When is universal quantification decidable?

Martín Escardó
University of Birmingham

Logic Seminar, Leeds, 23 Feb 2011

Organization of the talk

1. In the first 1/2 of the talk, I formulate questions.
2. In the following 1/4 of the talk, I formulate theorems answering them.
3. In the following 1/8 of the talk, I sketch proofs.
4. In the following 1/16 of the talk, I run programs trying to surprise you.

   What I say cannot only be done in principle, but also fast.
5. In the following 1/32 of the talk, I answer your questions.

   But I'd rather have you interrupting me at any point in the talk.

⋮

This will be a never-ending talk, but within the allocated time, thanks to Zeno.

Consider the following problem

Given $p\colon \mathbb{N} \to 2$, where $2 = \{0, 1\}$, decide whether or not

$$\forall n \in \mathbb{N}(p(n) = 1).$$

By reduction to the Halting problem, no algorithm solves this uniformly in p.

In terms of higher-type recursion, the required algorithm has type

$$\underbrace{(\mathbb{N} \to 2)}_{\text{input } p} \longrightarrow \underbrace{2}_{\text{answer yes/no}}$$

Consider the following seemingly harder problem

Given $p\colon 2^\omega \to 2$ continuous, decide whether or not

$$\forall \alpha \in 2^\omega (p(\alpha) = 1).$$

This time, there is an algorithm that solves this problem uniformly in p.

The problem is actually easier! But why? What is the algorithm?

In terms of higher-type recursion, the required algorithm has type

$$\underbrace{((\mathbb{N} \to 2) \to 2)}_{\text{input } p} \longrightarrow \underbrace{2}_{\text{answer yes/no}}$$

Impossible again

Given $p\colon \mathbb{N}^\omega \to 2$ continuous, decide whether or not

$$\forall \alpha \in \mathbb{N}^\omega (p(\alpha) = 1).$$

There is no algorithm that solves this uniformly in p.

In terms of higher-type recursion, the required algorithm has type

$$\underbrace{((\mathbb{N} \to \mathbb{N}) \to 2)}_{\text{input } p} \longrightarrow \underbrace{2}_{\text{answer yes/no}}$$

Making the problem more complex

Given $p\colon ((((\mathbb{N} \to 2) \to \mathbb{N}) \to 2) \to 2) \to 2$ continuous, decide whether or not

$$\forall F\colon ((\mathbb{N} \to 2) \to \mathbb{N}) \to 2) \to 2(p(F) = 1).$$

There is an algorithm that solves this uniformly in p.

Make a little change: $p\colon ((((\mathbb{N} \to 2) \to \mathbb{N}) \to 2) \to \mathbb{N}) \to 2$

Still there is an algorithm.

Make a little change: $p\colon ((((\mathbb{N} \to 2) \to \mathbb{N}) \to \mathbb{N}) \to 2) \to 2$.

Now there is no algorithm.

Yet another instance of the problem we study

Given $p\colon \mathbb{N}^\omega \to 2$ continuous, and $S \subseteq \mathbb{N}^\omega$, decide whether or not

$$\forall \alpha \in S(p(\alpha) = 1).$$

For some subsets S, there is an algorithm that solves this uniformly in p.

E.g. $S = \{0,1\} \times \{0,1,2\} \times \cdots \times \mathbb{Z}_n \times \cdots \subseteq \mathbb{N}^\omega$.

For others, there isn't.

E.g. $S = \mathbb{N}^\omega$, as we have mentioned.

General problem considered in this work

Problem. Given a Kleene–Kreisel type X, a set $S \subseteq X$, and a continuous $p\colon X \to 2$, decide whether or not

$$\forall x \in S(p(x) = 1).$$

Question. We ask which sets $S \subseteq X$ admit universal decision algorithms:

$$A_S\colon (X \to 2) \to 2$$

This is equivalent to asking which sets S admit existential decision algorithms:

$$E_S\colon (X \to 2) \to 2$$

Definition. We call exhaustible the sets S for which such algorithms exist.

This can be seen as an exhaustive search problem

The main difference is that we move from finite to infinite sets.

There are two general forms of the exhaustive search problem:

1. Given $p\colon X \to 2$ and $S \subseteq X$, determine whether or not there is some $x \in S$ with $p(x) = 1$.

   This has type $(X \to 2) \to 2$.

2. Given $p\colon X \to 2$ and $S \subseteq X$, find $x \in S$ such that $p(x) = 1$, or tell there isn't any such x.

   This has type $(X \to 2) \to X + 1$.

They often get confused.

Presumably because they are evidently equivalent in the finite case.

Terminology

1. Given $p\colon X \to 2$ and $S \subseteq X$, determine whether or not there is some $x \in S$ with $p(x) = 1$.

   This is done by an algorithm $E_S\colon (X \to 2) \to 2$.

   Definition. If such an algorithm exists, the set S is called exhaustible.

2. Given $p\colon X \to 2$ and $S \subseteq X$, find $x \in S$ such that $p(x) = 1$, or tell there isn't any such x.

   This is done by an algorithm $\varepsilon\colon (X \to 2) \to X + 1$.

   Definition. If such an algorithm exists, the set S is called searchable.

Another natural question

Searchability is stronger than exhaustibility, at least in principle.

This is because searchability asks for witnesses, and exhaustibility for yes/no.

Question. Are there exhaustible sets that are not searchable?

If every exhaustible set is also searchable,
a stronger question is whether there is an algorithm

$$\Phi\colon ((X \to 2) \to 2) \longrightarrow ((X \to 2) \to X + 1)$$

such that

$$\Phi(E_S) = \varepsilon_S.$$

If we can answer yes/no questions about existential quantification,
can we automatically get witnesses, when they exist?

Variation of the notion of searchable set

We postulated an algorithm

$$\varepsilon_S \colon (X \to 2) \to X + 1.$$

Sometimes it is technically more convenient to instead have (a total)

$$\varepsilon_S \colon (X \to 2) \to X.$$

This second version forces ε_S to tell lies sometimes.

But they can be detected by checking

$$p(\varepsilon_S(p)) = 1.$$

Only for the empty set $S = \emptyset$ the two variations don't agree.

Official definition of searchable set

Definition. A set $S \subseteq X$ is searchable if there is an algorithm

$$\varepsilon_S \colon (X \to 2) \to X$$

such that

1. $\varepsilon(p) \in S$, and

2. If there is $x \in S$ such that $p(x) = 1$, then $p(\varepsilon_S(p)) = 1$.

This definition is equivalent, except for the empty set.
This because the first condition can't hold for the empty set.

This is good, and we'll see why.

Questions

So far we've asked, explicitly or implicitly:

1. Which sets are exhaustible? Characterize exhaustible sets.

2. Which sets are searchable? Characterize searchable sets.

3. Are there exhaustible sets that are not searchable? Do characterizations help?

We can also ask:

4. Are there ways of systematically constructing a rich supply of exhaustible and searchable sets? Establish closure properties.

5. Can we systematically construct all of them? Constructively characterize.

These two kinds of questions 1-3 and 4-5 turn out to be intimately related.

We are almost ready to formulate a theorem

But what do we mean by the type hierarchy

2

$\mathbb{N}$,

$\mathbb{N} \to \mathbb{N}$

$(\mathbb{N} \to 2) \to \mathbb{N}$

$(\mathbb{N} \to 2) \to \mathbb{N}) \to 2$

$\vdots$

What are we talking about?

The Kleene–Kreisel continuous functionals

We are talking about the Kleene–Kreisel spaces of continuous functionals.

Kleene and Kreisel gave two competing definitions in the 1950's.

They were eventually understood to be equivalent.

Many more equivalent definitions have been proposed, giving different insights.

See Dag Normann's "50 years of continuous functionals" invited CiE'2008 talk.

We are interested in two particular characterizations

One, due to Hyland, in the early 1970's.

The Kleene–Kreisel spaces are certain compactly generated Hausdorff spaces.

Classical topology.

Another, due to Ershov, again in the early 1970's.

The Kleene–Kreisel spaces arise from Scott domains.

Domain theory.

For the formulations of our theorems, Hyland's characterization is neater.

For the proofs of our theorems, the two characterizations are essential.

Moreover, the proofs crucially rely on the deep relationship between them.

Towards Hyland's characterization

The category of compactly generated Hausdorff spaces is cartesian closed:

It has finite products $I \times X$, for any I and X.

Kelleification of the product topology.

It has function spaces $(X \to Y)$, for any X and Y.

Kelleification of the compact-open topology on the continuous maps.

They interact in the right way with products:

$(I \times X \to Y) \cong (I \to (X \to Y))$. Currying and uncurrying.

This theorem is due to Hurewicz, and written down by Kelley, in the 1950's, and popularized by Steenrod in the 1960's.

Their motivation: homotopies are paths in function spaces.

Hyland's characterization

Theorem. The Kleene–Kreisel spaces form the least collection of compactly generated Hausdorff spaces

1. containing the discrete spaces 2 and $\mathbb{N}$,

2. closed under finite products and function spaces.

The theorem is stated in Hyland's PhD thesis, 1970's.

A full proof is given by Dag Normann in "Recursion on the countable functionals", Springer LNM 811, 1980.

Our theorems hold in more generality

They apply to the least collection of compactly generated spaces

1. containing the discrete finite and countable spaces,

2. closed under finite and countably infinite products,

3. closed under function spaces,

4. closed under retracts.

These spaces are contained in the larger collection of QCB spaces.

Proposed by A. Simpson and M. Schroeder for semantics and computability.

QCB spaces form the largest known category of spaces that
admit a (rather natural) theory of computability, including Scott domains.

Theorems about searchable and exhaustible sets

1. Finite sets are both exhaustible and searchable, of course.

2. Exhaustible and searchable sets are closed under

   (a) finite products,
   (b) computable images,
   (c) computable retracts.

3. Searchable sets are exhaustible, trivially.

4. Searchable sets are closed under countable products.

5. The Cantor space 2^ω is searchable, as a consequence.

6. Non-empty exhaustible sets are computable images of the cantor space 2^ω.

7. Non-empty exhaustible sets are searchable, as a consequence.

How are these theorems proved?

Firstly, there is a topological

Crucial Lemma. Exhaustible sets are topologically compact.

Hence so are searchable sets.

Then there is the

Crucial insight. The converse must be true, to the extent it can be true.

This is why Hyland's topological characterization is pivotal to this development.

In fact, before we proceed, see what the theorems looks like if we replace "exhaustible" and "searchable" by compact, and make minor, natural adjustments.

Well known topological facts

1. Finite sets are compact.

2. Compact sets are closed under

   (a) arbitrary products,
   (b) continuous images,
   (c) retracts.

3. The Cantor space 2^ω is compact, as a consequence.

4. Non-empty, second countable, compact Hausdorff spaces are computable images of the cantor space 2^ω.

The above computational theorems mimick these topological theorems.

But they have a subtle aspect: they take into account the potential difference between "exhaustible" and "searchable", and resolve it.

We turn topological theorems into computational theorems

And, moreover, our proofs apply non-trivial topological technology to prove the computational theorems. Check papers to see how this works.

So there are two uses of topology: inspirational and technical.

But the proofs use deep computational theorems too, in particular the Kleene–Kreisel density theorem, among others.

More aspects of the proofs

Ultimately, the computational version of the topological theorem amounts to writing down algorithms.

I will not write them down in these slides.

But I will write down their types
and maybe write some of the algorithms in the board if there is time.

Useful notation

$J(X) = ((X \to 2) \to X)$

Type of search operators.

$K(X) = ((X \to 2) \to 2.$

Type of exhaustion operators.

Related by

$$\begin{array}{rcl} JX & \to & KX \\ \varepsilon & \mapsto & \overline{\varepsilon} \\ \varepsilon & \mapsto & \lambda p.p(\varepsilon(p)). \end{array}$$

One question (answered) was whether we can go in the opposite direction.

Computable images

You are given $f : X \to Y$.

Computable images of exhaustible sets are exhaustible

$$\begin{array}{rcl} ((X \to 2) \to 2) & \to & ((Y \to 2) \to 2)) \\ KX & \to & KY. \end{array}$$

Computable images of searchable sets are searchable

$$\begin{array}{rcl} ((X \to 2) \to X) & \to & 9(Y \to 2) \to Y)) \\ JX & \to & JY. \end{array}$$

Finite products

The product of two exhaustible sets is exhaustible

$$\begin{array}{rcl}((X \to 2) \to 2) \times ((Y \to 2) \to 2) & \to & ((X \times Y \to 2) \to 2) \\ KX \times KY & \to & K(X \times Y).\end{array}$$

The product of two searchable sets is searchable

$$\begin{array}{rcl}((X \to 2) \to X) \times ((Y \to 2) \to Y) & \to & ((X \times Y \to 2) \to X \times Y) \\ JX \times JY & \to & J(X \times Y).\end{array}$$

Countably infinite products

The product of countably many serchable sets is searchable

$$\prod_i((X_i \to 2) \to X_i) \quad \to \quad (\prod_i X_i \to 2) \to \prod_i X_i)$$

$$\prod_i J(X_i) \quad \to \quad J(\prod_i X_i).$$

There is no corresponding functional for exhaustible sets.

The crux of the problem are empty sets. Problem with continuity.

This explains the somewhat convoluted formulation of the computational theorems that correspond to topological theorems.

One more topological input to computation

In topology, we have that if C is compact and D is discrete, then

1. $(C \to D)$ is discrete.

2. $(D \to C)$ is compact.

Define the following types by induction:

$$\begin{array}{rcl} \text{discrete} & ::= & \mathbb{N} \mid 2 \mid \text{discrete} \times \text{discrete} \mid \text{compact} \to \text{discrete}, \\ \text{compact} & ::= & 2 \mid \text{compact} \times \text{compact} \mid \text{discrete} \to \text{compact}\,. \end{array}$$

Theorem. The compact types are exhaustible and searchable, and the discrete types have decidable equality.

Efficiency and complexity

One may have the impression that the preceding results would be of theoretical interest only. But:

1. There are fast algorithms.
   (a) And some general techniques to get them fast,
   (b) and for quantifying their efficiency.

2. There are interesting applications
   (a) integration when real numbers are represented by infinite sequences of digis,
   (b) automatics program verification.

3. This work has been implemented in practice.

4. We have run-time complexity theorems.

Two main references, among others

1. Infinite sets that admit fast exhaustive search, LICS'2007.

2. Exaustible sets in higher-type computation, LMCS'2008.

Ramifications of this work, joint with Oliva

Miracle! The program that implements Tychonoff does much more.

Game theory. Optimal strategies.

Proof theory. Computational content of classical choice.

And many other exciting things, both done and under development.

Very fruitful ramification.

Moreover, much better understanding in a more general setting.

Very fruitful collaboration too! 10 references, where 8 of them are published in conferences and journals, and many more to come, under development.

Summary

We have answered the question

Which kinds of infinite sets admit exhaustive search?

And related questions too.

The answers are motivated by topology,
but can be formulated in purely computational terms,
although the proofs crucially rely on topology.