Operational domain theory and topology of a sequential programming language

Martín Escardó

Joint work with Ho Weng Kin

School of Computer Science, University of Birmingham, UK

GEOCAL, Paris 7
22nd June 2005

Papers

M.H. Escardó. Synthetic topology of data types and classical spaces. ENTCS, vol 87, pages 21-156, 2004.

M.H. Escardó and W.K. Ho. Operational domain theory and topology of a sequential programming language. LICS 2005.

Summary

Domain theory and topology in programming language semantics:

Manufacture and study denotational models.

Imprecise for sequential languages:

Computational adequacy holds but full abstraction fails.

This work:

Reconciliation of a good deal of domain theory and topology with sequential computation.

Applications to correctness proofs of non-trivial programs that manipulate infinite data.

How the reconciliation is performed

1. Side-step denotational semantics.

2. Reformulate domain-theoretic and topological notions directly in terms of programming concepts, interpreted in an operational way.

Previous work in this direction: Mason, Smith, Talcot, Sands, Pitts, . . .

- They consider domain-theoretic techniques.
- We further develop this.
- We additionally consider topological techniques.

  (open sets, compact sets, continuity and uniform continuity)

Setting for operational domain theory and topology

In this talk we consider call-by-name PCF extended with

1. product types,
2. base types Σ (Sierpinski) and $\overline{\omega}$ (vertical natural numbers).

This can be seen as a subset of Haskell:

```
data Sierp = T
data OmegaBar = Succ OmegaBar
```

Notation and fundamental operational properties

$x \in \sigma$ x is an element (closed term) of type σ
$x = y$ contextual equivalence
$x \sqsubseteq y$ contextual preorder

Congruence:

$$f = g \wedge x = y \implies f(x) = g(y)$$
$$f \sqsubseteq g \wedge x \sqsubseteq y \implies f(x) \sqsubseteq g(y)$$

Extensionality:

$$\forall x \in \sigma. f(x) = g(x) \implies f = g$$
$$\forall x \in \sigma. f(x) \sqsubseteq g(x) \implies f \sqsubseteq g$$

Fixed-point approximation: $h(\mathrm{fix}\, g) = \bigsqcup_n h(g^n(\bot))$

where $\bot = \mathrm{fix}\, \lambda x.x.$

Evaluation relation: We never use it! (But of course $M \Downarrow v \iff M = v$.)

Rational chains

Definition. A sequence x_n is a rational chain if there are g and h with

$$x_n = h(g^n(\perp)).$$

Lemma. The sequence $0, 1, 2, \ldots, n, \ldots$ in $\overline{\omega}$ is a rational chain with

(i) $\infty = \bigsqcup_n n$,

(ii) $l(\infty) = \bigsqcup_n l(n)$ for every $l \in (\overline{\omega} \to \sigma)$.

Characterization. A sequence $x_n \in \sigma$ is a rational chain iff there is

$$l \in (\overline{\omega} \to \sigma)$$

such that

$$x_n = l(n), \qquad \text{and hence } \bigsqcup_n x_n = l(\infty).$$

Rational continuity

What ought be true is easily true:

Proposition. If $f \in (\sigma \to \tau)$ and x_n is a rational chain in σ, then

(i) $f(x_n)$ is a rational chain in τ, and

(ii) $f(\bigsqcup_n x_n) = \bigsqcup_n f(x_n)$.

Corollary. If f_n is a rational chain in $(\sigma \to \tau)$ and $x \in \sigma$,

(i) $f_n(x)$ is a rational chain in τ, and

(ii) $(\bigsqcup_n f_n)(x) = \bigsqcup_n f_n(x)$.

Proof. Apply the proposition to $F \in ((\sigma \to \tau) \to \tau)$ defined by $F(f) = f(x)$. $\Box$

Open sets

Definition. A set U of elements of a type σ is called open if there is

$$\chi_U \in (\sigma \to \Sigma)$$

such that for all $x \in \sigma$,

$$\chi_U(x) = \top \iff x \in U.$$

Proposition (Open sets form a rational topology).

> For any type, the open sets are closed under the formation of finite intersections and rational unions.

Finite unions would require weak parallel-or (aka parallel convergence test).

Open sets

Again, what ought be true is easily true:

Proposition (Topological continuity). For any $f \in (\sigma \to \tau)$ and any open subset V of τ, the set $f^{-1}(V)$ is open in σ.

Proposition (Specialization preorder). For $x, y \in \sigma$, the relation $x \sqsubseteq y$ holds iff $x \in U$ implies $y \in U$ for every open subset U of σ.

Proposition (Rational Scott openness). For any open set U in a type σ,

1. if $x \in U$ and $x \sqsubseteq y$ then $y \in U$,
2. if x_n is a rational chain with $\bigsqcup x_n \in U$, then $x_n \in U$ for some n.

Finite elements

We continue the programme of showing that what ought to be true is true.

But now the proofs start to get non-trivial.

Definition. An element b is called finite if for every rational chain x_n with

$$b \sqsubseteq \bigsqcup_n x_n,$$

there is n such that already

$$b \sqsubseteq x_n.$$

Theorem (Algebraicity). Every element of any type is the least upper bound of a rational chain of finite elements.

Finite elements

Theorem (Algebraicity). Every element of any type is the least upper bound of a rational chain of finite elements.

Corollaries.

1. b is finite iff for every rational chain x_n with $b = \bigsqcup_n x_n$, there is n such that already $b = x_n$.

2. $f = g$ holds in $(\sigma \to \tau)$ iff $f(b) = g(b)$ for every finite $b \in \sigma$.

3. For any $f \in (\sigma \to \tau)$, any $x \in \sigma$ and any finite $c \sqsubseteq f(x)$, there is a finite $b \sqsubseteq x$ such that already $c \sqsubseteq f(b)$.

4. If U is open and $x \in U$, then there is a finite $b \sqsubseteq x$ in U.

Finite elements

In order to establish algebraicity, we invoke the following concepts:

Definition.

1. A deflation on a type σ is an element of type $(\sigma \to \sigma)$ that
   - is below the identity of σ,
   - has finite image.
2. An SFP structure on a type σ is a rational chain id_n of idempotent deflations with $\bigsqcup_n \mathrm{id}_n = \mathrm{id}$.
3. A type is SFP if it has an SFP structure.

Finite elements

Theorem.

1. Each type of the language is SFP.
2. For any SFP structure, b is finite iff $b = \mathrm{id}_n(b)$ for some n.

In particular,

3. $\mathrm{id}_n(x)$ is finite,
4. any $x \in \sigma$ is the lub of the rational chain $\mathrm{id}_n(x)$.

Therefore the algebraicity theorem follows.

Finite elements

The SFP structures can be chosen in such a way that it is easy to see that:

Proposition.

1. Every element of any finitary type is finite.
2. If $f \in (\sigma \to \tau)$ and $x \in \sigma$ are finite then so is $f(x) \in \tau$.
3. If $x \in \sigma$ and $y \in \tau$ are finite then so is $(x, y) \in (\sigma \times \tau)$.

Finite elements

To prove the SFP theorem, we construct programs $\mathrm{d}^\sigma : \overline{\omega} \to (\sigma \to \sigma)$ by induction on σ:

$\mathrm{d}^{\mathtt{Bool}}(x)(p) = p$,

$\mathrm{d}^{\Sigma}(x)(p) = p$,

$\mathrm{d}^{\mathtt{Nat}}(x)(k) = \text{if } x > 0 \text{ then if } k == 0 \text{ then } 0 \text{ else } 1 + \mathrm{d}^{\mathtt{Nat}}(x-1)(k-1),$

$\mathrm{d}^{\overline{\omega}}(x)(y) = \text{if } x > 0 \wedge y > 0 \text{ then } 1 + \mathrm{d}^{\overline{\omega}}(x-1)(y-1),$

$\mathrm{d}^{\sigma\to\tau}(x)(f)(y) = \mathrm{d}^{\tau}(x)(f(\mathrm{d}^{\sigma}(x)(y))),$

$\mathrm{d}^{\sigma\times\tau}(x)(y,z) = (\mathrm{d}^{\sigma}(x)(y), \mathrm{d}^{\tau}(x)(z)).$

Lemma. The rational chain $\mathrm{id}^\sigma_n \stackrel{\text{def}}{=} \mathrm{d}^\sigma(n)$ is an SFP structure on σ.

Finite elements

In our operational context, the following topological characterization is non-trivial:

Theorem. $b \in \sigma$ is finite iff the set $\uparrow b \stackrel{\text{def}}{=} \{x \in \sigma \mid b \sqsubseteq x\}$ is open.

Hence the open sets $\uparrow b$ with b finite form a base of the rational topology:

Corollary. Every open set is a union of open sets $\uparrow b$ with b finite.

Finite elements

Definition (Hereditary totality).

1. An element of ground type is total iff it is maximal.
2. An element $f \in (\sigma \to \tau)$ is total iff $f(x) \in \tau$ is total whenever $x \in \sigma$ is total.
3. An element of type $(\sigma \times \tau)$ is total iff its projections into σ and τ are total.

Theorem (Density of total elements).

Every inhabited open set has a total element.

Equivalently, every finite element is below some total element.

Finite elements

Definition (δ-closeness). For $\delta \in \mathbb{N}$,

$$x =_\delta y \iff \mathrm{id}_\delta(x) = \mathrm{id}_\delta(y).$$

Proposition (ϵ–δ continuity). For total $f \in (\sigma \to \tau)$ and $x \in \sigma$,

$$\forall \epsilon \in \mathbb{N} \quad \exists \delta \in \mathbb{N} \quad \forall y \in \sigma \text{ total}, \quad x =_\delta y \implies f(x) =_\epsilon f(y).$$

We know this for

1. σ arbitrary,
2. $\tau \in \{\mathtt{Bool}, \mathtt{Nat}, \mathtt{Cantor}, \mathtt{Baire}\}$, where

$$\mathtt{Cantor} = (\mathtt{Nat} \to \mathtt{Bool}), \qquad \mathtt{Baire} = (\mathtt{Nat} \to \mathtt{Nat}).$$

Finite elements

Second formulation of ϵ–δ continuity (but in practice we use the previous).

Definition. $d(x, y) = \inf\{1/n \mid x =_n y\}$.

Proposition.

1. d is a metric, in fact an ultrametric.
2. $x =_\delta y \iff d(x, y) \leq 1/\delta$.
3. For total $f \in (\sigma \to \tau)$ and $x \in \sigma$,

$$\forall \epsilon > 0 \quad \exists \delta > 0 \quad \forall y \in \sigma \text{ total}, \quad d(x, y) < \delta \implies d(f(x), f(y)) < \epsilon.$$

Here τ has the same restriction as before.

Side remarks

Every open set is open in the metric topology. The converse of course fails.

For the language extended with first-order oracles, parallel-or and Plotkin's existential quantifier,

1. the contextual preorder order is directed complete,

2. rational chains are the same thing as ω-chains,

3. the open sets are precisely the Scott open sets,

4. the metrically open sets are precisely the Lawson open sets.

But we are interested in the sequential language, where this match fails.

Compact sets

Definition. A set Q of elements of a type σ is called compact if there is

$$(\forall_Q) \in ((\sigma \to \Sigma) \to \Sigma)$$

such that

$$\forall_Q(p) = \top \iff p(x) = \top \text{ for all } x \in Q.$$

Proposition (rational Heine–Borel property).

If Q is compact and U_n is a rational chain of open sets with $Q \subseteq \bigcup_n U_n$, then there is n such that already $Q \subseteq U_n$.

Compact sets

Proposition (Basic classical properties).

1. If $Q \subseteq \sigma$ is compact then so is $f(Q) \subseteq \tau$ for any $f \in (\sigma \to \tau)$.

2. If $Q \subseteq \sigma$ and $R \subseteq \tau$, then so is $Q \times R \subseteq \sigma \times \tau$.

3. If $Q \subseteq \sigma$ is compact and $V \subseteq \tau$ is open,

   then $\{f \in (\sigma \to \tau) \mid f(Q) \subseteq V\}$ is open.

PROOF. 1. $\forall y \in f(Q).p(y) = \forall x \in Q.p(f(x))$.

2. $\forall z \in Q \times R.p(z) = \forall x \in Q.\forall y \in R.p(x, y)$.

3. $\chi_{\{\cdots\}}(f) = \forall x \in Q.\chi_V(f(x))$. $\square$

Compact sets

Proposition. The set of all elements of any type σ is compact.

PROOF. By monotonicity, $\forall x \in \sigma . p(x) = p(\bot)$. So this is trivial. $\square$

Proposition. The sets of total elements of Nat and Baire $= (\texttt{Nat} \to \texttt{Nat})$ are not compact.

PROOF. Otherwise rational continuity would be violated. $\square$

Proposition. An open set is compact iff it has finite characteristic.

Proposition. Every open set is a rational union of compact open sets.

Compact sets

First non-trivial example of a compact set:

Proposition. The set $\mathbb{N}_\infty$ of sequences of the forms $0^n 1^\omega$ and 0^ω is compact in $\texttt{Baire} = (\texttt{Nat} \to \texttt{Nat})$.

Proof. Recursively define $\forall_{\mathbb{N}_\infty} : (\texttt{Baire} \to \Sigma) \to \Sigma$ by

$$\forall(p) = p(\text{if } p(1^\omega) \wedge \forall s.p(0 :: s) \text{ then } t),$$

where t is an arbitrary element of $\mathbb{N}_\infty$.

Then use the previously developed machinery to show that this works. $\Box$

Compact sets

Theorem (Continuity). For total $f \in (\sigma \to \tau)$ and T a set of total elements of σ.

$$\forall x \in T \quad \forall \epsilon \in \mathbb{N} \quad \exists \delta \in \mathbb{N} \quad \forall y \in T, \quad x =_\delta y \implies f(x) =_\epsilon f(y).$$

Theorem (Uniform continuity). For $f \in (\sigma \to \tau)$ total and T a compact set of total elements of σ,

$$\forall \epsilon \in \mathbb{N} \quad \exists \delta \in \mathbb{N} \quad \forall x \in T \quad \forall y \in T, \quad x =_\delta y \implies f(x) =_\epsilon f(y).$$

We know this for σ arbitrary and $\tau \in \{\mathtt{Bool}, \mathtt{Nat}, \mathtt{Cantor}, \mathtt{Baire}\}$.

Compact sets

We need this variation for the applications we have in mind:

Lemma. For $\gamma \in \{\texttt{Nat}, \texttt{Bool}\}$, $f \in (\sigma \to \gamma)$ total and T a compact set of total elements of σ,

1. $\exists \delta \in \mathbb{N} \quad \forall x \in T, \quad f(x) = f(\mathrm{id}_\delta(x))$,
2. $\exists \delta \in \mathbb{N} \quad \forall x, y \in T, \quad x =_\delta y \implies f(x) = f(y)$.

Definition. For f and T as above, we refer to the least $\delta \in \mathbb{N}$ such that

(1) holds as the big modulus of uniform continuity of f at T,

(2) holds as the small modulus of uniform continuity of f at T.

A data language

Definition.

1. Let $\mathcal{P}$ be our programming language.

2. Let $\mathcal{D}$ be $\mathcal{P}$ extended with oracles (arbitrary input tapes).

3. We take $\mathcal{D}$ as a data language for $\mathcal{P}$.

(Higher-type) data: closed terms defined from oracles.

Theorem.

1. For terms in $\mathcal{P}$, equivalence w.r.t. ground program contexts and equivalence w.r.t. ground data contexts coincide.

But:

2. There are programs that are total w.r.t. $\mathcal{P}$ but not w.r.t. $\mathcal{D}$.

Sample applications

The following is our main tool (to get uniform continuity):

Theorem. The total elements of `Cantor` $= {\scriptstyle (\mathtt{Nat}\to\mathtt{Bool})}$ form a compact set.

PROOF. Recursively define $\forall(p) = p(\text{if}\, \forall s.p(0 :: s) \wedge \forall s.p(1 :: s) \text{ then } t)$, where t is an arbitrary programmable total element of `Cantor`. $\square$

If the data language is taken to be $\mathcal{P}$ itself, the above theorem fails.

It is easier to universally quantify over all total elements of the Cantor type than just over the programmable ones:

the former can be achieved by a program but the latter cannot.

Sample applications

Theorem (Gandy, Ulrich Berger).

There is a total program

$$\varepsilon \colon (\texttt{Cantor} \to \texttt{Bool}) \to \texttt{Cantor}$$

such that for any total

$$p \in (\texttt{Cantor} \to \texttt{Bool}),$$

if $p(s) = \mathrm{true}$ for some total $s \in \texttt{Cantor}$, then $\varepsilon(p)$ is such an s.

The original specification and proof are based on the Scott model.

They can be directly understood in our operational setting.

Sample applications

PROOF. Define $\varepsilon\colon (\texttt{Cantor} \to \texttt{Bool}) \to \texttt{Cantor}$ by

$$\varepsilon(p) = \text{if } p(\varepsilon(\lambda s.p(0 :: s))) \begin{array}{ll} \text{then} & 0 :: \varepsilon(\lambda s.p(0 :: s)) \\ \text{else} & 1 :: \varepsilon(\lambda s.p(1 :: s)). \end{array}$$

Proceed by induction on the big modulus of uniform continuity of p total.

If p has modulus $\delta + 1$ then $\lambda s.p(0 :: s)$ and $\lambda s.p(1 :: s)$ have modulus δ.

If p has modulus zero, $p(\bot)$ is total and hence p is constant. $\square$

Sample applications

Corollary. There is a total $\forall\colon (\texttt{Cantor} \to \texttt{Bool}) \to \texttt{Bool}$ such that [. . .]

PROOF. Define $\exists(p) = p(\varepsilon(p))$ and $\forall(p) = \neg\exists s.\neg p(s)$. $\square$

Corollary. The function type $(\texttt{Cantor} \to \texttt{Nat})$ has decidable equality for total elements.

PROOF. $(f == g) = \forall$ total $s \in \texttt{Cantor}.f(s) == g(s)$. $\square$

Infinitely many other function types have the same property, e.g.

$((\texttt{Cantor} \to \texttt{Nat}) \to \texttt{Bool}) \to \texttt{Nat})$.

Sample applications

Theorem (Alex Simpson). There is a total program

$$\sup\colon (\texttt{Cantor} \to \texttt{Baire}) \to \texttt{Baire}$$

such that for every total $f \in (\texttt{Cantor} \to \texttt{Baire})$,

$$\sup(f) = \sup\{f(s) \mid s \in \texttt{Cantor} \text{ is total}\},$$

where the supremum is taken in the lexicographic order.

Again the original denotational specification and proof can be directly understood in our operational setting.

Sample applications

PROOF. Let $t \in \texttt{Cantor}$ be a programmable total element and define $\sup\colon (\texttt{Cantor} \to \texttt{Baire}) \to \texttt{Baire}$ by

$$\begin{aligned}\sup(f) \quad = \quad & \text{let}\, h = \text{hd}(f(t))\,\text{in} \\ & \quad \text{if}\,\forall\,\textsf{total}\ s \in \texttt{Cantor}.\,\text{hd}(f(s)) == h \\ & \quad \text{then}\ h :: \sup(\text{tl} \circ f) \\ & \quad \text{else}\ \max(\sup(\lambda s. f(0 :: s)), \sup(\lambda s. f(1 :: s))).\end{aligned}$$

To establish correctness, proceed by induction on the small modulus of uniform continuity of $\text{hd} \circ f\colon \texttt{Cantor} \to \texttt{Nat}$. $\square$

Open problems and further developments

1. The Tychonoff theorem (we have a sequential program but . . .).

2. Sequences ($\texttt{Nat} \to \sigma$) can be easily replaced by lazy lists.

3. Call-by-value easy.

4. Ho Weng Kin is working on recursive types (minimal invariants etc.).

5. State and control, and non-determinism and probability seem to pose genuine challenges. But this is the case at the denotational level too.

6. With probability or abstract data types for real numbers, types won't be algebraic: need operational $\ll$.

Papers

M.H. Escardó. Synthetic topology of data types and classical spaces. ENTCS, vol 87, pages 21-156, 2004.

M.H. Escardó and W.K. Ho. Operational domain theory and topology of a sequential programming language, 10pp. Submitted for publication.

`http://www.cs.bham.ac.uk/~mhe/`