Quantitative Continuity and Computable Analysis in Coq

Florian Steinberg
INRIA Saclay, France
https://floriansteinberg.github.io
florian.steinberg@inria.fr

Laurent Théry
INRIA Sophia-Antipolis, France
http://www-sop.inria.fr/marelle/thery.html
Laurent.Thery@inria.fr

Holger Thies
Kyushu University, Japan
http://www.holgerthies.com
thies@inf.kyushu-u.ac.jp

Abstract

We give a number of formal proofs of theorems from the field of computable analysis. Many of our results specify executable algorithms that work on infinite inputs by means of operating on finite approximations and are proven correct in the sense of computable analysis. The development is done in the proof assistant Coq and heavily relies on the Incone library for information theoretic continuity. This library is developed by one of the authors and the results of this paper extend the library. While full executability in a formal development of mathematical statements about real numbers and the like is not a feature that is unique to the Incone library, its original contribution is to adhere to the conventions of computable analysis to provide a general purpose interface for algorithmic reasoning on continuous structures. The paper includes a brief description of the most important concepts of Incone and its sub libraries mf and Metric.

The results that provide complete computational content include that the algebraic operations and the efficient limit operator on the reals are computable, that the countably infinite product of a space with itself is isomorphic to a space of functions, compatibility of the enumeration representation of subsets of natural numbers with the abstract definition of the space of open subsets of the natural numbers, and that continuous realizability implies sequential continuity. We also describe many non-computational results that support the correctness of definitions from the library. These include that the information theoretic notion of continuity used in the library is equivalent to the metric notion of continuity on Baire space, a complete comparison of the different concepts of continuity that arise from metric and represented space structures and the discontinuity of the unrestricted limit operator on the real numbers and the task of selecting an element of a closed subset of the natural numbers.

2012 ACM Subject Classification Mathematics of computing → Continuous functions; Theory of computation → Models of computation; Software and its engineering → Formal methods

Keywords and phrases computable analysis, Coq, contionuous functionals, discontinuity, closed choice on the naturals

Digital Object Identifier 10.4230/LIPIcs.ITP.2019.28

Related Version A full version is available on HAL: https://hal.inria.fr/hal-02088293.

Supplement Material The project page of this paper: https://holgerthies.github.io/continuity

Funding *Florian Steinberg*: Supported by the ANR project FastRelax (ANR-14-CE25-0018-01) of the French National Agency for Research and by EU-MSCA-RISE project 731143 "Computing with Infinite Data" (CID).

Laurent Théry: Supported by the ANR project FastRelax (ANR-14-CE25-0018-01) of the French National Agency for Research.
Holger Thies: Supported by JSPS KAKENHI Grant Number JP18J10407 and by the Japan Society for the Promotion of Science (JSPS), Core-to-Core Program (A. Advanced Research Networks).

Acknowledgements The first and last authors would like to thank Hugo Férée, Akitoshi Kawamura and Matthias Schröder for discussion on the topics of this paper.

1 Introduction

Computable analysis is the theory of computing on continuous structures. Its roots are often cited as going back to Turing's fundamental paper from 1936 in which he introduced his mathematical model of computation later known as Turing machine [62]. Turing's original definitions rely on the binary representation and he adapted them to the ones still used today in his 1937 correction [63] with a pointer to earlier work by Brouwer. The theory of computable functions on the real numbers was further developed in the 1950s by Grzegorczyk and Lacombe in parallel [21, 33]. Later on, Kreitz and Weihrauch extended the theory to apply to more general spaces and introduced the formal framework of representations that is standard today [32, 65, 34]. The basic idea behind computable analysis is fairly simple: To make uncountable structures available to computation, one encodes them by infinitary objects that can still be operated on mechanically. Most commonly infinite strings are used, but more conveniently one may use functions between discrete structures. An example for a reasonable encoding of a real number is a function that provides arbitrarily accurate approximations. To compute functions on the real numbers, one operates on such codes by means of allowing calls to their values. Since the inputs and outputs of such functions can be chosen rational and thus be described by finite means, this leads to a computational model that can properly handle infinite inputs while remaining realistic in the sense of being implementable.

The model used in computable analysis is by far not the only popular model for computing with functional inputs. Alternative approaches use similar access models but assume all inputs to be computable, or deal with functional input by encoding via a Gödel numbering. Many of these models are special cases from the perspective of computable analysis [2, 34]. The former of the two mentioned above should for instance be understood to impose a weaker notion of correctness of algorithms as they are only required to behave appropriately on a countable subset of all possible inputs, namely the computable ones. Yet another take on computation on the real numbers are algebraic approaches like the BSS model [8]. These models have the disadvantage of not providing directly implementable algorithms and the advantage that they closely resemble how numerical analysts proceed in practice: the mathematical proof of correctness of the algorithm underlying an implementation often uses mathematical methods that assume the capability to carry out exact operations on real numbers. For the actual implementation, real variables are substituted with machine numbers so that highly optimized and hardware-supported floating-point operations can be used for fast computations. As machine numbers fail to satisfy basic mathematical properties like associativity, the mathematical proof of correctness of the algorithm does not need to have any direct implications for validity of the values the implementation returns even if everything is done correctly. This problem is well aware to algorithm designers and in applications that demand high reliability, correctness may be recovered in an additional step by estimation of rounding errors. These often lead to laborious computations that are error-prone themselves and quickly become infeasible to do by hand.

Recent advances in formal proofs provide a toolset that can be used to make the loop back from numerical practice to the theory of computation [11, 9, 5, 10]. A popular tool in these works is the classical formalization of real numbers in Coq's standard library. This is because working conservative over this axiomatization in Coq provides capabilities fairly similar to working in the BSS model. The computable analysis community has shown an increase of interest in these developments [41]. Algorithms from computable analysis are notoriously difficult to implement in a way that makes them competitive in terms of speed and memory consumption [7, 30] and thus applications often highlight reliability which naturally goes well with verification. Furthermore, popular methods to overcome the efficiency problems use a toolset similar to that used by the verified numerics community [40].

As a step of bringing formal methods and computable analysis closer together, this paper formulates some more theoretical algorithms in the proof assistant Coq. The produced code is fully executable and proven correct in the sense of computable analysis. Where the real numbers turn up, the axiomatization from Coq's standard library is used. We do not make attempts to make these algorithms competitive in terms of speed or memory usage. For example, we currently use rational numbers for approximating reals and no kind of efficiency can be expected before these are not at least replaced by arbitrary precision floating-point numbers. However, it should be kept in mind that this is possible in principle and we believe our framework to have realistic applications. Indeed, for the formalization we use the Incone library whose long term goal is to provide an environment in which the intersection of formal proofs, computable and numerical analysis can conveniently be investigated in Coq and their merits can be combined in attempts to prove efficient algorithms with practical relevance correct.

1.1 Proofs about continuous structures in Coq and related research

Few if any of our results are mathematically original, but most are known facts from computable analysis. Parts of our development of real numbers has previously been covered by fully constructive developments such as the C-CoRn library. Some of these results are also covered by a smaller project that implemented Cauchy reals to use them and the Mathematical Components library to give a definition of the algebraic real numbers in Coq [13]. To the best of our knowledge most of the rest of our results falls outside of the scope of any other formal development in Coq or in other proof assistants for that matter. We consider these formalizations original to this paper.

As our development heavily relies on the Incone library, we make some effort to describe its central concepts and how they were formalized. We tried to keep the presentation of the background theory from computable analysis close to the formal development in the Incone library. The standard references for computable analysis are [48, 65, 29]. The main topics are also presented in a way somewhat closer to how this paper proceeds in [52, 46, 3]. Due to the page restriction, we had to cut some corners in the presentation of the internals of the Incone library and point to the full version of this article for a more exhaustive treatment [60]. By relying on the toolset that the library provides, most of our proofs went quite smoothly and stayed close to the informal proofs from computable analysis. Where the proofs turned out to be more complicated, this paper includes informal descriptions of the formal proofs and the difficulties encountered. For a more thorough description of the interesting parts of the proofs and some details of the simpler proofs we also point the interested reader to the full version.

The C-CoRn library for constructive analysis is by far the most advanced fully computational Coq development that deals with real numbers [14]. It provides a wide range of results about functions on real numbers and some about operators on function spaces and includes an exhaustive treatment of metric spaces and uniformly continuous functions between metric spaces [44]. While the mathematical contents that are the topics of the C-CoRn library, this paper and the Incone library are similar, the approach and scope are quite different. The C-CoRn library is inspired by, and roughly follows the development of constructive analysis by Bishop and Bridges [6]. Executability is achieved by restricting to constructive proofs. This constructive focus makes the C-CoRn library and the publications related to it difficult to access for some classically trained mathematicians. The Incone library follows the tradition of computable analysis where computational content is extra information that should follow a mathematical understanding of the structures under consideration. For the formulation of a clean mathematical theory, classical reasoning and well justified axioms may be used where they simplify the proofs and clean up the statement of theorems. It should thus be understood as a complementary approach.

The use of axioms always comes with the danger of introducing inconsistencies. We attempted to minimize their use in many places and only use axioms from Coq's standard library which are commonly used in the Coq community. The parts that involve real numbers rely on the axiomatization of the real numbers as an archimedian ordered field from Coq's standard library. Other axioms that we use fairly often include classical reasoning, functional extensionality and weak choice principles like countable choice or choice principles on countable types, some parts also use proof irrelevance. Throughout the paper we make some effort to discuss where we believe the use of axioms to be essential and why. How much work we put into minimizing the use of axiom depends on the use cases of the results. For instance, there exists a line of lemmas that mostly act as sanity checks for the library and are best understood when interpreted in the sense of category theory (universal properties of products etc.). The parts of these that do not feature computational content were given a lower priority in optimizations for axiom use.

1.2 Realizability, computable analysis and computing on infinite data

In computable analysis the elements of an abstract set X to compute over are encoded over Baire space by use of a partial surjective function from Baire space to X that is called a representation. An element of Baire space that is mapped to $x \in X$ by the representation is considered to provide on demand information about x. The description of real numbers via functions that take rational accuracy requirements and return rational approximations is an example for such a representation. A set with a designated representation is called a represented space and there exist natural notions of what it means for a function between represented spaces to be continuous and to be computable. Both of these notions, and in particular where they diverge, are central points to computable analysis. An informal rule of thumb is that any function that is continuous and whose definition is sufficiently 'natural' is also computable.

The Incone library follows these ideas to provide a formal definition of represented spaces in Coq. However, as implicitly done in the example of real numbers, it adds an additional layer of abstraction where the inputs and outputs of a description need not always be explicitly encoded as natural numbers but are allowed to use any countable and inhabited types. The Incone library includes a definition of continuity of functions between represented spaces and, if Coq's types are interpreted as sets and a classical setting is assumed, the continuity part of computable analysis is captured. If one wants to reason about computability as refinement of continuity, more care has to be taken. For instance, to avoid difficulties with the input and output types, one should guarantee that these types are either finite or there is an effective bijection with the natural numbers. This may be forced by requiring the construction of Mathematical Components `countType` structure for the input and output types [38]. In presence of this additional information, the Incone library provides tools to capture the notion of computability used in computable analysis in Coq. It provides a way to specify functions on Baire space such that the functions computable in the sense of computable analysis are exactly those that can be instantiated with pure Coq terms, i.e., Coq terms that do not involve any axioms. This construction is compatible with Coq's code-extraction capabilities.

However, the Incone library does not give a formal definition of computability of functions between represented spaces or even Baire space, but only reasons about it on the meta-level. This is due to a reflection problem where checking a term for use of axioms can not be done internally. The additional value of such a definition would be the possibility to give computability-theoretic proofs of incomputability. Such proofs are fairly rare in computable analysis due to the rule of thumb mentioned above: any sufficiently natural function that is incomputable should already be discontinuous. This is in particular true for all instances where we have proven incomputability so far. Once computability theoretic proofs of incomputability move to the center of our attention, a formal definition of computability may be added either using a self-reflection library or more directly by relying on a full formalization of a model of computation [20, 66].

1.3 Structure of the paper and pointers to the main results

All theorems, propositions and lemmas in this paper have been formalized in Coq and were made part of the Incone library. They come with explicit pointers to their name in the library. The statements of the results in the library and in the paper are fairly close. The only notable exception is what was discussed at the end of the last section: whenever the paper claims computability, the formal version proves continuity by explicitly specifying a term that witnesses the continuity and this term is axiom-free as can be checked by the user. Many of the claims that are stated in the plain text, as corollaries or as examples are also supported by formal proofs and occasionally library names are put in brackets after the statement. The identifiers of the exact versions of the mf, Rlzrs, Metric and Incone libraries that this paper refers to can be found in the references [58, 57, 59, 56] or downloaded from the project homepage.

The results whose formalization we consider the main contributions are that the algebraic operations and the efficient limit operator on the reals are computable (Examples 3 and 5), that the countably infinite product is isomorphic to a space of functions (Theorem 7), compatibility of the enumeration representation of subsets of natural numbers with the abstract definition of the space of open subsets of the natural numbers (Theorem 16), and that continuous realizability implies sequential continuity. The previous results are fully algorithmic, but we also describe many non-computational theorems. These include that Incone's information theoretic notion of continuity is equivalent to the metric notion on Baire space (Theorem 14), a complete comparison of the different concepts of continuity that arise from metric and represented space structures (Corollary 9 and Lemma 10) and the discontinuity of the unrestricted limit operator on the real numbers (Example 5) and the task of selecting an element of a closed subset of the natural numbers (Corollary 18).

Coq uses a type-theoretic setting, while the mathematics that we formalize is more commonly formulated over a set theoretic background theory. As is very common in these situations, the paper uses a mix of set-theoretic and type-theoretic notations. In particular we identify subsets of a given type T with functions of type $T \to$ `Prop` and borrow the elementhood notation from set theory, i.e., we write $t \in T$ for $T(t)$. We also use the mathematical notation for subsets, subset inclusion and partial functions. Finally, we avoid the use of the colon for typing when referring to elementhood of certain function spaces. This is because of the confusing ambiguity in interpretation of function types. For our purposes it is more natural to consider elements of some function spaces as mathematical functions and not elements of a function type. This is because we do not want to restrict to the computable functions only and also expressed through regular use of the functional extensionality axiom and of choice principles to construct elements.

2 Multifunctions and partial operators on Baire space

In computable analysis, the computability and topological structure of Baire space are carried over to more general spaces by means of encodings that are called representations. Before we go into detail about how this is done, this section describes the structure on Baire space that we need. Classically, Baire space is the space of all total functions from natural numbers to natural numbers, i.e., functions of type $\mathbb{N} \to \mathbb{N}$. We more generally refer to any space of the form $\mathbf{Q} \to \mathbf{A}$ as Baire space if $\mathbf{Q}$ and $\mathbf{A}$ are countable and inhabited types. Classically these assumption imply that the types are either finite or bijectively related to the natural numbers. Of course, constructively this is far from true. Indeed, if computability considerations come in, one has to be more careful as the bijections with the natural numbers need not be computable. In the applications considered in this paper, however, the substitution by natural numbers are extremely simple, obviously computable and can even be carried out by hand. The critical reader may therefore replace any occurrence of $\mathbf{Q}, \mathbf{A}$ and their dashed variants in the following by $\mathbb{N}$ and assume that the difference in naming is merely for easy distinction of different in- and outputs and readability.

Computable analysis heavily relies on the theory of continuous partial operators on Baire space. In Coq, functions are always total and to find an appropriate notion of partiality, which is important for a proper treatment of continuity, we first need to discuss how functions can be specified through relations [1, 47, 45, 15]. A multivalued function $F\colon S \rightrightarrows T$ (notation `_ ->> _` in the library) is a function that assigns to each $s\colon S$ a possibly empty subset $F(s)$ of T. While this gives F the type $S \to T \to$ `Prop` and one could identify F with a binary relation, the intuition behind a multivalued function is different as S is treated as input type and T as output type. The **domain** of a multifunction F is given by $\mathrm{dom}(F) := \{s\colon S \mid \exists t\colon T, t \in F(s)\}$ and for $s \in \mathrm{dom}(F)$ the set $F(s)$ should be interpreted as the set of eligible return values. A multivalued function is called **total** if its domain is all of S, and **single-valued** if each of the sets $F(s)$ has at most one element.

A multivalued function can be considered a specification for functions: A function $f\colon S \to T$ fulfills the specification $F\colon S \rightrightarrows T$ if $s \in \mathrm{dom}(F) \implies f(s) \in F(s)$ for all $s\colon S$. In this case we say that f **is a choice for** F (`icf` in the library with notation `_ \is_choice_for _`). The operations on multivalued functions are chosen such that they behave well with respect to the interpretation as specifications. For instance, the **composition** $F \circ G$ of two multivalued functions $G\colon R \rightrightarrows S$ and $F\colon S \rightrightarrows T$ is given by

$$F \circ G(r) := \{t\colon T \mid G(r) \subseteq \mathrm{dom}(F) \wedge \exists s, t \in F(s) \wedge s \in G(r)\}.$$

(notation `_ \o _` in the library). This is an associative operation and the second half, namely $F \circ_R G(r) := \{t\colon T \mid \exists s, t \in F(s) \wedge s \in G(r)\}$, is what is commonly used as composition for relations. The domain condition is a modifier that addresses the difference in interpretations and in particular leads to a loss of the symmetry under exchange of the input and output types. In particular for the multifunction composition it is true that if f is a choice for F and g is a choice for G then $f \circ g$ is a choice for $F \circ G$, which may fail for the relational composition (compare Figure 1a).

There is a very straightforward way to generate multifunctions from functions or partial functions. Namely, for a function $f\colon S \to T$ just use the specification `F2MF`$f\colon S \rightrightarrows T$ that uniquely determines it, i.e., `F2MF`$f(s) := \{t\colon T \mid t = f(s)\}$. Clearly, this multifunction is always total and single-valued and assuming that T is not empty each total single-valued multifunction arises in this way (`fun_spec`). This construction can be extended to partial functions by assigning to $g\colon S \to \operatorname{opt} T$ the function `PF2MF`$g(s) := \{t\colon T \mid g(s) = \operatorname{Some} t\}$, which is still single-valued but need not be total anymore. We are mostly interested in operators on Baire spaces, whose domains are rarely decidable. Coding a partial function as a function to an option type may be understood to indicate that the domain of the function should be decidable and we thus avoid it. Instead, we choose the mathematical notation $g\colon \subseteq S \to T$ for partial functions and in the Incone library they are usually treated as single-valued multifunctions right away. The assignments `F2MF` and `PF2MF` are compatible with the multifunction composition and many other operations.

Note that in contrast to functions, any multifunction can be assigned a reverse multifunction where the input and output is simply switched. All properties of a multifunction have a co-version that requires the same property for the reverse multifunction. Many of the co-properties have nice characterizations for the special case of functions. For instance, a function f is injective if and only if `F2MF`f is co-single-valued and a partial function f is surjective if and only if `PF2MF`f is co-total.

An important concept for our purposes is the notion of a tightening (`tight` in the library with notation `_ \tightens _`). For multifunctions $F, G\colon S \rightrightarrows T$ we say that F **tightens** G if it is more restrictive as a specification. That is, if

$$\operatorname{dom}(G) \subseteq \operatorname{dom}(F) \quad \text{and} \quad \forall s \in \operatorname{dom}(G), F(s) \subseteq G(s).$$

Indeed, under appropriate assumptions F tightens G if and only if being a choice for F implies being a choice for G (`icf_tight` and `tight_icf`, also compare Figure 1b). A function f is a choice for a multifunction F if and only if `F2MF`f tightens F (`icf_spec`) and if `PF2MF`f tightens F we say that f is a **partial choice** for F. An exhaustive overview over the concepts and notations for multifunctions the mf library provides can be found in the preamble of the `mf.v` file [58].

For the purposes of this paper another construction is important. A multifunction Φ_N of type $S \rightrightarrows T$ can be obtained from a function N of type $\mathbb{N} \times S \to \operatorname{opt} T$ via

$$\Phi_N(s) := \{t\colon T \mid \exists n, N(n, s) = \operatorname{Some} t\}.$$

In the special case where $S = \mathbb{N} = T$ the specification of any partial computable function can be expressed using a primitive recursive function N and this is particularly interesting to us as any primitive recursive function has a definition in Coq that is closed under the global context [43]. The core idea behind why this is true is a version of the Kleene normal-form theorem [55], although there are some technical differences. For a fixed partial computable function, a primitive recursive function N that works can be obtained from any Turing machine computing the function as follows: on input (n, s) return $\operatorname{Some} t$ if the machine on input s terminates within the first n time-steps and returns t and None otherwise. Under the reasonable assumption that any Coq-function is computable we obtain a characterization of the partial computable functions. Thus, the above correspondence can be used to talk about computable functions in Coq at least on a meta-level. A priori, the multifunction Φ_N need neither be total nor single-valued but a single-valued tightening $\Phi_{N'}$ of Φ_N can be obtained.

2.1 Continuity of partial operators between Baire spaces

Fix some types $\mathbf{Q}$, $\mathbf{A}$, $\mathbf{Q}'$ and $\mathbf{A}'$ and set $\mathcal{B} := \mathbf{Q} \to \mathbf{A}$ and $\mathcal{B}' := \mathbf{Q}' \to \mathbf{A}'$. An important class of objects of investigation in computable analysis are computable, or at least continuous partial operators on Baire space, or in our generalized setting of type $F \colon \subseteq \mathcal{B} \to \mathcal{B}'$. One way to produce specifications of such operators is to relativize the Φ assignment from the previous section and assign to a function $M : \mathbb{N} \times \mathcal{B} \times \mathbf{Q}' \to \operatorname{opt} \mathbf{A}'$ the specification $F_M \colon \mathcal{B} \rightrightarrows \mathcal{B}'$ such that

$$\psi \in F_M(\varphi) \qquad \iff \qquad \forall q' \colon \mathbf{Q}', \exists n \colon \mathbb{N}, M(n, \varphi, q') = \operatorname{Some} \psi(q').$$

(`operator` in the library with notation `\F_( _ )`, compare Example 15). The relativization adds complexity as it can for instance be seen on the example of composition: on the one hand it is easy to realize composition for the Φ assignment, finding a tightening of $F_M \circ F_{M'}$ from M and M' alone, on the other hand,this is problematic : M' allows to produce arbitrary good approximations to a functional input for M, but no information is known about how good this approximation must be for M to return a correct value. Indeed, these approximations are sufficient to obtain correct values of the composition only if F_M is continuous.

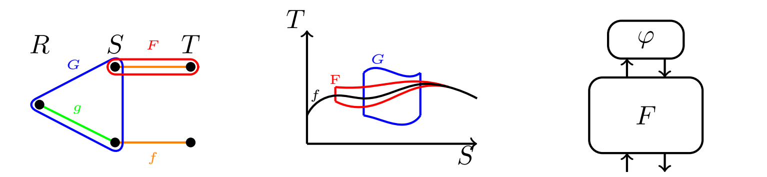
Continuity of F_M can be made sense of by equipping $\mathcal{B}$ and $\mathcal{B}'$ with the topologies of pointwise convergence, or equivalently by using an appropriate metric on these spaces. For our purposes a slightly different, information based description of the same concept is more adequate. Intuitively continuity means that the return-values of an operator $F \colon \mathcal{B} \to \mathcal{B}'$ interpreted as functional of type $F \colon \mathcal{B} \times \mathbf{Q}' \to \mathbf{A}'$ do only depend on finite information about the values of the functional input from $\mathcal{B}$ and thus can be thought of as being represented by a diagram as depicted in Figure 1c. Mathematically, a function $F \colon \mathcal{B} \to \mathcal{B}'$ is **continuous** if for any element φ of $\mathcal{B}$ and any $q' \colon \mathbf{Q}'$ there exists a **certificate**, i.e., a finite list $L \colon \operatorname{seq} \mathbf{Q}$ such that for any ψ that coincides with φ on L it holds that $F(\psi)(q') = F(\varphi)(q')$. Here, two functions are said to **coincide on** a finite list L if $\varphi(q) = \psi(q)$ for any q contained in L. A partial operator $F \colon \subseteq \mathcal{B} \to \mathcal{B}'$ is continuous if for all $\varphi \in \operatorname{dom}(F)$ and $q' \colon \mathbf{Q}'$ there exists a certificate, i.e., a finite list $L \subseteq \mathbf{Q}$ such that the above statement holds for any $\psi \in \operatorname{dom}(F)$.

The definition of continuity in the INCONE library follows the mathematical definition given above mostly literally. The only notable difference is that instead of a separate list for each $q' \colon \mathbf{Q}'$ a Skolem-function $\mu \colon \mathbf{Q}' \to \operatorname{seq} \mathbf{Q}$ is used. This is equivalent to the above definition whenever an appropriate choice principle is available (`choice_cont`) and avoids assuming any axioms in the proof that the composition of continuous operators is continuous. From a meta-level many of the proofs of continuity that can be found in the INCONE library proceed by specifying an axiom-free COQ-function interpreted either through the `F2MF` or through the $F_\cdot$ assignment and may thus be understood as proofs of computability. All claims of computability in the rest of the paper should be understood in this sense.

Partiality is treated by using multifunctions and the statement of continuity of a multifunction is chosen in such a way that continuity implies the function to be single-valued (`cont_sing`). This definition works well with the composition of multivalued functions:

▶ **Theorem 1** (`cont_comp`). *Let $F \colon \subseteq \mathcal{B} \to \mathcal{B}'$ and $G \colon \subseteq \mathcal{B}' \to \mathcal{B}''$ be continuous partial operators. The operator $G \circ F \colon \subseteq \mathcal{B} \to \mathcal{B}''$ is continuous.*

The idea behind the proof is that the certificate functions μ and ν whose existence is guaranteed by the continuity of F and G can be interpreted as multivalued functions and composed relationally to obtain a certificate function for the composition of the operators. The necessary relational composition can be realized constructively.



$$\begin{tikzcd} \mathbf{X} \arrow[r, "f"] & \mathbf{Y} \\ \mathcal{B} \arrow[u, "\delta_{\mathbf{X}}"] \arrow[r, "F"'] & \mathcal{B}' \arrow[u, "\delta_{\mathbf{Y}}"'] \end{tikzcd}$$

(a) f chooses through F, g chooses through G. $f \circ g$ does not choose through $F \circ_R G$, but through $F \circ G$ which is empty.

(b) f chooses through F which tightens G. Thus f also chooses through G.

(c) A continuous operator F can be thought of as determining its return value by asking finitely many questions to the input function φ.

(d) $F\colon\subseteq \mathcal{B} \to \mathcal{B}'$ is a realizer of $f\colon \mathbf{X} \to \mathbf{Y}$.

Figure 1

2.2 Represented spaces and continuous realizability

A **representation** δ of a space X is a partial surjective mapping $\delta\colon\subseteq \mathcal{B} \to X$. If $\delta(\varphi) = x$ then φ is called a δ-name, or just name, of x. A pair $\mathbf{X} = (X, \delta_{\mathbf{X}})$ of a set and a representation of that set is called a **represented space**. The definition of represented spaces in the Incone library replaces the Baire space $\mathbb{N}^{\mathbb{N}}$ from the definition used in computable analysis with some space $\mathcal{B} = \mathbf{Q} \to \mathbf{A}$, where $\mathbf{Q}$ and $\mathbf{A}$ should be countable inhabited types, i.e., with a Baire space according to the conventions we fixed. Thus, a represented space $\mathbf{X}$ is defined as a record containing a type X (with a coercion from $\mathbf{X}$ to X) together with types $\mathbf{Q}_{\mathbf{X}}$ and $\mathbf{A}_{\mathbf{X}}$ and proofs that these are countable and inhabited and additionally a multivalued function $\delta_{\mathbf{X}}\colon (\mathbf{Q}_{\mathbf{X}} \to \mathbf{A}_{\mathbf{X}}) \rightrightarrows X$ and proofs that it is single-valued and co-total, where the last requirement is equivalent to being surjective for partial functions. We use the notation $\mathcal{B}_{\mathbf{X}} := \mathbf{Q}_{\mathbf{X}} \to \mathbf{A}_{\mathbf{X}}$.

As an example let us equip the real numbers with the representation that is used for motivation and as a point of reference throughout this section.

▶ **Example 2** (`examples/Q_reals.v`). Choose $\mathbf{Q}_{\mathbb{R}}, \mathbf{A}_{\mathbb{R}} := \mathbb{Q}$, i.e., $\mathcal{B}_{\mathbb{R}} = \mathbb{Q} \to \mathbb{Q}$. It is straight forward to prove that the rational numbers provided by Coq's standard library are countable and inhabited. The multifunction $\delta_{\mathbb{R}}\colon \mathcal{B}_{\mathbb{R}} \rightrightarrows \mathbb{R}$ (`rep_RQ` in Incone) specified by:

$$\delta_{\mathbb{R}}(\varphi) = x \quad\iff\quad \forall \varepsilon \in \mathbb{Q}, 0 < \varepsilon \implies |x - \varphi(\varepsilon)| < \varepsilon$$

is a representation. Indeed, using the axiomatization of the real numbers provided by Coq's standard library $\delta_{\mathbb{R}}$ can be proven single-valued and surjective and we refer to the represented space $(\mathbb{R}, \delta_{\mathbb{R}})$ (`RQ` in the library) simply by $\mathbb{R}$.

The topological and computability structure of Baire space can be pushed forward through a representation: A partial operator on Baire space is a **realizer** of a function $f\colon \mathbf{X} \to \mathbf{Y}$ between represented spaces if it assigns to each name of x a name of $f(x)$ (compare Figure 1d). A function between represented spaces is **continuous** if it has a continuous realizer and **computable** if it has a computable realizer. Represented spaces form a Cartesian closed category both with the continuous and the computable functions as morphisms.

With little effort, the definition of being a realizer can be made sense of if both operators on Baire space and functions between represented spaces are multivalued. For the full definitions we point the interested reader to [34] or the Rlzrs library. While we are mostly interested in continuous, and therefore single-valued, realizers the case where f is multivalued is of interest to us as it is needed for the concrete example of closed choice on the natural numbers that we discuss in Section 3.3. We call a multifunction between represented spaces **continuously realizable** if there exists a continuous realizer that maps any name of an input to a name of some eligible return-value. Multivaluedness can also be used to recover continuity: the sign function on the real numbers is discontinuous, but can be approximated by the family of continuously realizable ε-sign multifunctions whose set of eligible return values is increased to $\{-1, 0, 1\}$ whenever $|x|$ is smaller than ε. Another popular and similar example is the use of an ε-equality test to account for the undecidability of equality on the real numbers. That continuity and continuous realizability are preserved under composition follows from content of the Rlzrs library together with the fact that continuity of operators on Baire space is preserved under composition from Theorem 1.

Any Baire space can be made a represented space by using the identity function as a representation. While a partial operation between Baire spaces is continuous if and only if it is continuously realizable with respect to these representations, there are many multivalued functions between Baire spaces that are continuously realizable but not continuous. This is because continuity implies single-valuedness and continuous realizability, to the contrary, is stable under increasing the set of eligible return values. On Baire spaces continuous realizability of a multifunction is equivalent to the existence of a continuous choice function. However, this is specific to Baire spaces and fails for more general represented spaces as can for instance be seen at the example of an ε-sign function or an ε-equality test as given above.

2.3 Basic constructions and examples for represented spaces

Now that we can talk about continuity and computability on the real numbers, a reasonable next step is to attempt to prove addition and multiplication computable.

▶ **Example 3** (`examples/Q_reals.v`). The arithmetic operations on the real numbers are of type $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and to make sense of continuity of functions of this type we need to specify a represented space structure on $\mathbb{R} \times \mathbb{R}$. The Incone library automatically generates a represented space $\mathbf{X} \times \mathbf{Y}$ from arbitrary represented spaces $\mathbf{X}$ and $\mathbf{Y}$ and proves correctness of this construction and continuity of the basic functions. Relying on this one can prove that addition and multiplication of real numbers is continuous (`Rplus_cont` and `Rmult_cont`). The realizers are defined directly using the `F2MF` assignment and are computable (in the sense described in Section 2.1).

Let I be a countable inhabited type and let $\mathbf{X}$ be a represented space. Consider the set of families $(x_i)_{i \in I}$ indexed over I. A reasonable description of such a family would be a function that takes an additional argument from I and if this argument is fixed to i results in a name for x_i. Formally this can be captured by defining a represented space $\prod_I \mathbf{X}$, where the underlying set are the functions of type $I \to X$, the questions given by $\mathbf{Q}_{\prod_I \mathbf{X}} := I \times \mathbf{Q}_{\mathbf{X}}$, the answers by $\mathbf{A}_{\prod_I \mathbf{X}} := \mathbf{A}_{\mathbf{X}}$ and using the representation

$$(x_i) \in \delta_{\prod_I \mathbf{X}}(\varphi) \qquad \iff \qquad \forall i \colon I, x_i \in \delta_{\mathbf{X}}(q \mapsto \varphi(i, q)),$$

where (x_i) is short for the function $i \mapsto x_i$.

For the understanding of the following proposition recall that the universal property that is required from an infinite product $\prod_I \mathbf{X}_i$ is that for each represented space $\mathbf{Y}$ and family (f_i) of continuous functions $f_i \colon \mathbf{Y} \to \mathbf{X}_i$ there exists a unique continuous function $F \colon \mathbf{Y} \to \prod_I \mathbf{X}_i$ such that for all $i \in I$ and $y \in \mathbf{Y}$ it holds that $F(y)_i = f_i(y)$. The following proposition says that in the special case where all the spaces $\mathbf{X}_i$ coincide and $I = \mathbb{N}$, the space constructed above has this universal property.

▶ **Proposition 4** (`rep_Iprod_sing`, `rep_Iprod_sur` and `cprd_uprp_cont`). $\prod_I \mathbf{X}$ *is a represented space and* $\mathbf{X}^\omega := \prod_\mathbb{N} \mathbf{X}$ *is a countably infinite product in the category of represented spaces and continuous functions.*

The use of the symbol ω instead of $\mathbb{N}$ is to differentiate the space $\mathbf{X}^\omega$ of sequences (notation `_ \^w` in the library) from a function space. The proof of single-valuedness of the infinite product representation assumes functional extensionality and the proof of surjectivity needs a choice principle over the index set I. Since $I = \mathbb{N}$ is by far the most common use-case and I is assumed countable, this will usually boil down to the axiom of countable choice. The proof of the universal property relies on stronger choice principles, classical reasoning and proof irrelevance. Since the category of represented spaces with computable functions fails to have countably infinite products, the universal property should not be provable without axioms. This makes this result more of a sanity result than something that may actually be of use, and minimizing the strength of the axioms used is not our highest priority.

The limit operator is a good example of a multi-function whose natural source space is the space of sequences. Consider the multivalued function $\lim_\mathbf{X} : \mathbf{X}^\omega \rightrightarrows \mathbf{X}$ where $x \in \lim_\mathbf{X}(x_n)$ if and only if there is a convergent sequence of names $(\varphi_n) \subseteq \mathcal{B}_\mathbf{X}$ and some φ such that φ is a name of x, each φ_n is a name for x_n and the sequence (φ_n) converges to φ in $\mathcal{B}_\mathbf{X}$, i.e., $\lim_{\mathcal{B}_\mathbf{X}}(\varphi_n) = \varphi$ where $\mathcal{B}_\mathbf{X}$ is given the topology of pointwise convergence of functions between discrete spaces. A function $f : \mathbf{X} \to \mathbf{Y}$ between represented spaces is called sequentially continuous if it preserves this notion of a limit, i.e., if $\lim_\mathbf{X} x_n = x$ implies that $\lim_\mathbf{Y} f(x_n) = f(x)$. While the limit operator on Baire space is single-valued, this may not be true for the limit operator on a general represented space, as can be seen at the example of Sierpiński space that is discussed in Section 3.3. In most spaces that are relevant for numerical analysis, the limit operator is single-valued but discontinuous and has an appropriate computable restriction.

▶ **Example 5** (`examples/Q_reals.v`). On the real numbers $\mathbb{R}$ the limit operator $\lim_\mathbb{R}$ captures the usual notion of convergence of sequences of real numbers. Furthermore, $\lim_\mathbb{R}$ is discontinuous (`lim_not_cont`), but its restriction to those sequences (x_n) that are efficiently Cauchy in that $|x_n - x_m| \leq 2^{-n} + 2^{-m}$ is computable (`lim_eff_hcr`).

The Incone library defines and proves correct a continuous universal U (one may think of either Kleene-Kreisel associateship [28, 31, 17] or Weihrauch's η [65]). Let $\mathbf{X}$ and $\mathbf{Y}$ be represented spaces and consider the collection of all continuously realizable functions from $\mathbf{X}$ to $\mathbf{Y}$. In Incone, the continuous universal U is used to construct a representation for this collection of functions by

$$f \in \delta_{\mathbf{YX}}(\psi) \quad \iff \quad F_{U(\psi)} \text{ realizes } f.$$

Since functions (as opposed to partial or multifunctions) are uniquely determined by each of their realizers, $\delta_{\mathbf{YX}}$ is single-valued. That $\delta_{\mathbf{YX}}$ is co-total is the distinguishing property of continuous universals like U. Thus, $\delta_{\mathbf{YX}}$ is a representation and we refer to the represented space of continuously realizable functions from $\mathbf{X}$ to $\mathbf{Y}$ with this representation as $\mathbf{Y}^\mathbf{X}$ (notation `_ c-> _` in Incone).

Recall that spaces are called isomorphic, in symbols $\mathbf{X} \simeq \mathbf{Y}$, if they are connected by a continuous bijection with continuous inverse and computably isomorphic if there exists a computable bijection and with computable inverse. The natural numbers come with a natural representation and the space $\mathbf{X}^\mathbb{N}$ of functions with respect to this structure is isomorphic to the space $\mathbf{X}^\omega$ of sequences that was constructed as an infinite product at the beginning of this section. I.e. $\mathbf{X}^\mathbb{N} \simeq \mathbf{X}^\omega$. This means that there is an overlap in scope between the function space construction and the infinite product. To understand this in more detail, let I be any countable and inhabited type. Set $\mathbf{Q_I} := \{\star\}$ and $\mathbf{A_I} := I$. Then $\delta_\mathbf{I}(\varphi) := \varphi(\star)$ makes $\mathbf{I} := (I, \delta_\mathbf{I})$ a represented space that is discrete in the following sense:

▶ **Lemma 6** (`cs_id_dscrt`). *For any countable, inhabited type I the represented space $\mathbf{I}$ from above is discrete in the sense that any function that has $\mathbf{I}$ as its domain is continuous.*

The set underlying the space $\prod_I \mathbf{X}$ is the set of functions from I to $\mathbf{X}$. Since $\mathbf{I}$ is discrete all functions from $\mathbf{I}$ to $\mathbf{X}$ are continuous and the sets underlying $\prod_I \mathbf{X}$ and $\mathbf{X}^\mathbf{I}$ are identical. Indeed these spaces are computably isomorphic and we formalized the proof of this.

▶ **Theorem 7** (`sig_iso_fun`). *For any represented space $\mathbf{X}$ and countable inhabited type I the space $\prod_I \mathbf{X}$ is computably isomorphic to $\mathbf{X}^\mathbf{I}$, where $\mathbf{I}$ is the discrete space over I.*

The realizer that translates from sequences to functions is defined using the simpler `F2MF` assignment, but relies on the details of how Incone implements the universal. This may be attributed to the fact that the above theorem need not be true in an arbitrary Cartesian closed category. The construction of a sequence from a continuous function proceeds by using a variation of the realizer of the evaluation operation that is proven computable for arbitrary represented spaces in the Incone library. On the one hand this makes it mostly independent of the implementation of the universal. On the other hand it means that the universal has to be executed and is thus an instance where a realizer uses the more complicated $F.$ assignment. An axiom-free definition of a realizer using the `F2MF` assignment is likely to be impossible. This is related to the fact that the construction of the reals from Dedekind cuts and Cauchy sequences are not fully equivalent in a constructive setting [35].

3 Metric spaces and closed choice on the naturals

A function $d\colon M \times M \to \mathbb{R}$ is called a **pseudo-metric** on a set M if it is positive, symmetric and fulfills $d(x,x) = 0$ and the triangle inequality $d(x,z) \le d(x,y) + d(y,z)$. It is called a **metric** if $d(x,y) = 0$ implies $x = y$. A pair (M,d) is called a **pseudo-metric space** if d is a pseudo-metric on M and a **metric space** if d is a metric. Every pseudo-metric space comes with a topology that is generated by the open balls and therefore with notions of continuity of functions between and limits of sequences in such spaces. The latter is of particular importance since any pseudo-metric space is first-countable and thus knowing the limits is sufficient for characterizing continuity. A more accessible definition of continuity can be given using the well-known ε-δ-criterion that does not require any knowledge about topology. A function $f\colon N \to M$ between pseudo-metric spaces (N, d_N) and (M, d_M) is called **continuous in** x if

$$\forall \varepsilon, \exists \delta, \forall y, d_N(x,y) \le \delta \implies d_M(f(x), f(y)) \le \varepsilon.$$

The function is called **continuous** if it is continuous in any point of M. Here, ε and δ are a priori reals but may be replaced by rationals for density reasons. An element x of a pseudo-metric or metric space (M,d) is said to be the **limit** of a sequence (x_n) in M, in symbols $\lim_{(M,d)}(x_n) = x$, if

$$\forall \varepsilon, \exists N, \forall n, N \le n \implies d(x, x_n) \le \varepsilon.$$

The function f is then called **sequentially continuous** if $\lim_{(N,d_N)}(x_n) = x$ implies $\lim_{(M,d_M)}(f(x_n)) = f(x)$.

Metric spaces have received considerable attention in their formal treatment [36]. There exists a definition of the concept of a metric space and continuity of functions between metric spaces in the standard library of Coq. Several external libraries come with their own versions of metric spaces and continuity. Metric spaces and uniformly continuous functions are some of the core concepts of the C-CoRn library [44]. Another example is the Coquelicot library, which uses a concept it refers to as uniform space that closely resembles pseudo-metric spaces (`cntp_cntp`). The Incone library comes with its own version of metric spaces that is kept close to the classical mathematical treatment and is thus most similar to the metric spaces that can be found in Coq's standard library. It provides interfaces with both the standard library of Coq (`MS2M_S`, `M_S2MS`, `Uncv_lim`, `cont_limin`, etc.) and the Coquelicot library (`US2MS`, `MS2US`, `cntp_cntp`, etc.) so that it is possible to reuse results proven there. In contrast to the Coquelicot library, the metric library does not attempt to be conservative over the background theory of the real numbers.

While the naming of notions for metric spaces is identical to what we used for represented spaces, there are some conceptual differences. First off, a function between metric spaces is continuous if and only if it is sequentially continuous, where for represented spaces the backward implication can fail. A sufficient condition to recover it is admissibility of the involved representations [50]. Secondly, metric continuity can be recovered from a pointwise notion while continuous realizability can not. The pointwise notion introduces subtle problems in the treatment of subspaces. Even in the most well-behaved cases like a closed interval as subspace of the real numbers there is a difference between a function on the reals being continuous in each point of the interval and the restriction of the function to the interval being continuous. The statements of important theorems from the standard library (for instance the mean value theorem) do not account for this difference and diverge slightly from what a mathematician would expect. The metric library assumes proof-irrelevance to allow for a treatment of subspaces as dependent types.

3.1 Comparing continuity in represented and in metric spaces

Metric spaces are well investigated in computable analysis [64]. In particular in the case where (M, d) is a metric space and (r_n) is a designated dense sequence in M, M can be made a represented space $\mathbf{M} := (M, \delta_{\mathbf{M}})$ using the representation defined by

$$x \in \delta_{\mathbf{M}}(\varphi) \quad \iff \quad \forall n, d(x, r_{\varphi(n)}) \leq 2^{-n}.$$

Note that the idea behind this construction is nearly identical to that behind the representations of the reals: A name takes a precision requirement, now encoded as integer, and returns an approximation, or rather an index of an approximation.

A metric space is **separable** if there exists a dense sequence and even though the sequence goes into the definition of the corresponding Cauchy representation, we decide to not mention it explicitly in the following. This is justified in a continuity setting as different choices of dense sequences lead to isomorphic represented spaces. The situation is more complicated if computability is considered and the proofs in the library explicitly carry the sequences along.

▶ **Theorem 8** (`lim_mlim`). *Whenever (M, d) is a separable metric space and $\mathbf{M}$ as above then* $\lim_{(M,d)} = \lim_{\mathbf{M}}$.

The proof that the sequential notions of continuity on metric and represented space coincide follows immediately from this theorem. Each direction of the proof requires to translate limits in both directions and is thus as constructive or non-constructive as the worse direction of the previous theorem (which requires to assume mild choice principles).

▶ **Corollary 9** (`scnt_mscnt`). *If (M, d) and (M', d') are separable metric spaces, then a function $f : M \to N$, is sequentially continuous as a function between metric spaces if and only if it is sequentially continuous as function $f\colon \mathbf{M} \to \mathbf{M}'$.*

For the equivalence of ε-δ-continuity and continuous realizability one direction needs stronger assumptions and for the Incone library we have thus separated the proofs.

▶ **Lemma 10** (`cont_mcont` and `mcont_cont`). *Let (M, d) and (M', d') be two separable metric spaces. A function $f\colon M \to M'$ is ε-δ-continuous if and only if $f\colon \mathbf{M} \to \mathbf{M}'$ is continuous.*

The proof that continuous realizability implies ε-δ-continuity is straight forward, the proof of the other implication has turned out to be more complicated. We sketch some of the details.

Call a function $\mu\colon \mathbb{N} \to \mathbb{N}$ a **modulus of metric continuity** of f in x if

$$\forall y, d(x, y) \le 2^{-\mu(n)} \implies d(f(x), f(y)) \le 2^{-n}$$

And call such a modulus **minimal** if it is minimal in the obvious way.

▶ **Lemma 11** (`exists_minmod_met`). *For any continuous function f between metric spaces and any argument x for f there exists a minimal modulus of f in x.*

The proof relies on classical reasoning. This is a case where the use of axioms can not be avoided: In the special case where the metric space is Baire space the existence of a minimal modulus cannot be proven constructively [61].

If the source space is Baire space, it can be shown that the minimal modulus function is continuous in each x and from this Lemma 10 can be deduced. For general metric spaces, however, this strategy is bound to fail: If the metric space is connected, the function assigning to each x the minimal modulus function of f in x cannot be continuous as it takes values in the totally disconnected space $\mathbb{N}^{\mathbb{N}}$. One might expect that this is due to the awkward typing, and that making μ have type $\mathbb{R} \to \mathbb{R}$ instead would help, but it does not. It is known that also in this case the minimal modulus need not be continuous and that a construction of a continuous modulus of continuity, while possible in general, takes considerably more effort [16]. Our proof that ε-δ-continuity implies continuous realizability therefore proceeds differently and uses a notion of being almost-selfmodulating instead, where the value of the minimal modulus on slightly disturbed input from the metric space is bounded in terms of a shift of the minimal modulus in the original value.

Interestingly, similar tools as those in the above proof turn out to be useful in other parts of the Incone library. More specifically, the proof of correctness of the continuous universal that the library uses for the construction of function spaces also makes use of minimal moduli.

3.2 Recovering continuity on Baire space from a metric structure

Fix some types $\mathbf{Q}$ and $\mathbf{A}$ and set $\mathcal{B} := \mathbf{Q} \to \mathbf{A}$. Recall from the discussion in Section 2.1 that if $\mathcal{B}$ is a Baire space, then there exists a canonical way to make this space a represented space and that the elementary notion of continuity coincides with the represented space notion for partial functions. The limit operator $\lim_{\mathcal{B}}$ that this space gets as a represented space captures pointwise convergence with respect to the discrete topology on $\mathbf{A}$. The information theoretic notion of continuity on $\mathcal{B}$ from Section 2.1 is equivalent to sequential continuity in the associated represented space and a proof of this can be found in the Incone library.

For each function $\mathrm{cnt}\colon \mathbb{N} \to \mathbf{Q}$ define a mapping $d_{\mathrm{cnt}}\colon \mathcal{B} \times \mathcal{B} \to \mathbb{R}$ by

$$d_{\mathrm{cnt}}(\varphi, \psi) := \begin{cases} 2^{-k} & \text{if } \varphi \neq \psi \text{ and } k = \min\{n, \varphi(\mathrm{cnt}(n)) \neq \psi(\mathrm{cnt}(n))\} \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $\mathcal{B}$ is a Baire space, then $\mathbf{Q}$ is countable and there exists some surjective function $\mathrm{cnt}\colon \mathbb{N} \to \mathbf{Q}$. This makes the above mapping a metric.

▶ **Proposition 12** (`dst_pos`, `dst_sym`, `dstxx`, `dst_trngl`, `dst_eq`)**.** *Whenever* $\mathrm{cnt}\colon \mathbb{N} \to \mathbf{Q}$ *is surjective,* $(\mathcal{B}, d_{\mathrm{cnt}})$ *is a metric space.*

The core of the proof is an implementation of a function that approximates an unbounded search and developing some of its properties.

▶ **Theorem 13** (`lim_lim`)**.** *Let* $\mathcal{B}$ *be a Baire space and* cnt *surjective, then* $\lim_{(\mathcal{B},d_{\mathrm{cnt}})} = \lim_{\mathcal{B}}$.

The above theorem directly implies that the concepts of sequential continuity between Baire spaces coincides with the corresponding metric notion. For metric spaces sequential continuity and continuity are equivalent by combination of Corollary 9 and Lemma 10, thus:

▶ **Corollary 14** (`cont_cont`)**.** *Whenever* $\mathcal{B}$ *and* $\mathcal{B}'$ *are Baire spaces and* cnt *and* cnt' *are appropriate surjective functions then* $F\colon\subseteq \mathcal{B} \to \mathcal{B}'$ *is continuous in the sense of Section 2.1 if and only if it is continuous as function from* $(\mathrm{dom}(F), d_{\mathrm{cnt}})$ *to* $(\mathcal{B}', d_{\mathrm{cnt}'})$.

▶ **Example 15** (`examples/continuous_search.v`)**.** An instructive example is the search operator whose domain are those functions from $\mathbb{N}^{\mathbb{N}}$ that eventually return zero and whose value is the first argument on which such an input returns zero. This operator is continuous and does not have a continuous total extension. As the regular notion of continuity on the original Baire space $\mathbb{N}^{\mathbb{N}}$ is captured by the continuity introduced in Section 2.1, this is true for both the metric notion as well as the information-theoretic notion. This operator is not only continuous but computable and this is witnessed by the function that was used in the proof of Proposition 12 to approximate an unbounded search and is a good example for the mechanisms discussed in Section 2.1.

3.3 Sierpiński space and closed choice on the naturals

This section describes the content of the file `examples/closed_choice.v` from the Incone library. Sierpiński space $\mathbb{S}$ (`cs_Sirp` in the library) is the space whose base set is the two point set $\{\bot, \top\}$ equipped with the total representation $\delta_{\mathbb{S}}\colon \subseteq (\mathbb{N} \to \mathbb{B}) \to \mathbb{S}$ specified by

$$\delta_{\mathbb{S}}(\varphi) = \top \iff \exists n \in \mathbb{N}\ \varphi(n) \neq \text{false}.$$

For a subset $U \subseteq \mathbf{X}$ denote by $\chi_U\colon \mathbf{X} \to \mathbb{S}$ its characteristic function. One reason for the importance of Sierpiński space in computable analysis is that a set $U \subseteq \mathbf{X}$ is open if and only if this characteristic function χ_U is continuous as a function from $\mathbf{X}$ to $\mathbb{S}$. Thus we can identify the space $\mathcal{O}(\mathbf{X})$ of open subsets of $\mathbf{X}$ with the function space $\mathbb{S}^{\mathbf{X}}$ [46]. Similarly, the space $\mathcal{A}(\mathbf{X})$ of closed subsets of $\mathbf{X}$ is represented as the complements of opens.

For many concrete spaces $\mathbf{X}$ simpler descriptions of $\mathcal{O}(\mathbf{X})$ and $\mathcal{A}(\mathbf{X})$ are available. If the represented space $\mathbf{X} = \mathbb{N}$ are the natural numbers, for instance, the infinite product construction and in particular of the special case $I = \mathbb{N}$ and $\mathbf{X} = \mathbb{S}$ of the statement $\mathbf{X}^{\mathbf{I}} \simeq \prod_I \mathbf{X}$ of Lemma 7 guarantees that $\mathcal{O}(\mathbb{N}) = \mathbb{S}^{\mathbb{N}} \simeq \prod_{\mathbb{N}} \mathbb{S} = \mathbb{S}^{\omega}$. There exists a fully concrete description that is often used for reasoning about $\mathcal{O}(\mathbb{N})$. Consider the enumeration representation of the open subsets of the natural numbers, where a name of an open set enumerates its elements. We call the corresponding space $\mathcal{O}_{\mathbb{N}}$. The representation of the concrete space $\mathcal{A}_{\mathbb{N}}$ of the closed subsets of the natural numbers is given by $\delta_{\mathcal{A}_{\mathbb{N}}}(\varphi) = \mathbb{N} \setminus \{n\colon \mathbb{N} \mid \exists m\colon \mathbb{N},\ \varphi(m) = n+1\}$. The information a name specifies about a closed set is an enumeration of its complement. The underlying sets of the spaces of opens and closeds of $\mathbb{N}$ are all subsets, but the information about such sets that is made available by names differs.

We provide a formal proof that the enumeration representations of the open and closed subsets of the natural numbers capture the abstract structure these spaces can be given through the exponential in the category of represented spaces and Sierpiński space.

▶ **Theorem 16** (`AN_iso_Anat`, `ON_iso_Onat` and `clsd_iso_open`). *$\mathcal{A}(\mathbb{N}) \simeq \mathcal{A}_{\mathbb{N}}$, $\mathcal{O}(\mathbb{N}) \simeq \mathcal{O}_{\mathbb{N}}$ and $\mathcal{A}(\mathbb{N}) \simeq \mathcal{O}(\mathbb{N})$.*

The last of these isomorphies is trivial: the isomorphism is taking the complement and it is realized by the identity function. The isomorphism of $\mathcal{O}(\mathbb{N})$ and $\mathcal{O}_{\mathbb{N}}$ is proven by first replacing $\mathcal{O}(\mathbb{N})$ by $\mathbb{S}^\omega$ as described above. The realizers for the isomorphism between $\mathbb{S}^\omega$ and $\mathcal{O}_{\mathbb{N}}$ uses the Cantor paring function provided by the Mathematical Components library.

As an application let us consider some choice operators that are popular for classification of computational tasks with respect to their Weihrauch degree. Solving the task of choosing an element of a non-empty closed subset of a represented space $\mathbf{X}$ can be formalized as asking for a realizer for the multivalued function $C_{\mathbf{X}}$ defined by

$$C_{\mathbf{X}}\colon \mathcal{A}(\mathbf{X}) \rightrightarrows \mathbf{X}, \quad a \in C_{\mathbf{X}}(A) \iff a \in A.$$

Or in words: a is an acceptable return value of $C_{\mathbf{X}}$ on input A if and only if a is an element of A. The domain of $C_{\mathbf{X}}$ are the non-empty subsets of $\mathbf{X}$ and this means that a realizer can behave arbitrarily on names of the empty set and may even diverge. As the input A is given as a closed set where a name specifies negative information about element-hood, this task does not have a continuous, let alone computable, solution for most spaces $\mathbf{X}$.

Consider the case $\mathbf{X} = \mathbb{N}$. The domain of the multivalued function $C_{\mathbb{N}}$ is $\mathcal{A}(\mathbb{N})$ but the same definition also specifies a multifunction $C'_{\mathbb{N}}\colon \mathcal{A}_{\mathbb{N}} \rightrightarrows \mathbb{N}$ whose source space $\mathcal{A}_{\mathbb{N}}$ uses the enumeration representation. A mathematician may even consider it pointless to give this function a different name as isomorphic spaces are regularly identified. For the question whether $C_{\mathbb{N}}$ has a continuous realizer $\mathcal{A}(\mathbb{N})$ may be substituted with $\mathcal{A}_{\mathbb{N}}$ (`CN_CN'_hcr`).

▶ **Proposition 17** (`CN'_not_cont`). *$C'_{\mathbb{N}}$ does not have a continuous realizer.*

Our formal proof follows the standard proof by contradiction literally: Assume that to the contrary that F was a continuous realizer of $C'_{\mathbb{N}}$. Pick any name φ of the one point set $\{0\}$. As F is a realizer, it has to return a name of 0 on input φ, i.e., $F(\varphi)(\star) = 0$. Since F is continuous there is a list $L \subseteq \mathbb{N}$ such that $F(\varphi)(\star) = F(\psi)(\star)$ for all $\psi\colon \mathbb{N} \to \mathbb{N}$ that coincide with φ on L. Consider the name φ' of the non-empty set $A := \mathbb{N} \setminus (\{n \mid \exists m \in L, \varphi(m) = n+1\} \cup \{0\})$ defined by $\varphi'(n) := \varphi(n)$ if $n \in L$ and 1 otherwise. On the one hand, $F(\varphi')(\star) \in A$ since F is a realizer. On the other hand $F(\varphi')(\star) = F(\varphi)(\star) = 0$ as φ and φ' coincide on L. By definition of A it holds that $0 \notin A$, which is a contradiction and completes the proof.

From the previous result together with exchangeability of $C_{\mathbb{N}}$ and $C'_{\mathbb{N}}$ it is follows that:

▶ **Corollary 18** (`CN_not_cont`). *Closed choice on the natural numbers is discontinuous.*

4 Conclusion

The INCONE library formalizes ideas from computable analysis in COQ. There exists some overlap with other developments, in particular with C-CORN. However, the emphasis of the library is different and many of our examples fall outside of the scope of C-CORN and similar developments. It may be considered complementary as it provides general purpose tools for enriching abstract mathematical objects with computational structure. We feel that the example from the last section of this paper showcases the capabilities of the library well. The abstract definition of the space of open subsets is based on INCONE's function space construction and the proof that it is equivalent to the concrete representation relies on the libraries results about infinite products of represented spaces. We believe the INCONE library to be reasonably accessible to people familiar with the setting that computable analysis works in. We hope that combination with recent developments in computable analysis [41] could open it to an even wider audience including parts of the numerical analysis community.

The INCONE library keeps close to recent work about complexity theory for computable analysis [25, 18, 42, 27] such that it should be possible to add capabilities to at least do qualitative complexity theory in terms of tracking the rate of decrease in accuracy of approximations. Recently there has been a lot of progress on the formalization of models of computation [20, 66] and methods from implicit complexity theory [19] that may even allow to do quantitative complexity theory. Another way to gain insight into such efficiency considerations would be to capture the trace of the basic feasible functionals on the operators on Baire space [39, 23, 24].

The replacement of Baire space by more general spaces means that we maintain the ability to benefit from COQ's machinery in the low-level manipulations of data. From an abstract point of view this makes our approach look like an attempt to interpret a class of generalized Kleene-Kreisel continuous functionals as a computational model in presence of an ambient model of computation. This is a backwards approach to the more common idea of identifying a sub-algebra that captures computability in a given partial combinatory algebra, in this case K_2 [34, 3]. Most of the methods from the RLZRS library are not original and have been implemented independently of a specific proof assistant before [4]. An implementation in other proof assistants one could trade convenience in computationally operating on discrete data against bigger mathematical libraries.

We feel that this paper provides sufficient evidence that the concepts developed in the INCONE library can be used as a foundation for proving statements from computable analysis in COQ. The possible applications we are interested to look into are manifold. One that would be a particularly fitting extension of the contents of this paper is a proof that $C([0,1]) \simeq \mathbb{R}^{[0,1]}$. This statement is called the Computable Weierstraß Theorem [49]: $C([0,1])$ is represented as separable metric space with supremum norm and the rational polynomials as dense sequence and $\mathbb{R}^{[0,1]}$ is a function space. Other possibilities include:

- A more computation-efficient representation of real numbers and results about ODE solving [22, 37, 26]. This may be done by providing an interface with C-CORN, parts of it could also be done separately by relying on libraries like COQ-Interval.
- Duality theory for spaces of summable sequences (ℓ_p-spaces) which provide a pool of examples where subspaces of exponentials can be treated complexity theoretically [53, 51]. Additionally it constitutes a step towards capturing popular methods for solving partial differential equations [12, 54, 10].
- A characterization of continuity via preimages of open sets and similar results [46, 52].

References

1 Klaus Ambos-Spies, Ulrike Brandt, and Martin Ziegler. Real Benefit of Promises and Advice. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *The Nature of Computation. Logic, Algorithms, Applications*, pages 1–11, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

2 Jeremy Avigad and Vasco Brattka. *Computability and analysis: the legacy of Alan Turing*, page 1–47. Lecture Notes in Logic. Cambridge University Press, 2014. `doi:10.1017/CBO9781107338579.002`.

3 Andrej Bauer. *The Realizability Approach to Computable Analysis and Topology*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2000. AAI3002721.

4 Andrej Bauer and C.A. Stone. RZ: A tool for bringing constructive and computable mathematics closer to programming practice. *Computation and Logic in the Real World. CiE 2007. Lecture Notes in Computer Science, vol 4497.*, 2007.

5 Yves Bertot, Laurence Rideau, and Laurent Théry. Distant decimals of π. *Journal of Automated Reasoning*, pages 1–45, 2017. URL: `https://hal.inria.fr/hal-01582524`.

6 Errett Bishop and Douglas Bridges. *Constructive analysis*, volume 279. Springer Science & Business Media, 2012.

7 Jens Blanck. Exact real arithmetic systems: Results of competition. In *Computability and complexity in analysis. 4th international workshop, CCA 2000. Swansea, GB, September 17–19, 2000. Selected papers*, pages 389–393. Berlin: Springer, 2001.

8 Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

9 Sylvie Boldo, François Clément, Jean-Christophe Filliâtre, Micaela Mayero, Guillaume Melquiond, and Pierre Weis. Wave Equation Numerical Resolution: a Comprehensive Mechanized Proof of a C Program. *Journal of Automated Reasoning*, 50(4):423–456, April 2013. `doi:10.1007/s10817-012-9255-4`.

10 Sylvie Boldo, François Clément, Florian Faissole, Vincent Martin, and Micaela Mayero. A Coq Formal Proof of the Lax–Milgram theorem. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2017, pages 79–89, Paris, France, January 2017. ACM. `doi:10.1145/3018610.3018625`.

11 Sylvie Boldo and Guillaume Melquiond. *Computer Arithmetic and Formal Proofs*. ISTE Press - Elsevier, December 2017. URL: `https://hal.inria.fr/hal-01632617`.

12 Vasco Brattka and Atsushi Yoshikawa. Towards computability of elliptic boundary value problems in variational formulation. *Journal of Complexity*, 22(6):858–880, 2006. Computability and Complexity in Analysis. `doi:10.1016/j.jco.2006.04.007`.

13 Cyril Cohen. Construction of real algebraic numbers in Coq. In Lennart Beringer and Amy Felty, editors, *ITP - 3rd International Conference on Interactive Theorem Proving - 2012*, Princeton, United States, August 2012. Springer.

14 Luís Cruz-Filipe, Herman Geuvers, and Freek Wiedijk. C-CoRN, the constructive Coq repository at Nijmegen. In *International Conference on Mathematical Knowledge Management*, pages 88–103. Springer, 2004.

15 K. Deimling. *Multivalued Differential Equations*. De Gruyter series in nonlinear analysis and applications. W. de Gruyter, 1992.

16 Ali Enayat. δ as a continuous function of x and ε. *The American Mathematical Monthly*, 107(2):151–155, 2000.

17 Martín Escardó and Chuangjie Xu. A constructive manifestation of the Kleene–Kreisel continuous functionals. *Annals of Pure and Applied Logic*, 167(9):770–793, 2016. Fourth Workshop on Formal Topology (4WFTop). `doi:10.1016/j.apal.2016.04.011`.

18 Hugo Feree. Game Semantics Approach to Higher-order Complexity. *J. Comput. Syst. Sci.*, 87(C):1–15, August 2017. `doi:10.1016/j.jcss.2017.02.003`.

19 Hugo Férée, Samuel Hym, Micaela Mayero, Jean-Yves Moyen, and David Nowak. Formal proof of polynomial-time complexity with quasi-interpretations. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 146–157. ACM, 2018.

20 Yannick Forster and Gert Smolka. Call-by-Value Lambda Calculus as a Model of Computation in Coq. *Journal of Automated Reasoning*, 2018.

21 A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

22 Fabian Immler and Johannes Hölzl. Numerical Analysis of Ordinary Differential Equations in Isabelle/HOL. In *ITP*, volume 7406 of *LNCS*, pages 377–392, 2012.

23 Bruce M. Kapron and Stephen A. Cook. A New Characterization of Type-2 Feasibility. *SIAM J. Comput.*, 25:117–132, 1996.

24 Bruce M. Kapron and Florian Steinberg. Type-two polynomial-time and restricted lookahead. In *LICS*, 2018.

25 Akitoshi Kawamura and Stephen Cook. Complexity theory for operators in analysis. *ACM Transactions in Computation Theory*, 4(2):Article 5, 2012.

26 Akitoshi Kawamura, Florian Steinberg, and Holger Thies. Parameterized Complexity for Uniform Operators on Multidimensional Analytic Functions and ODE Solving. In *International Workshop on Logic, Language, Information, and Computation*, pages 223–236. Springer, 2018.

27 Akitoshi Kawamura, Florian Steinberg, and Holger Thies. Second-order linear-time computability with applications in computable analysis. *15th Annual Conference on Theory and Applications of Models of Computation*, 2019. extended abstract accepted for presentation at TAMC 2019.

28 S.C. Kleene. Countable functionals. *Constructivity in Mathematics: proceedings of the colloquium held at Amsterdam*, 1959.

29 Ker-I Ko. *Complexity theory of real functions.* Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1991.

30 Michal Konecný and Eike Neumann. Representations and evaluation strategies for feasibly approximable functions. *CoRR*, abs/1710.03702, 2017. `arXiv:1710.03702`.

31 Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite type, Constructivity in Mathematics, 1959.

32 Christoph Kreitz and Klaus Weihrauch. Theory of representations. *Theoretical computer science*, 38:35–53, 1985.

33 Daniel Lacombe. Sur les possibilités d'extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles. In *Le raisonnement en mathématiques et en sciences expérimentales*, Colloques Internationaux du Centre National de la Recherche Scientifique, LXX, pages 67–75. Editions du Centre National de la Recherche Scientifique, Paris, 1958.

34 John Longley and Dag Normann. *Higher-order computability*, volume 100. Springer, 2015.

35 Robert S. Lubarsky and Michael Rathjen. On the constructive Dedekind reals. *Logic and Analysis*, 1(2):131–152, May 2008. `doi:10.1007/s11813-007-0005-6`.

36 Marco Maggesi. A Formalization of Metric Spaces in HOL Light. *J. Autom. Reasoning*, 60(2):237–254, 2018.

37 Evgeny Makarov and Bas Spitters. The Picard Algorithm for Ordinary Differential Equations in Coq. In *ITP*, volume 7998 of *Lecture Notes in Computer Science*, pages 463–468. Springer, 2013.

38 The Mathematical Components library. `https://math-comp.github.io/math-comp/`.

39 Kurt Mehlhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Sciences*, 12(2):147–178, 1976. `doi:10.1016/S0022-0000(76)80035-9`.

40 Norbert Th. Müller. The iRRAM: Exact arithmetic in C++. In *Computability and complexity in analysis. 4th international workshop, CCA 2000. Swansea, GB, September 17–19, 2000. Selected papers*, pages 222–252. Berlin: Springer, 2001.

41 Norbert Th. Müller, Sewon Park, Norbert Preining, and Martin Ziegler. On Formal Verification in Imperative Multivalued Programming over Continuous Data Types. *CoRR*, abs/1608.05787, 2016. `arXiv:1608.05787`.

42 Eike Neumann and Florian Steinberg. Parametrised second-order complexity theory with applications to the study of interval computation. *CoRR*, abs/1711.10530, 2017. submitted for publication. `arXiv:1711.10530`.

43 Russell O'Connor. Essential Incompleteness of Arithmetic Verified by Coq. In Joe Hurd and Tom Melham, editors, *Theorem Proving in Higher Order Logics*, pages 245–260, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

44 Russell O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. PhD thesis, Radboud Universiteit Nijmegen, 2009.

45 Arno Pauly. Multi-valued functions in computability theory. In *Conference on Computability in Europe*, pages 571–580. Springer, 2012.

46 Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016.

47 Arno Pauly and Martin Ziegler. Relative computability and uniform continuity of relations. *J. Logic & Analysis*, 5, 2013.

48 Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*, volume Volume 1 of *Perspectives in Mathematical Logic*. Springer-Verlag, Berlin, 1989.

49 Marian Boykan Pour-El and Jerome Caldwell. On a simple definition of computable function of a real variable-with applications to functions of a complex variable. *Mathematical Logic Quarterly*, 21(1):1–19, 1975.

50 Matthias Schröder. Extended admissibility. *Theoretical computer science*, 284(2):519–538, 2002.

51 Matthias Schröder and Florian Steinberg. Bounded time computation on metric spaces and Banach spaces. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. `doi:10.1109/LICS.2017.8005139`.

52 Matthias Schröeder. *Admissible Representations for Continuous Computations*. PhD thesis, FernUniversität Hagen, 2002.

53 Matthias Schröder. Spaces allowing Type-2 Complexity Theory revisited. *Math. Log. Q.*, 50:443–459, September 2004. `doi:10.1002/malq.200310111`.

54 Svetlana Selivanova and Victor Selivanov. Computing Solutions of Symmetric Hyperbolic Systems of PDE's. *Electron. Notes Theor. Comput. Sci.*, 221:243–255, December 2008. `doi:10.1016/j.entcs.2008.12.021`.

55 Robert I Soare. Recursively enumerable sets and degrees. *Bulletin of the American Mathematical Society*, 84(6):1149–1181, 1978.

56 Florian Steinberg. The Incone library. `https://github.com/FlorianSteinberg/incone`, 2019. release v1.0.

57 Florian Steinberg. The Metric library. `https://github.com/FlorianSteinberg/metric`, 2019. release v1.0.

58 Florian Steinberg. The Mf library. `https://github.com/FlorianSteinberg/mf`, 2019. release v1.0.

59 Florian Steinberg. The Rlzrs library. `https://github.com/FlorianSteinberg/rlzrs`, 2019. release v1.0.

60 Florian Steinberg, Laurent Thery, and Holger Thies. Quantitative continuity and computable analysis in Coq. working paper or preprint, April 2019. URL: `https://hal.inria.fr/hal-02088293`.

61 A. S. Troelstra and D. van Dalen. *Constructivism in mathematics. Vol. II*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988.

62 A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230–265, 1936. `doi:10.1112/plms/s2-42.1.230`.

63 Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. A correction. *Proceedings of the London Mathematical Society*, 2(1):544–546, 1938.

64 Klaus Weihrauch. Computability on computable metric spaces. *Theoretical Computer Science*, 113(2):191–210, 1993.

65 Klaus Weihrauch. *Computable Analysis*. Springer, Berlin/Heidelberg, 2000.

66 Maximilian Wuttke. Verified Programming of Turing Machines in Coq. Master's thesis, Saarland University, 2018.