Topology in higher-type computation

Martín Escardó

Analytic Topology in Mathematics and Computer Science seminar, Oxford, 17th May 2010

Plan

Part I. Topological spaces for computation.

Part II. Turning topological theorems into computational theorems.

Part III. 5-minute excursion in game theory and proof theory.

Plan

Part I. Topological spaces for computation.

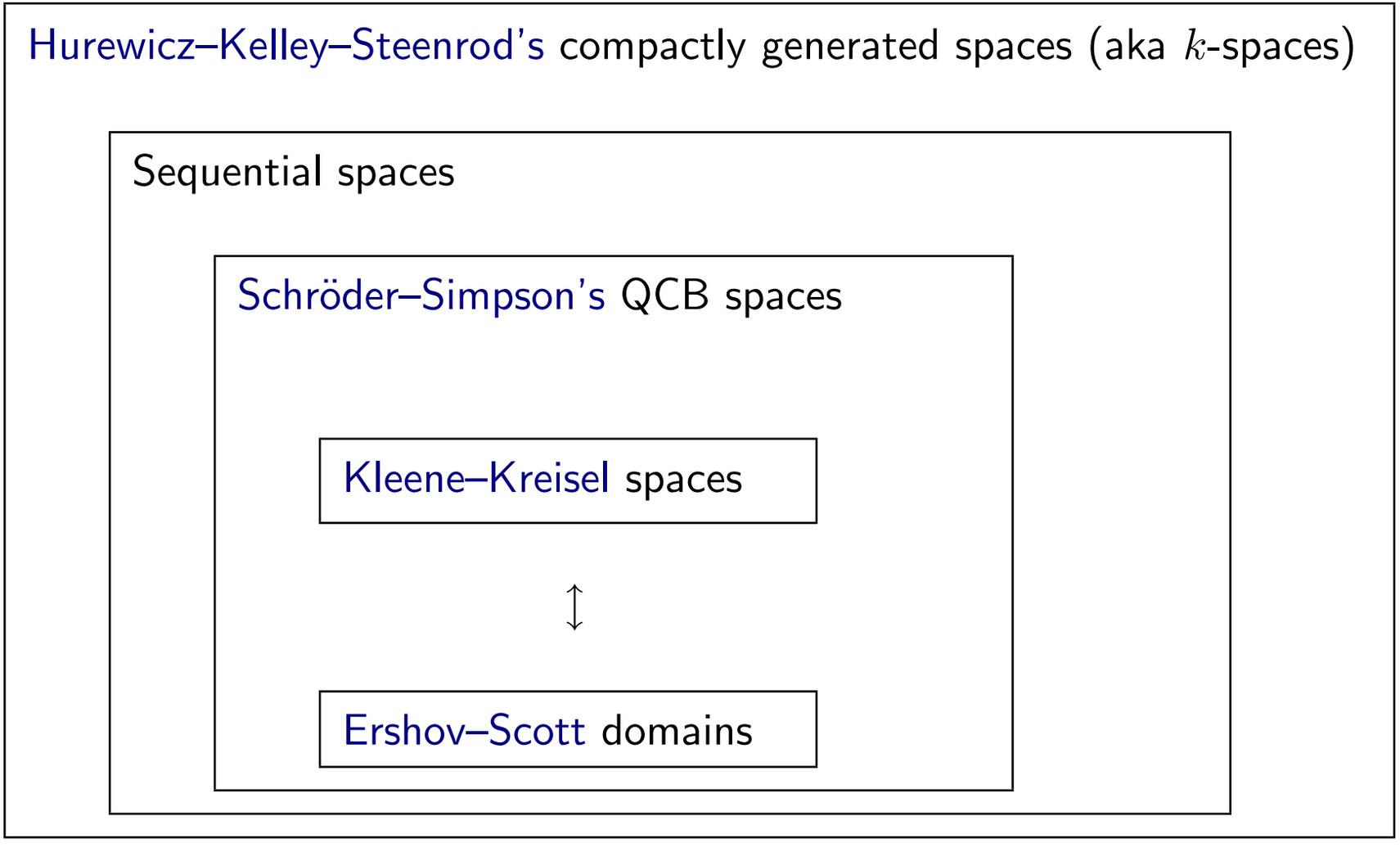
Part II. Turning topological theorems into computational theorems.

Part III. 5-minute excursion in game theory and proof theory.

If you attended my MFPS talk in Ottawa two weeks ago, you have seen a shortened version of (II) and an extended version of (III), but not (I).

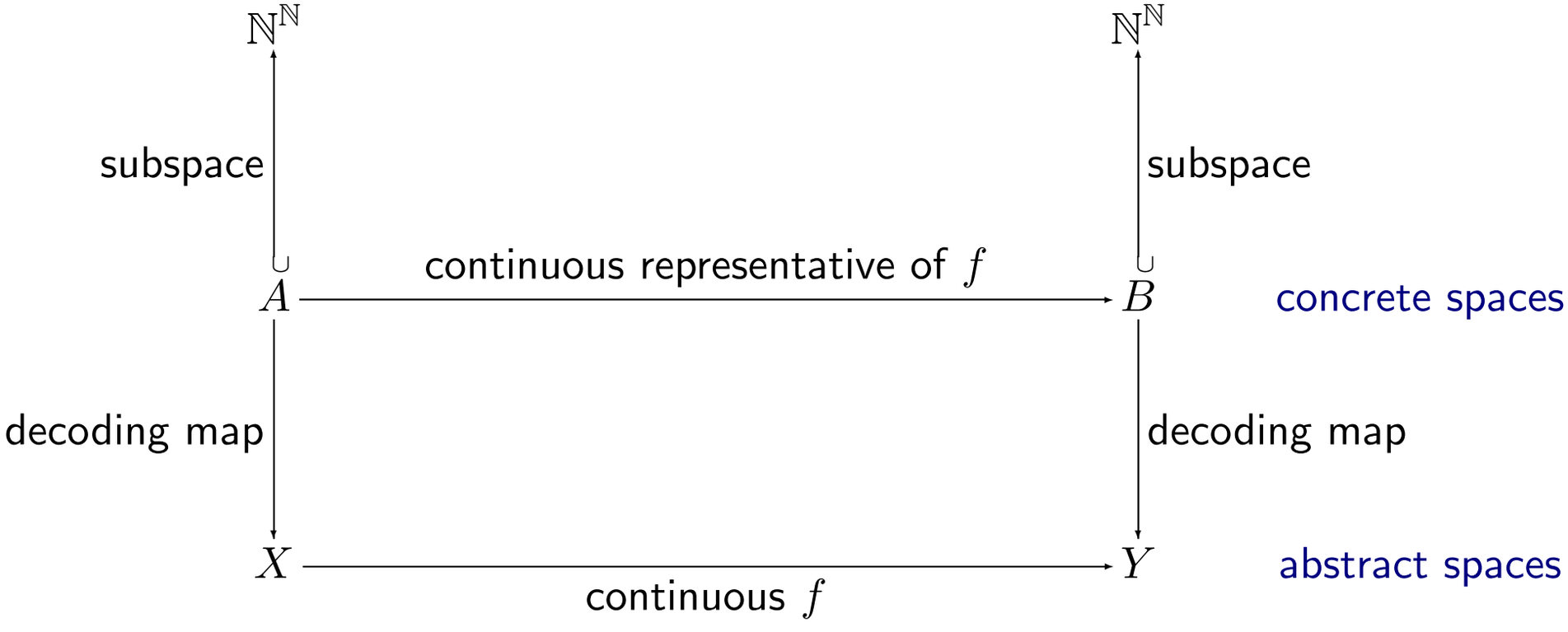
If you attended my 2007 talk at this seminar, you have seen a preliminary version of (II), but not (I) or (II).

A topological zoo for computation

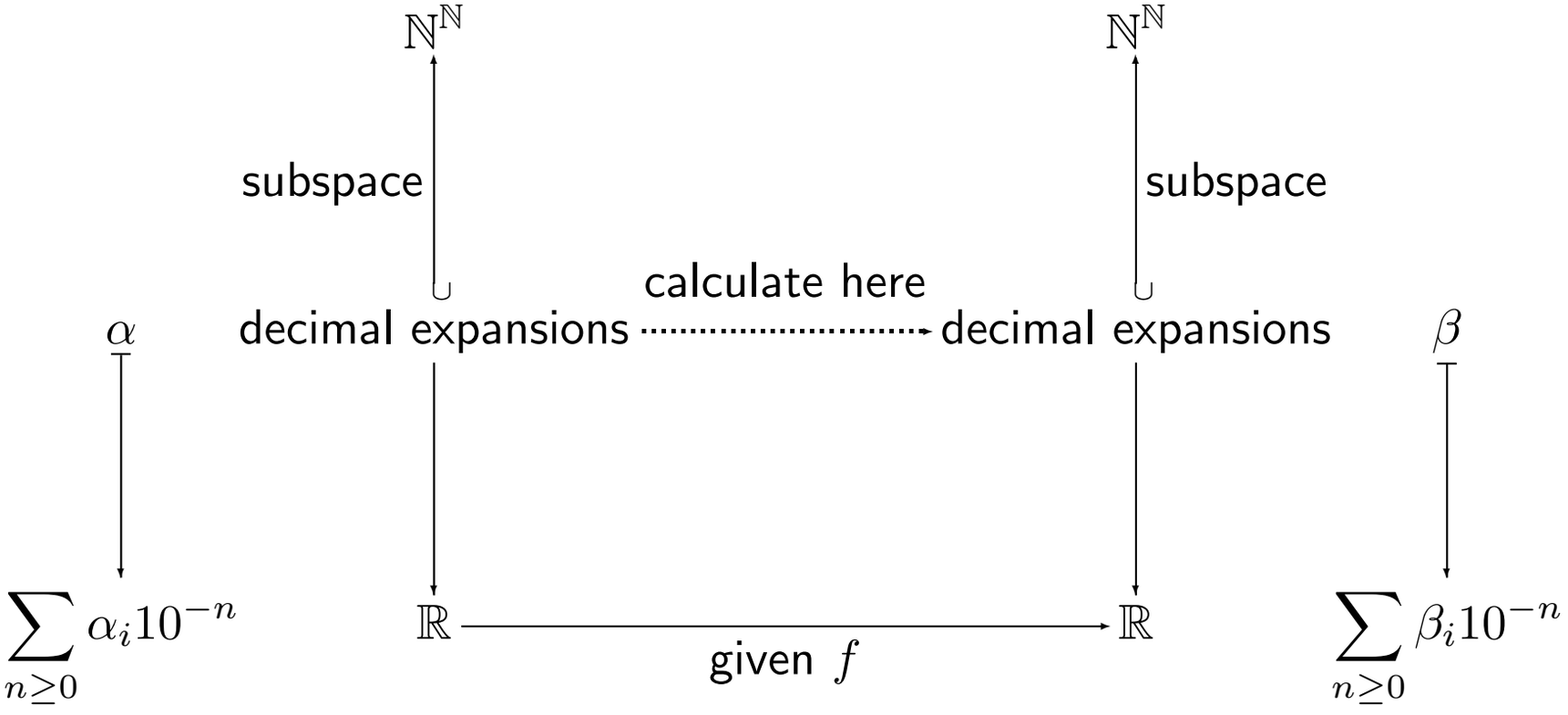


1. Cartesian closed categories.
2. QCB is suitable for computation.
3. Countable products and function spaces coincide in super- and sub-categories.

Concrete representations of QCB spaces



Example: decimal notation



Assume $\alpha_0 \in \mathbb{Z}$ and $\alpha_{i+1} \in \{0, \ldots, 9\}$.

Computational significance of continuity

For a function $f\colon \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ to be computable, it is necessary that finite parts of the output depend only on finite parts of the input.

Same for functions $f\colon A \to B$ with A and B subspaces of $\mathbb{N}^{\mathbb{N}}$.

For every $\alpha \in A$ and every output precision n, there is an input precision m such that $\alpha' =_m \alpha$ implies $f(\alpha') =_n f(\alpha)$ for all $\alpha' \in A$.

This is continuity for the product topology, or the metric

$$d(\alpha, \alpha') = \inf\{2^{-n} \mid \alpha =_n \alpha'\}.$$

$\ldots \alpha_3, \alpha_2, \alpha_1, \alpha_0 \longrightarrow$

In order to calculate f, repeatedly do:

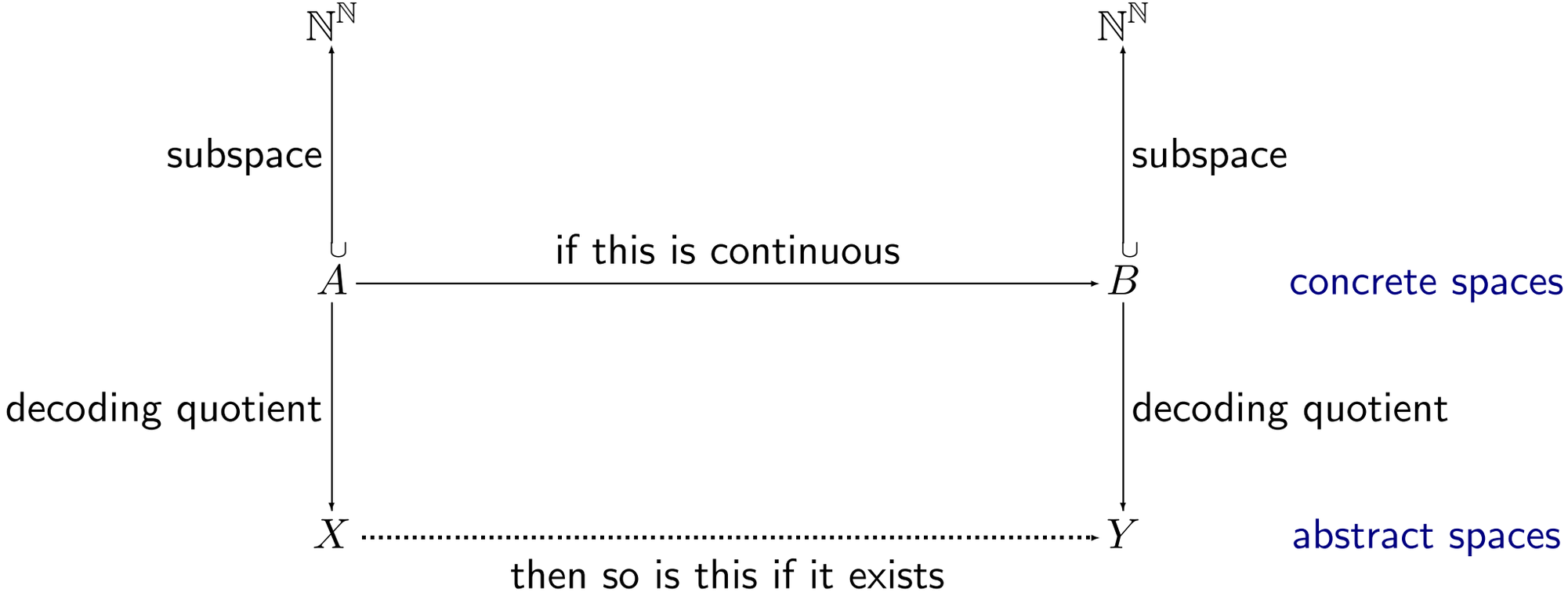
1. read finitely many digits
2. calculate for a finite amount of time
3. write finitely many digits

$\longrightarrow \ldots \beta_3, \beta_2, \beta_1, \beta_0$

Concrete versus abstract continuity

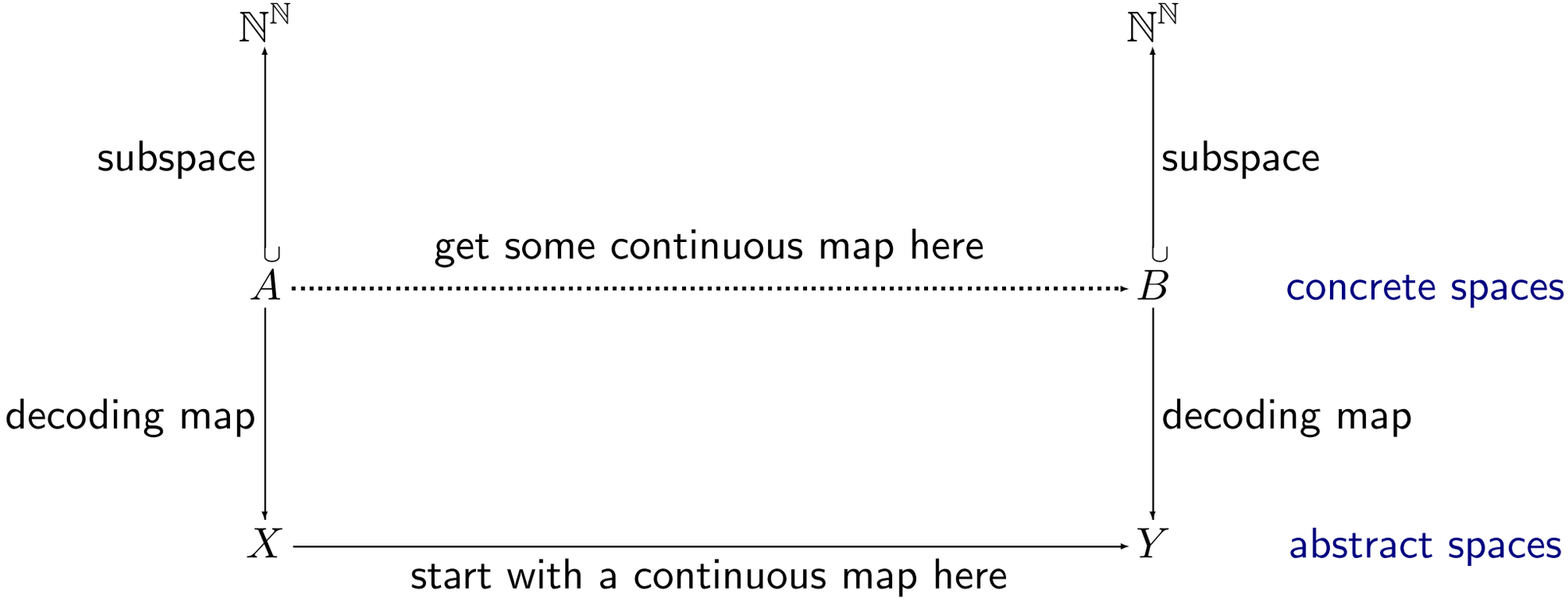
We assume the decoding map to be a topological quotient.

Then for every concrete map there corresponds at most one abstract map.



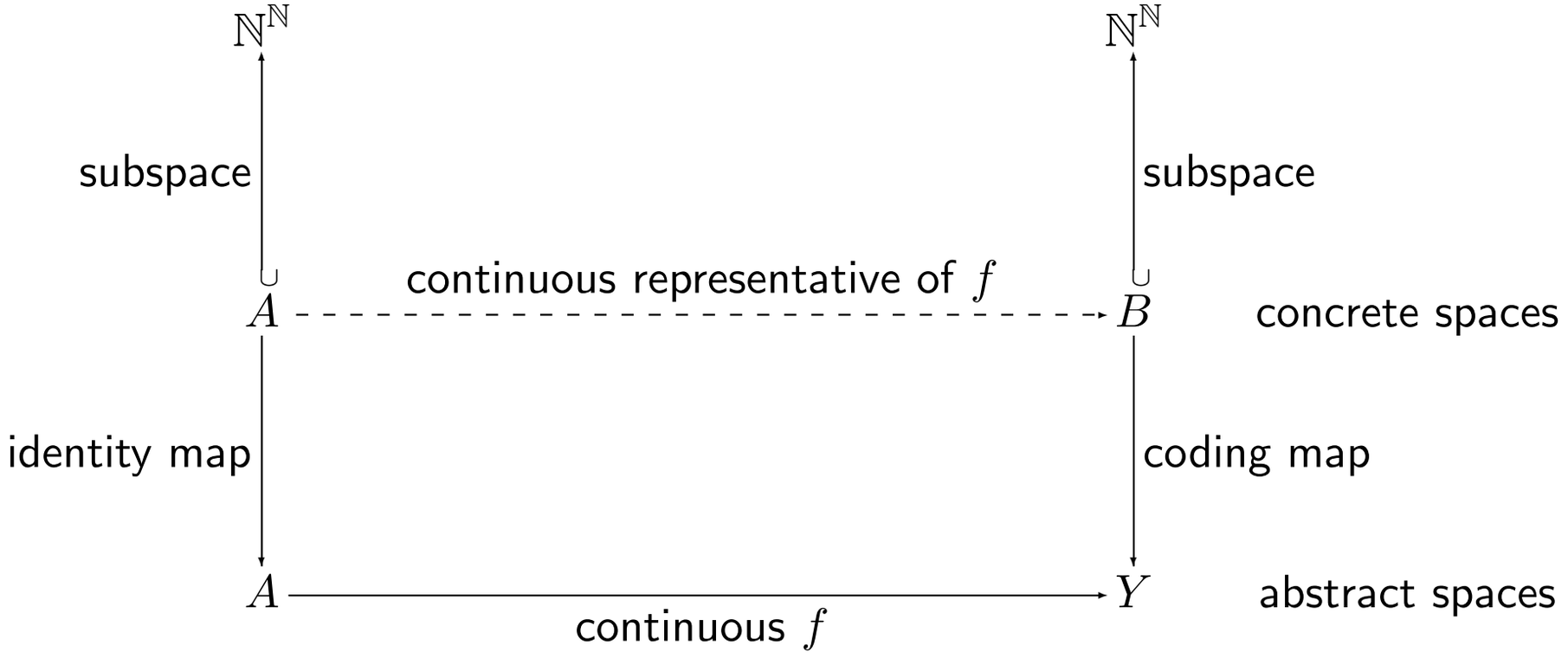
Admissible representations of topological spaces

For every continuous abstract map there is at least one corresponding continuous concrete map.



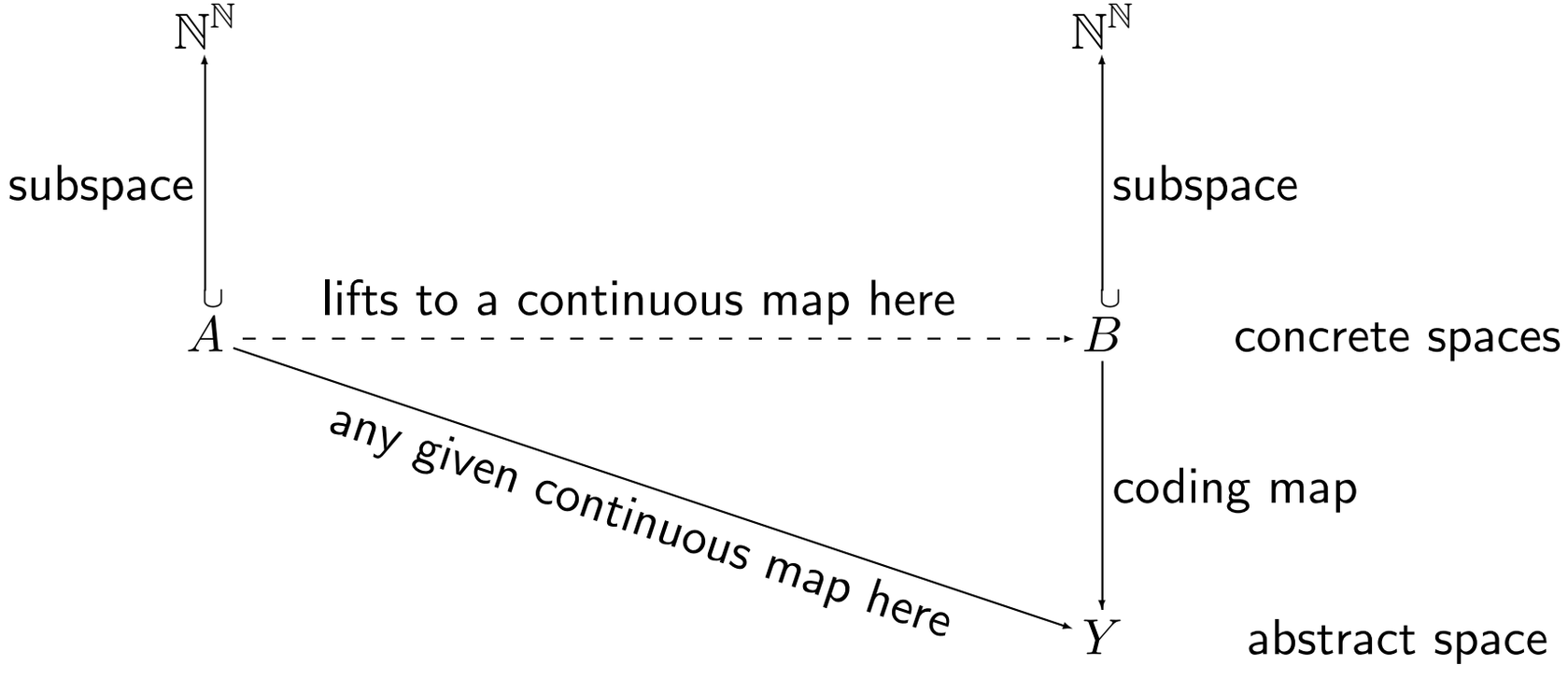
To make this precise we take $X = A$ and the identity as the decoding map.

Admissible representation of a space Y



Admissible representation

An admissible representation of a space Y is a continuous surjection $\mathbb{N}^{\mathbb{N}} \supseteq B \to Y$ with the following lifting property:



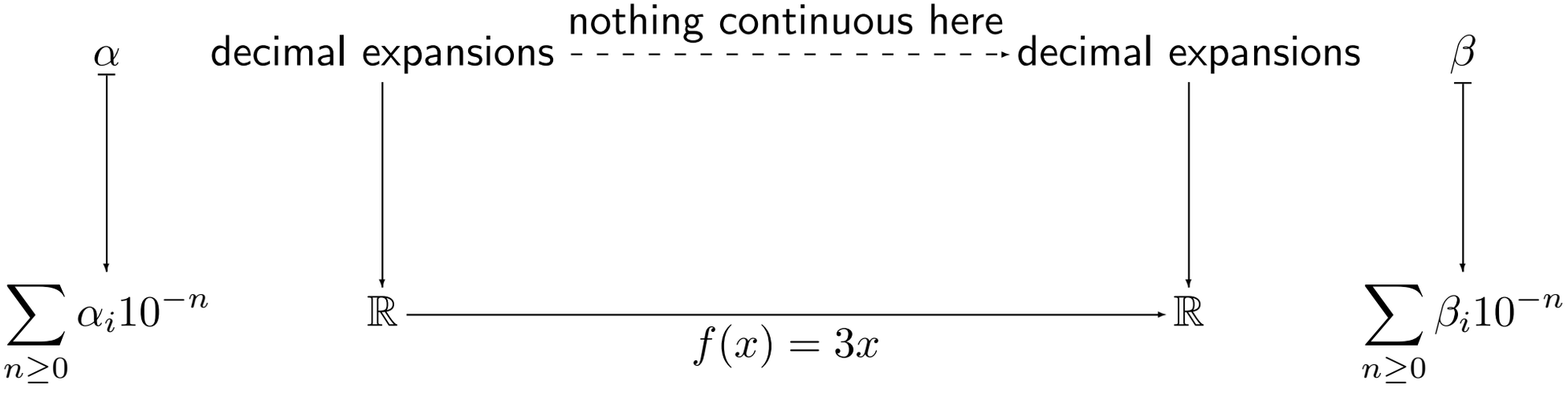
Can continuously translate from A-notation to B-notation.

Admissibility is a maximality condition.

Any two admissible representations are continuously reducible to each other.

Surprising counter-example (Brouwer 1920's)

Decimal notation is not admissible.



Suppose the input starts 0.333333.

The first digit of the output must be 0 if we eventually read a digit < 3.

The first digit of the output must be 1 if we eventually read a digit > 3.

While we read digits $= 3$, we cannot produce any output for lack of information.

$\Longrightarrow$ If the input is $1/3$, the next output digit depends on *all* input digits.

This is not a reason to despair

The real line does have admissible representations:

1. Allow negative digits (Gauss).
2. Use binary notation, but with a non-integral base (Brouwer).
3. Nested sequences of rational intervals (Grzegorczyk).
4. Rational cauchy sequences with fixed rate of convergence (Bishop).
5. Continued fractions with rational entries (Vuillemin).

QCB spaces

TFAE for any T_0 topological space X:

1. X is a quotient of some countably based space.
2. X is sequential and has a countable pseudo-base.
3. X has an admissible quotient representation.

The equivalence of (1) and (2) doesn't need the T_0 separation assumption.

QCB is equivalent to a full subcategory of Scott's equilogical spaces.

These results are due to Schröder, Simpson, Menni, Bauer.

QCB spaces

TFAE for any T_0 topological space X:

1. X is a quotient of some countably based space.
2. X is sequential and has a countable pseudo-base.
3. X has an admissible quotient representation.

A pseudo-base is a collection $\mathcal{B}$ of sets such that

for any convergent sequence $x_n \longrightarrow x$ and

any open neighbourhood U of x,

there is $B \subseteq U$ in $\mathcal{B}$ such that

$x \in B$ and

eventually $x_n \in B$ for all n.

Computation on QCB spaces

Two equivalent approaches:

1. Via admissible representations.
2. Via equilogical spaces.

Give different insight.

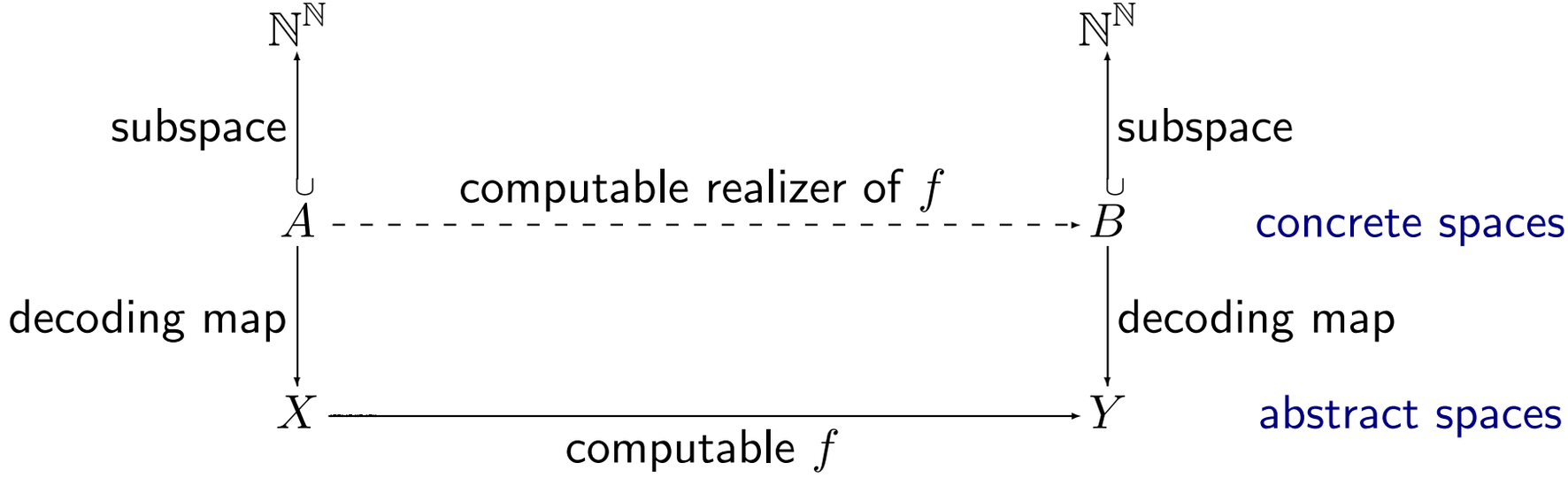
Both are very useful for various purposes.

But both require more investigation in my opinion.

Computation on QCB spaces via admissible representations

Assume that X and Y are endowed with addmissible representations A and B.

$f : X \to Y$ is called computable if it has at least one computable realizer $A \to B$.



The point is that computability on subspaces of $\mathbb{N}^{\mathbb{N}}$ is well understood since the 1950's.

One is reducing the general situation to this particular case. This is the so-called TTE approach of Weihrauch's school, with important contributions by Schröder.

Kleene associates

They are admissible representations for Kleene–Kreisel spaces.

This is how Kleene defined computability.

He also studied formal systems for computation and related the two.

The full relationship was fully elucidated later (Tait, Berger).

Computation on QCB spaces via equilogical spaces

We instead reduce computability on general QCB spaces to computations on effectively given domains.

This is Scott's school approach, with important contributions by Bauer.

Types

For computational purposes, one often works with inductively defined collections of spaces.

Example 1. Kleene–Kreisel hierarchy.

Start with $\mathbb{N}$.

Close under finite products and function spaces.

We get $\mathbb{N}, \mathbb{N}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}^{\mathbb{N}}}, \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}^{\mathbb{N}}} \ldots$

Example 2. Generalized Kleene–Kreisel hierarchy.

Further close under countable products and retracts.

Or perhaps r.e. products and computable retracts.

Example 3. Add the reals $\mathbb{R}$ to Example (2).

Formal systems for computation

Typically based on the typed lambda-calculus.

Can be interpreted thanks to cartesian closedness.

Example 1. Gödel's system T.

Example 2. Platek–Scott–Plotkin's PCF and FPC.

Example 3. Escardó's Real PCF and Real FPC.

Example 4. Practical programming languages based on (2).

ML, OCaml, Haskell, . . .

Milner is mainly responsible making these

theoretical tools available practical use with many insights.

PCF

Based on the Ershov–Scott hierarchy.

Allows algorithm definitions by recursion (fixed-point theorem).

Hereditarily total functionals match the Kleene–Kreisel hierarchy.

Types. Topological spaces or domains.

Programs. Continuous maps.

Kleene was also interested in sequential computation.

Game theoretic models by Abramsky–Jagadeesan–Malacaria and Hyland-Ong-Nickau.

I think the connection between topological and games models deserves attention.

Normann showed that every total functional is equivalent to a sequential one.

FPC

Further allows type definitions by recursion (domain equations).

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.

However, you can often come up with a proof that works for both topology and computation. (I called this *synthetic topology*.)

Exhaustive search

Algorithmically check all possibilities in finite time.

Either find something.

Or else report that there is nothing to be found.

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).

Computational common wisdom

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

Can't always trust common wisdom

E.g. The Cantor space $2^{\mathbb{N}}$ is exhaustively searchable.

Many other examples.

Corollary

The type $\mathbb{N}^{2^{\mathbb{N}}} = ((\mathbb{N} \to 2) \to \mathbb{N})$ has decidable equality.

Proof. Given $f, g \colon (\mathbb{N} \to 2) \to \mathbb{N}$.

Apply exhaustibility of the Cantor space.

Check whether whether or not $f(\alpha) = g(\alpha)$ for every $\alpha \colon \mathbb{N} \to 2$.

Exhaustible set

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

1. *Input:* $p\colon X \to 2$ decidable. $\qquad 2 = \{0, 1\} = \{\mathrm{False}, \mathrm{True}\}.$
2. *Output:* True or False.
3. *Specification:* output True iff $\exists k \in K.p(k) = \mathrm{True}$.

The algorithm has type $(X \to 2) \to 2$.

Searchable set

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

1. *Input:* $p\colon X \to 2$ 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 type $(X \to 2) \to 1 + X$.

Of course, by definition,

Searchable $\Longrightarrow$ exhaustible.

Searchable set, slightly different notion and formulation

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

1. *Input:* $p\colon X \to 2$ 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.

(This has to do with the computable Tychonoff theorem discussed below.)

The algorithm has type $(X \to 2) \to X$.

Still

Searchable $\Longrightarrow$ 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 $\exists_K \colon (X \to 2) \to 2$.

The boolean existential quantification functional is computable.

Searchable: algorithm $\varepsilon_K \colon (X \to 2) \to X$.

The set K has a computable a selection function.

Derived functionals:

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

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

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 $\alpha\colon \mathbb{N} \to \mathbb{N}$ is *not* searchable.

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

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

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

More examples (LMCS'2008)

Consider the types defined by induction as follows:

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

Theorem.

1. Compact types are searchable.

2. Discrete types have decidable equality.

E.g. $(((\mathbb{N} \to 1+1) \to \mathbb{N}) \to 1+1+1) \to ((\mathbb{N} \to 1+1+1+1) \to \mathbb{N})$ has decidable equality.

Theorems (LMCS'2008)

1. The non-empty exhaustible sets are the computable images of the Cantor space $(\mathbb{N} \to 2)$.

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.

7. Arzela–Ascoli type characterization of searchable subspace of function spaces. (For details see publications.)

Is this feasible?

I have implemented this.

I have some counter-intuitively fast examples.

See publications.

Computational Tychonoff theorem in more detail

Countable products of searchable sets are searchable.

Write $JX = ((X \to 2) \to X)$ for the type of selection functions.

Our algorithm has type $\prod_i JX_i \to J\prod_i X_i$.

Given selection functions $\varepsilon_0, \varepsilon_1, \varepsilon_2, \ldots$ for sets $K_0 \subseteq X_0, K_1 \subseteq X_1, K_2 \subseteq X_2, \ldots$,
our algorithm produces a single selection function δ for the set $\prod_i K_i \subseteq \prod_i X_i$.

So the input is a sequence $\varepsilon \in \prod_i JX_i$ of selection functions,
and the output is a single selection function $\delta \in J\prod_i X_i$.

Flavour of how the product algorithm works

Let's consider the binary case first. We want an algorithm $JX \times JY \to J(X \times Y)$.

Given two selection functions

$\varepsilon \in JX$ for $K \subseteq X$, and

$\delta \in JY$ for $L \subseteq Y$,

we want to construct a selection function

$\varepsilon \otimes \delta \in J(X \times Y)$ for $K \times L \subseteq X \times Y$.

This is what we do:

Given $p \colon X \times Y \to 2$, we

find $a \in K$ such that there is $y \in K$ with $p(a, y) = \mathrm{True}$,

find $b \in L$ such that $p(a, b) = \mathrm{True}$.

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

In summary

We define an algorithm/continuous functional $(\otimes)\colon JX \times JY \to J(X \times Y)$ by, for all $p\colon X \times Y \to 2$,

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

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

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

Why is this continuous?

Because we are working in a cartesian closed category of spaces, and hence all maps that are λ-definable from continuous functions are themselves continuous.

Why is this computable?

Because the definition can also be interpreted in a cartesian closed category of computable maps.

Countable case

We define a functional $\bigotimes\colon \prod_i JX_i \to J\prod_i X_i$ using a fixed-point theorem.

Intuitively, we iterate the binary case infinitely often:

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

Countable case – mathematical details

Formally, we find a continuous solution F to the equation

$$F(\varepsilon) = \varepsilon_0 \otimes F(\lambda i.\varepsilon_{i+1})$$

and show it is unique by bar induction.

We can do this for QCB spaces.

We use the fact that QCB is fully embedded into a category of partial equivalence relations of domains, with an embedding that preserves products and function spaces.

We use bar induction to argue that the domain theoretic solution is invariant under the partial equivalence relation.

By fullness we get the map in QCB.

(Bar induction is used for both uniqueness and existence.)

Countable case – computational details

With $X_i = X$ for all i, this can be written down in PCF.

(For the general case we require more sophisticated formal systems.)

The above argument tell us that we get a *total functional*.

Hence the computable functional lives not only in the Ershov–Scott hierarchy, but also in the Kleene–Kreisel one.

I have also written down this functional in practical versions of PCF.

I have got surprisingly fast examples.

Recent directions with Paulo Oliva

I originally considered the case $X_i = X$, and the generalization is in joint work with him. More interestingly:

1. We made precise how the product functional is in fact a form of bar recursion.

2. We showed that J is a strong monad with the binary product of selection functions coming from the strength. There is a monad morphism into the continuation monad.

3. We showed that products of selection functions calculate optimal strategies and Nash equilibria for sequential games.

4. We showed that the infinite product of selection functions realizes a generalization of Spector's double-negation shift.

5. We have applied the infinite-product functional to proof theory, to computationally extract witnesses from classical proofs in the presence of countable and dependent choice.

6. We have a proof translation of classical logic into intuitionistic logic based on the monad J.

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, Nash equilibria.

IV. Proof theory.

Computational extraction of witnesses from classical proofs.

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, and Nash equilibria.

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

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.