

Types and Type-Free λ -Calculus

Dana S. Scott, FBA, FNAS

University Professor Emeritus
Carnegie Mellon University

Visiting Scholar
University of California, Berkeley

My Personal Motivation

- **Denotational semantics** started in Oxford in late 1969.
- It was hoped that **domain theory** would provide a basis both for **recursive definitions** in programs and recursive definitions of **semantical structures**.
- Early troubles were encountered in using **tops** and **bottoms**.
 - Soon researchers turned to **operational semantics**.
 - Others wanted to expand work to **communicating** and **parallel processes**.
- **Axiomatic** and **synthetic theories** did not solve the problems.
 - As a result much research effort in domain theory **faded**.
 - Late work by Reynolds and collaborators, however, has opened up **new** and **promising** approaches.
- Perhaps the much simplified modeling using **enumeration operators** can spark new investigations.

Church's λ -Calculus

Definition. λ -calculus — as a formal theory — has rules for the *explicit definition* of functions via well known equational axioms:

α -conversion

$$\lambda X. [\dots X \dots] = \lambda Y. [\dots Y \dots]$$

β -conversion

$$(\lambda X. [\dots X \dots]) (T) = [\dots T \dots]$$

η -conversion

$$\lambda X. F(X) = F$$

NOTE: The third axiom will be dropped in favor of a theory employing properties of a **partial ordering**.

The Enumeration Operator Model

Definitions. (1) *Pairing*: $(n, m) = 2^n(2m+1)$.

(2) *Sequence numbers*: $\langle \rangle = 0$ and

$$\langle n_0, n_1, \dots, n_{k-1}, n_k \rangle = (\langle n_0, n_1, \dots, n_{k-1} \rangle , n_k).$$

(3) *Sets*: $\text{set}(0) = \emptyset$ and $\text{set}((n, m)) = \text{set}(n) \cup \{m\}$.

(4) *Kleene star*: $X^* = \{n \mid \text{set}(n) \subseteq X\}$, for sets $X \subseteq \mathbb{N}$.

Definition. The *model* is given by these definitions on **sets** of integers:

Application:

$$F(X) = \{ m \mid \exists n \in X^* . (n, m) \in F \}$$

Abstraction:

$$\lambda X . [\dots X \dots] =$$

$$\{0\} \cup \{ (n, m) \mid m \in [\dots \text{set}(n) \dots] \}$$

What is the Secret?

- (1) The powerset $\mathcal{P}(\mathbb{N}) = \{X \mid X \subseteq \mathbb{N}\}$ is a **topological space** with the sets $\mathcal{U}_n = \{X \mid n \in X^*\}$ as a **basis** for the topology.
- (2) Functions $\Phi: \mathcal{P}(\mathbb{N})^n \rightarrow \mathcal{P}(\mathbb{N})$ are **continuous** iff, for all $m \in \mathbb{N}$, we have $m \in \Phi(X_0, X_1, \dots, X_{n-1})$ iff there are $k_i \in X_i^*$ for each of the $i < n$, such that $m \in \Phi(\text{set}(k_0), \text{set}(k_1), \dots, \text{set}(k_{n-1}))$.
- (3) The application operation $F(X)$ is continuous as a function of **two** variables.
- (4) If the function $\Phi(X_0, X_1, \dots, X_{n-1})$ is continuous, then the abstraction term $\lambda X_0. \Phi(X_0, X_1, \dots, X_{n-1})$ is continuous in all of the **remaining variables**.
- (5) If $\Phi(X)$ is continuous, then $\lambda X. \Phi(X)$ is the **largest set** F such that for all sets T , we have $F(T) = \Phi(T)$. And, therefore, generally $F \subseteq \lambda X. F(X)$.

NOTE: This model could easily have been defined in 1957!!
It clearly satisfies the rules of **α , β -conversion** (but not η).

THIS LECTURE IS DEDICATED TO THE MEMORIES OF

John R. Myhill

Born: 11 August 1923, Birmingham, UK

Died: 15 February 1987, Buffalo, NY

John Shepherdson

Born: 7 June 1926, Huddersfield, UK

Died: 8 January 2015, Bristol, UK

Hartley Rogers, Jr.

Born: 6 July, 1926, Buffalo, NY

Died: 17 July, 2015, Waltham, MA

- John Myhill and John C. Shepherdson, *Effective operations on partial recursive functions*, **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik**, vol. 1 (1955), pp. 310-317.
- Richard M. Friedberg and Hartley Rogers Jr., *Reducibility and completeness for sets of integers*, **Mathematical Logic Quarterly**, vol. 5 (1959), pp. 117-125. Some earlier results are presented in an abstract in **The Journal of Symbolic Logic**, vol. 22 (1957), p. 107.
- Hartley Rogers, Jr., **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, 1967, xix + 482 pp.

Some Lambda Properties

Theorem. For all sets of integers F and G we have:

$$\lambda X.F(X) \subseteq \lambda X.G(X) \text{ iff } \forall X.F(X) \subseteq G(X),$$

$$\lambda X.(F(X) \cap G(X)) = \lambda X.F(X) \cap \lambda X.G(X),$$

and

$$\lambda X.(F(X) \cup G(X)) = \lambda X.F(X) \cup \lambda X.G(X).$$

Definition. A continuous operator $\Phi(X_0, X_1, \dots, X_{n-1})$

is **computable** iff in the model this set is **RE**:

$$F = \lambda X_0 \lambda X_1 \dots \lambda X_{n-1} . \Phi(X_0, X_1, \dots, X_{n-1}).$$

How to do Recursion?

Three Basic Theorems.

- All pure λ -terms define *computable* operators.
- If $\Phi(X)$ is continuous and if we let $\nabla = \lambda X. \Phi(X(X))$, then the set $P = \nabla(\nabla)$ is the *least fixed point* of Φ .
- The least fixed point of a *computable* operator is computable.

A Principal Theorem. These computable operators:

$$\text{Succ}(X) = \{n+1 \mid n \in X\},$$

$$\text{Pred}(X) = \{n \mid n+1 \in X\}, \text{ and}$$

$$\text{Test}(Z)(X)(Y) = \{n \in X \mid 0 \in Z\} \cup \{m \in Y \mid \exists k. k+1 \in Z\},$$

together with λ -calculus, suffice for defining **all RE sets**.

Gödel Numbering

Theorem. There is a computable $V = \lambda x. V(x)$ where

- (i) $V(\{0\}) = \lambda y. \lambda x. y,$
- (ii) $V(\{1\}) = \lambda z. \lambda y. \lambda x. z(x)(y(x)),$
- (iii) $V(\{2\}) = \text{Test},$
- (iv) $V(\{3\}) = \text{Succ},$
- (v) $V(\{4\}) = \text{Pred},$ and
- (vi) $V(\{4 + (n, m)\}) = V(\{n\})(V(\{m\})).$

Theorem. Every *recursively enumerable set* is of the form $V(\{n\})$.

NOTE: The operator V is the analogue of the
Universal Turing Machine.

Inseparable Sets?

Definition. Modify the definition of \mathbf{V} via *finite approximations*:

- (i) $\mathbf{V}_k(\{n\}) = \mathbf{V}(\{n\}) \cap \{i \mid i < k\}$ for $n < 5$, and
- (ii) $\mathbf{V}_k(\{4 + (n, m)\}) = \mathbf{V}_k(\{n\}) (\mathbf{V}_k(\{m\}))$.

Theorem. Each $\mathbf{V}_k(\{n\}) \subseteq \mathbf{V}_{k+1}(\{n\})$ is *finite*,
the predicate $j \in \mathbf{V}_k(\{n\})$ is *recursive*,
and we have:

$$\mathbf{V}(\{n\}) = \bigcup_{k < \infty} \mathbf{V}_k(\{n\}).$$

Theorem. The sets \mathcal{L}_0 and \mathcal{L}_1 are *recursively enumerable*,
disjoint, and *recursively inseparable*:

$$\mathcal{L}_0 = \{n \mid \exists j [0 \in \mathbf{V}_j(\{n\})(\{n\}) \wedge 1 \notin \mathbf{V}_j(\{n\})(\{n\})]\}$$

$$\mathcal{L}_1 = \{n \mid \exists k [1 \in \mathbf{V}_k(\{n\})(\{n\}) \wedge 0 \notin \mathbf{V}_k(\{n\})(\{n\})]\}$$

What Happened to the Scott Domains?

Definition. A set $c = \lambda x. c(x)$ represents a **closure operator** iff for all $x \subseteq \mathbb{N}$ we have

$$x \subseteq c(x) = c(c(x)).$$

Theorem. The fixed points of closure operators give examples (up to isomorphism) of **all** countably based **algebraic lattices**.

Moreover we have a construction of a **cartesian closed category** out of the class of closure operators using these definitions:

$$F : C \rightarrow D \text{ iff } F = D \circ F \circ C$$

and

$$(C \rightarrow D) = \lambda F. D \circ F \circ C,$$

$$\text{where } F \circ G = \lambda X. F(G(X)).$$

Are There More General Types?

Definition. *Pairing functions* for sets in $\mathcal{P}(\mathbb{N})$ can be defined by these **computable** enumeration operators:

$$\mathbf{Pair}(X)(Y) = (X, Y) = \{2n \mid n \in X\} \cup \{2m+1 \mid m \in Y\},$$

$$\mathbf{Fst}(Z) = \{n \mid 2n \in Z\}, \text{ and } \mathbf{Snd}(Z) = \{m \mid 2m+1 \in Z\}.$$

Moreover, we may now regard

$$\mathcal{P}(\mathbb{N}) = \mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N}), \text{ and for } \mathcal{A} \subseteq \mathcal{P}(\mathbb{N}) \text{ we write}$$

$$x \mathcal{A} y \text{ iff } (x, y) \in \mathcal{A}.$$

Definition. By a **type** over $\mathcal{P}(\mathbb{N})$ we understand a **(partial) equivalence relation**

$\mathcal{A} \subseteq \mathcal{P}(\mathbb{N})$ where, for all $x, y, z \in \mathcal{P}(\mathbb{N})$, we have

$x \mathcal{A} y$ implies $y \mathcal{A} x$, and

$x \mathcal{A} y$ and $y \mathcal{A} z$ imply $x \mathcal{A} z$.

Additionally we often write $x:\mathcal{A}$ for $x \mathcal{A} x$.

Let \mathcal{T} be the **class** of all such types.

The Category of Types

Definition. The *exponentiation* of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ is defined as that equivalence relation where

$F(\mathcal{A} \rightarrow \mathcal{B})G$ *iff* $\forall X, Y. X \mathcal{A} Y$ implies $F(X) \mathcal{B} G(Y)$.

When we write $F: \mathcal{A} \rightarrow \mathcal{B}$, we should understand the

F as standing for its *whole* equivalence class.

Note that $F: \mathcal{A} \rightarrow \mathcal{B}$ *implies* $\forall X. X: \mathcal{A}$ implies $F(X): \mathcal{B}$.

Definition. The *product* of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$

is defined as that equivalence relation where

$X(\mathcal{A} \times \mathcal{B})Y$ *iff* $\mathbf{Fst}(X) \mathcal{A} \mathbf{Fst}(Y)$ and $\mathbf{Snd}(X) \mathcal{B} \mathbf{Snd}(Y)$.

And we have

$X: (\mathcal{A} \times \mathcal{B})$ *iff* $\mathbf{Fst}(X): \mathcal{A}$ and $\mathbf{Snd}(X): \mathcal{B}$.

Note: Types do form a cartesian closed category.

Isomorphic Types

Definition. Two types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ are *isomorphic*, in symbols $\mathcal{A} \cong \mathcal{B}$, provided there are sets

$F: \mathcal{A} \rightarrow \mathcal{B}$ and $G: \mathcal{B} \rightarrow \mathcal{A}$ where

$\forall x: \mathcal{A}. x \in \mathcal{A} \iff G(F(x))$ and $\forall y: \mathcal{B}. y \in \mathcal{B} \iff F(G(y))$.

Definition. The *sum* of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ is defined as that equivalence relation where $x \in (\mathcal{A} + \mathcal{B}) \iff y \in (\mathcal{A} + \mathcal{B})$ iff

either $\exists x_0, y_0 [x_0 \in \mathcal{A} \wedge y_0 \in \mathcal{B} \ \& \ x = (\{0\}, x_0) \ \& \ y = (\{0\}, y_0)]$

or $\exists x_1, y_1 [x_1 \in \mathcal{A} \wedge y_1 \in \mathcal{B} \ \& \ x = (\{1\}, x_1) \ \& \ y = (\{1\}, y_1)]$.

And we have

$x \in (\mathcal{A} + \mathcal{B})$ iff either **Fst**(x) = $\{0\}$ & **Snd**(x) : \mathcal{A}

or **Fst**(x) = $\{1\}$ & **Snd**(x) : \mathcal{B} .

Some Properties of Isomorphism

Theorem. Isomorphism is an *equivalence relation* on types,
and if $\mathcal{A}_0 \cong \mathcal{B}_0$ and $\mathcal{A}_1 \cong \mathcal{B}_1$,

then $(\mathcal{A}_0 \times \mathcal{A}_1) \cong (\mathcal{B}_0 \times \mathcal{B}_1)$, $(\mathcal{A}_0 + \mathcal{A}_1) \cong (\mathcal{B}_0 + \mathcal{B}_1)$, and $(\mathcal{A}_0 \rightarrow \mathcal{A}_1) \cong (\mathcal{B}_0 \rightarrow \mathcal{B}_1)$.

Theorem. We have these *algebraic laws* for all $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{T}$:

$(\mathcal{A} \times \mathcal{B}) \cong (\mathcal{B} \times \mathcal{A})$ and $(\mathcal{A} + \mathcal{B}) \cong (\mathcal{B} + \mathcal{A})$, and

$((\mathcal{A} \times \mathcal{B}) \times \mathcal{C}) \cong (\mathcal{A} \times (\mathcal{B} \times \mathcal{C}))$,

$((\mathcal{A} + \mathcal{B}) + \mathcal{C}) \cong (\mathcal{A} + (\mathcal{B} + \mathcal{C}))$,

$(\mathcal{A} \times (\mathcal{B} + \mathcal{C})) \cong (\mathcal{A} \times \mathcal{B}) + (\mathcal{A} \times \mathcal{C})$,

$((\mathcal{A} \times \mathcal{B}) \rightarrow \mathcal{C}) \cong (\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{C}))$,

$(\mathcal{A} \rightarrow (\mathcal{B} \times \mathcal{C})) \cong (\mathcal{A} \rightarrow \mathcal{B}) \times (\mathcal{A} \rightarrow \mathcal{C})$, and

$((\mathcal{A} + \mathcal{B}) \rightarrow \mathcal{C}) \cong (\mathcal{A} \rightarrow \mathcal{C}) \times (\mathcal{B} \rightarrow \mathcal{C})$.

Types, Subspaces, and Closures

Definition. Every $\mathcal{X} \subseteq \mathcal{P}(\mathbb{N})$ can be considered as a **topological subspace** of the powerset. The **type** corresponding to \mathcal{X} is this equivalence relation:

$$x[\mathcal{X}]y \text{ iff } x = y \in \mathcal{X}.$$

Theorem. The **topological** mappings $F: \mathcal{X} \rightarrow \mathcal{Y}$ are just the **restrictions** of enumeration operators.

Theorem. The **range** of a closure operator uniquely determines it.

Definition. The **type** corresponding to a closure operator C is this equivalence relation:

$$x[C]y \text{ iff } x = C(x) = y.$$

Note: The category of **closure operators** can now be considered as a **full subcategory** of the topological category of **subspaces**. The category of subspaces can also be considered as a **full subcategory** of the category of **types**.

Embedding Spaces

Theorem. Every countably based T_0 -space \mathcal{X} is homeomorphic to a **subspace** of $\mathcal{P}(\mathbb{N})$.

Proof Sketch: Let a subbasis for the topology of \mathcal{X} be $\{ \mathcal{O}_n \mid n \in \mathbb{N} \}$.

Define $\varepsilon: \mathcal{X} \rightarrow \mathcal{P}(\mathbb{N})$ by $\varepsilon(x) = \{ n \in \mathbb{N} \mid x \in \mathcal{O}_n \}$.

By the T_0 -axiom, this mapping is one-one onto a subspace of $\mathcal{P}(\mathbb{N})$.

Check first that the **inverse image** of opens of $\mathcal{P}(\mathbb{N})$ are open in \mathcal{X} .

Notice next that $\varepsilon(\mathcal{O}_n) = \varepsilon(\mathcal{X}) \cap \{ S \in \mathcal{P}(\mathbb{N}) \mid n \in S \}$.

Hence, the **image** of a open of \mathcal{X} is an open of the subspace.

Therefore, ε is a homeomorphism to a subspace.

Note: This theorem is originally due to:

- P. Alexandroff, *Zur Theorie der topologischen Raume*, C.R. (Doklady) Acad. Sci. URSS, vol. 11 (1936), pp, 55-58.

Dependent Products

Definition. For each $\mathcal{A} \in \mathcal{T}$ the *identity type* on \mathcal{A} is defined as that relation such that

$$z (x \equiv_{\mathcal{A}} y) w \text{ iff } z \mathcal{A} x \mathcal{A} y \mathcal{A} w.$$

Definition. For each $\mathcal{A} \in \mathcal{T}$, an \mathcal{A} -indexed family of types is a function $\beta: \mathcal{P}(\mathbb{N}) \rightarrow \mathcal{T}$, such that

$$\forall x_0, x_1. x_0 \mathcal{A} x_1 \text{ implies } \beta(x_0) = \beta(x_1).$$

Definition. The *dependent product* of an \mathcal{A} -indexed family of types, β , is this equivalence relation:

$$F_0 (\prod x: \mathcal{A}. \beta(x)) F_1 \text{ iff}$$

$$\forall x_0, x_1. x_0 \mathcal{A} x_1 \text{ implies } F_0(x_0) \beta(x_0) F_1(x_1).$$

Dependent Sums

Definition. The *dependent sum* of an \mathcal{A} -indexed family of types, \mathcal{B} , is this equivalence relation:

$$\begin{aligned} & Z_0 (\sum X : \mathcal{A}. \mathcal{B}(X)) Z_1 \text{ iff} \\ & \exists X_0, Y_0, X_1, Y_1 [X_0 \mathcal{A} X_1 \ \& \ Y_0 \mathcal{B}(X_0) Y_1 \ \& \\ & \quad Z_0 = (X_0, Y_0) \ \& \ Z_1 = (X_1, Y_1)] \end{aligned}$$

Theorem. The dependent products and dependent sums of indexed families of types are always again types.

Exercise: Relate dependent sums and products to products and exponents of single types.

Systems of Dependent Types

Definition. We say that $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ form
a **system of dependent types** iff

- $\forall X_0, X_1. [X_0 \mathcal{A} X_1 \Rightarrow \mathcal{B}(X_0) = \mathcal{B}(X_1)]$, and
- $\forall X_0, X_1, Y_0, Y_1. [X_0 \mathcal{A} X_1 \ \& \ Y_0 \mathcal{B}(X_0) Y_1 \Rightarrow \mathcal{C}(X_0, Y_0) = \mathcal{C}(X_1, Y_1)]$, and
- $\forall X_0, X_1, Y_0, Y_1, Z_0, Z_1. [X_0 \mathcal{A} X_1 \ \& \ Y_0 \mathcal{B}(X_0) Y_1 \ \& \ Z_0 \mathcal{C}(X_0, Y_0) Z_1 \Rightarrow$
 $\mathcal{D}(X_0, Y_0, Z_0) = \mathcal{D}(X_1, Y_1, Z_1)]$,

provided that $\mathcal{A} \in \mathcal{T}$, and $\mathcal{B}, \mathcal{C}, \mathcal{D}$ are functions on $\mathcal{P}(\mathbb{N})$ to \mathcal{T}
of the indicated number of arguments.

Theorem. Under the above assumptions on the
system $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$, we will always have

$$\prod X : \mathcal{A} . \sum Y : \mathcal{B}(X) . \prod Z : \mathcal{C}(X, Y) . \mathcal{D}(X, Y, Z) \in \mathcal{T}.$$

Polymorphism Anyone?

Theorem. The class \mathcal{T} of all types is a **complete lattice**, because it is closed under **arbitrary intersections**.

Example: $\lambda X. \lambda Y. (X, Y) : \bigcap_{\mathcal{A}, \mathcal{B}} (\mathcal{A} \rightarrow (\mathcal{B} \rightarrow (\mathcal{A} \times \mathcal{B})))$

Theorem. Any **monotone** $\Phi : \mathcal{T} \rightarrow \mathcal{T}$ has a least **fixed point**.

Definition. The **Scott numerals** (1963) in the λ -calculus are:
 $\underline{0} = \lambda X. \lambda F. X$, $\underline{1} = \lambda X. \lambda F. F(\underline{0})$, $\underline{2} = \lambda X. \lambda F. F(\underline{1})$, etc., and
 $\underline{\text{succ}} = \lambda Y. \lambda X. \lambda F. F(Y)$, and
 $\underline{\text{pred}} = \lambda Y. Y(\underline{0}) (\lambda X. X)$.

Example: $\mathcal{I}_{\text{scott}} = \bigcap_{\mathcal{A}} (\mathcal{A} \rightarrow ((\mathcal{I}_{\text{scott}} \rightarrow \mathcal{A}) \rightarrow \mathcal{A}))$ types the numerals.

Propositions as Types?

Definition. Every type $\mathcal{P} \in \mathcal{T}$ can be regarded as a *proposition*, where *asserting* (or *proving* \mathcal{P}) means finding *evidence* $E : \mathcal{P}$.

Convention: Under this interpretation of logic, asserting $(\mathcal{P} \times \mathcal{Q})$ means asserting a *conjunction*, asserting $(\mathcal{P} + \mathcal{Q})$ means asserting a *disjunction*, asserting $(\mathcal{P} \rightarrow \mathcal{Q})$ means asserting an *implication*, asserting $(\prod x : \mathcal{A}. \mathcal{P}(x))$ means asserting a *universal quantification*, and asserting $(\sum x : \mathcal{A}. \mathcal{B}(x))$ means asserting an *existential quantification*.

Example: Given $F : (\mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{A}))$, then asserting

$$\prod x : \mathcal{A}. \prod y : \mathcal{A}. \prod z : \mathcal{A}. F(x)(F(y)(z)) \equiv_{\mathcal{A}} F(F(x)(y))(z)$$

is the same as asserting that F is an *associative binary operation*.

Computability as a Modality?

Definition. For $\mathcal{A} \in \mathcal{T}$ define $\#\mathcal{A} = \mathcal{A} \cap \mathbf{RE}$.

Definition. $[\mathbb{N}]$ is this equivalence relation:
 $x[\mathbb{N}]y$ iff $\exists n \in \mathbb{N} . x = \{n\} = y$.

Theorem. $([\mathbb{N}] \rightarrow [\mathbb{N}])$, $([\mathbb{N}] \rightarrow ([\mathbb{N}] \cup \{\emptyset\}))$,
 $\#([\mathbb{N}] \rightarrow [\mathbb{N}])$, and $\#([\mathbb{N}] \rightarrow ([\mathbb{N}] \cup \{\emptyset\}))$
correspond to the **Baire space**, the
superspace of **partial functions**, the space
of **total recursive functions**, and the space
of **partial recursive functions**.

Example: For types $\mathcal{A}, \mathcal{B} \in \mathcal{T}$, asserting
 $\# \sum F: \mathcal{A} \rightarrow \mathcal{B} . \sum G: \mathcal{B} \rightarrow \mathcal{A} . (\prod X: \mathcal{A} . X \equiv_{\mathcal{A}} G(F(X)) \times \prod Y: \mathcal{B} . Y \equiv_{\mathcal{B}} F(G(Y)))$
is the same as asserting that \mathcal{A} and \mathcal{B} are **recursively isomorphic**.

Toward Descriptive Set Theory?

Wikipedia: Every *Polish space* is obtained as a *continuous image* of *Baire space*; in fact, every Polish space is the image of a continuous bijection defined on a closed subset of Baire space.

Similarly, every *compact* Polish space is a continuous image of *Cantor space*.

Because of these *universality properties*, and because the Baire space $(\mathbb{N} \rightarrow \mathbb{N})$ has the convenient property that it is homeomorphic to

$$(\mathbb{N} \rightarrow (\mathbb{N} \rightarrow \mathbb{N})),$$

many descriptive set-theory results are proved in the context of Baire space alone.

And, a subspace of a Polish space is *analytic* iff it is either empty or a continuous image of the Baire space.

Therefore, our category of types is a convenient arena for introducing notions and proofs from this area.

Any Conclusions?

- Enumeration operators over $\mathcal{P}(\mathbb{N})$ are characterized by a simple **topology**.
- The large category of types over $\mathcal{P}(\mathbb{N})$ **inherits** much topology.
 - λ -calculus over $\mathcal{P}(\mathbb{N})$ plus the arithmetic combinators provides a basic notion of **computability**.
- The category of types over $\mathcal{P}(\mathbb{N})$ thus **inherits** aspects of computability.
 - Polymorphism for types then provides an abstract foundation for defining inductive and co-inductive **data structures**.
 - Propositions-as-types then will enforce using **constructive logic**.

The model can in this way function as a **laboratory** for exploring these ideas in a very concrete fashion.