\ \text{if } i < n \text{ then } 1 \text{ else } 0).$$

Now form a sequence

$$\begin{aligned} x_0 &= y_0, \\ x_{n+1} &= \min(x_n, y_{n+1}). \end{aligned}$$

Then

$$x_0 \geq x_1 \geq x_2 \geq x_3 \geq \cdots \geq x_n \geq x_{n+1} \geq \ldots$$

Hence x can be of one of the forms

$1^k 0^\omega$ or 1^ω.

A computation with infinite objects

Given $p\colon 2^{\mathbb{N}} \to 2$, construct sequences

$$\begin{aligned} y_n &= p(1^n 0^\omega), \\ x_n &= \min(y_0, y_1, y_2, \ldots, y_n). \end{aligned}$$

The sequence x is

$$\begin{array}{rll} 1^k 0^\omega & \text{if} & \forall n < k.\, y_n = 1 \text{ and } y_k = 0, \\ 1^\omega & \text{if} & \forall n.\, y_n = 1. \end{array}$$

A computation with infinite objects

Given $p\colon 2^{\mathbb{N}} \to 2$, construct a sequence

$$x_n \quad = \quad \min(p(0^\omega), p(10^\omega), p(110^\omega), \ldots, p(1^n 0^\omega)).$$

The sequence x is

$$\begin{array}{rll} 1^k 0^\omega & \text{if} & \forall n < k.\, p(1^n 0^\omega) = 1 \text{ and } p(1^k 0^\omega) = 0, \\ 1^\omega & \text{if} & \forall n.\, p(1^n 0^\omega) = 1. \end{array}$$

We now evaluate $p(x)$:

If we get 0, we have found a decreasing root of p.

If we get 1, then we know that $\forall k.\, p(1^k 0^\omega) = 1$ and $p(1^\omega) = 1$. Why?

Hence in this case we know that p has no decreasing root.

Summary of what we have proved

Let $\mathbb{N}_\infty \subseteq 2^{\mathbb{N}}$ denote the set of decreasing sequences.

Let $p\colon 2^{\mathbb{N}} \to 2$ be given.

Theorem
We can calculate a sequence $x \in \mathbb{N}_\infty$ such that p has a root in the set $\mathbb{N}_\infty$ if and only if x is a root.

Corollary
It is decidable whether $p\colon 2^{\mathbb{N}} \to 2$ has a root in the set $\mathbb{N}_\infty$.

Notice that the set $\mathbb{N}_\infty$ is infinite.
We've found a way of checking infinitely many cases in finite time.

A selection functional

From $p\colon 2^{\mathbb{N}} \to 2$ we have calculated $x \in 2^{\mathbb{N}}$.

So we have a functional $\varepsilon\colon (2^{\mathbb{N}} \to 2) \to 2^{\mathbb{N}}$, defined by

$$\varepsilon(p) = \lambda i.\ \min_{n \leq i} p(1^n 0^\omega),$$

such that

$$p(\varepsilon(p)) = 0 \iff \exists x \in \mathbb{N}_\infty.\, p(x) = 0,$$

and hence such that

$$p(\varepsilon(p)) = 1 \iff \forall x \in \mathbb{N}_\infty.\, p(x) = 1.$$

Because the conditions in the left-hand sides are decidable, so are the conditions in the right-hand sides.

Equivalent coinductive definition

First something that doesn't quite work:

$$\begin{array}{rll} \varepsilon(p) = & \text{if} & p(0^\omega) = 0 \text{ then } 0^\omega \\ \text{else} & \text{if} & p(10^\omega) = 0 \text{ then } 10^\omega \\ \text{else} & \text{if} & p(110^\omega) = 0 \text{ then } 110^\omega \\ \text{else} & \text{if} & p(1110^\omega) = 0 \text{ then } 1110^\omega \\ \text{else} & \text{if} & p(11110^\omega) = 0 \text{ then } 11110^\omega \\ & \dots & \end{array}$$

The problem is that the case 1^ω is never checked or produced.

Equivalent coinductive definition

With the following modification, the case 1^ω is not checked but is produced if no other decreasing sequence is a root:

$$\begin{array}{rll} \varepsilon(p) = & \text{if} & p(0^\omega) = 0 \text{ then } 0^\omega \\ \text{else} & 1 : \text{ if} & p(10^\omega) = 0 \text{ then } 0^\omega \\ \text{else} & 1 : \text{ if} & p(110^\omega) = 0 \text{ then } 0^\omega \\ \text{else} & 1 : \text{ if} & p(1110^\omega) = 0 \text{ then } 0^\omega \\ \text{else} & 1 : \text{ if} & p(11110^\omega) = 0 \text{ then } 0^\omega \\ & \dots & \end{array}$$

So to see whether there is a root, it remains to check $p(\varepsilon(p)) = 0$.

In finite form

$$\begin{aligned}
\varepsilon(p) = & \quad \text{if} \quad p(0^\omega) = 0 \text{ then } 0^\omega \\
\text{else} & \quad 1 : \text{ if} \quad p(10^\omega) = 0 \text{ then } 0^\omega \\
\text{else} & \quad 1 : \text{ if} \quad p(110^\omega) = 0 \text{ then } 0^\omega \\
\text{else} & \quad 1 : \text{ if} \quad p(1110^\omega) = 0 \text{ then } 0^\omega \\
\text{else} & \quad 1 : \text{ if} \quad p(11110^\omega) = 0 \text{ then } 0^\omega \\
& \quad \ldots
\end{aligned}$$

The finite folding of this is

$$\varepsilon(p) = \text{if } p(0^\omega) = 0 \text{ then } 0^\omega \text{ else } 1 : \text{ if } \varepsilon(\lambda x.\, p(1 : x)).$$

By induction, $\varepsilon(p)(i) = \min_{n \leq i} p(1^n 0^\omega)$, as above.

The point of the original definition is that it can be written in system T.

Using booleans rather than binary digits

Of course, rather than finding a root, as above, we can solve the satisfiability problem in the same manner:

$$\begin{aligned} \varepsilon &: (2^{\mathbb{N}} \to \mathrm{Bool}) \to 2^{\mathbb{N}} \\ \varepsilon(p) &= \text{if } p(0^\omega) \text{ then } 0^\omega \text{ else } 1 : \text{ if } \varepsilon(\lambda x.\, p(1:x)). \end{aligned}$$

Then $p(x) = \mathrm{True}$ for some $x \in \mathbb{N}_\infty$ if and only if $p(\varepsilon(p)) = \mathrm{True}$.

Searchable set

We say that a subset S of a type X is searchable if there is a function $\varepsilon\colon (X \to \mathrm{Bool}) \to X$ such that

1. $\varepsilon(p) \in S$ (so S must be non-empty),

2. $p(x) = \mathrm{True}$ for some $x \in S$ if and only if $p(\varepsilon(p)) = \mathrm{True}$.

Weaker notion of searchability

We say that a subset S of a type X is exhaustible if there is a function

$$E\colon (X \to \mathrm{Bool}) \to \mathrm{Bool}$$

such that $p(x) = \mathrm{True}$ for some $x \in S$ if and only if $E(p) = \mathrm{True}$.

This function doesn't tell what the witness x is.

Fact. Any searchable set is exhaustible.

Proof. Define $E(p) = p(\varepsilon(p))$.

Fact. We also get $A\colon (X \to \mathrm{Bool}) \to \mathrm{Bool}$ such that $p(x) = \mathrm{True}$ for all $x \in S$ if and only if $A(p) = \mathrm{True}$.

Proof. Define $A(p) = \neg E(\lambda x. \neg p(x))$.

Exercise

For subsets of the Cantor space $2^{\mathbb{N}}$, the weaker notion of searchability implies the stronger one.

This can be done in system T.

How about subsets of the Baire space $\mathbb{N}^{\mathbb{N}}$?

Solution in the next slide.

Solution

Define a functional $F\colon ((2^{\mathbb{N}} \to \mathrm{Bool}) \to \mathrm{Bool}) \to (2^{\mathbb{N}} \to \mathrm{Bool}) \to 2^{\mathbb{N}})$ as follows

$$\begin{array}{rcl} F(E)(p) = & \text{if} & E(\lambda x.\, p(0 : x)) \\ & \text{then} & 0 : F(E)(\lambda x.\, F(E)(\lambda x\colon p(0 : x))) \\ & \text{else} & 1 : F(E)(\lambda x.\, F(E)(\lambda x\colon p(1 : x))). \end{array}$$

Images of searchable sets

Haskell programs (in the system T fragment)

Are in separate files.

Let's look at them and run some experiments.

Foundation for computation with infinite objects

1. Constructive mathematics?

   Brouwer's intuitionism.
   Bishop mathematics.
   Martin-Löf type theory (Agda).
   Coquand's calculus of constructions (Coq).
   Aczel's constructive ZF theory of sets (interpretation in ML type theory).

2. Computability theory within constructive mathematics?

   Markov's school (Hyland's effective topos).

3. Computability theory within classical mathematics?

   Recursion theory.
   (Write programs in your favourite programming language,
   and use classical logic to reason about them.
   This is what most computer scientists do.)

Foundation for computation with infinite objects

1. Constructive mathematics?

2. Computability theory within constructive mathematics?

3. Computability theory within classical mathematics?

Cf. Richman and Bridges' book *Varieties of constructive mathematics*.

Foundation for computation with infinite objects

Important practical aspects of two constructive systems I particularly like:

1. Bishop mathematics.

   It is informal but rigorous in the usual mathematical sense.

   Every theorem and proof of Bishop mathematics is literally a theorem and proof in classical mathematics, and hence can be undertood by everybody.

2. Martin-Löf type theory.

   Also compatible with classical mathematics.

   Its theorems look (mostly) familiar to classical mathematicians.

   But their proofs look rather peculiar (they are functional programs).

   You can directly check and run Martin–Löf proofs in computers (Agda, Coq).

   It is claimed that Bishop mathematics can be formalized in ML type theory.

Technical questions now

Question 1. Given an arbitrary set $X \subseteq 2^{\mathbb{N}}$, can we decide whether any given function $p\colon 2^{\mathbb{N}} \to 2$ has a root in the set X, and if so exhibit one?

1. If X is finite, this this clear. Just check each case, one by one.
2. If X is the infinite set $\mathbb{N}_\infty$, this is possible, as we have seen.
3. For some infinite X, this amounts to a solution of the Halting problem.
4. If $X = 2^{\mathbb{N}}$, this cannot be done in system T.

   Or in Bishop mathematics, ML type theory, Markov's school.

   But can be done in a terminating extension of system T.

   And in Brouwer's constructive mathematics.

   What is going on here?

Questions

Question 1. Given a set $X \subseteq 2^{\mathbb{N}}$, can we decide whether the function p has a root in the set X, and if so exhibit one?

Refined questions:

Question 2. What are the subsets $X \subseteq 2^{\mathbb{N}}$ for which the previous question has a positive answer in all settings discussed above?

Question 3. Which kinds of subsets $X \subseteq 2^{\mathbb{N}}$ distinguish the various constructive settings in terms of the first question?

Topology

Topology is about spaces, limits, open sets, closed sets, continuous functions, compact sets, Hausdorff separation, connectivity, dimension, and much more.

And all this, including the “much more”, is naturally and neatly related to computation.

1. The crucial connection with continuity was discovered by Brouwer: “Computable functions” are necessarily continuous.

   Brouwer did this well before Turing machines, Church’s thesis, recursion theory, and electronic computers.

2. In turns out the “searchability” property is related to compactness.

Continuity in computation

Intuitively, a function is continuous iff every finite piece of information about the output of the function can depend only on a finite part of its input.

And this intuition is quite close to the rigorous mathematical definition.

In this sense, computable functions ought to be continuous.

For how could you, otherwise, completely survey the input of the function if it is infinite, in order to get a finite part of the output?

Continuity in computation

But then there is a disturbing aspect of our motivating construction.

To calculate a potential root $x \in \mathbb{N}_\infty$ of a given $p \colon 2^{\mathbb{N}} \to 2$, we built x by evaluating p at infinitely many arguments.

Namely $1^n 0^\omega$ for every natural number n.

Recall we checked whether $p(\lambda i.\ \min_{n \leq i} p(1^n 0^\omega)) = 0$ to know whether p has a root in $\mathbb{N}_\infty$.

This seems to contradict computability and our decision procedure:

1. The definition of invokes infinitely many values of p.

2. Hence it is dubious that the binary numeral $p(\lambda i.\ \min_{n \leq i} p(1^n 0^\omega))$ can be algorithmically calculated, unless the evaluation of $p(\lambda i.\ \min_{n \leq i} p(1^n 0^\omega))$ queries finitely many positions of the sequence $\lambda i.\ \min_{n \leq i} p(1^n 0^\omega)$.

Continuity in computation: solution to the apparent paradox

So, can we really assert, as we did, that finite parts of the output can depend only on finite parts of the input?

Yes, as follows:

The definition of x does invoke the values of p for infinitely many arguments of p.

However, the evaluation engine (executed by a person or by a computer), when it evaluates $p(x)$, to know whether p has a root, can only query finitely many positions of the sequence x.

We can refer to this as meta-theoretical continuity.

Can this be sensibly turned into object-level continuity?

Should this be done or avoided?

This completes the introduction to the course

1. Some infinite sets can be “surveyed” in finite time.

2. This can be undertood in various ways (Brouwer, Bishop, Martin-Löf, Markov, classical recursion theory, . . .).

3. This understanding helps us to figure out which infinite sets can(not) be surveyed in various conceptions of, or foundations for, computation.

This completes the introduction to the course

1. Some infinite sets can be “surveyed” in finite time.

2. This can be undertood in various ways (Brouwer, Bishop, Martin-Löf, Markov, classical recursion theory, . . .).

3. This understanding helps us to figure out which infinite sets can(not) be surveyed in various conceptions of, or foundations for, computation.

Remark

The terminology surveyable was suggested to me by Fred Richman.

I have used the terminology searchable in a number of papers.

Both terminologies are good (and also imperfect).

Exercises

Fix a natural number n.

1. The set of decreasing sequences of natural numbers bounded by n is searchable.

2. The set of binary sequences with at most n ones is searchable.

3. Write a program in system T (maybe using a practical functional language).

4. Come up with your own examples of searchable sets of sequences.

5. If you are familiar with any variety of constructive mathematics, write a proof in that variety.

6. Write formal proofs in e.g. Agda or Coq.

Searchable ordinals