Synthetic topology
of program types and classical spaces

Martín Escardó

School of Computer Science, University of Birmingham, UK

Oxford Advanced Seminar on Informatic Structures
CONCEPTS OF SPACE
Michaelmas 2004, Friday 22 October

What is topology?

Geometry

invariance under continuous deformations.

Analysis

approximation, limits, closed sets, continuous functions, . . .

We are interested in the analytic aspect.

(But the geometric aspect is relevant too, e.g. fractals, image compression, digital topology, concurrency via homotopy,. . .)

Why topology in computation?

It gives insight regarding the nature of *infinite* computational objects.

Computers are finite, but the universe of discourse has infinite entities.

Syntax: loops, recursion.

Time: non-terminating computations.

Data: stream and higher-type computation.

Precision: real-number computation.

Abstraction: probabilistic descriptions of computational processes, continuous descriptions of digital images.

Illustration with a non-trivial computational problem

Computational set-up. Consider e.g. the programming language PCF.

(Simply typed λ-calculus with $\mathbb{N}$ and recursion.)

We include

Sierpinski type $\mathbb{S}$ of "results of semidecisions", with elements

$\top$,	"observable true",	terminating computation,
$\bot$,	"unobservable false",	non-terminating computation.

A computational problem

If Q is a finite set of programmable elements $x_1, \ldots, x_n$ of a type σ then

$$\begin{array}{rccl} \forall_Q\colon & (\sigma \to \mathbb{S}) & \to & \mathbb{S} \\ & p & \mapsto & \text{truth value of } \text{“}\forall x \in Q.p(x)\text{”} \end{array}$$

is clearly programmable:

$$\forall_Q(p) = p(x_1) \wedge \cdots \wedge p(x_n).$$

Problem I. Are there σ and infinite Q such that $\forall_Q$ is programmable?

A more ambitious computational problem

Consider the product type $\sigma^\omega \stackrel{\text{def}}{=} (\mathtt{nat} \to \sigma)$.

Problem II. Is there a programmable "fold" function

$$\begin{array}{lccc} t\colon & ((\sigma \to \mathbb{S}) \to \mathbb{S})^\omega & \to & ((\sigma^\omega \to \mathbb{S}) \to \mathbb{S}) \\ \text{s.t.} & (\forall_{Q_i})_{i\in\omega} & \mapsto & (\forall_{\prod_{i\in\omega} Q_i}) \quad ? \end{array}$$

That is, given the quantifiers for the factors, can we construct the quantifier for the product?

N.B. We don't care what the functional t does when its input is not a sequence of universal quantification functionals.

Positive answer to II gives positive answer to I

Consider $\sigma = \mathtt{nat}$.

$$\begin{array}{rccc} t\colon & ((\mathtt{nat} \to \mathbb{S}) \to \mathbb{S})^\omega & \to & ((\mathtt{nat}^\omega \to \mathbb{S}) \to \mathbb{S}) \\ & (\forall_{Q_i})_{i\in\omega} & \mapsto & (\forall_{\prod_{i\in\omega} Q_i}). \end{array}$$

Taking $Q_i = 2 \stackrel{\text{def}}{=} \{0,1\}$, we get $t\,((\forall_2)_{i\in\omega}) = (\forall_{2^\omega})$.

Hence the infinite set 2^ω admits computable universal quantification.

Corollary *The type* $((\mathtt{nat} \to \mathtt{bool}) \to \mathtt{nat})$ *has semidecidable equality for hereditarily total functionals.* (More about this later.)

Positive solution to Problem II

Use topology with the aid of the Scott model of PCF.

Rather brief sketch. Use the following two lemmas and a bit more work to derive the program t and prove its correctness.

Lemma (Synthetic formulation of compactness)

$\forall_Q : (D \to \mathbb{S}) \to \mathbb{S}$ *is continuous iff* Q *is a topologically compact subspace of the domain* D *under the Scott topology.*

Lemma (Tychonoff) *A product of compact spaces is compact.*

Sketch of solution

We want an algorithm to compute a functional t such that

$$\begin{array}{rccc} t\colon & ((D \to \mathbb{S}) \to \mathbb{S})^\omega & \to & ((D^\omega \to \mathbb{S}) \to \mathbb{S}) \\ & (\forall_{Q_i})_{i\in\omega} & \mapsto & (\forall_{\prod_{i\in\omega} Q_i}). \end{array}$$

Non-solution:
$\forall \vec{x} \in \prod_{i\in\omega} Q_i.\, p(\vec{x}) = \forall x_0 \in Q_0\, \forall x_1 \in Q_1\, \forall x_2 \in Q_2 \ldots p(x_0, x_1, x_2, \ldots).$

Truncate this to a finite iteration of the quantifiers:

$$t_n\colon \quad ((D \to \mathbb{S}) \to \mathbb{S})^\omega \quad \to \quad ((D^\omega \to \mathbb{S}) \to \mathbb{S})$$

so that, for all $\vec{\Phi} \in ((D \to \mathbb{S}) \to \mathbb{S})^\omega$ and $p \in (D^\omega \to \mathbb{S})$,

$$t(\vec{\Phi})(p) = \exists n.t_n(\vec{\Phi})(p).$$

Sketch of solution concluded

- To compute this, we use search (existential quantifier).
- To show that the search terminates, we use the Tychonoff theorem.
- For this, we first have to apply the synthetic characterization of compactness.

Aside. From the assumption that the search terminates, we conclude that the countable version of the Tychonoff theorem holds, at least for second countable spaces.

All details of the construction and proof

See Chapter 13 of the monograph

> M.H. Escardó. *Synthetic topology of data types and classical spaces*, vol. **87** of ENTCS, in press, 150pp.

Part I Topology of data types.

Part II Topology of classical spaces.

Part III Domain theory, topology and denotational semantics.

Sketch in more detail

Consider

- an infinite sequence Q_i of compact subsets of a domain D (with $\bot$), and
- a predicate $p \in (D^\omega \to \mathbb{S})$ s.t. $\forall \alpha \in \prod_i Q_i . p(\alpha) = \top$.

1. By continuity of p, for each $\alpha \in \prod_i Q_i$, there is $\beta = \beta(\alpha) \ll \alpha$ s.t. already $p(\beta) = \top$.

2. Then $\{\Uparrow \beta(\alpha) \mid \alpha \in \prod_i Q_i\}$ is an open cover of $\prod_i Q_i$, and, by Tychonoff, there is a finite subset F of $\prod_i Q_i$ s.t. $\{\Uparrow \beta(\alpha) \mid \alpha \in F\}$ already covers $\prod_i Q_i$.

3. If $\beta \ll \alpha$ then $\beta_i = \bot$ for all but finitely many $i \in \omega$.

Denote by $|\beta|$ the smallest k s.t. $\beta_i = \bot$ for all $i \geq k$.

4. Let $n = \max\{|\beta(\alpha)| \mid \alpha \in F\}$.

5. Then "$\forall \alpha \in \prod_i Q_i . p(\alpha) = \top$" can be finitely computed as "$\forall \alpha_0 \in Q_0 \forall \alpha_1 \in Q_1 \ldots \forall \alpha_n \in Q_n p(\alpha_0 \alpha_1 \ldots \alpha_n \bot \bot \bot \ldots)$"

6. Parallel computational strategy

Try $k = 0$, $k = 1$, $k = 2$,. . . in parallel until one of the computations

"$\forall \alpha_0 \in Q_0 \forall \alpha_1 \in Q_1 \ldots \forall \alpha_n \in Q_k p(\alpha_0 \alpha_1 \ldots \alpha_k \bot \bot \bot \ldots)$"

succeeds.

7. Sequential computational strategy

Follows the same topological idea but requires a new computational idea.

A Haskell program will be flashed in a moment.

In summary:

Theorem (Computational Tychonoff) *A product of an r.e. sequence of computationally compact spaces is computationally compact.*

Tychonoff program in Haskell

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

type Seq a = Nat -> a
type Quant a = (a -> S) -> S

tych :: (Seq a, Seq (Quant a)) -> (Quant (Seq a))
tych(u,quants)(p) = forall(\x->p(ifs(forall'(\s->p(cons(x,s))),u)))
   where forall  = hd(quants)
         u'      = tl(u)
         quants' = tl(quants)
         forall' = tych(u',quants')
```

Application: quantifier of the Cantor space 2^ω

```
forall_2 :: Quant Nat
forall_2 p = p(0) /\ p(1)

c :: Baire
c = \i -> 0

forall_C :: Quant Baire
forall_C = tych(c, \i -> forall_2)
```

Negative solution to problem II

"for all elements" is stronger than "for all *computable* elements".

There are programs $\mathtt{p}\colon \mathtt{nat}^\omega \to \mathbb{S}$ such that

1. $\mathtt{p}(\mathtt{x})$ holds for all computable $\mathtt{x}\colon 2^\omega$, but
2. $[\![p]\!](x)$ fails for some non-computable $x \in [\![2^\omega]\!]$.

They are constructed using so-called Kleene trees.

For such a "pathological" predicate, it is not possible to perform the universal quantification in finite time.

With the weaker interpretation of "for all", we get a negative answer.

Summary of solutions to Problem II

Positive solution uses topology (Tychonoff theorem).

"for all" means "for all elements in the Scott model".

Negative solution uses recursion theory (Kleene trees).

"for all" means "for all programmable elements".

"for all programmable" is harder than "for all"!

N.B. We get positive operational solutions by considering the notion of *data language* for a programming language.

Which solution is correct?

Both. And there are more solutions (and questions).

This is where synthetic topology enters into the picture.

Synthetic topology

Addresses various vexing questions in a non-committal way:

- Computation with computable data vs. arbitrary data.
- Sequential vs. parallel computation.
- Denotational vs. operational semantics.
- Continuous vs. computable. • open vs. observable vs. semidecidable.
- Classical vs. computational topology.
- Domains vs. topological spaces vs. locales vs. sequential spaces, vs. compactly generated spaces vs. convergence spaces vs. . . .

Flavours of synthetic topology

1. Via the λ-calculus,
2. via the internal language of a topos.

(1) is the subject of the monograph cited above, and of this talk.

(2) is work in progress with some collaborators.

Synthetic topology

Take the notion of continuity as primitive.

Let "continuous" mean "definable in a suitable language".

Flavour (1): Some form of the simply typed λ-calculus with $\mathbb{S}$.

Flavour (2): The internal language of a topos with a dominance Σ.

$U \subseteq X$ is open iff the function

$$\begin{array}{ccc} X & \xrightarrow{\chi_U} & \mathbb{S} \\ x & \mapsto & \text{“}x \in U\text{”} \end{array}$$

is continuous.

Possible languages and the resulting open sets

E.g. $\underbrace{\lambda\text{-calculus over } \widehat{\mathrm{Top}}}_{\text{topologically open}}$ $\underbrace{\mathrm{PCF}^{++}}_{\text{semidecidable}}$ $\underbrace{\mathrm{PCF}}_{\text{“sequentially” semidecidable}}$

$\underbrace{\mathrm{PCF}^{++}_{\Omega}}_{\text{observable} = \text{topologically open}}$ $\underbrace{\mathrm{PCF}_{\Omega}}_{\text{“sequentially” observable}}$, . . .

$$\begin{cases} ++ & = \text{parallel-or and } \exists, \\ \Omega & = \text{first-order “oracles”} \\ & = \text{input streams provided by e.g. Geiger counters.} \end{cases}$$

Idea: consider a powerful “attacker”, get stronger specifications.

PCF_{Ω} is a data language, PCF is a programming language.

Synthetic formulation of topological notions

$U \subseteq X$ open	$X \to \mathbb{S}$ $x \mapsto$ "$x \in U$"
$C \subseteq X$ closed	$X \to \mathbb{S}$ $x \mapsto$ "$x \notin C$"
$O \subseteq X$ overt	$(X \to \mathbb{S}) \to \mathbb{S}$ $p \mapsto$ "$\exists x \in O.p(x)$"
$Q \subseteq X$ compact	$(X \to \mathbb{S}) \to \mathbb{S}$ $p \mapsto$ "$\forall x \in Q.p(x)$"
X discrete	$X \times X \to \mathbb{S}$ $(x, y) \mapsto$ "$x = y$"
X Hausdorff	$X \times X \to \mathbb{S}$ $(x, y) \mapsto$ "$x \neq y$"

We can again profitably let the notions vary by letting the underlying language vary.

Topology via first-order logic

Not quite the same as Vickers' "*Topology via* [propositional] *logic*".

Technology. λ-definability preserves continuity.

To show that a set is open, λ-define its characteristic function.

To show that a set if closed, λ-define the charateristic function of its complement.

To show that a set is compact, λ-define its universal quantifier.

To show that a space is Hausdorff, λ-define its apartness map.

To show that a space is discrete, λ-define its equality map.

⋮

A product of two compact spaces is compact

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

Synthetic proof. Given the universal quantifiers of X and Y, we have to λ-define the universal quantifier of $X \times Y$.

But $\forall z \in X \times Y.\, p(z) \iff \forall x \in X.\, \forall y \in Y.\, p(x, y)$.

Hence the λ-definition $\forall_{X \times Y}(p) = \forall_X(\lambda x.\forall_Y(\lambda y.p(x, y))$ works. Q.E.D.

Magic?

Remarks

(i) This proves the binary case of the Tychonoff theorem.

And at the same time computational versions.

(ii) Our solution to Problem II is a subtle iteration of the binary case.

(The crude iteration cannot be written down!)

But, to show that the resulting program has the required termination property, we invoke the topological Tychonoff theorem.

Compact subsets of Hausdorff spaces are closed

Proposition *If* $\overbrace{X \times X \xrightarrow{\neq} \mathbb{S}}^{X \text{ is Hausdorff}}$ *and* $\overbrace{(X \to \mathbb{S}) \xrightarrow{\forall_Q} \mathbb{S}}^{Q \subseteq X \text{ is compact}}$, *then* $\overbrace{X \xrightarrow{\chi_{X \setminus Q}} \mathbb{S}}^{Q \text{ is closed}}$.

Synthetic proof. $x \notin Q \iff \forall q \in Q.\, x \neq q$.

Hence $\chi_{X \setminus Q}(x) = \forall_Q(\lambda q.x \neq q)$ is continuous,

because it is λ-definable from continuous maps. Q.E.D.

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

Synthetic proofs are programs in a literal sense.

Overt subsets of discrete spaces are open

Proposition *If* $\overbrace{X}^{X \times X \xrightarrow{=} \mathbb{S}}$ *is discrete and* $\overbrace{O \subseteq X}^{(X \to \mathbb{S}) \xrightarrow{\exists_O} \mathbb{S}}$ *is overt, then* $\overbrace{O}^{X \xrightarrow{\chi_O} \mathbb{S}}$ *is open.*

Synthetic proof. $x \in O \iff \exists y \in O.\, x = y$. Q.E.D.

Not very exciting to topologists. But definitely to locale theorists.

Computationally, it confirms what is expected from a discrete set over which we are able to existentially quantify in a computational fashion:

it must be r.e.

Closed subsets of compact spaces are compact

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

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

$\forall x \in F.p(x)$ iff $\forall x \in X.x \in F \Rightarrow p(x)$ 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 "weak parallel-or". Q.E.D.

If we can computationally quantify over X and semidecide the complement of F, then we can also computationally quantify over F.

This is use of parallelism is an exception — topology is very sequential.

Continuous images of compact sets are compact

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

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

Similarly, continuous images of overt sets are overt.

Topological theorems on function spaces

Proposition *If X is overt and Y is Hausdorff then $(X \to Y)$ is Hausdorff.*

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

Corollary $((\mathtt{nat} \to \mathtt{bool}) \to \mathtt{nat})$ *has semidecidable apartness for total functionals.*

Proposition *If X is compact and Y is discrete then $(X \to Y)$ is discrete.*

Synthetic proof. $f = g$ iff $\forall x \in X. f(x) = f(x)$. Q.E.D.

Corollary $((\mathtt{nat} \to \mathtt{bool}) \to \mathtt{nat})$ *has semidecidable equality for total functionals.*

The compact-open topology

How does one observe a function?

Simple idea. Evaluate it for a particular input and then check whether its output lands in a given semidecidable set.

Better:

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

$$N(Q,V) \stackrel{\text{def}}{=} \{f \in (X \to Y) \mid f(Q) \subseteq V\}$$

is open in $(X \to Y)$.

Synthetic proof. $f \in N(Q,V)$ iff $\forall x \in Q.f(x) \in V$. Q.E.D.

Closure properties of synthetic open sets

Theorem *Open sets are closed under the formation of compact intersections and overt unions.*

Rather than finite intersections and arbitrary unions.

$$x \in \bigcap \mathcal{Q} \iff \forall U \in \mathcal{Q}.\, x \in U.$$

$$x \in \bigcup \mathcal{O} \iff \exists U \in \mathcal{O}.\, x \in U.$$

Compact = generalization of finite.

Overt = taming of arbitrary. (No taming at all in the classical case!)

Conclusion

The synthetic formulations of topological notions allow us to

1. easily develop the core of classical topology,
2. extract computational content from the theorems,
3. resolve the mismatch between classical continuity and computability,
4. see that topology can be applied to sequential computation, even though the open sets are not even closed under *finite unions* in a sequential setting.

"Proofs are programs."

This holds quite literally: "Proofs are their own realizers."

Synthetic topology

addresses various vexing questions in a non-committal way:

- Computation with computable data vs. arbitrary data.
- Sequential vs. parallel computation.
- Denotational vs. operational semantics.
- Continuous vs. computable. • open vs. observable vs. semidecidable.
- Classical vs. computational topology.
- Domains vs. topological spaces vs. locales vs. sequential spaces, vs. compactly generated spaces vs. convergence spaces vs. . . .

The End