Topology, computation, monads, games and proofs

Martín Escardó

$\frac{3}{4}$ of this talk is joint work with Paulo Oliva from Queen Mary, London.

MFPS 2010, Ottawa, May 6-10, 2010

Contents plan

I. Topology in computation.

Exhaustive search.

II. The selection monad.

Selection functions for generalized quantifiers.

III. Game theory.

Optimal plays and strategies.

IV. Proof theory.

Computational extraction of witnesses from classical proofs.

Unifying concept: selection functions for quantifiers

A certain *countable product of selection functions* implements:

I. Topology in computation: Tychonoff theorem.

II. The selection monad: Strength.

III. Game theory: Optimal plays and strategies.

IV. Proof theory: Double-negation shift. Bar recursion.

Time plan

Each part has half of the time allocated to the previous part.

Even though it probably deserves twice as much.

I. Topology and computation.

Old theorem. A set of natural numbers is exhaustively searchable iff it is finite.

Intuition: How could one possibly check infinitely many cases in finite time?

Proof: Removed from this intuition (Halting problem, diagonalization).

Common wisdom

A set of whatever-you-can-think-of is exhaustively searchable iff it is finite.

E.g. types in

1. System T, PCF
2. FPC, ML, Haskell etc.

Can't always trust common wisdom

E.g. The total elements of $\mathrm{Nat} \to \mathrm{Bool}$ are exhaustively searchable.

Many other examples.

Corollary

The type $(\mathrm{Nat} \to \mathrm{Bool}) \to \mathrm{Nat}$ has decidable equality.

Proof. Given $f, g \colon (\mathrm{Nat} \to \mathrm{Bool}) \to \mathrm{Nat}$.

Check whether $f(\alpha) = g(\alpha)$ for every $\alpha \colon \mathrm{Nat} \to \mathrm{Bool}$.

Exhaustible set

A set $K \subseteq X$ is *exhaustible* iff there is an an algorithm s.t.

1. *Input:* $p\colon X \to \mathrm{Bool}$ decidable.
2. *Output:* True or False.
3. *Specification:* output True iff $\exists k \in K . p(k) = \mathrm{True}$.

The algorithm has higher type $(X \to \mathrm{Bool}) \to \mathrm{Bool}$.

Searchable set

A set $K \subseteq X$ is *searchable* iff there is an an algorithm s.t.

1. *Input:* $p\colon X \to \mathrm{Bool}$ decidable.

2. *Output:* Either fail or some $k \in K$.

3. *Specification:* output fail if $\forall k \in K . p(k) = \mathrm{False}$,
   or else $k \in K$ with $p(k) = \mathrm{True}$.

The algorithm has higher type $(X \to \mathrm{Bool}) \to 1 + X$.

Of course

Searchable $\implies$ exhaustible.

This is so by definition.

Searchable set, slightly different notion and formulation

A set $K \subseteq X$ is *searchable* iff there is an an algorithm s.t.

1. *Input:* $p\colon X \to \mathrm{Bool}$ decidable.
2. *Output:* $k \in K$.
3. *Specification:* If $\exists x \in K . p(x) = \mathrm{True}$ then $p(k) = \mathrm{True}$.
   Otherwise $p(k) = \mathrm{False}$, of course.

Only difference: previous accounts for the empty set, this doesn't.

The algorithm has higher type $(X \to \mathrm{Bool}) \to X$.

Still

Searchable $\implies$ exhaustible.

Given the potential example $k \in K$,

check whether $p(k) = \mathrm{True}$ or $p(k) = \mathrm{False}$.

Better: the answer to the exhaustion procedure is just $p(k)$.

Summary of the two notions

$K \subseteq X$

Exhaustible: algorithm $\exists_K \colon (X \to \mathrm{Bool}) \to \mathrm{Bool}$.

The boolean existential quantifier is computable.

Searchable: algorithm $\varepsilon_K \colon (X \to \mathrm{Bool}) \to X$.

The set K has a computable a selection function.

Derived functions:

$$\exists_K(p) = p(\varepsilon_K(p)).$$

$$\forall_K(p) = \neg\exists_K(\neg \circ p).$$

Theorem (LMCS'2008, ENTCS'2004)

Exhaustible sets (hence searchable sets) are topologically compact.

Types with decidable equality are topologically discrete.

First examples and counter-examples

1. A set of natural numbers is compact iff it is finite.

2. The maximal elements of the lazy natural numbers are searchable.

   (Which amount to the one-point compactification of discrete natural numbers.)

3. The set of all sequences $\alpha\colon \mathrm{Nat} \to \mathrm{Nat}$ is *not* searchable.

4. The set of sequences $\alpha\colon \mathrm{Nat} \to \mathrm{Nat}$ such that $\alpha_k < 17$ is searchable.

5. The set of sequences $\alpha\colon \mathrm{Nat} \to \mathrm{Nat}$ such that $\alpha_k < k$ is searchable.

6. The set of sequences $\alpha\colon \mathrm{Nat} \to \mathrm{Nat}$ such that $\alpha_k < \beta_k$ is searchable, for any given sequence $\beta\colon \mathrm{Nat} \to \mathrm{Nat}$.

More examples (LMCS'2008)

Consider the types defined by the following grammar:

$$\begin{aligned}\text{compact} &::= 1 \,|\, \text{compact} + \text{compact} \,|\, \text{compact} \times \text{compact} \,|\, \text{discrete} \to \text{compact},\\ \text{discrete} &::= 1 \,|\, \text{Nat} \,|\, \text{discrete} + \text{discrete} \,|\, \text{discrete} \times \text{discrete} \,|\, \text{compact} \to \text{discrete}\,.\end{aligned}$$

Theorem.

1. Compact types are searchable.
2. Discrete types have decidable equality of total elements.

E.g. $(((\text{Nat} \to 1+1) \to \text{Nat}) \to 1+1+1) \to ((\text{Nat} \to 1+1+1+1) \to \text{Nat})$ has decidable equality.

Dictionary between topology and computation

Open set.	Semi-decidable set.
Closed and open set.	Decidable set.
Continuous map.	Computable function.
Compact set.	Exhaustively searchable set.
Discrete space.	Type with decidable equality.
Hausdorff space.	Space with semi-decidable apartness.

Take a theorem in topology,
apply the dictionary,
get a theorem in computability theory.
Unfortunately you have to come up with a new proof.

Theorems (LMCS'2008)

1. The non-empty exhaustible sets are the computable images of the Cantor space $(\mathrm{Nat} \to \mathrm{Bool})$.

2. Searchable sets are closed under computable images.

3. Hence hence the non-empty exhaustible sets are searchable.

   (Given yes/no algorithm, get an algorithm for witnesses.)

4. Searchable sets are closed under countable products.

   (Tychonoff theorem.)

5. And under intersections with decidable sets.

6. They are retracts of the types where they live.

Is this feasible?

I have some counter-intuitively fast examples to show you.

II. Selection functions for generalized quantifiers.

Pause to look at some motivating examples.

Mean-value theorem

$\int_0^1 f = f(a)$.

The mean value is attained.

If you travelled from London to Ottawa and your journey took 12 hours, then at some point you were travelling at $5379/12 \approx 440$km/h.

Maximum-value theorem

$\sup_0^1 f = f(a)$.

The maximum value is attained.

Universal-value theorem

$\forall p = p(a)$.

The universal value is attained.

Known as Drinker paradox:

In every pub there is a person a such that everybody drinks iff a drinks.

Existential-value theorem

$\exists p = p(a)$.

The existential value is attained.

Another version of the Drinker paradox:

In every pub there is a person a such somebody drinks iff a drinks.

General pattern

$\phi(p) = p(a)$.

R type of results.

$p \colon X \to R$.

$\phi \colon (X \to R) \to R$. Lives in the continuation monad.

$a \in X$.

We want to find a from given p, as $a = \varepsilon(p)$.

$\varepsilon \colon (X \to R) \to X$. Lives in the selection monad.

$\phi(p) = p(\varepsilon(p))$.

Continuation monad

$KX = ((X \to R) \to R)$.

Well known, with many theoretical and practical uses.

Selection monad

$JX = ((X \to R) \to X)$.

Images of searchable sets are searchable:

$f\colon X \to Y$

$Jf\colon JX \to JY, \qquad Jf\varepsilon = \lambda q.f(\varepsilon(\lambda x.f(q(x)))$.

Singletons are searchable:

$\eta\colon X \to JX, \qquad \eta(x) = \lambda p.x$.

The union of a searchable set of searchable sets is searchable:

$\mu\colon JJX \to JX, \qquad \mu(E) = \lambda p.E(\lambda\varepsilon.p(\varepsilon(p)))(p)$.

Monad morphism $J \to K$

$\varepsilon \mapsto \phi$ where $\phi(p) = p(\varepsilon(p))$.

We write $\phi = \overline{\varepsilon}$.

Then $\overline{\varepsilon}(p) = p(\varepsilon(p))$.

Definition.

A quantifier $\phi \in KX$ is attainable if it has a selection function $\varepsilon \in JX$:

$$\phi = \overline{\varepsilon}.$$

J and K are strong

The strengths

$X \times TY \to T(X \times Y)$.

extend to

$\otimes\colon TX \times TY \to T(X \times Y)$.

NB. The monads are not commutative.

The extension of the co-strengths $TX \times Y \to T(X \times Y)$

give different maps $\otimes'\colon TX \times TY \to T(X \times Y)$.

Terminology and examples

1. $\otimes\colon KX \times KY \to K(X \times Y)$.

   Product of quantifiers.

   $$(\forall_X \otimes \exists_Y)(p) = \forall x \in X . \exists y \in Y . p(x,y).$$

   $$\exists_X \otimes \exists_Y = \exists_{X \times Y}.$$

2. $\otimes\colon JX \times JY \to J(X \times Y)$.

   Product of selection functions.

   $$(\varepsilon \otimes \delta)(p) = (a, b(a))$$

   where $b(x) = \delta(\lambda y . p(x,y))$,

   $$a = \varepsilon(\lambda x . p(x, b(x))).$$

Theorem

Attainable quantifiers are closed under finite products.

Proof.

The monad morphism gives

$$\overline{\varepsilon} \otimes \overline{\delta} = \overline{\varepsilon \otimes \delta}.$$

Hence if $\phi = \overline{\varepsilon}$ and $\gamma = \overline{\delta}$,

then $\phi \otimes \gamma = \overline{\varepsilon \otimes \delta}$,

and so $\varepsilon \otimes \delta$ is a selection function for $\phi \otimes \gamma$.

Example 1

Binary Tychonoff theorem.

The product of two searchable sets is searchable.

Proof.

$\varepsilon \in JX$ selection function for $\exists_K \in KX$ with $K \subseteq X$.

$\delta \in JY$ selection function for $\exists_L \in KY$ with $L \subseteq Y$.

$\varepsilon \otimes \delta$ selection function for $\exists_K \otimes \exists_L = \exists_{K \times L} \in K(X \times Y)$.

Example 2

In every pub there are a man a and a woman b such that

every man buys a drink to some woman iff a buys a drink to b.

Proof.

This amounts to $(\forall x \in X. \exists y \in Y. p(x,y)) = p(a,b)$.

By the Drinker paradoxes, the quantifiers $\forall_X$ and $\exists_Y$

have selection functions A and E.

By the above theorem, the quantifier $\forall_X \otimes \exists_Y$ has a selection function $A \otimes E$.

Hence we can take $(a,b) = (A \otimes E)(p)$.

Countable Tychonoff theorem for searchable sets

Arbitrary products of compact sets are compact.

Countable products of searchable sets are searchable.

Countable “strength”:

$$\bigotimes \colon \prod_i JX_i \to J \prod_i X_i,$$

characterized by

$$\bigotimes_i \varepsilon_i = \varepsilon_0 \otimes \bigotimes_i \varepsilon_{i+1}.$$

NB. This exists only in particular categories.

III. Game theory.

Products of selection functions calculate

1. optimal plays, and

2. optimal strategies.

Example 1

1. Two-person game that finishes after exactly n moves.
2. Eloise starts and alternates playing with Abelard. One of them wins.
3. i-th move is an element of the set X_i.
4. A predicate $p\colon \prod_{i=0}^{n-1} X_i \to \mathrm{Bool}$ tells whether Eloise wins.
5. Eloise can win iff

$$\exists x_0 \in X_0 \quad \forall x_1 \in X_1 \quad \exists x_2 \in X_2 \quad \forall x_3 \in X_3 \cdots p(x_0, \ldots, x_{n-1}).$$

6. If $\phi_{2i} = \exists_{X_{2i}}$ and $\phi_{2i+1} = \forall_{X_{2i+1}}$, this amounts to $\left(\bigotimes_{i=1}^{n} \phi_i\right)(p)$.

Example 2

1. Two-person game that finishes after exactly n moves.

2. Eloise starts and alternates playing with Abelard. Lose, draw, win.

3. i-th move is an element of the set X_i.

4. A predicate $p\colon \prod_{i=0}^{n-1} X_i \to \{-1, 0, 1\}$.

5. The optimal outcome of the game is

$$\sup_{x_0 \in X_0} \inf_{x_1 \in X_1} \sup_{x_2 \in X_2} \inf_{x_3 \in X_3} \cdots \quad p(x_0, \ldots, x_{n-1}).$$

6. If $\phi_{2i} = \sup_{X_{2i}}$ and $\phi_{2i+1} = \inf_{X_{2i+1}}$, this again amounts to $\left(\bigotimes_{i=1}^{n} \phi_i\right)(p)$.

Sequential game of length n

$X_0, \ldots, X_{n-1}$ sets of possible moves at rounds $0, \ldots, n-1$.

$p \colon \prod_{i=0}^{n-1} X_i \to R$ outcome (or pay-off) function.

$\phi_0 \in KX_0$, . . . , $\phi_{n-1} \in KX_{n-1}$ quantifiers for each round.

We don't stipulate who plays at each round.

This is implicit in the choice of quantifiers.

Subgame

Determined by a partial play $a = (a_0, \ldots, a_{n-1}) \in \prod_{i=0}^{k-1} X_i$ for $k \leq n$:

$$(X_i, p_a, \phi_i).$$

Here $p_a\colon \prod_{i=k}^{n-1} X_i \to R$ is defined by

$$p_a(x_k, \ldots, x_{n-1}) = p(a_0, \ldots, a_{k-1}, x_k, \ldots, x_{n-1}),$$

Like the original game but starts at the position determined by the moves a.

Optimal outcomes and plays

1. The *optimal outcome* of the game is $w = \left(\bigotimes_{i=0}^{n-1} \phi_i\right)(p)$.

2. A play $(a_0, \ldots, a_{n-1})$ is *optimal* if

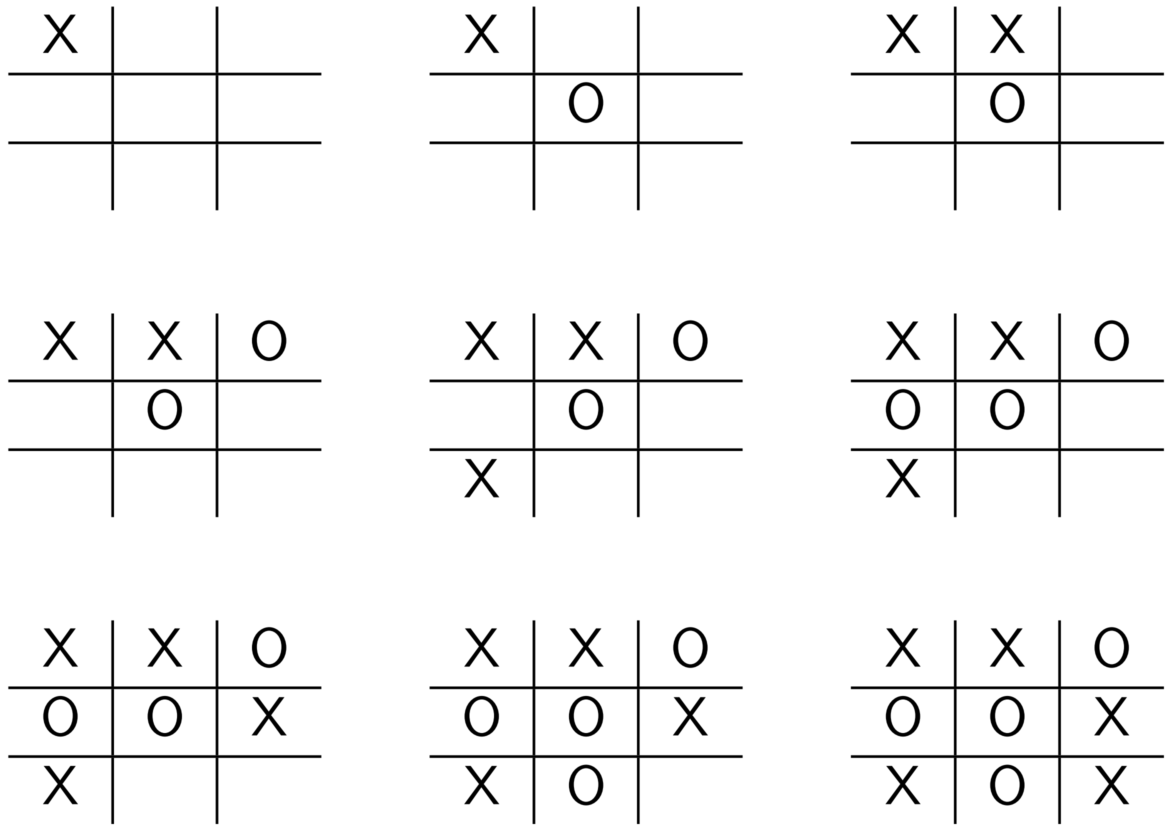
$$w_{()} = w_{(a_0)} = w_{(a_0,a_1)} = w_{(a_0,a_1,a2)} = \cdots = w_{(a_0,a_1,\ldots,a_{n-1})}.$$

All players have played as best as they could.

Example

The optimal outcome for tic-tac-toe is a draw.

An optimal play is



Optimal moves and strategies

1. A move $a_k \in X_k$ is *optimal* for a subgame $(a_0, \ldots, a_{k-1}) \in \prod_{i=0}^{k-1} X_i$ if it doesn't change the optimal outcome.

$$w_{(a_0,\ldots,a_{k-1})} = w_{(a_0,\ldots,a_{k-1},a_k)}.$$

2. A *strategy* is a family of functions,

$$\mathrm{next}_k : \prod_{i=0}^{k-1} X_i \to X_k.$$

3. A strategy is *optimal* if the move $\mathrm{next}_k(a)$ is optimal for every partial play a.

Policy functions for the game

A *policy* is a sequence of selection functions $\varepsilon_i\colon (X_i \to R) \to X_i$ for the game quantifiers.

E.g., if the policy of the player is to maximize the payoff, then $\varepsilon(p)$ is a point where p attains its maximum value.

Calculating optimal plays and strategies

Theorem. Let (X_i, p, ϕ_i) be a game with policy functions ε_i.

1. An optimal play is given by

$$a = \left(\bigotimes_{i=k}^{n-1} \varepsilon_i \right)(p).$$

2. An optimal strategy is given by

$$\mathrm{next}_k(a) = \left(\left(\bigotimes_{i=k}^{n-1} \varepsilon_i \right)(p_a) \right)_0.$$

Nash equilibria for sequential games

Calculated as in the theorem, with $R = \mathbb{R}^n$ and $\phi_i = \sup$.

(Simultaneous games are a completely different story.)

Dependent product of selection functions

Sometimes the allowed moves depend on the played moves at previous rounds.

Consider “dependent product” $TX \times (X \to TY) \to T(X \times Y)$.

E.g., $(\exists x \in X.\forall y \in Y_x.p(x,y)) = (\phi \otimes \gamma)(p)$ for $\phi = \exists_X$ and $\gamma(x) = \forall_{Y_x}$.

Iterating this (in)finitely often, we get

$$\prod_i \left(\prod_{k<i} X_k \to TX_i \right) \to T\left(\prod_i X_i \right).$$

Optimal plays and strategies calculated using this.

Let's run an example in the computer

If there is enough time left.

IV. Proof theory.

Algebras of the monad J:

$JA \to A$.

$((A \to R) \to A) \to A$.

Propositions that satisfy Peirce's Law.

Get proof translation that eliminates Peirce's Law directly.

Connection with the double-negation translation via the morphism $J \to K$.

Double negation shift

$$\forall i \in \mathbb{N}. \neg\neg A(i) \implies \neg\neg \forall i \in \mathbb{N}. A(i).$$

Used by Spector (1962) to interpret the classical axiom of countable choice.

Can be written as a K-shift:

$$\forall i \in \mathbb{N}. K A(i) \implies K \forall i \in \mathbb{N}. A(i).$$

Non-intuitionistic principle, realized by *Spector bar recursion*.

J-shift

The countable product functional $\bigotimes$: $\prod_i JX_i \to J \prod_i X_i$ realizes the J-shift

$$\forall i \in \mathbb{N}.JA(i) \implies J\forall i \in \mathbb{N}.A(i).$$

More general than the K-shift.

$\bigotimes$ is yet another form of bar recursion.

Unifying concept: selection functions for quantifiers

I. Topology in computation.

Exhaustive search.

II. The selection monad.

Selection functions for generalized quantifiers.

III. Game theory.

Optimal plays and strategies.

IV. Proof theory.

Computational extraction of witnesses from classical proofs.

Unifying concept: selection functions for quantifiers

The product of selection functions $\bigotimes$: $\prod_i JX_i \to J \prod_i X_i$ gives:

I. Topology in computation: Tychonoff theorem.

II. The selection monad: Strength.

III. Game theory: Optimal plays and strategies.

IV. Proof theory: Generalized double-negation shift, bar recursion.

Unifying concept: selection functions for quantifiers

The product of selection functions $\bigotimes\colon \prod_i JX_i \to J\prod_i X_i$ gives:

I. Topology in computation: Tychonoff theorem.

II. The selection monad: Strength.

III. Game theory: Optimal plays and strategies.

IV. Proof theory: Generalized double-negation shift, bar recursion.

Thanks!

References

1. MHE. Synthetic topology of data types and classical spaces. ENTCS'2004.

2. MHE. Infinite sets that admit fast exhaustive search. LICS'2007.

3. MHE. Exhaustible sets in higher-type computation. LMCS'2008.

4. MHE. Computability of continuous solutions of higher-type equations, LNCS'2009.

5. MHE & PO. Selection functions, bar recursion, and backward induction, MSCS'2010.

6. MHE & PO. Searchable Sets, Dubuc-Penon Compactness, Omniscience Principles, and the Drinker Paradox. CiE'2010.

7. MHE & PO. The Peirce translation and the double negation shift. LNCS'2010.

8. MHE & PO. Computational interpretations of analysis via products of selection functions, LNCS'2010.

Links

`http://math.andrej.com/2007/09/28/seemingly-impossible-functional-programs/`

`http://math.andrej.com/2008/11/21/a-haskell-monad-for-infinite-search-in-finite-time/`

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

Maybe add links to the Haskell programs here.