Topology of data types

and the computational content of topology

Martín Escardó

School of Computer Science, University of Birmingham, UK

LFCS seminar, Edinburgh, 14th Oct 2003

Overview

This is based on my recent paper

Topology of data types and computability concepts.

Two complementary issues are addressed:

1. Use of topological technology to explain computational phenomena.

2. Use of computational technology to explain topological phenomena.

I hope we have a good mix of people with and without background on topology in the audience.

History

In intuitionistic mathematics:

Theorem (Brouwer 1920's) *All functions $f : \mathbb{R} \to \mathbb{R}$ are continuous.*

Exported to classical mathematics, Brouwer's arguments prove:

Theorem (Folklore) *Computable functions $f : \mathbb{R} \to \mathbb{R}$ are continuous.*

More generally

Work on

> intuitionistic and constructive mathematics,
> recursion theory,
> domain theory, programming-language semantics,
> type-two theory of effectivity

has led to

Folklore Data types are topological spaces and computable functions are continuous.

Moreover, the topologies that arise are familiar.

> (E.g. real line and Cantor space.)

When topology plays no role in computation

Most of the time, actually.

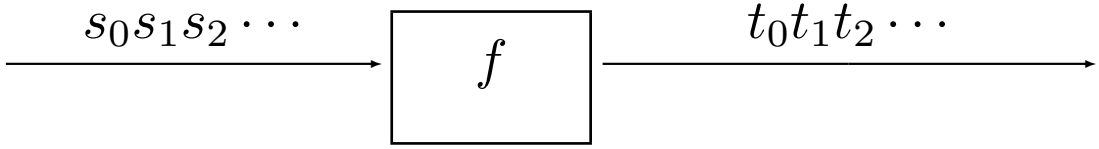
That's because most data types that one encounters in practice are discrete.

Topology becomes non-trivial and significant in the presence of (potentially) infinite data.

Typical examples and applications occur in higher-type computation, process algebra and exact real-number computation.

Example — the Cantor space

Consider computations of functions $f\colon 2^\omega \to 2^\omega$ where $2 = \{0, 1\}$:



$$s = s_0s_1s_2\cdots \qquad t = t_0t_1t_2\cdots$$
$$f(s) = t.$$

The black box alternates between reading some digits from the input, performing some internal computations and writing some digits to the output.

Functions of finite character

Computable functions $f : 2^\omega \to 2^\omega$ have to be of *finite character*:

> Finitely many digits of the output can depend only on finitely many digits of the input.

Theorem (Folklore) *Endow 2 with the discrete topology and 2^ω with the product topology.*

Then $f : 2^\omega \to 2^\omega$ is of finite character iff it is continuous.

The space 2^ω is known as the *Cantor space*.

Sample application

Application Compactness of the Cantor space allows one to conclude that the data type of integer valued functions on the Cantor space has decidable equality.

This will be elaborated after we have developed some machinery.

The Sierpinski space

From a computational point of view, this is the space of results of observations or semidecisions:

$$\mathbb{S} = \{\bot, \top\}, \text{ with open sets } \emptyset, \{\top\}, \{\bot, \top\},$$

$\top = \mathrm{true},$	observable,	terminating computation,
$\bot = \mathrm{false},$	not observable,	non-terminating computation.

The asymmetric nature of semidecisions is reflected in the asymmetry of the topology:

$\{\top\}$ is open but $\{\bot\}$ is not.

(Otherwise we would be able to solve the halting problem.)

Topological notions via the Sierpinski space

Well known and easy:

Lemma (Sierpinski) *A subset U of a topological space X is open iff its characteristic function is continuous.*

$$\begin{array}{rcl} \chi_U \colon X & \to & \mathbb{S} \\ x & \mapsto & [\![x \in U]\!] = \begin{cases} \top & \text{if } x \in U, \\ \bot & \text{if } x \notin U. \end{cases} \end{array}$$

This provides a fundamental link with computability:

U is semidecidable if and only if χ_U is computable.

Hausdorff, discrete and compact spaces

Perhaps not so well known:

> The notions of Hausdorff, discrete, and compact space can also be reduced to continuity of certain maps involving the Sierpinski space.

Replacing continuity by computability, we get

> computationally Hausdorff,
>
> computationally discrete,
>
> computationally compact.

Hausdorff separation via Sierpinski

Lemma *A space X is Hausdorff iff its* apartness map *is continuous.*

$$\begin{array}{rcl}(\neq)\colon X \times X & \to & \mathbb{S} \\ (x, y) & \mapsto & [\![x \neq y]\!].\end{array}$$

We say that X is *computationally Hausdorff* if this map is computable.

For example, the Cantor space and the real line are computationally Hausdorff.

Discreteness via Sierpinski

The notions of discrete and Hausdorff space turn out to be dual:

Lemma *A space X is discrete iff its* equality map *is continuous.*

$$\begin{array}{rcl}(=)\colon X \times X & \to & \mathbb{S} \\ (x, y) & \mapsto & [\![x = y]\!].\end{array}$$

The Cantor space and the real line are not discrete.

Hence they are not computationally discrete.

Interesting: There are computationally discrete, non-computationally Hausdorff spaces.

Compactness via Sierpinski

Let $(X \to Y)$ denote the set of continuous maps from X to Y topologized in the *natural* way. (Don't worry too much about what the natural way is.)

The following remarkable fact is a reformulation of a well known property of function-space topologies:

Lemma *A subset Q of a topological space X is compact iff its universal-quantification functional is continuous.*

$$\begin{array}{rccl} \forall_Q : & (X \to \mathbb{S}) & \to & \mathbb{S} \\ & p & \mapsto & [\![\forall x \in Q.p(x) = \top]\!] \end{array}$$

Compactness generalizes finiteness

Because binary disjunction is continuous (and computable) any finite space is (computationally) compact:

$$\forall_{\{x_1,\ldots,x_n\}}(p) = p(x_1) \wedge \cdots \wedge p(x_n).$$

Perhaps counter-intuitively, the are computationally compact spaces of infinite (indeed uncountable) cardinality.

The Cantor space is one.

Computational compactness of the Cantor space

A program in Haskell:

```
data S = T                              -- Sierpinski-space machinery

(/\) :: S -> S -> S
T /\ T = T

ifs :: (S,a) -> a
ifs(T,x) = x

data Two = Zero | One                   -- Cantor-space machinery
type Cantor = [Two]

forall :: (Cantor -> S) -> S
forall(p) = p(ifs(forall(\s -> p(Zero : s)), arbitrary))
          /\ p(ifs(forall(\s -> p(One  : s)), arbitrary))
          where arbitrary = Zero : arbitrary   -- say
```

A classically invisible notion

As Hausdorff is dual to discrete, compactness should be dual to something.

Lemma *For $F \subseteq X$, the existential quantification functional is always continuous.*

$$\begin{array}{rccl} \exists_F : & (X \to \mathbb{S}) & \to & \mathbb{S} \\ & p & \mapsto & [\![\exists x \in F. p(x) = \top]\!]. \end{array}$$

But it is not always computable.

E.g. consider $X = \mathbb{N}$. Then $\exists_F$ is computable iff F is r.e.

Cf. Joyal's open locales and Taylor's overt spaces in ASD.

Topology via the λ-calculus

We use the above five lemmas to

1. easily develop basic topology and

2. extract computational content from the theorems.

Topological technique: A function which is λ-definable from continuous functions is itself continuous.

This is not the whole story, but it suffices for the purposes of this talk.

Computational technique: A function which is λ-definable from computable functions is itself computable.

Proposition *If X is Hausdorff and $Q \subseteq X$ is compact, then Q is closed.*

Proof Because $x \notin Q \iff \forall y \in Q.x \neq y$, the characteristic function of the complement of Q is λ-definable from continuous functions as $\chi_{X \setminus Q}(x) = \forall_Q(\lambda y.x \neq y)$. Q.E.D.

Thus, both the *formulation* of the classical proposition and its *proof* are seen to have computational content.

Computational reading of the formulation: If we can computationally tell points of X apart and we can computationally quantify over Q, then we can computationally semidecide the complement of Q.

Computational reading of the proof: Programs for the first two tasks give a program for the third.

Proposition *If X is compact and $F \subseteq X$ is closed then F is compact.*

Proof We λ-define $\forall_F \colon \mathbb{S}^X \to \mathbb{S}$ from continuous maps.

Notice that $\forall x \in F.p(x)$ iff $\forall x \in X.x \in F \implies p(x)$ iff $\forall x \in X.x \notin F \vee p(x)$.

Hence $\forall_F(p) = \forall_X(\lambda x.\chi_{X \setminus F}(x) \vee p(x))$.

$(- \vee -) \colon \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ is the evident (continuous) disjunction map. Q.E.D.

Computationally, this has the drawback of relying on the so-called "weak parallel-or". (This is the only use of parallel-or in this talk.)

In any case, it says that if we can computationally quantify over X and semidecide the complement of F, then we can also computationally quantify over F.

Proposition *If $f : X \to Y$ is continuous and $Q \subseteq X$ is compact then $f(Q)$ is compact.*

Proof $\forall y \in f(Q).p(y)$ iff $\forall x \in Q.p(f(x))$. Q.E.D.

Proposition *If X and Y are compact then so is $X \times Y$.*

Proof $\forall z \in X \times Y.p(z)$ iff $\forall x \in X.\forall y \in Y.p(x, y)$. Q.E.D.

Magic? No. Cut elimination gives the familiar proofs.

Examples involving function spaces

Proposition *If Y is Hausdorff then so is the function space Y^X.*

Proof $f \neq g$ iff $\exists x \in X.f(x) \neq g(x)$. Q.E.D.

The λ-proof of the above proposition suggests the following dual version.

A dual theorem

We start from the proof and then try to figure out what it proves:

$$f = g \text{ iff } \forall x \in X.f(x) = g(x).$$

To get continuous equality on the function space, we need continuous equality on Y and continuous universal quantification over X.

Proposition *If X is compact and Y is discrete, then the function space Y^X is discrete.*

Corollary *For any Hausdorff and discrete data type D, the data type of D-valued functions on the Cantor space has decidable equality.*

To get *sequentially* decidable equality, more work is needed (cf. Berger).

A dual Tychonoff theorem?

The Tychonoff theorem gives:

$$Y \text{ compact}, \quad X \text{ discrete} \implies Y^X \text{ compact}.$$

We have just proved a symmetric consequence:

$$Y \text{ discrete}, \quad X \text{ compact} \implies Y^X \text{ discrete}.$$

It would be interesting to formulate this as a consequence of a "dual" Tychonoff theorem, but we don't know what such a theorem would state.

The topology of Y^X

Notice that we didn't need to know what the topology of Y^X is in the proofs of the above two propositions!

Everything is encapsulated in the proofs of continuity of $\forall$ and $\exists$

(which will not be provided in this talk).

In fact, we can easily construct plenty of open sets of Y^X using λ-technology and the quantifiers.

This is what we do next.

Observing functions

How does one observe a function?

A simple idea is that we evaluate it for a particular input and then check whether its output lands in a given open set: $f(x) \in V$.

But we can do better than that:

Proposition *If $Q \subseteq X$ is compact and $V \subseteq Y$ is open then the set*

$$N(Q,V) = \{f \in Y^X \mid f(Q) \subseteq V\}$$

is open in Y^X.

Proof Because $f \in N(Q,V)$ iff $\forall x \in Q.f(x) \in V$, we conclude that $\chi_{N(Q,V)}(f) = \forall_Q(\lambda x.\chi_V(f(x)))$. Q.E.D.

A computable functional

Proposition *If X is compact then the supremum functional is continuous.*

$$\begin{array}{rcl} \sup\colon \mathbb{R}^X & \to & \mathbb{R} \\ f & \mapsto & \sup f(X). \end{array}$$

Proof λ-define the characteristic functions of $\sup^{-1}(-\infty, b)$ and $\sup^{-1}(a, \infty)$ using the existential and universal quantifiers. Q.E.D.

Here we get algorithms for semideciding the relations $a < \sup f$ and $\sup f < b$, which shows that $\sup$ is computable.

To get a *sequential* algorithm, more work is needed (cf. Simpson).

The Kuratowski–Mrova theorem

Recall that a function is called *closed* if it maps closed sets to closed sets.

Theorem *A space X is compact iff, for every space Y, the projection $\pi\colon X \times Y \to Y$ is closed.*

Proof π is closed iff $W \in \mathcal{O}(X \times Y)$ implies $Y \setminus \pi(X \times Y \setminus W) \in \mathcal{O}\,Y$.

But $Y \setminus \pi(X \times Y \setminus W) = \{y \in Y \mid \forall x \in X.(x, y) \in W\}$.

This immediately gives $(\Rightarrow)$.

To prove $(\Leftarrow)$, choose $Y = \mathbb{S}^X$ and $W = \{(x, p) \in X \times \mathbb{S}^X \mid p(x) = \top\}$.

Then $Y \setminus \pi(X \times Y \setminus W) = \{p \in \mathbb{S}^X \mid \forall x \in X.p(x) = \top\} = (\forall_X)^{-1}(\top)$.

This shows that $\forall_X\colon \mathbb{S}^X \to \mathbb{S}$ is continuous. Q.E.D.

More computer programs

Omitted. E.g. Countable Tychonoff theorem:

Proposition *A product of an r.e. sequence of computationally compact spaces is itself computationally compact.*

We have a two-line Haskell program that produces the quantifier of the product from the quantifiers of the factors.

(With a three-page proof, which uses denotational semantics and the classical Tychonoff theorem in order to argue that the program has the correct termination property. NB. For compactness of the Cantor space, we have an operational proof.)

Summary

Data types are topological spaces.

Semidecidable sets are open, and computable functions are continuous.

Data types with semidecidable equality are discrete.

Data types with semidecidable apartness are Hausdorff.

Data types with computable universal quantifiers are compact.

The classical theorems hold for the computational notions.

Their proofs are literally programs.

The programs also take care of the classical case!

Concluding remarks

Topology via first-order logic cf. Vickers' work.

Synthetic topology of data types

Start from an undefined notion of "continuity".

Use the λ-calculus.

Many of the definitions of topology arise as theorems.

If we don't have parallel-or, we don't lose much.

So even e.g. game semantics can be said to have topological content.

Data language for a base programming language.