Quantitative aspects of ODT in verification of program correctness

Weng Kin, Ho
Mathematics and Mathematics Education
National Institute of Education, Singapore
wengkin.ho@nie.edu.sg

22 Jan 2010

Outline

1. Motivation

Outline

1. Motivation

2. Operational/algorithmic topology

Outline

1. Motivation
2. Operational/algorithmic topology
3. Separation axioms

Outline

Outline

Outline

Outline

Outline

Subtopics to be covered

Subtopics to be covered

1. Compactness and its computational content

Subtopics to be covered

1. Compactness and its computational content
2. Quantitative domain theory of types

Subtopics to be covered

1. Compactness and its computational content
2. Quantitative domain theory of types
3. Sample applications: contextual equivalence and program correctness

Highlights

In today's talk, we encounter

1. the notion of operational compactness

Highlights

In today's talk, we encounter

1. the notion of operational compactness
2. the computational intuition of searchability

Highlights

In today's talk, we encounter

1. the notion of operational compactness
2. the computational intuition of searchability
3. quantitative domain theory of types

Highlights

In today's talk, we encounter

1. the notion of operational compactness
2. the computational intuition of searchability
3. quantitative domain theory of types
4. how ODT can be applied to prove contextual equivalence and program correctness

Domain theory

Domain theory can in fact be seen as topology of partial orders. So,

Maxim

no topology = no domain theory.

Domain theory

Many preceding works have pointed towards the use of ‘topology’ in computation:

Domain theory

Many preceding works have pointed towards the use of ‘topology’ in computation:

- M. Smyth: open set to express ‘an observable/affirmative predicate’.

Domain theory

Many preceding works have pointed towards the use of 'topology' in computation:

- M. Smyth: open set to express 'an observable/affirmative predicate'.
- S. Vickers: locales to express geometric logic.

Domain theory

Many preceding works have pointed towards the use of ‘topology’ in computation:

- M. Smyth: open set to express ‘an observable/affirmative predicate’.
- S. Vickers: locales to express geometric logic.
- S. Abramsky: Stone-duality to express program logic.

Domain theory

Many preceding works have pointed towards the use of 'topology' in computation:

- M. Smyth: open set to express 'an observable/affirmative predicate'.
- S. Vickers: locales to express geometric logic.
- S. Abramsky: Stone-duality to express program logic.
- Yu. Ershov: continuous maps to express computability.

Some photos



Figure: Some famous people in domains and semantics

Continuous maps

Following Ershov's ideas, one can start with a very natural definition:

Definition

A function $f : \sigma \to \tau$ is continuous if it is definable in the language.

Continuous maps

The definability of functions dictates the nature of the hierarchy of 'topologies' on types!

Continuous maps

The definability of functions dictates the nature of the hierarchy of 'topologies' on types!

Maxim

Functions are first-class citizens in functional programming paradigm.

Opens

Next question: What are then the open sets?

Opens

Next question: What are then the open sets?

Definition ((Operational) opens)

A subset $U \subseteq \sigma$ is (operationally) open in type σ if its characteristic function $\chi_U : \sigma \to \Sigma$ is continuous.

Note that

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

Opens as semi-decidable sets

Open subsets of a data type is precisely the *semi-decidable* (with respect to the language) subsets of that data type.

Opens as semi-decidable sets

Open subsets of a data type is precisely the *semi-decidable* (with respect to the language) subsets of that data type.
Clearly, conjunction of two semi-decisions is still a semi-decision.

Operational topology

Because open sets are characterized by definability of extensional functions, we have:

Main spirit of OT

Writing topological proofs become writing programs.

Opens aren't really opens

Proposition

The opens of a data type do not form a topology!

Opens aren't really opens

Proof.

Suppose not, then in particular any two opens U and V creates a new open $U \cup V$.

Opens aren't really opens

Proof.

Suppose not, then in particular any two opens U and V creates a new open $U \cup V$.

Then we have the (extensional) characteristic function of $U \cup V$ can be realized by an (intensional) program given by

$$p : \sigma \longrightarrow \Sigma.$$

Opens aren't really opens

Proof.

Suppose not, then in particular any two opens U and V creates a new open $U \cup V$.
Then we have the (extensional) characteristic function of $U \cup V$ can be realized by an (intensional) program given by

$$p : \sigma \longrightarrow \Sigma.$$

But notice that p gives the algorithmic weak parallel-or $\vee$.
(Contradiction!) □

Opens aren't really opens

Here a weak parallel-or $\vee$ means

$$p \vee q = \top \iff p = \top \text{ or } q = \top.$$

Opens aren't really opens

So operational topology isn't a topology!

Opens aren't really opens

So operational topology isn't a topology!
But it behaves very much like one.

Specialization (pre-)order

Proposition

For any $x, \; y : \sigma$,

$$x \sqsubseteq_{\sigma} y \iff \forall \text{ open } U.x \in U \implies y \in U.$$

The contextual pre-order coincides with the specialization order induced by operational opens.

Specialization (pre-)order

Proposition

All opens are upper with respect to the contextual pre-order.

Specialization (pre-)order

Proposition

All opens are upper with respect to the contextual pre-order.

Proof.

By definition! □

Specialization (pre-)order

Proposition

Every continuous function is monotone w.r.t. the specialization (pre-)orders.

Corollary (Halting problem)

There is no continuous function $f : \Sigma \to \Sigma$ such that

$$f(\bot) = \top \text{ and } f(\top) = \bot.$$

Data types as d-spaces

Operational opens are not just 'opens'. They are somewhat intrinsic to the operational semantics of the language!

Data types as *d*-spaces

Because the intrinsic topology on types is the Scott topology, we expect the operational opens to behave like Scott-open sets.

Analogy

directed complete $\rightsquigarrow$ rational-chain complete

Scott open $\rightsquigarrow$?

Data types as d-spaces

Proposition

Opens are operationally Scott open in the sense that they are

Data types as d-spaces

Proposition

Opens are operationally Scott open in the sense that they are

1. *upper with respect to $\sqsubseteq$, and*
2. *inaccessible by joins of rational chains.*

Data types as d-spaces

Proof.

For the second property, suppose that there is a rational chain x_n whose join is in an open U.

Data types as d-spaces

Proof.

For the second property, suppose that there is a rational chain x_n whose join is in an open U. Then we have

$$\chi_U(\bigsqcup_n x_n) = \chi_U(l(\infty)) = \top$$

where $l : \overline{\omega} \to \sigma$ is the realizer of the rational chain x_n.

Data types as d-spaces

Proof.

For the second property, suppose that there is a rational chain x_n whose join is in an open U. Then we have

$$\chi_U(\bigsqcup_n x_n) = \chi_U(l(\infty)) = \top$$

where $l : \overline{\omega} \to \sigma$ is the realizer of the rational chain x_n. Thus by rational continuity, $\chi_U(l(\bigsqcup_n n)) = \bigsqcup_n \chi_U(l(n))$.

Data types as d-spaces

Proof.

For the second property, suppose that there is a rational chain x_n whose join is in an open U. Then we have

$$\chi_U(\bigsqcup_n x_n) = \chi_U(l(\infty)) = \top$$

where $l : \overline{\omega} \to \sigma$ is the realizer of the rational chain x_n. Thus by rational continuity, $\chi_U(l(\bigsqcup_n n)) = \bigsqcup_n \chi_U(l(n))$. Since $\top$ is finite, it follows that $\chi(l(n)) = \top$ for some $n < \infty$, i.e.,

Data types as d-spaces

Proof.

For the second property, suppose that there is a rational chain x_n whose join is in an open U. Then we have

$$\chi_U(\bigsqcup_n x_n) = \chi_U(l(\infty)) = \top$$

where $l : \overline{\omega} \to \sigma$ is the realizer of the rational chain x_n. Thus by rational continuity, $\chi_U(l(\bigsqcup_n n)) = \bigsqcup_n \chi_U(l(n))$. Since $\top$ is finite, it follows that $\chi(l(n)) = \top$ for some $n < \infty$, i.e., there is $n \in \mathbb{N}$ such that already $x_n \in U$. □

Data types as d-spaces

Recall that a *d-space* is a topological space in which every open set is Scott-open. These are also known as *monotone convergence spaces*.

Data types as d-spaces

Recall that a *d-space* is a topological space in which every open set is Scott-open. These are also known as *monotone convergence spaces*.

So from the above result, we have shown that every data type is an operational d-space.

Data types as algebraic domains

Theorem

The following are equivalent:

Data types as algebraic domains

Theorem

The following are equivalent:

1. b *is finite.*

Data types as algebraic domains

Theorem

The following are equivalent:

1. b *is finite.*
2. $\uparrow b$ *is open.*

Data types as algebraic domains

Corollary

Every open set is the union of open sets of the form $\uparrow b$ with b finite.

In other words, the sets $\uparrow b$ with b finite forms a basis for the operational topology.

T_0-space

For any type σ, the following are equivalent:

T_0-space

For any type σ, the following are equivalent:

1. $x \neq_\sigma y$

T_0-space

For any type σ, the following are equivalent:

1. $x \neq_\sigma y$
2. $\exists$ open $U \subseteq \sigma.(x \in U \ \wedge \ y \notin U) \ \vee \ (y \in U \ \wedge \ x \notin U)$

Proof.

Trivial by definition! □

T_0-space

Proposition

Every type as an operational topological space is automatically T_0.

Subspace

Definition (Subspace)

Any subset X of a type σ is called a subspace of Y.
Given any subspaces X of σ and Y of τ, a function

$$f : X \to Y$$

is relatively continuous if there is at least one continuous function

$$g : \sigma \to Y$$

such that $g(x) = f(x)$ for all $x \in X$.

Subspace

Definition

A subset of a space is relatively open if its Σ-valued characteristic map is relatively continuous.

Subspace

Proposition

For a subspace X of a data type σ, a subset U of X is relatively open in X iff there is an open U' of σ such that $X \cap U' = U$.

Separation axioms

If opens are observables, then the ability to separate distinct programs amounts to different degrees of separation.

Separation axioms

We take this as an example:

Definition (Hausdorff space)

X is a Hausdorff subspace of σ if the ability to tell any given pair of inequivalent programs in X apart is a semidecision, i.e., $(\neq) : \sigma \times \sigma \to \Sigma$ such that

$$\forall x,\ y : \sigma.(\neq)(x, y) = \top \iff x \neq_\sigma y$$

is continuous (i.e., definable in the language).

Separation axioms

We take this as an example:

Definition (Hausdorff space)

X is a Hausdorff subspace of σ if the ability to tell any given pair of inequivalent programs in X apart is a semidecision, i.e., $(\neq) : \sigma \times \sigma \to \Sigma$ such that

$$\forall x,\ y : \sigma.(\neq)(x, y) = \top \iff x \neq_\sigma y$$

is continuous (i.e., definable in the language).

Can you see why this is a good definition?

Separation axioms

Non-examples

Every non-trivial data type is **NOT** Hausdorff. (Why?)

Separation axioms

Example

Following are examples of Hausdroff subspaces:

Separation axioms

Example

Following are examples of Hausdroff subspaces:

1. The subspace N of `Nat` of (non-divergent) natural numbers

Separation axioms

Example

Following are examples of Hausdroff subspaces:

1. The subspace N of `Nat` of (non-divergent) natural numbers
2. The subspace B (called the *Baire* space) consisting of all strict total functions of `Baire` := `Nat` → `Nat` type

Separation axioms

Example

Following are examples of Hausdroff subspaces:

1. The subspace *N* of `Nat` of (non-divergent) natural numbers
2. The subspace *B* (called the *Baire* space) consisting of all strict total functions of `Baire` := `Nat` $\rightarrow$ `Nat` type
3. The subspace *C* (called the *Cantor* space) consisting of all strict total sequences of 0's and 1's of `Baire`-type.

Break (5 mins)

Let us have a short break.

Operational compactness

Compactness is a crucial topological property.
But what is the computational parallel of this notion?

Operational compactness

Definition ((Operational) Compactness)

A set Q of elements of a data type σ is compact if there is a program $\forall_Q : (\sigma \to \Sigma) \to \Sigma$ such that

$$\forall_Q(p) = \top \iff \forall x \in Q . p(x) = \top .$$

Why such a weird definition?

Proposition

For any set Q *of* σ*-elements, the following are equivalent:*

Why such a weird definition?

Proposition

For any set Q of σ-elements, the following are equivalent:

1. $\{U \text{ open} \mid Q \subseteq U\}$ *is open in* $(\sigma \to \Sigma)$

Why such a weird definition?

Proposition

For any set Q of σ-elements, the following are equivalent:

1. $\{U \text{ open} \mid Q \subseteq U\}$ *is open in* $(\sigma \to \Sigma)$
2. *There is a program* $\forall_Q : ((\sigma \to \Sigma) \to \Sigma)$ *such that*

$$\forall_Q(p) = \top \iff \forall x \in Q . p(x) = \top .$$

Why such a weird definition?

Proposition

For any set Q of σ-elements, the following are equivalent:

1. $\{U \text{ open } \mid Q \subseteq U\}$ *is open in* $(\sigma \to \Sigma)$
2. *There is a program* $\forall_Q : ((\sigma \to \Sigma) \to \Sigma)$ *such that*

$$\forall_Q(p) = \top \iff \forall x \in Q. p(x) = \top.$$

Proof.

$\forall_Q = \chi_{\mathcal{U}}$ where $\mathcal{U} := \{\chi_U \mid Q \subseteq U\}$, because if $p = \chi_U$ then $Q \subseteq U \iff \forall x \in Q. p(x) = T$. □

Why such a weird definition?

Notice that the first condition that

$$\{U \text{ open } \mid Q \subseteq U\} \text{ is open in } (\sigma \to \Sigma)$$

is parallel to the domain-theoretic statement:

$$\{U \in \mathcal{O}X \mid Q \subseteq U\} \text{ is Scott open in } \mathcal{O}X.$$

Why such a weird definition?

Do you now recognize the characterizing property on Q for which:

$\{U \in \mathcal{O}X \mid Q \subseteq U\}$ is Scott open in $\mathcal{O}X$?

Why such a weird definition?

Do you now recognize the characterizing property on Q for which:

$$\{U \in \mathcal{O}X \mid Q \subseteq U\} \text{ is Scott open in } \mathcal{O}X?$$

Answer

Q is a *compact* subspace of X.

Computational content of compactness

The existence of a program $\forall_Q : (\sigma \to \Sigma) \to \Sigma$ such that

$$\forall_Q(p) = \top \iff \forall x \in Q . p(x) = \top$$

tells us that the process of searching through and verifying a given predicate over all the possible elements of a compact set terminates in finite time.

Computational content of compactness

Example

Computational content of compactness

Example

1. N is not a compact set, otherwise number-theorists will be put out of job!

Computational content of compactness

Example

1. N is not a compact set, otherwise number-theorists will be put out of job!
2. C is compact w.r.t the data language PCF_Ω but not the programming language PCF.

Topological properties involving compactness

The following are very well-known elementary results in topology, but operationally manifested!

Proposition

Topological properties involving compactness

The following are very well-known elementary results in topology, but operationally manifested!

Proposition

1. *Continuous image of compact sets are compact.*

Topological properties involving compactness

The following are very well-known elementary results in topology, but operationally manifested!

Proposition

1. *Continuous image of compact sets are compact.*
2. *Every compact subspace of a Hausdorff space is closed.*

Topological properties involving compactness

The following are very well-known elementary results in topology, but operationally manifested!

Proposition

1. *Continuous image of compact sets are compact.*
2. *Every compact subspace of a Hausdorff space is closed.*
3. *Intersection of compactly many open sets is open.*

Topological properties involving compactness

Proof of (1).

$$\forall_{f(Q)} := \lambda p.\forall_Q(p \circ f)$$

□

The rest are homework exercises for my diligent PhD/MSc students!

Total elements

Definition (Totality)

Total elements

Definition (Totality)

- *An element of ground type* γ *is total.*

Total elements

Definition (Totality)

- *An element of ground type* γ *is total.*
- *An element of product type is total if its projections are.*

Total elements

Definition (Totality)

- *An element of ground type* γ *is total.*
- *An element of product type is total if its projections are.*
- *An element of function type is total if application to totals yields totals.*

Total elements

It turns out that

Theorem

The set of total elements of a given type are dense in it in the sense that every inhabited open set must contain at least one total element.

Total elements

It turns out that

Theorem

The set of total elements of a given type are dense in it in the sense that every inhabited open set must contain at least one total element.

Proof.

Hint: First show that every finite element is below a total element. □

Moduli of (uniform) continuity

Theorem

For total $f : \sigma \to \texttt{Baire}$ and Q a compact set of total elements of σ,

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

Omitting this proof, we inspect another similar result for types other than Baire.

Moduli of (uniform) continuity

Proposition

For ground types γ, $f : \sigma \to \gamma$ total and Q a compact set of total elements of σ,

Moduli of (uniform) continuity

Proposition

For ground types γ, $f : \sigma \to \gamma$ total and Q a compact set of total elements of σ,

1. *Big m.o.c. of f at Q:*
   $\exists \delta \in \mathbb{N}. \forall x \in Q. f(x) = f(\mathrm{id}_\delta(x))$.

Moduli of (uniform) continuity

Proposition

For ground types γ, $f : \sigma \to \gamma$ *total and* Q *a compact set of total elements of* σ,

1. *Big m.o.c. of* f *at* Q:
   $\exists \delta \in \mathbb{N}. \forall x \in Q. f(x) = f(\mathrm{id}_\delta(x))$.
2. *Small m.o.c. of* f *at* Q:
   $\exists \delta \in \mathbb{N}. \forall x, y \in Q. x =_\delta y \implies f(x) = f(y)$.

Moduli of (uniform) continuity

Recall that for total elements in `Nat` one has an equality semidecision e, which can be written as

```
e: (Nat,Nat) -> \Sierp
e (s,t) = if s == 0 then (t == 0)
                    else e(s-1,t-1)
```

Then the program e behaves as follows:
If $s,\ t \in$ `Nat` are total, then $s = t \iff e(s,t) = \top$.

Moduli of (uniform) continuity

We aim to show

$$\exists \delta \in \mathbb{N}. \forall x \in Q. f(x) = f(\mathrm{id}_\delta(x)).$$

Proof

Let $p(x) := e(f(x), f(x))$. Clearly, because $x \in Q$ and f are total, we always have

$$\forall_Q(p) = \top.$$

Moduli of (uniform) continuity

Proof.

So by the finiteness of $\top$, there is $\delta \in \mathbb{N}$ such that already

$$\forall_Q(\mathrm{id}_\delta(p)) = \top$$

which implies that $\forall_Q(p(\mathrm{id}_\delta(x))) = \top$, i.e.,

$$\forall x \in Q. e(\mathrm{id}_\delta(x), \mathrm{id}_\delta(x)) = \top.$$

Applying monotonicity applied to $\mathrm{id}_\delta(x) \sqsubseteq \mathrm{id}(x)$, we must have

$$\exists \delta \in \mathbb{N}. \forall x \in Q. e(x, \mathrm{id}_\delta(x)) = \top.$$

as desired. □

Universal quantification for boolean-valued predicates

Theorem (U. Berger)

There is a total program

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

such that for any total $p : (\texttt{Cantor} \to \texttt{Bool})$*, if* $p(s) = 0$ *for some total* $s : \texttt{Cantor}$*, then* $\varepsilon(p)$ *is such an* s*.*

Here 0 stands for true.

Proof.

By simple induction on the big m.o.c. of p at C. □

Conclusion

In today's tutorial, we have

Conclusion

In today's tutorial, we have

1. talked about the computational parallel of topological compactness,

Conclusion

In today's tutorial, we have

1. talked about the computational parallel of topological compactness,
2. explored some of its uses in programming and

Conclusion

In today's tutorial, we have

1. talked about the computational parallel of topological compactness,
2. explored some of its uses in programming and
3. used a 'quantitative domain theoretic' approach to check program correctness.

References

References

1. M.H. Escardó. *Synthetic topology of data types and classical spaces*. ENTCS, 87, pp. 21–156, 2004.

References

1. M.H. Escardó. *Synthetic topology of data types and classical spaces*. ENTCS, 87, pp. 21–156, 2004.
2. M.H. Escardó & W.K. Ho. *An operational domain theory and topology of sequential languages*. Information and Computation, 207(3), pp. 411–437, 2009.

References

1. M.H. Escardó. *Synthetic topology of data types and classical spaces*. ENTCS, 87, pp. 21–156, 2004.
2. M.H. Escardó & W.K. Ho. *An operational domain theory and topology of sequential languages*. Information and Computation, 207(3), pp. 411–437, 2009.
3. W.K. Ho. *Operational Domain Theory and Topology of Sequential Functional Languages*. PhD Thesis, School of Computer Science, University of Birmingham, Oct 2006.

All these can be downloaded from my webpage at http://math.nie.edu.sg/wkho/pubtalk.htm.