Global Optimisation with Constructive Reals

Dan R. Ghica and Todd Waugh Ambridge

School of Computer Science, University of Birmingham, UK

***Abstract*—We draw new connections between deterministic, complete, and general global optimisation of continuous functions and a generalised notion of regression, using constructive type theory and computable real numbers. Using this foundation we formulate novel convergence criteria for regression, derived from the convergence properties of global optimisations. We see this as possibly having an impact on optimisation-based computational sciences, which include much of machine learning. Using computable reals, as opposed to floating-point representations, we can give strong theoretical guarantees in terms of both precision and termination. The theory is fully formalised using the safe mode of the proof assistant AGDA. Some examples implemented using an off-the-shelf constructive reals library in JAVA indicate that the approach is algorithmically promising.**

I. INTRODUCTION

A. Global optimisation

For some given *objective function* (f) and set of equalities, inequalities, or arbitrary constraints (S), the central goal of *optimisation* is to compute, with mathematical guarantees, the minimum of f subject to S. The applications are numerous and obvious, in all areas of computational sciences. In particular, this problem subsumes many aspects of *machine learning* if the objective function is an error (or "loss") function representing a kind of distance between a parameterised model and a reference model (or just some sampled data).

Much of the recent literature in this area has focussed on *local optimisation* and methods for computing it efficiently, such as *gradient descent* and supporting techniques such as *automatic differentiation*. Local optimisation is attractive because it can be computed efficiently, even for functions with high dimensionality. However, the only guarantee that a local minimum can make is that small variations of the input will not deteriorate the output.

There exists a mathematically attractive alternative, computing the *global minimum* of f, which can give much stronger correctness guarantees than a local minimum. Subject to continuity and compactness constraints, which are reasonable for many problem domains, the global minimum, up to any desired precision, is computable. The most rigorous global optimisation algorithms, which we will concern ourselves with in this paper, are *general* (no assumptions regarding a particular shape of the function graph), *complete* (guaranteed to find the solution within a specified margin of error), and *deterministic* (strong termination guarantees). In particular, we focus on algorithms in the style of Piyavskii, which apply to Lipschitz-continuous functions, which discretise the domain of the function so that an efficient branch-and-bound-style optimisation algorithm can be then used [1], [2]. The continuity property is then used to guarantee that the granularity of the domain can be used to bound the imprecision of the value of the objective function.

In this paper we are revisiting the problem of global optimisation using a general constructive and type-theoretic approach which generalises the computable real numbers. We call this *constructive reals*. The foundation on which we build is Escardó's work on *searchability* [3], [4], which uses a constructive version of Tychonoff's theorem to search over infinite products of finite spaces, which can be used to represent reals as infinite sequences of (finite) digits.

The first contribution of our paper is of a technical nature. We generalise the class of functions to which the key theorems concerning searchable types apply, by adding an explicit requirement of continuity rather than relying on the fact that functions defined in a constructive setting are implicitly continuous. With this spelled-out assumptions proofs become more intricate but we are enjoying an additional benefit: proofs can be fully formalised in constructive type theory. Concretely, we are formalising all the theory in the "safe" mode of AGDA. Unlike earlier formalisations, ours passes the termination checker and does not rely on the axiom K.

A second technical contribution is showing that our representation of the reals, along with key arithmetic operations, satisfies a constructive specification. While not surprising in itself, the formalisation requires a certain degree of sophistication which should make it of interest.

B. Regression analysis

Regression analysis is a set of algorithms for estimating the relationship between a dependent variable ("outcome") and several independent variables ("features"). A parameterised function is proposed as a *model*, for example a linear function in which the slope and the intercept are the parameters, and the value of the parameters is computed so that a "loss" function (commonly least squares) between the values predicted by the model and some reference data is minimised. A model can be arbitrarily complex, for example a neural network with millions of weights as parameters. Regression analysis is therefore an instance of the global optimisation problem, and guarantees that can be made on computing the global optimum of the loss function can be automatically considered as guarantees on the precision of the regressed model.

Surprisingly to us, although the connection between global optimisation and regression analysis is intuitively very strong, there is little literature studying the connection between the two. We are in this paper remedying this disconnect using computable real numbers and constructive type theory. This gives us a fresh and principled perspective on the problem of regression analysis. We will investigate convergence properties for regression analysis. Convergence properties of interpolation, another way of constructing models out of data, have been studied extensively by Weierstrass-style theorems [5]. In this paper we are proposing several convergence theorems instead for regression. The contribution here is more methodological than technical, as the proofs follow naturally from the mathematical properties of global search. The statements on the other hand are not obvious and require a different conceptual, not just mathematical, perspective on regression analysis.

We need to re-think regression as the process of approximating a black-box *oracle* with a *(parameterised) model* generated from a finite number of data points. A model is said to be *essentially correct* if it is extensionally equal to the oracle for some chosen values of its parameters.

Remember that the convergence of interpolation guarantees that any (subject to sanity conditions) oracle can be reconstituted to any desired precision if the number of samples is large enough. A similar theorem cannot hold for regression, for the simple reason that in regression we must commit to a model which may or may not be essentially correct. For example, if our oracle has quadratic behaviour, no amount of data will yield a precise linear approximation of it. This commitment to a particular model must be taken into account in formulating convergence properties for regression, and this is where our methodological contribution lies. If the model is completely wrong then, of course, convergence cannot be achieved. But a model may not be completely wrong, just idealised, whereas data from the oracle can be distorted by noise or measurement errors. We call the composition of an oracle with unknown error-inducing factors a "distorted oracle". We state and prove the fact that models and oracles satisfy three regression theorems:

- *Perfect regression.* If the underlying model is essentially correct, regression will always compute the correct parameters up to any desired precision using a finite number of samples.
- *Imperfect regression.* The loss between the true oracle and the regressed model is bound by the loss between the true and distorted oracles.
- *Interpolated oracle.* A practically important situation is when regression does not have access to the oracle, but uses data sampled ahead of time. We show how this is an instance of imperfect regression.

C. Motivation

The stone of constructive reals helps us kill two birds.

The first one is that of numerical errors in implementation of optimisation algorithms. Constructive reals have a built-in notion of approximation which is neatly propagated throughout all (continuous) operations. The second one is the formulation of convergence properties which characterise essentially correct models, which can lead to new methods of certifying the correctness of optimisation-based techniques such as machine learning. The finding of global rather than local minima is essential in obtaining such guarantees.

These results are of a foundational nature and do not have the algorithmic maturity of other much-researched techniques. However, preliminary numerical experiments using an off-the-shelf library [6] are yielding results that indicate that these techniques could be in the future of practical interest.

II. Preamble: Constructive Type Theory

Our mathematical framework is that of constructive type theory (MLTT), a common framework for formal mathematics and the backbone of univalent foundations [7], [8]. The notional style is therefore that of MLTT's 'propositions as types' methodology: proving a proposition corresponds to inhabiting a type, dependent products represent universal quantifiers, dependent sums represent constructive existentials, etc. Specifically, we borrow much notation from Escardó's Agda framework for formal mathematics, itself based upon MLTT [9]. This paper's formal Agda development[1] uses Escardó's as a backbone.

A. Notations

The base type theory consists of an empty type $\mathbb{0}$, unit type $\mathbb{1}$, two-element type $\mathbb{2} :\equiv \{0,1\}$, natural numbers type $\mathbb{N}$ and type formers for functions, dependent product types $\prod$, dependent sums $\sum$, co-products $+$, products $\times$ and equality $=$. Every type is a term in one of countably-many type universes $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$ such that $\mathcal{U}_i : \mathcal{U}_{i+1}$; throughout, we leave the index of a type universe implicit.

The type X is not inhabited when the type $\neg X :\equiv X \to \mathbb{0}$ is. A type X is *decidable* if it constructively either holds or does not, $\mathsf{decidable}\ X :\equiv X + \neg X$. A type family $Y : X \to \mathcal{U}$ is *detachable* if it is everywhere decidable, $\mathsf{detachable}\ Y :\equiv \prod_{(x:X)} \mathsf{decidable}(Yx)$. A type X is *discrete* if it has decidable equality, $X\text{-}\mathsf{is}\text{-}\mathsf{discrete} :\equiv \prod_{(x,y:X)} \mathsf{decidable}(x = y)$.

The type of non-empty n-ary products of a type X are labelled X^n such that, by induction, $X^1 :\equiv X$ and $X^{(n+2)} :\equiv X \times X^{(n+1)}$. The type of infinitary sequences of a type X are labelled $X^{\mathbb{N}} :\equiv \mathbb{N} \to X$. The functions $\mathsf{map} : (A \to B) \to (A^{\mathbb{N}} \to B^{\mathbb{N}})$ and $\mathsf{map2} : (A \times B \to C) \to (A^{\mathbb{N}} \times B^{\mathbb{N}} \to C^{\mathbb{N}})$ are the canonical functions of those types. Two infinitary sequences $x, y : X^{\mathbb{N}}$ are *equal-up-to-point* $n : \mathbb{N}$ if their first n elements are pointwise equal, $(x \approx y)\ n :\equiv \prod_{(k:\mathbb{N})} \big((k < n) \to (x_n = y_n)\big)$.

B. Continuity via codistances

Given some type for non-negative real numbers $\mathbb{R}_{\geq 0}$, a *metric space* is a type X equipped with a metric $d_X : X \times X \to \mathbb{R}_{\geq 0}$ such that (i) $d_X(x,y) = 0 \Leftrightarrow x = y$, (ii) $d_X(x,y) = d_X(y,x)$ and (iii) $d_X(x,z) \leq d_X(x,y) + d_X(y,z)$. By replacing the triangle inequality property (iii) with the ultrametric property (iv) $d_X(x,z) \leq \mathsf{max}(d_X(x,y), d_X(y,z))$, we instead define X as a generalised *ultrametric space*. Clearly, every ultrametric space yields a metric space. The ϵ-δ definition of continuous functions is defined on metric spaces.

[1]Accessible at https://github.com/tnttodda/RegressionInType/tree/e930592

The usual definition of continuity which we aim to emulate is the standard one (since this definition is only used as a model it is not written in type-theoretic style). Given two metric spaces X and Y, a function $f : X \to Y$ is *uniformly continuous* if and only if $\forall \epsilon > 0 . \exists \delta > 0 . \forall x, y : X .\ d_X(x, y) < \delta \implies d_Y(fx, fy) < \epsilon$.

For a formulation within our constructive type theory framework, which does not contain a data type for the set of all non-negative real nunbers, we consider ultrametric spaces instead defined by a *co-ultrametric* $c_X : X \times X \to \mathbb{N}_\infty$ where X is a type and $\mathbb{N}_\infty$ is a type for extended natural numbers (the natural numbers extended with a single point $\infty : \mathbb{N}_\infty$) [10]. The extended naturals are implemented in our framework as infinitary binary sequences where each element is less than or equal to the previous.

The extended natural numbers type $\mathbb{N}_\infty$ is defined as, $\mathbb{N}_\infty :\equiv \sum_{(\alpha:2^{\mathbb{N}})}(\text{is-decreasing } \alpha)$, where is-decreasing $\alpha :\equiv \prod_{(i:\mathbb{N})}(\alpha_i \geq \alpha_{i+1})$.
By the above, we can lift any natural $n : \mathbb{N}$ by defining under $n : \mathbb{N}_\infty$ as the sequence of n-many 1s followed by infinitely-many 0s. $\infty : \mathbb{N}_\infty$ is therefore defined as the sequence of infinitely-many 1s.

Given two extended naturals $u, v : \mathbb{N}_\infty$, u is *less-than-or-equal-to* v if $u \preceq v :\equiv \prod_{(n:\mathbb{N})}\big((u_n = 1) \to (v_n = 1)\big)$.

The *minimum* of two extended naturals $\mathsf{min}(u, v) : \mathbb{N}_\infty$ is given via $\mathsf{min} :\equiv \mathsf{map2}\ \mathsf{min}\mathbb{2}$, where $\mathsf{min}\mathbb{2}(0, b) = 0$ and $\mathsf{min}\mathbb{2}(1, b) = b$. Clearly the minimum is decreasing if the arguments are decreasing, so we leave out the construction of the type is-decreasing $\mathsf{min}(u, v)$.

The type for extended naturals allows us to define the type of *codistances* on a given type. A codistance on a type X is a function $c_X : X \times X \to \mathbb{N}_\infty$.

Definition 1. A codistance $c_X : X \times X \to \mathbb{N}_\infty$ on X is a *co-ultrametric* on X if the following types are inhabited,

1) $\prod_{(x,y:X)} c_X(x, y) = \infty \leftrightarrow x = y$,
2) $\prod_{(x,y:X)} c_X(x, y) = c_X(y, x)$,
3) $\prod_{(x,y,z:X)} \mathsf{min}(c_X(x, y), c_X(y, z)) \preceq c_X(x, z)$.

We often refer to a type equipped with a co-ultrametric as a *co-ultrametric type*. We now give three classes of co-ultrametric types: discrete types, products of co-ultrametric types and infinitary sequences of discrete types.

Definition 2. Given a discrete type X with $d : X$-is-discrete, the *discrete codistance* $c_X : X \times X \to \mathbb{N}_\infty$ is defined by case analysis of $d(x, y) : \mathsf{decidable}(x = y)$ for all $x, y : X$,

- If $x = y$ then $c_X(x, y) = \infty$,
- If $x \neq y$ then $c_X(x, y) = \mathsf{under}\ 0$.

Lemma 3. *The discrete codistance is a co-ultrametric for all discrete types.*

Definition 4. The *binary-product codistance* $c_{(X\times Y)} : (X \times Y) \times (X \times Y) \to \mathbb{N}_\infty$ of codistances $c_X : X \times X \to \mathbb{N}_\infty$ and $c_Y : Y \times Y \to \mathbb{N}_\infty$, is

$$c_{(X\times Y)}((x_1, y_1), (x_2, y_2)) :\equiv \mathsf{min}\ (c_X(x_1, x_2), c_Y(y_1, y_2)).$$

From the binary-product codistance, we can define the finite-product codistance for any type X^n, where X has codistance $c_X : X \times X \to \mathbb{N}_\infty$, by induction on $n : \mathbb{N}$.

Lemma 5. *If the composite codistances are co-ultrametrics, the binary/finite-product codistance is a co-ultrametric.*

Definition 6. Given a discrete type X with $d : X$-is-discrete, the *discrete-sequence codistance* $c_{X^{\mathbb{N}}} : X^{\mathbb{N}} \times X^{\mathbb{N}} \to \mathbb{N}_\infty$ is defined coinductively and by case analysis of $d(\mathsf{head}\ x, \mathsf{head}\ y) : \mathsf{decidable}(\mathsf{head}\ x = \mathsf{head}\ y)$ for all $x, y : X^{\mathbb{N}}$:

- If $\mathsf{head}\ x = \mathsf{head}\ y$ then $c_{X^{\mathbb{N}}}(x, y) = 1 :: c_{X^{\mathbb{N}}}(\mathsf{tail}\ x, \mathsf{tail}\ y)$,
- If $\mathsf{head}\ x \neq \mathsf{head}\ y$ then $c_{X^{\mathbb{N}}}(x, y) = \mathsf{under}\ 0$.

Lemma 7. *The discrete-sequence codistance is a co-ultrametric for all discrete types.*

Corollary 8. $\mathbb{N}_\infty$ *is a co-ultrametric type.*

Proof: By Lem. 7, $2^{\mathbb{N}}$ is a co-ultrametric type with co-ultrametric $c_{2^{\mathbb{N}}} : 2^{\mathbb{N}} \times 2^{\mathbb{N}} \to \mathbb{N}_\infty$. Defining $c_{\mathbb{N}_\infty} : \mathbb{N}_\infty \times \mathbb{N}_\infty \to \mathbb{N}_\infty$ as $c_{\mathbb{N}_\infty}((x, i), (y, j)) = c_{2^{\mathbb{N}}}(x, y)$ completes the proof. ∎

Lemma 9. *Given a discrete type X with discrete-sequence codistance $c_{X^{\mathbb{N}}} : X^{\mathbb{N}} \times X^{\mathbb{N}} \to \mathbb{N}_\infty$, the type $(\mathsf{under}\ \epsilon \preceq c_{X^{\mathbb{N}}}(x, y))$ is equivalent to the type $(x \approx y)\ \epsilon$ for all $x, y : X^{\mathbb{N}}$ and $\epsilon : \mathbb{N}$.*

An ultrametric space can be defined by a co-ultrametric type by setting, for example, $d_X(x, y) := 2^{-c_X(x,y)}$, with the usual convention that $2^{-\infty} = 0$. This therefore allows us to implement a definition of continuity on co-ultrametrics within our framework that relates directly to a definition of continuity on a metric space defined by that co-metric.

Definition 10. Given types X and Y, with codistances $c_X : X \times X \to \mathbb{N}_\infty$ and $c_Y : Y \times Y \to \mathbb{N}_\infty$, the function $f : X \to Y$ is *uniformly continuous* if the following type is inhabited:

$$\prod_{(\epsilon:\mathbb{N})} \sum_{(\delta:\mathbb{N})} \prod_{(x,y:X)} (\mathsf{under}\ \delta \preceq c_X(x, y)) \to (\mathsf{under}\ \epsilon \preceq c_Y(fx, fy)).$$

Of course, if one or more of the codistances on X and Y used above are *not* co-ultrametrics, we can still utilise the above definition of 'uniform continuity', but we have no guarantee that what we are utilising relates to a verified definition of continuous functions. As expected, it follows from the above that the composition of multiple continuous functions is continuous.

We can specialise uniform continuity to predicates:

Definition 11. Given a type X with codistance $c_X : X \times X \to \mathbb{N}_\infty$, the predicate $p : X \to \mathcal{U}$ is *uniformly continuous* if the type $\sum_{(\delta:\mathbb{N})} \prod_{(x,y:X)}(\mathsf{under}\ \delta \preceq c_X(x, y)) \to (px \to py)$ is inhabited.

We sometimes refer to the constructed $\delta : \mathbb{N}$ in continuity types as the 'modulus of continuity' of the function or predicate.

A codistance $c_X : X \times X \to \mathbb{N}_\infty$ is *everywhere self-indistinguishable* if, given any $x : X$ and $n : \mathbb{N}$, $c_X(x, x)_n = 1$. By Def. 1.(1), every co-ultrametric is trivially everywhere self-indistinguishable.

A codistance $c_X : X \times X \to \mathbb{N}_\infty$ is right-continuous if (under $\epsilon \preceq c_X(x,y)$) implies (under $\epsilon \preceq c_{\mathbb{N}_\infty}(c_X(z,x), c_X(z,y))$. Of course, the definition of 'right-continuity' also suggests a definition of 'left-continuity'. For simplicity, we only utilise right-continuity in this paper, though the choice is arbitrary as any symmetric codistance (such as any co-ultrametric by Def. 1.(2)) that is either right-continuous or left-continuous is both.

Lemma 12. *The discrete-sequence codistance is everywhere self-indistinguishable and right-continuous for all discrete types.*

Lemma 13. *If the composite codistances are everywhere self-indistinguishable and right-continuous, so is the binary/finite-product codistance.*

C. *Searchable and continuously-searchable types*

A *predicate* on a type X is a detachable type family on that type, $\mathsf{predicate}\ X :\equiv \sum_{(p:X\to\mathcal{U})}(\mathsf{detachable}\ p)$.

Definition 14. A type X is *searchable* if a *search functional* $\mathcal{E}$ (or '*searcher*') can be constructed that, when given a predicate on X, returns a construction of X that satisfies the predicate, if such a construction exists [3], [4].

$$\begin{aligned}\mathsf{searchable}\ X &:\equiv \sum_{(\mathcal{E}:\mathsf{searcher} X)}(\mathsf{search\text{-}condition}\ \mathcal{E}),\\ \mathsf{searcher}\ X &:\equiv \mathsf{predicate}\ X \to X,\\ \mathsf{search\text{-}condition}\ X\ \mathcal{E} &:\equiv \prod_{(p:\mathsf{predicate} X)}\Big(\sum_{(x_0:X)} p(x_0) \to p(\mathcal{E}p)\Big).\end{aligned}$$

Any non-empty finite type is trivially searchable, and Escardó proved the following Tychonoff-style theorem that states countable products of searchable types are searchable [11].

Proposition. *We can construct a term of the following type,*

$$\prod_{(X:\mathbb{N}\to\mathcal{U})}\Big(\prod_{(n:\mathbb{N})}\mathsf{searchable}\ (Xn)\Big) \to \mathsf{searchable}\ \Big(\prod_{(n:\mathbb{N})}(Xn)\Big).$$

This theorem was proved in MLTT under the assumption of a *Brouwerian continuity principle* which states that all predicates are continuous. This result has been written in AGDA, but cannot be formalised safely, meaning computable content cannot be extracted from the proof [11].

Using our definition of uniform continuity on predicates (Def. 11), we can provide a safe alternative to this result by restricting predicates to be searched to only those that are provably 'uniformly continuous' (rather than assuming all predicates are continuous).

A *continuous predicate* on a type X with codistance $c_X : X \times X \to \mathbb{N}_\infty$ is a predicate on X that is continuous via c_X, $\mathsf{c\text{-}predicate}\ X :\equiv \sum_{(p:X\to\mathcal{U})}(\mathsf{detachable}\ p) \times (\mathsf{continuous}\ c_X\ p)$.

Definition 15. A type X is *continuously searchable* via codistance $c_X : X \times X \to \mathbb{N}_\infty$ if a *continuous searcher* can be constructed that, when given a continuous predicate on (X, c_X), returns a construction of X that satisfies the predicate, if such a construction exists.

$$\begin{aligned}&\mathsf{c\text{-}searchable}\ (X, c_X) :\equiv\\ &\qquad \sum_{(\mathcal{E}:\mathsf{c\text{-}searcher})}(X, c_X)(\mathsf{c\text{-}search\text{-}condition}\ \mathcal{E}),\\ &\mathsf{c\text{-}searcher}\ (X, c_X) :\equiv \mathsf{c\text{-}predicate}\ (X, c_X) \to X,\\ &\mathsf{c\text{-}search\text{-}condition}\ (X, c_X)\ \mathcal{E} :\equiv\\ &\qquad \prod_{(p:\mathsf{c\text{-}predicate}(X,c_X))}\Big(\sum_{(x_0:X)} p(x_0) \to p(\mathcal{E}p)\Big).\end{aligned}$$

Lemma 16. *If a type X is searchable, then it is continuously searchable via any codistance $c_X : X \times X \to \mathbb{N}_\infty$.*

Proof: Immediate, as every continuous predicate is a predicate. ■

Corollary 17. *Any non-empty finite type is continuously searchable.*

Proof: The proof follows either from the above Lem. 16 (as any non-empty finite type is searchable) or by trivially showing that every predicate on a non-empty finite type is continuous by the associated discrete codistance. ■

Theorem 18. *Given any searcher $\mathcal{E}_X : \mathsf{searcher}\ X$, $X^{\mathbb{N}}$ is continuously searchable via the discrete-sequence codistance.*

Proof: By induction, we construct the searcher $\mathcal{E}^{\delta}_{X^{\mathbb{N}}}$, which searches continuous predicates $p : \mathsf{c\text{-}predicate}\ (X^{\mathbb{N}}, c_{X^{\mathbb{N}}})$ with modulus of continuity $\delta : \mathbb{N}$, and show that it satisfies the search condition.

The searcher $\mathcal{E}^{0}_{X^{\mathbb{N}}}$ can return any element of $X^{\mathbb{N}}$, as given any $\alpha_0 : X^{\mathbb{N}}$ such that $p(\alpha_0)$, any other $\alpha : X^{\mathbb{N}}$ will satisfy p by the vacuous continuity of p; i.e. under $0 \preceq c_{X^{\mathbb{N}}}(\alpha\_0, \alpha)$. Thus, the search condition for $\mathcal{E}^{0}_{X^{\mathbb{N}}}$ is satisfied.

When $\delta = \delta' + 1$, we construct a new predicate $p'(x) = \lambda\alpha.p(x :: \alpha)$ for any $x : X$. For any argument, this is a continuous predicate with modulus of continuity δ' (one lower than δ, as the head of the sequence is already fixed). The searcher $\mathcal{E}^{\delta}_{X^{\mathbb{N}}}$ returns $(x' :: \alpha') : X^{\mathbb{N}}$, where $x' = \mathcal{E}_X(\lambda x.p(x :: \mathcal{E}^{\delta'}_{X^{\mathbb{N}}}(p'(x))))$, and $\alpha' = \mathcal{E}^{\delta'}_{X^{\mathbb{N}}}(p'(x'))$. As we have some $\alpha_0 = (x'_0 :: \alpha'_0) : X^{\mathbb{N}}$ that satisfies the predicate, the search condition on $\mathcal{E}^{\delta}_{X^{\mathbb{N}}}$ follows from the search conditions on $\mathcal{E}_X$ and $\mathcal{E}^{\delta'}_{X^{\mathbb{N}}}$ ■

Theorem 19. *Given types X and Y that are continuously searchable via some codistances, $X \times Y$ is continuously searchable via the binary-product codistance of those codistances (up to some expected continuity conditions on one of the composite continuous searchers).*

Proof: The proof is similar to the above in that we construct the searcher by splitting the searched continuous predicate on $X \times Y$ in to a continuous predicate on X and another on Y, which we then search with the composite searcher. As above, one of the predicates must have a fixed argument; but because we do not have this 'sequence' structure which ensures the modulus of continuity decreases, we require a continuity condition that effectively says one of the searchers themselves is a continuous function. ■

III. CONSTRUCTIVE REAL ARITHMETIC

The concepts in the previous sections are enough to formulate the general results we are after; but they are set in an abstract framework. In this section we show that this framework is inhabited by at least one essential instance: representations of exact real numbers. This will set our applications on solid theoretical ground.

We utilise a particularly convenient representation of compact real intervals as infinite sequences of signed digits, and we show that the motivating example representation of the interval $[-1, 1]$ corresponds to the Escardó-Simpson specification of that interval. This is new, albeit expected.

We also describe, without proofs, a practical JAVA implementation of the constructive reals from the literature [6]. It follows a similar conceptual development with the simpler signed-digit representation but it is much more sophisticated in order to achieve algorithmic efficiency. Even though the AGDA proofs have computational content and are in principle executable, they are impractical. Our experimental implementation will be therefore based on this JAVA library.

A. *Escardó-Simpson interval object in type theory*

The classical specification of the real numbers is as the unique complete Archimidean field. For ratifying our signed-digit implementation of the real interval $[-1, 1]$, we instead utilise the Escardó-Simpson interval object specification of closed intervals [12]. The completeness axiom is replaced by an iteration property, and the interval is specified by a universal property that gives a recursion principle for real numbers. This specification supports constructive mathematics by design, making it attractive to the theory of computation – and its applications. It has previously been shown that in the category of sets any closed and bounded interval is an interval object, that in the category of topological spaces any closed and bounded interval with the expected Euclidean topology is an interval object, and that the HoTT book reals coincide with the Escardó-Simpson reals [8], [12], [13]. We introduce a formalised implementation of the interval object specification within constructive type theory.

Interval objects are conceived to represent any line segment – a bounded, convex subset of the real line. In order to define convexity, interval objects utilise the concept of taking the midpoint between two numbers, and furthermore that taking such a midpoint can be infinitely iterated. We start with the structure of a *midpoint algebra*, which is a binary operation $\oplus : \mathbb{I} \times \mathbb{I} \to \mathbb{I}$ where the underlying type $\mathbb{I}$ is a *set* (elements can be equal in only one way) and the *midpoint operator* is idempotent ($a = a \oplus a$), commutative ($a \oplus b = b \oplus a$) and transpositional ($(a \oplus b) \oplus (c \oplus d) = (a \oplus c) \oplus (b \oplus d)$). Given two midpoint algebras $(A, \oplus_A)$ and $(B, \oplus_B)$, a map $h : A \to B$ is a *midpoint homomorphism* if $h(x \oplus_A y) = h(x) \oplus_B h(y)$.

We next add two further properties to the above structure: *cancellation* and *iteration*. The cancellation property says that if $a \oplus c = b \oplus c$ then $a = b$; adding this gives us a *cancellative midpoint algebra*. The classical set $\mathbb{R}^n$ is a cancellative midpoint algebra closed under the usual binary midpoint function $\lambda xy.\frac{1}{2}(x + y)$; as are various subsets of $\mathbb{R}^n$, such as the rationals. Furthermore, given two rational endpoints, e.g. -1 and 1, we could use the midpoint function to generate any dyadic rational number in $[-1, 1]$ – but we cannot generate *any* particular point on the convex line. For this, we require our version of the completeness axiom: the iteration property.

The iteration property states that there is an operator $\mathrm{M} : \mathbb{I}^{\mathbb{N}} \to A$ that gives the infinitely iterated midpoint of a stream of points of $\mathbb{I}$. Formally, this operator is defined by two sub-properties: $\mathrm{M}(\alpha) = \mathrm{head}\,\alpha \oplus \mathrm{M}(\mathrm{tail}\,\alpha)$ and $\left(\prod_{(i:\mathbb{N})} \alpha i = \beta i \oplus \alpha(i+1)\right) \to \mathrm{head}\ \ \alpha = \mathrm{M}\,\beta$. From these sub-properties, we can prove the M operator is (i) idempontent $\mathrm{M}(\lambda -.x) = x$, (ii) symmetric $\mathrm{M}(\lambda i.\, \mathrm{M}(\lambda j.xij)) = \mathrm{M}(\lambda i.\, \mathrm{M}(\lambda j.xji))$ and (iii) a midpoint homomorphism $\mathrm{M}(\lambda i.xi \oplus yi) = \mathrm{M}\,x \oplus \mathrm{M}\,y$. Furthermore, every midpoint homomorphism $h : A \to A$ is automatically an M homomorphism, i.e. $\mathrm{M}(\lambda n.h(x_n)) = h(\mathrm{M}(x))$. Adding iteration to a cancellative midpoint algebra gives us the structure we call an *abstract convex body*; every line segment of $\mathbb{R}^n$ is an abstract convex body.

By adding two endpoints $u, v : \mathbb{I}$, we arrive at a *bi-pointed convex body*. A bi-pointed convex body $(A, \oplus_A, M_A, u, v)$ is an interval object if, given any other bi-pointed convex body $(B, \oplus_B, M_B, s, t)$, there is a unique midpoint homomorphism $h^* : A \to B$ that preserves the bi-pointed structure; meaning h satisfies (i) $h^*(u) = s$, (ii) $\prod_{(x,y:A)} h^*(x \oplus_A y) = h^*(x) \oplus_B h^*(y)$.

Thus, given an interval object as above, there is a unique map $\mathsf{affine}_A : A \to A \to A \to A$ where $\mathsf{affine}_A(a, b)$ is defined as the unique map h of the universal property on the interval object $(A, \oplus_A, \mathrm{M}_A, u, v)$ and the sub-interval $(A, \oplus_A, \mathrm{M}_A, a, b)$. This affine map transforms a point $x : A$ on the interval object with endpoints u, v into the relative point $\mathsf{affine}_A(a, b, x) : A$ on the interval object with the same underlying convex body, but with endpoints a, b.

We now axiomatise the interval object $(\mathbb{I}, \oplus, \mathrm{M}, -1, +1)$ for representing the interval $[-1, 1]$, where $\mathbb{I}$ is a set and $-1, +1 : \mathbb{I}$ are two elements representing the endpoints. The term $-1 \oplus +1 : \mathbb{I}$ clearly represents the number 0, and all other numbers in the interval can be represented by terms of $\mathbb{I}$ iteratively generated from these endpoints by $\oplus : \mathbb{I} \times \mathbb{I} \to \mathbb{I}$ and $\mathrm{M} : \mathbb{I}^{\mathbb{N}} \to \mathbb{I}$. The universal property gives us the unique map $\mathsf{affine}(a, b) : \mathbb{I} \to \mathbb{I}$ for any $a, b, : \mathbb{I}$, which transforms a point in $[-1, 1]$ to a point in the sub-interval with endpoints a and b.

The negation operator can be defined as $x^{-1} :\equiv \mathsf{affine}(+1, -1, x)$, which is a midpoint homomorphism satsifying $-1^{-1} = +1$ and $+1^{-1} = -1$. From the uniqueness of affine and the fact that the composition of any two midpoint homomorphisms is a midpoint homomorphism, it can be proved that $(x^{-1})^{-1} = x$. The multiplication operator is defined as $x * y :\equiv \mathsf{affine}(x^{-1}, x, y)$; commutativity and associtativity are again formalised using the uniqueness of affine, and fixing either argument gives a midpoint homomorphism. The fact that these operations and properties are derived from the specification, rather than axioms of it, highlights the conciseness of the interval object specification.

The final concept we introduce is that of *finite approximations* – in the next subsection, showing our functions on encodings for real numbers realise functions on the interval object specification is primarily achieved using finite approximations.

Definition 20. Given two sequences $x, y : \mathbb{I}^{\mathbb{N}}$ and any $n : \mathbb{N}$, x and y are *n-approximately equal* if there are $w, z : \mathbb{I}$ such that $(x_0 \oplus (x_1 \oplus ...(x_n \oplus w))) = (y_0 \oplus (y_1 \oplus ...(y_n \oplus z)))$.

Definition 21. A convex body has *finite approximations* if given two sequences $x, y : \mathbb{I}^{\mathbb{N}}$, showing they are n-approximately equal for all $n : \mathbb{N}$ implies $\mathrm{M}(x) = \mathrm{M}(y)$.

As noted by Escardó and Simpson, the cancellation property is equivalent to having finite approximations and, thus, our interval object $\mathbb{I}$ has finite approximations [12].

Lemma 22. *Given two functions $f, g : X \to \mathbb{I}^{\mathbb{N}}$ if we can show for all $n : \mathbb{N}$ that for any $x_1 : X$ $f(x_1)$ and $g(x_1)$ being n-approximately equal implies for any $x_2 : X$ $f(x_2)$ and $g(x_2)$ are $(n+1)$-approximately equal, then for any $x_3 : X$ $M(f(x_3)) = M(g(x_3))$.*

Proof: This inductive argument holds by finite approximations, as the base case is trivial. ■

B. Exact real arithmetic via signed-digit encodings

In our framework for regression on exact real numbers, we utilise a *signed-digit representation* to encode real numbers in compact intervals [14]. The idea behind this is that, given some $d : \mathbb{N}$, an infinitary sequence $\alpha : \mathbb{N} \to \{-d, ..., 0, ..., d\}$ is a signed-digit representation of the real number $\langle\!\langle \alpha \rangle\!\rangle : [-d, d]$, such that $\langle\!\langle \alpha \rangle\!\rangle = \sum_{n=0}^{\infty} \frac{\alpha_n}{2^{n+1}}$.

Our motivating example utilises functions of type $\mathbb{3}^{\mathbb{N}} :\equiv \mathbb{N} \to \mathbb{3}$ where $3 :\equiv \{-1, 0, 1\}$ to encode real numbers in the interval $[-1, 1]$. We can define the relevant *realisability map*, which maps encodings of $[-1, 1]$ to the numbers they encode in the interval object for $[-1, 1]$.

The realisability map $\langle\!\langle - \rangle\!\rangle : \mathbb{3}^{\mathbb{N}} \to \mathbb{I}$ is defined as $\langle\!\langle \alpha \rangle\!\rangle :\equiv \mathrm{M}(\mathsf{map}\ \langle - \rangle\ \alpha)$, where $\langle - \rangle : \mathbb{3} \to \mathbb{I}$ is the pointwise realisability map defined by cases $\langle -1 \rangle :\equiv -1$, $\langle 0 \rangle :\equiv -1 \oplus +1$ and $\langle 1 \rangle :\equiv +1$.
Because M is idempotent, for each $t : \mathbb{3}$ we have $\langle\!\langle \mathsf{repeat}\ t \rangle\!\rangle = \langle t \rangle$, giving us encodings of the interval's endpoints -1 and $+1$, as well as the midpoint $-1 \oplus +1$.

We wish define the arithmetic operations for negation, midpoint, infinitary midpoint and multiplication on our type of encodings $\mathbb{3}^{\mathbb{N}}$, and confirm that they are both *continuous* and *realise* the relevant functions on the interval $\mathbb{I}$. First, we introduce our realiable definition of continuity on the type of encodings.

Corollary 23. *$\mathbb{3}^{\mathbb{N}}$ is a co-ultrametric type. Furthermore, for any $n : \mathbb{N}$, $(\mathbb{3}^{\mathbb{N}})^n$ is a co-ultrametric type.*

Proof: By Lem. 7, the discrete-sequence codistance is a co-ultrametric for $\mathbb{3}^{\mathbb{N}}$. The finite-product case follows by Lem. 5. ■
By Cor. 23, functions of type $(\mathbb{3}^{\mathbb{N}})^n \to \mathbb{3}^{\mathbb{N}}$ have a corresponding definition of uniform continuity from Def. 10, that relates directly to a definition of uniform continuity metric spaces.

Lemma 24. *An n-ary function $f : (\mathbb{3}^{\mathbb{N}})^n \to \mathbb{3}^{\mathbb{N}}$ is uniformly continuous if for all $\epsilon : \mathbb{N}$ there is a $\delta : \mathbb{N}$ such that, for $(x_1, ..., x_n) : (\mathbb{3}^{\mathbb{N}})^n$ and $(y_1, ..., y_n) : (\mathbb{3}^{\mathbb{N}})^n$, if for each $i \le n$ $(x_i \approx y_i)\ \delta$ then $(f(x_1, ..., x_n) \approx f(x_1, ..., x_n))\ \epsilon$.*

Proof: By Lem. 9, this is equivalent to the definition of uniform continuity given by Def. 10 for the co-ultrametric given by Corollary 23. ■
Put simply, a function on signed-digit codes is continuous if, in order to compute some finite prefix of the function's output, we only require knowledge of some finite prefix of each argument's digits. As expected, it follows from the above that composition preserves continuity. The use of this lemma is implicit in the following.

Lemma 25. *Given any function $f : \mathbb{3} \to \mathbb{3}$, the function $(\mathsf{map}\ f) : \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is uniformly continuous.*

Proof: For all $\epsilon : \mathbb{N}$ and $x, y : \mathbb{3}^{\mathbb{N}}$ we trivially have $(x \approx y)\ \epsilon$ implies $(\mathsf{map}\ f\ x \approx \mathsf{map}\ f\ y)\ \epsilon$. ■

Lemma 26. *Given any function $f : \mathbb{3} \times \mathbb{3} \to \mathbb{3}$, the function $(\mathsf{map2}\ f) : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is uniformly continuous.*

Proof: Similar argument to Lem. 25. ■
Of course, Cor. 23 through to Lem. 26 apply not just to $\mathbb{3}$, but to any discrete type.

We turn our attention to continuous arithmetic realisers for $[-1, 1]$. The functions $\mathsf{neg} : \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$, $\mathsf{mid} : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ and $\mathsf{mul} : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ are intended to encode the negation, midpoint, infinitary midpoint and multiplication functions on the interval $[-1, 1]$, respectively. Their definitions are taken from Escardó's literate Haskell file '*Real number computation in Haskell with real numbers represented as infinite sequences of digits*' [15].

We say that an n-ary function on codes $\overline{f} : X^n \to X$ *realises* a function on the interval object $f : \mathbb{I}^n \to \mathbb{I}$ relative to some realisability map $[\![-]\!] : X \to \mathbb{I}$ if the following diagram commutes:

$$\begin{tikzcd}
X^n \arrow[r, "\overline{f}"] \arrow[d, "{[\![-]\!]^n}"] & X \arrow[d, "{[\![-]\!]}"] \\
\mathbb{I}^n \arrow[r, "f"] & \mathbb{I}
\end{tikzcd}$$

For functions on our type of encodings $\mathbb{3}^{\mathbb{N}}$ we use the term '*realises*' to mean '*realises* relative to the realisability map $\langle\!\langle - \rangle\!\rangle$', and for pointwise functions on the type $\mathbb{3}$ we use the term '*realises*' to mean '*realises* relative to the pointwise realisability map $\langle - \rangle$'.

An n-ary function $\overline{f} : (\mathbb{3}^{\mathbb{N}})^n \to \mathbb{3}^{\mathbb{N}}$ realises a function $f : \mathbb{I}^n \to \mathbb{I}$ if the type $\prod_{(\alpha : (\mathbb{3}^{\mathbb{N}})^n)} (\langle\!\langle \overline{f} \alpha \rangle\!\rangle = f \langle\!\langle \alpha \rangle\!\rangle)$ is inhabited.

A unary pointwise function $\overline{f} : \mathbb{3} \to \mathbb{3}$ realises a function $f : \mathbb{I} \to \mathbb{I}$ if the type $\prod_{(t : \mathbb{3})} (\langle f t \rangle = f \langle t \rangle)$ is inhabited.

Lemma 27. *Given any function $\overline{f} : \mathbb{3} \to \mathbb{3}$ that realises a midpoint homomorphism $h : \mathbb{I} \to \mathbb{I}$, the function $(\mathsf{map}\ \overline{f}) : \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ also realises h.*

Proof: Because h is also an M-homomorphism, we get $\langle\!\langle \mathsf{map}\ \overline{f}\ \alpha \rangle\!\rangle = M(\lambda n. \langle \overline{f} \alpha_n \rangle) = M(\lambda n. h \langle \alpha_n \rangle) = h \langle\!\langle \alpha \rangle\!\rangle$ ■

The negation function is straightforward, it simply 'flips' every digit in the infinitary sequence. The negation operation $\mathsf{neg} : \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is defined as $\mathsf{neg} :\equiv \mathsf{map}\ \mathsf{flip}$, where flip :

$\mathbb{3} \to \mathbb{3}$ is the pointwise negation operation defined by cases $\mathsf{flip}(-1) :\equiv 1$, $\mathsf{flip}(0) :\equiv -1 \oplus +1$ and $\mathsf{flip}(1) :\equiv -1$.

Corollary 28. *The negation operation* neg *is continuous.*

Proof: By Lem. 25. ■

Corollary 29. *The negation operation on codes* neg *realises the negation operation on the interval object* $(-^{-1})$.

Proof: Clearly, flip realises the midpoint homomorphism $(-^{-1})$; thus, the result follows by Lem. 27. ■

The midpoint function is more complex to define, and uses an intermediary signed-digit representation of $[-2, 2]$; i.e. functions of type $\mathbb{5}^{\mathbb{N}} :\equiv \mathbb{N} \to \mathbb{5}$ where $\mathbb{5} :\equiv \{-2, -1, 0, 1, 2\}$. In the following, we utilise the 'affine' realisability map $\mathsf{half}' : \mathbb{5} \to \mathbb{I}$ that sends each digit to its half on the interval object, and an extention of this map $\mathsf{half}(\alpha) :\equiv \mathrm{M}(\mathsf{map}\ \mathsf{half}'\mathbb{5}\ \alpha)$ that sends signed-digit representations of numbers in $[-2, 2]$ to their half on the interval object for $[-1, 1]$.

Lemma 30. *There is a function* $\mathsf{div2\text{-}aux} : \mathbb{5} \times \mathbb{5} \to \mathbb{3} \times \mathbb{5}$ *such that for any* $x, y : \mathbb{5}$ *and* $z : \mathbb{I}$,

$$\mathsf{half}'(x) \oplus (\mathsf{half}'(y) \oplus z) = \langle a \rangle \oplus (\mathsf{half}'(b) \oplus z),$$

where $(a, b) :\equiv \mathsf{div2\text{-}aux}(x, y)$.

Proof: By case analysis on x and y, this is a simple midpoint-arithmetic exercise left for the reader. ■

Definition 31. The halving operation $\mathsf{div2} : \mathbb{5}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is defined as $\mathsf{div2}(x :: (y :: \alpha)) :\equiv a :: \mathsf{div2}(b :: \alpha)$, where $(a, b) :\equiv \mathsf{div2\text{-}aux}(x, y)$.

Operation $\mathsf{add2} : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{5}^{\mathbb{N}}$ is $\mathsf{add2} :\equiv \mathsf{map2}\ \mathsf{add}$, where $\mathsf{add} : \mathbb{3} \times \mathbb{3} \to \mathbb{5}$ sums two elements of $\mathbb{3}$.

Definition 32. The midpoint operation $\mathsf{mid} : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is defined as, $\mathsf{mid}(\alpha, \beta) :\equiv \mathsf{div2}(\mathsf{add2}(\alpha, \beta))$.

Two sequences $\alpha, \beta : \mathbb{3}^{\mathbb{N}}$ will be summed pointwise into a single sequence using add, and then halved into a single sequence by the operation $\mathsf{div2} : \mathbb{5}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$.

Corollary 33. *The midpoint operation* mid *is continuous.*

Proof: The addition operation add2 is continuous by Lem. 26. The halving operation div2 is also continuous: its definition by the function $\mathsf{div2\text{-}aux} : \mathbb{5} \times \mathbb{5} \to \mathbb{3} \times \mathbb{5}$ shows that to know the ϵ-digits of output we must know $(\epsilon + 1)$-digits of input. The composition of these two functions is thus continuous. ■

Lemma 34. *Given a signed-digit encoding* $\alpha : \mathbb{5}^{\mathbb{N}}$ *of a real number in* $[-2, 2]$, *the signed-digit encoding* $\mathsf{div}(\alpha) : \mathbb{3}^{\mathbb{N}}$ *realises the number* $\mathsf{half}(\alpha) : \mathbb{I}$ *on the interval object; i.e.* $\prod_{(\alpha : \mathbb{5}^{\mathbb{N}})} \big(\langle\!\langle \mathsf{div2}(\alpha) \rangle\!\rangle = \mathsf{half}(\alpha)\big)$.

Proof: By the technique outlined in Lem. 22.

Set $f(\alpha) :\equiv \mathsf{map}\ \langle - \rangle\ \mathsf{div2}(\alpha)$ and $g(\alpha) :\equiv \mathsf{map}\ \mathsf{half}'\ \alpha$.

Assume for all $\alpha : \mathbb{5}^{\mathbb{N}}$, $f(\alpha)$ and $g(\alpha)$ are n-approximately equal, and show that for all $x, y : \mathbb{5}$ and $\beta : \mathbb{3}^{\mathbb{N}}$, $f(x :: (y :: \beta))$ and $g(x :: (y :: \beta)))$ are $(n+1)$-approximately equal.

By definition of div2, $f(x :: (y :: \beta)) = \langle a \rangle :: f(b :: \beta)$ where $(a, b) :\equiv \mathsf{div2\text{-}aux}(x, y)$.

We need to prove that $\langle a \rangle \oplus (f(b :: \beta)_0 \oplus ... \oplus (f(b :: \beta)_n \oplus w)) = \mathsf{half}'(x) \oplus (\mathsf{half}'(y) \oplus ... \oplus (\mathsf{half}'(\beta(n-1) \oplus z)))$ for some $w, z : \mathbb{I}$.

By the hypothesis, we have some $w, z : \mathbb{I}$ by which $f(b :: \beta)$ and $g(b :: \beta)$ are n-approximately equal.

Thus, we have $\langle a \rangle \oplus (f(b :: \beta)_0 \oplus ... \oplus (f(b :: \beta)_n \oplus w)) = \langle a \rangle \oplus (\mathsf{half}'(b) \oplus ... \oplus (\mathsf{half}(\beta(n-1)))) \oplus z))$.

We now only need to show that $\langle a \rangle \oplus (\mathsf{half}'(b) \oplus ... \oplus (\mathsf{half}(\beta(n-1)))) \oplus z)) = \mathsf{half}'(x) \oplus (\mathsf{half}'(y) \oplus ... \oplus (\mathsf{half}'(\beta(n-1) \oplus z)))$, which follows by Lem. 30. ■

Lemma 35. *Given two signed-digit encodings* $\alpha, \beta : \mathbb{3}^{\mathbb{N}}$, *the number* $\mathsf{half}(\mathsf{add}(\alpha, \beta)) : \mathbb{I}$ *realises the midpoint of the numbers realised by* α *and* β; *i.e.* $\prod_{(\alpha, \beta : \mathbb{3}^{\mathbb{N}})} \big(\mathsf{half}(\mathsf{add}(\alpha, \beta)) = \langle\!\langle \alpha \rangle\!\rangle \oplus \langle\!\langle \beta \rangle\!\rangle\big)$.

Proof: For any $a, b : \mathbb{3}$, it is trivial to show that $\mathsf{half}'(\mathsf{add2}(a, b)) = \langle a \rangle \oplus \langle b \rangle$. Thus, $\mathsf{half}(\mathsf{add}(\alpha, \beta)) = \mathrm{M}(\lambda n. \langle \alpha_n \rangle \oplus \langle \beta_n \rangle)$. The result follows by the fact M is a midpoint homomorphism. ■

Theorem 36. *The midpoint operation on codes* mid *realises the midpoint operation on the interval object* $\oplus$.

Proof: By Lems. 34 and 35. ■

The $\mathsf{bigMid} : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ operator uses another intermediary signed-digit representation, this time of $[-4, 4]$; i.e. functions of type $\mathbb{9}^{\mathbb{N}} :\equiv \mathbb{N} \to \mathbb{9}$ where $\mathbb{9} :\equiv \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$. We omit the definition of the operator $\mathsf{div4} : \mathbb{9}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ and the fact that it is continuous and realises an operator $\mathsf{quarter} : \mathbb{9}^{\mathbb{N}} \to \mathbb{I}$ that sends signed-digit representations of numbers in $[-4, 4]$ to their quarter on the interval object for $[-1, 1]$ (the reader can induce the details from those concerning div2).

Definition 37. The infinitary midpoint operator $\mathsf{bigMid} : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is defined as, $\mathsf{bigMid}(\alpha s) :\equiv \mathsf{div4}(\mathsf{bigMid}'(\alpha s))$, where $\mathsf{bigMid}' : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}} \to \mathbb{9}^{\mathbb{N}}$ is defined as

$$\begin{aligned}&\mathsf{bigMid}'((a :: b :: \alpha) :: ((c :: \beta) :: \gamma s)) :\equiv \\ &\quad \mathsf{add}'(\mathsf{add}(\mathsf{a}, \mathsf{a}), \mathsf{add}(\mathsf{b}, \mathsf{c})) :: (\mathsf{bigMid}'(\mathsf{mid}(\alpha, \beta) :: \gamma s))),\end{aligned}$$

and $\mathsf{add}' : \mathbb{5} \times \mathbb{5} \to \mathbb{9}$ has the obvious definition.

Lemma 38. *The infinitary midpoint operation* bigMid *is continuous.*

Proof: By composition of the continuity of div4 and bigMid′. The auxiliary operator bigMid′ is continuous as it is defined only by continuous processes (e.g. the midpoint operator). ■

Lemma 39. *Given any* $\alpha, \beta : \mathbb{3}^{\mathbb{N}}$ *and* $z : \mathbb{I}$, $\langle\!\langle \alpha \rangle\!\rangle \oplus (\langle\!\langle \beta \rangle\!\rangle \oplus z) = (\langle \alpha_0 \rangle \oplus (\langle \alpha_1 \rangle \oplus \langle \beta_0 \rangle)) \oplus (\langle\!\langle \mathsf{mid}(\mathsf{tail}(\mathsf{tail}(\alpha)), \mathsf{tail}(\beta)) \rangle\!\rangle \oplus z)$.

Theorem 40. *The infinitary midpoint operation on codes* bigMid *realises the iteration operator on the interval object* M.

Proof: The realisability map used here is $\mathrm{M}(\mathsf{map}\ \langle\!\langle - \rangle\!\rangle\ -) : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}} \to \mathbb{I}$. We set $f :\equiv \mathsf{map}\ \langle\!\langle - \rangle\!\rangle$ and $g(\alpha s) :\equiv \mathsf{map}\ \mathsf{quarter}'\ \mathsf{bigMid}'(\alpha s)$. We assume that for all $\alpha s : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}}$, $f(\alpha s)$ and $g(\alpha s)$ are n-approximately equal, and show that for all $a, b, c : \mathbb{3}$, $\alpha, \beta : \mathbb{3}^{\mathbb{N}}$ and

$\gamma s : (\mathbb{3}^{\mathbb{N}})^{\mathbb{N}}$, $f((a :: b :: \alpha) :: ((c :: \beta) :: \gamma s))$ and $g((a :: b :: \alpha) :: ((c :: \beta) :: \gamma s))$ are $(n+1)$-approximately equal. This is done by combining the induction hypothesis on $\mathsf{mid}(\alpha, \beta) :: \gamma s$ with Lem. 39. Using Lem. 22 gives us, for all $\alpha s : \mathbb{3}^{\mathbb{N}}$, $\mathrm{M}(\mathsf{map}\ \langle\!\langle - \rangle\!\rangle\ \alpha s) = \mathsf{quarter}(\mathsf{bigMid}'(\alpha s))$. The result follows by the fact div4 realises quarter, and thus $\mathrm{M}(\mathsf{map}\ \langle\!\langle - \rangle\!\rangle\ \alpha s) = \mathsf{bigMid}(\alpha s)$. ■

Finally, we use the bigMid operation to define the multiplication operation on codes.

Definition 41. The multiplication operation $\mathsf{mul} : \mathbb{3}^{\mathbb{N}} \times \mathbb{3}^{\mathbb{N}} \to \mathbb{3}^{\mathbb{N}}$ is $\mathsf{mul}(\alpha, \beta) :\equiv \mathsf{bigMid}(\mathsf{map2\ digitMul}\ \alpha\ (\lambda - .\beta))$, where $\mathsf{digitMul}(-1, x) = \mathsf{neg}(x)$, $\mathsf{digitMul}(0, x) = \lambda - .0$ and $\mathsf{digitMul}(1, x) = x$.

Lemma 42. *The multiplication operation* mul *is continuous.*

Proof: From the continuity of bigMid and the mapping of digitMul (by case analysis on $t : \mathbb{3}$, $\mathsf{digitMul}(t)$ is always continuous). ■

Lemma 43. *The pointwise multiplication operation on codes* digitMul *realises the multiplication operation on the interval object* $*$.

Proof: We wish to show that for all $t : \mathbb{3}$ and $\beta : \mathbb{3}^{\mathbb{N}}$ that $\langle\!\langle \mathsf{digitMul}(t, \beta) \rangle\!\rangle = \langle t \rangle * \langle\!\langle \beta \rangle\!\rangle$.

When $t = -1$, the result follows as $\langle\!\langle \mathsf{neg}(\beta) \rangle\!\rangle = \langle\!\langle \beta \rangle\!\rangle^{-1} = -1 * \langle\!\langle \beta \rangle\!\rangle$. When $t = 0$, the result follows as $\langle\!\langle 0 \rangle\!\rangle = (-1 \oplus +1) * \langle\!\langle \beta \rangle\!\rangle$. When $t = 1$, the result follows as $\langle\!\langle \beta \rangle\!\rangle = +1 * \beta$. ■

Theorem 44. *The multiplication operation on codes* mul *realises the multiplication operation on the interval object* $*$.

Proof: Recall that we wish to show for all $\alpha, \beta : \mathbb{3}^{\mathbb{N}}$ that $\mathsf{mul}(\alpha, \beta) = \langle\!\langle \alpha \rangle\!\rangle * \langle\!\langle \beta \rangle\!\rangle$. By Thm. 40, we have $\mathsf{mul}(\alpha, \beta) = \mathrm{M}(\lambda n.\mathsf{map}\ \langle\!\langle - \rangle\!\rangle\ (\mathsf{map2\ digitMul}\ (\alpha_n)\ (\lambda - .\beta)))$

For each $n : \mathbb{N}$, we utilise Lem. 43 and the fact that $\lambda y.\langle \alpha_n \rangle * y$ is a midpoint homomorphism to get $(\mathsf{map}\ \langle\!\langle - \rangle\!\rangle\ (\mathsf{map2\ digitMul}\ (\alpha_n)\ (\lambda - .\beta))) = (\lambda n.\langle \alpha_n \rangle * \langle\!\langle \beta \rangle\!\rangle)$.

The result $M(\lambda n.\langle \alpha_n \rangle * \langle\!\langle \beta \rangle\!\rangle) = \langle\!\langle \alpha \rangle\!\rangle * \langle\!\langle \beta \rangle\!\rangle$ follows by the fact that $\lambda x.x * \langle\!\langle \beta \rangle\!\rangle$ is a midpoint and M-homomorphism. ■

C. Boehm's constructive reals

The signed-digit sequence representation is handy for proofs but not efficiently implementable on conventional CPUs which require wider data representations, such as (finite) integers stretching over several bytes. This is why, for implementation and numerical experimentation, we use Boehm's JAVA library for constructive real arithmetic [6]. This is a sophisticated library in terms of data structures and algorithms, so we do not prove its correctness formally.

In Boehm's library, real numbers x are represented as computable functions $f_x : \mathbb{Z} \to \mathbb{Z}$ such that $|x - 2^n \times f_x(n)| \leq 2^n$. The type of computable reals is defined as CR. For some given object $x : CR$, we write $[\![x]\!]$ to refer to the real number represented by x. In the JAVA implementation, CR is a class with an abstract method $approximate() : int \to BigInteger$, with the argument (finite integers) and return (arbitrary integers) types natively supported by the JDK. The first argument is a precision and the second is a code, satisfying the inequation above.

Each concrete class must implement $approximate()$ as $f_{[\![x]\!]}$. For example, the library provides a representation for π which, when approximated, gives values $f_\pi(1) = 2$, $f_\pi(-5) = 101$, etc. Other sub-classes allow different sorts of real numbers to be represented by different implementations of the approximation function. One such sub-class lifts any $i : BigInteger$ to a CR by simply rounding $i \times 2^{-n}$. The addition sub-class is more interesting, ensuring the final error is less than 2^n:

$$f_{x+y}(n) = \mathsf{round} \frac{f_x(n-2) + f_y(n-2)}{4}.$$

Boehm's library provides a wide variety of operations in this manner. Its main application is the ANDROID calculator app [6]. The approximation functions here are sophisticated, efficient analogues to our signed-digit sequences, which are also functions that give approximations up to a specified degree of precision.

Our methodology is strongly tied to explicit notions of continuity defined on such finite approximations of real number encodings. The constructive ϵ-δ notion of continuity says a function $h : \mathbb{R} \to \mathbb{R}$ is *continuous* if there exists some *modulus of continuity* function $m : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that $|h(x) - h(y)| < m(x, |x - y|)$. Using Boehm's framework, we represent arbitrary continuous functions h with modulus of continuity m as objects of a class $CRFunction$, which has methods:

- `CR function(CR x)`, which should satisfy $[\![$`function(x)`$]\!] = h([\![x]\!])$,
- `CR modulus(CR x, CR e)`, which should satisfy $[\![$`modulus(x,e)`$]\!] = m([\![x]\!], [\![e]\!])$.

As well as representing real numbers that can be approximated to any degree of precision, we can also use the library to represent closed real intervals. All objects of CR whose $approximate()$ method returns the same output m for a given approximation-level input n lie in the real interval $[(2m-1) \times 2^{n-1}, (2m+1) \times 2^{n-1}]$. For example, $f_\pi(1) = 2$ and $f_4(1) = 2$ as both lie in the interval $[3, 5]$. We use this notion to add the class $CRIntervalCode$, whose objects (m, n) represent real intervals as above. We implement methods within this class that allow the interval's lower or upper bound to be compared with that of any other interval.

Using the classes CR, $CRIntervalCode$ and $CRFunction$, we can define a new method in $CRFunction$, $apply : CRIntervalCode \to CRIntervalCode$, which takes a code representing an input interval as an argument and returns a code that represents the interval fully encompassing the output region of the function in the input region. This is constructed using the continuity information provided by the $modulus()$ method.

IV. GENERALISED PARAMETRIC REGRESSION

Let X stand for a type equipped with a codistance $c_X : X \times X \to \mathbb{N}_\infty$, generally used for *parameters* $k : X$, and Y a (higher-order) type generally used to denote *oracles* $\Omega : Y$. A *model function*, usually labelled M, is an X-indexed family of models $(M : X \to Y)$.

A *loss function*, usually labelled Φ, is a codistance (a function of type $Y \times Y \to \mathbb{N}_\infty$) that is everywhere self-indistinguishable and right-continuous. Note that our loss functions, as codistances, in actuality measure 'sameness' towards infinity rather than the 'loss' towards zero – in the following, we use the terminology one would expect for loss functions in regression (i.e. that loss tends to zero and that we minimise loss).

A *distortion functional*, usually labelled Ψ, is a function of type $(Y \times Y \to \mathbb{N}_\infty) \to (Y \times Y \to \mathbb{N}_\infty)$.

Finally, a *parametric regressor*, usually labelled reg, is a functional that computes parameters X using a reference oracle Y and a model function $X \to Y$, so that $\mathsf{reg} : Y \to (X \to Y) \to X$. We show now how to construct a parametric regressor for every continuously searchable type.

Lemma 45. *The type family* $(\lambda x.\mathsf{under}\ \epsilon \preceq x) : \mathbb{N}_\infty \to \mathcal{U}$ *is a continuous predicate for any* $\epsilon : \mathbb{N}$.

Corollary 46. *For any codistance* $c_X : X \times X \to \mathbb{N}_\infty$*, the type family* $(\lambda x.\mathsf{under}\ \epsilon \preceq f(x)) : X \to \mathcal{U}$ *is a continuous predicate for any uniformly continuous* $f : X \to \mathbb{N}_\infty$.

Definition 47. Given a codistance $c_X : X \times X \to \mathbb{N}_\infty$ such that X is continuously searchable via c_X, a loss function $\Phi : Y \times Y \to \mathbb{N}_\infty$ and any precision value $\epsilon : \mathbb{N}$, we construct the *parametric regressor*,

$$\mathsf{reg} : \prod_{(M : X \to Y)} (\mathsf{continuous}\ c_X\ M) \to Y \to X,$$
$$\mathsf{reg}(M, \phi^M, y) :\equiv \mathcal{E}(\lambda x.\mathsf{under}\ \epsilon \preceq \Phi(y, M(x))).$$

In the above, the function $(\lambda x.\Phi(y, M(x))) : X \to \mathbb{N}_\infty$ is uniformly continuous by the uniform continuity of the model function (given by ϕ^M) and the right-continuity of the loss function. Thus, by Cor. 46, the type family $(\lambda x.\mathsf{under}\ \epsilon \preceq \Phi(y, M(x))) : X \to \mathcal{U}$ is a continuous predicate and can be searched.

A. Regression as minimisation

Definition 48. Given two extended naturals $u, v : \mathbb{N}_\infty$, u is *less-than-or-equal-to* v *up-to-point* $\epsilon : \mathbb{N}$ if $u \preceq^\epsilon v :\equiv \prod_{(k:\mathbb{N})} \big((k < \epsilon) \to (uk = 1) \to (vk = 1)\big)$.

Definition 49. A function $f : X \to \mathbb{N}_\infty$ has an *up-to-ϵ-maximum argument* for a given point $\epsilon : \mathbb{N}$ if $\sum_{(x_0:X)} \prod_{(x:X)} (fx \preceq^\epsilon fx_0)$.

Theorem 50. *Given a codistance* $c_X : X \times X \to \mathbb{N}_\infty$ *such that* X *is continuously searchable via* c_X*, any uniformly continuous function* $f : X \to \mathbb{N}_\infty$ *has an up-to-ϵ-maximum-argument for every* $\epsilon : \mathbb{N}$.

Proof: The proof follows by induction on ϵ. In the base case ($\epsilon = 0$), the proof is vacuous and we simply return any $x : X$ via the searcher. In the inductive case ($\epsilon = \epsilon' + 1$ for some $\epsilon' : \mathbb{N}$), we use the searcher to ask whether or not there is an $x : X$ satisfying the continuous predicate $(\lambda x.f(x)_{\epsilon'} = 1)$. If there is, then we can simply return this x, as we cannot consider positions in the domain of f beyond ϵ', and so this must be an up-to-ϵ-maximum-argument. If there is not, then we can invoke the inductive hypothesis to search for the up-to-ϵ'-maximum-argument, as this will also be the up-to-ϵ-maximum-argument, given $f(x)_{\epsilon'}$ is never equal to 1. ■

Theorem 51 (Regression as minimisation)**.** *Given a codistance* $c_X : X \times X \to \mathbb{N}_\infty$ *such that* X *is continuously searchable via* c_X*, an oracle* $\Omega : Y$*, loss function* $\Phi : Y \times Y \to \mathbb{N}_\infty$ *and uniformly continuous model function* $M : X \to Y$*, the function* $\lambda x.\Phi(\Omega, M(x)) : X \to \mathbb{N}_\infty$ *has an up-to-n-maximum-argument for every* $n : \mathbb{N}$.

Proof: By continuity of M and right-continuity of Φ, the function $\lambda x.\Phi(\Omega, M(x))$ is continuous. The result follows by Thm. 50. ■

We are generalising the problem of regression to that of finding the value of the parameter of a model that minimises a loss function. With this broad view of regression in our scope we consider the issue of *convergence*. For arbitrary oracles Ω and model function M the only guarantee we can provide is that the "best arguments" – those which minimise the loss function – can be computed, up to any given precision. What we consider next is whether these "best arguments" are any good in some absolute sense.

B. Regression with an essentially correct model

A general and absolute guarantee of precision can only be given if the model is *essentially correct* in the sense that it is the same as the oracle up to the value of model parameters. And indeed in many practical situations we may know the laws governing certain phenomena but we do not know the constants involved. If the model function is correctly chosen then, using regression, we cannot only minimise the loss function but also we can make it arbitrarily small. To express this property we introduce the concept of *synthetic oracle*, which is simply an oracle Ω "synthesised" from a model function by applying it to an arbitrary and unknown parameter $\Omega = M(k)$.

Theorem 52 (Convergence of perfect regression)**.** *The parametric regressor for given codistance* $c_X : X \times X \to \mathbb{N}_\infty$ *such that* X *is continuously searchable via* c_X*, loss function* $\Phi : Y \times Y \to \mathbb{N}_\infty$ *and any precision value* $\epsilon : \mathbb{N}$ *is such that for any uniformly continuous model function* $M : X \to Y$ *and synthetic oracle* $\Omega :\equiv M(k)$ *for any* $k : X$*, we have that* $\mathsf{under}\ \epsilon \preceq \Phi(\Omega, \omega)$*, where* $\omega = M(\mathsf{reg}(M, \Omega))$.

Proof: The result follows from the later Thm. 53 by setting $\Psi :\equiv \mathsf{id}$; i.e. the distortion functional is just the identity function, as the oracle we query is not distorted from the true oracle. ■

A raw intuition of this theorem statement can be misleading: the regressor sees the synthetic oracle as a black box process, so it is not simply searching for the k. It is searching for *any* parameter that makes the loss function ϵ-small.

The proof of the theorem is the consequence of a more general theorem discussed in the next section. However, conceptually this theorem is more straightforward and the result resonates with the intuition. In the sequel we consider the more complicated question of convergence when using imperfect models on distorted oracles.

C. Regression with imperfect models

There is no established notion of convergence of regression, even less so for regression with a mismatch between the true oracle and the model function. Clearly, when such a mismatch occurs we cannot expect for the loss function to converge to zero as in the case of regression with an essentially correct model. To formulate such a theorem we introduce the concept of a *distorted oracle*; an oracle which is obtained by applying a distortion functional Ψ to a true synthetic oracle Ω.

The strong convergence theorem for imperfect regression, our main result, guarantees that if the mismatch between the distorted oracle and the true oracle, as measured by the loss function, is bound by some ϵ, then the same epsilon will bound the mismatch, as measured by the same loss function, between the distorted oracle and the regressed model.

Theorem 53 (Strong convergence of imperfect regression)**.** *The parametric regressor for given codistance $c_X : X \times X \to \mathbb{N}_\infty$ such that X is continuously searchable via c_X, loss function $\Phi : Y \times Y \to \mathbb{N}_\infty$ and any precision value $\epsilon : \mathbb{N}$ is such that for any uniformly continuous model function $M : X \to Y$, distortion functional $\Psi : Y \to Y$ and synthetic oracle $\Omega :\equiv M(k)$ for any $k : X$, if* $\mathsf{under}\ \epsilon \preceq \Phi(\Psi\Omega, \Omega)$ *then* $\mathsf{under}\ \epsilon \preceq \Phi(\Psi\Omega, \omega)$*, where $\omega = M(\mathsf{reg}(M, \Psi\Omega))$.*

Proof: The regressor searches the continuous predicate $(\lambda x.\mathsf{under}\ \epsilon \preceq \Phi(\Psi\Omega, M(x)))$. By the antecedent, at least one element (i.e. $k : X$) that satisfies this predicate; thus, the element returned by the searcher will also satisfy it by the search condition on the regressor's underlying searcher. ■

By bounding the imprecision of regressing from a distorted oracle by the size of the distortion this theorem makes a clear and strong statement. Moreover, this guarantees that if there is no distortion applied to the oracle, i.e. $\Phi(\Psi\Omega, \Omega)$ converges to zero, then so will the loss between the distorted oracle and the regressed model. Thus, this theorem is the natural generalisation of regression with an essentially-correct model.

There is something unsatisfactory about the theorem. Its precision guarantee is between the regressed model ω and the distorted oracle $\Psi\Omega$. The size of the distortion, used in the antecedent is there to guarantee the success of the computation of the parameters by regression. But can we give a guarantee relating the non-distorted model and the regressed model? The next theorem provides such a guarantee.

Theorem 54 (Weak convergence of imperfect regression)**.** *The parametric regressor for given codistance $c_X : X \times X \to \mathbb{N}_\infty$ such that X is continuously searchable via c_X, loss function $\Phi : Y \times Y \to \mathbb{N}_\infty$ and any precision value $\epsilon : \mathbb{N}$ is such that for any uniformly continuous model function $M : X \to Y$, distortion functional $\Psi : Y \to Y$ and synthetic oracle $\Omega :\equiv M(k)$ for any $k : X$, if* $\mathsf{under}\ \epsilon \preceq \Phi(\Psi\Omega, \Omega)$ *then* $\mathsf{under}\ \epsilon \preceq c_{\mathbb{N}_\infty}(\Phi(\Omega, \Psi\Omega), \Phi(\Omega, \omega))$*, where $\omega = M(\mathsf{reg}(M, \Psi\Omega))$.*

Proof: By Thm. 53 we get $\mathsf{under}\ \epsilon \preceq \Phi(\Psi\Omega, \omega)$. The conclusion follows from the right-continuity of Φ. ■

We say this is the *weak* convergence theorem because it is a logical consequence of the strong convergence theorem. We consider this a key result because it gives a bound on the loss between the true oracle and the inferred model, even if the regression is applied to a distorted model. What is interesting about this theorem is that the error bound is not guaranteed directly, but indirectly: this is a "second order" guarantee that bounds the distance between the two loss values themselves: that between the true oracle and the distorted oracle, against that between the true model and the regressed model.

D. Regression with interpolated models

Oracles as used in these theorems are used "on-line", i.e. they are queried as needed. In common practice, however, regression is usually done "off-line", i.e. on data collected ahead of time, in the form of input-output pairs. This collection of data can be reasonably thought of as a distorted oracle, which approximates the true oracle, the phenomenon we are modelling. The error of this distorted oracle can be measured via sampling and, from the convergence properties of interpolation, can be made as small as desired. Thus, the overall imprecision of "off-line" regression can be made arbitrarily small by combining the convergence of interpolation with the convergence of imperfect regression.

Definition 55. Using a given codistance $c_{Y_2} : Y_2 \times Y_2 \to \mathbb{N}_\infty$ and points $ys : (Y_1)^n$, we construct a *sampled loss function* $L(ys) : (Y_1 \to Y_2) \times (Y_1 \to Y_2) \to \mathbb{N}_\infty$ by using the finite-product codistance $c_{(Y_2)^n} : (Y_2)^n \times (Y_2)^n \to \mathbb{N}_\infty$, $L(ys)(f, g) :\equiv c_{(Y_2)^n}(\{f(ys_0), ..., f(ys_n)\}, \{g(ys_0), ..., g(ys_n)\})$.

Lemma 56. *If the given codistance is everywhere self-indistinguishable and right-continuous, so is the sampled loss function for any points when computing the loss for two continuous functions.*

Proof: By composition of continuities and Lem. 13. ■

Definition 57. Given a decidable, reflexive and transitive order $\leq_{Y_1}: Y_1 \times Y_1 \to \mathcal{U}$, and some points $ys : (Y_1)^n$ that are ordered such that $\neg(ys_{i-1} \leq_{Y_1} ys_i)$, we construct the *constant interpolation function* $I(ys) : (Y_1 \to Y_2) \to (Y_1 \to Y_2)$ by induction on $n : \mathbb{N}$.

When $n = 1$, we return $I(y_0)(f, y) :\equiv f(y_0)$. When $n > 1$, we decide the value of $I((y_0, ys))(f, y)$ by case analysis on whether $y \leq y_0$; if it is, then we return $f(y_0)$, if not, then we return $I(ys)(f, y)$.

Lemma 58. *Given some everywhere self-indistinguishable codistance $c_{Y_2} : Y_2 \times Y_2 \to \mathbb{N}_\infty$, a decidable, reflexive and transitive order $\leq_{Y_1}: Y_1 \times Y_1 \to \mathcal{U}$, points $ys : (Y_1)^n$ that are ordered such that $\neg(ys_{i-1} \leq_{Y_1} ys_i)$, and an oracle $\Omega : Y_1 \to Y_2$, we have* $\mathsf{under}\ \epsilon \preceq L(ys)(I(ys)(\Omega), \Omega)$ *for any $\epsilon : \mathbb{N}$.*

Proof: The intuition here is clear: the loss function L compares the true oracle to itself interpolated at some points, but only at those same points. Hence, the compared values will always be identical. ■

Theorem 59 (Convergence of regression via interpolation)**.** *The parametric regressor for given codistance $c_X : X \times X \to \mathbb{N}_\infty$ such that X is continuously searchable via c_X and any precision value $\epsilon : \mathbb{N}$ is such that for any uniformly continuous model function $M : X \to (Y_1 \to Y_2)$ and synthetic oracle $\Omega :\equiv M(k)$ for any $k : X$,* under $\epsilon \preceq L(ys)(\Psi\Omega, \omega)$*, where $\omega = M(\mathsf{reg}(M, I(ys)\Omega))$, $L(ys)$ is a sampled loss function on some samples $ys : (Y_1)^n$ ordered such that $\neg(ys_{i-1} \leq_{Y_1} ys_i)$ for some decidable, reflexive and transitive order $\leq_{Y_1}$: $Y_1 \times Y_1 \to \mathcal{U}$ (and using some codistance on Y_2), and $I(ys)$ is a constant interpolation function based upon those same samples ys.*

Proof: By Thm. 53 and Lem. 58. ■

V. Examples and applications

A. Signed-digit encodings for the unit interval

Let us illustrate the general framework for parametric regression (Sec. IV) using the verified signed-digit encodings of real numbers in $[-1, 1]$ (Sec. III-B). By Thm. 19, Thm. 18 and Def. 47, we have parametric regressors for any continuous function $M : (\mathbf{3}^{\mathbb{N}})^{d_1} \to ((\mathbf{3}^{\mathbb{N}})^{d_2} \to (\mathbf{3}^{\mathbb{N}})^{d_3})$, a range of types that covers various traditional examples of parametric regression, for example linear or polynomial and much more. To be most concrete, we work out the "hello world" example of regression, *simple linear regression*. More complicated parametric regression models would be perfectly similar, so there is no profit in doing more than the simplest one.

The model is $\mathsf{linear\text{-}model} : \mathbf{3}^{\mathbb{N}} \times \mathbf{3}^{\mathbb{N}} \to (\mathbf{3}^{\mathbb{N}} \to \mathbf{3}^{\mathbb{N}})$ defined as $\mathsf{linear\text{-}model}(\alpha, \beta) :\equiv (\lambda x. \mathsf{mid}(\mathsf{mul}(\alpha, x), \beta))$. It is continuous by composition of Lem. 33 and Lem. 42.

Fixing any degree of precision $\epsilon : \mathbb{N}$ we can perform linear regression analysis using such a loss function on any black-box oracle $\Omega : \mathbf{3}^{\mathbb{N}} \to \mathbf{3}^{\mathbb{N}}$, either on-line or off-line via constant interpolation.

Whatever the oracle, the 'regression as minimisation' convergence property (Thm. 50) will guarantee that we can construct 'best-guess' parameters $(a_1, a_2) : \mathbf{3}^{\mathbb{N}} \times \mathbf{3}^{\mathbb{N}}$ such that $\mathsf{linear\text{-}model}(a_1, a_2) : \mathbf{3}^{\mathbb{N}} \to \mathbf{3}^{\mathbb{N}}$ minimises the loss function, up to precision ϵ.

If the oracle is a linear function, so that our choice of a linear model is *essentially correct*, the parametric regressor will (Thm. 52) compute parameters $(b_1, b_2) : \mathbf{3}^{\mathbb{N}} \times \mathbf{3}^{\mathbb{N}}$ so that $\mathsf{linear\text{-}model}(b_1, b_2) : \mathbf{3}^{\mathbb{N}} \to \mathbf{3}^{\mathbb{N}}$ is approximately equal to the oracle, up to ϵ. Otherwise, the convergence properties of regression with a distorted oracle (Thm. 53) will apply, bounding the loss of the regressed model by the loss of the distortion of the oracle.

Finally, the convergence of regression via interpolation (Thm. 59) means that, if the oracle is constructed by the constant interpolation process on the samples $xs : (\mathbf{3}^{\mathbb{N}})^n$, the regressed model matches the interpolated oracle to any degree of precision.

The results in this section, formalised in Agda, provide *theoretical* guarantees. However, the algorithmic content extracted from the constructive proofs is not geared towards efficient execution; the the code is too slow to run in practice. For practical examples we will change tack, to using the more optimised Boehm constructive reals (Sec. III-C).

B. Global optimisation with Boehm's constructive reals

In this subsection, we give an initial note about work towards developing a more efficient and experimentally versatile 'branch-and-bound'-style global optimisation algorithm on Boehm encodings for real numbers. We discuss only the unary case, though branch-and-bound algorithms can be extended to multiple variables.

Recall that, in general, a global optimisation algorithm seeks to find a minimum argument for a given black-box function $f : \mathbb{R} \to \mathbb{R}$ in some search area, to some degree of precision. A branch-and-bound algorithm consists of the following steps [2]:

1) Initialise the search area as some candidate interval(s),
2) *Select* some candidate from the search area using heuristic criteria,
3) *Branch* the candidate into multiple sub-intervals,
4) *Bound* the sub-intervals by computing the lower and upper bounds of f on each of them from its modulus of continuity,
5) *Discard* any candidates in the search area that cannot contain a global minimiser (i.e. those whose lower bound is strictly higher than another candidate's upper bound),
6) Repeat from step (2) until the difference between the lower and upper bounds of f containing a potential solution candidate are less than the desired precision.

After each iteration, the remaining search area is a potential solution for the global minimisation problem. In the next iteration, the potential solution will either be the same width or thinner than the current. Branch-and-bound algorithms converge given (i) the function f is continuous, (ii) the branching procedure ensures the width of the widest interval tends to 0, and (iii) the bounding procedure ensures that the distance between the lower and upper-bound estimates for each interval also tends to 0 [2]. To develop such an algorithm for the Boehm library, we represent intervals using our aforementioned class $\mathit{CRIntervalCode}$ and continuous functions using $\mathit{CRFunction}$.

The bounding process is performed via the $\mathit{apply}()$ method, which is applied to each interval in the search area, returning an interval encompassing the entire output of that function on any member of the input interval; these output intervals can then be cached and compared for the discarding and selection processes. The branching process is relatively straightforward: in order to branch an interval (m, n), we simply return the three intervals that fully cover that interval in the approximation-level $n-1$ (i.e. $(2m-1, n-1)$, $(2m, n-1)$ and $(2m+1, n-1)$). For example, the encoding of the interval $[3, 5]$, which has approximation-level 1, is fully covered by encodings of the intervals $[2.5, 3.5]$, $[3.5, 4.5]$ and $[4.5, 5.5]$ on approximation-level 0. Due to the nature of the representation, the branching process is not ideal as the three interval codes represent a wider area than the original interval. The algorithm takes care of this by removing any duplicated intervals, and removing any intervals that are completely outside of the original instantiated search area.

The selection heuristic has no impact on the correctness of the algorithm, but obviously affects efficiency. A simple but effective heuristic is to select the interval in the search area with the minimum current output lower-bound.

By implementing addition and multiplication we can experiment with model functions and oracles for any randomly generated polynomial functions. On a conventional laptop, for a precision of 10^{-2} and a timeout of 2 seconds, the algorithm consistently finds the minimum of polynomials of degree up to 5; for a timeout of 10 seconds the the degree increases to 7. Performance degrades, as it is to be expected, both with increase in polynomial degree and precision. For instance, a timeout of 2 seconds and precision of 10^{-2} succeeds only for about 58% of degree 10 polynomials.

Extending to other operations is possible, with the definition of the associated modulus of continuity posing the greatest challenge. A more thorough algorithmic analysis through numerical experiments is in preparation.

VI. Related Work

As an area of research, the study of global optimisation algorithms is mature, with a recent survey indicating more than twenty textbooks and research monographs in the last few decades [16].

Global optimisation algorithms fall under several categories, but in this paper we focused specifically on those algorithms which are *general, complete, continuous, and deterministic*, finding one guaranteed optimal-within-epsilon global minimum of a continuous function [17].

In contrast, incomplete algorithms make no guarantees regarding the quality of the solution it arrives at, focussing on efficiency via sophisticated heuristics, rather than correctness. The typical example of an incomplete algorithm is *gradient descent*, which will only find a local minimum of a function [18]. Another contrasting approach is that of randomized algorithms, which can offer an asymptotic guarantee that the optimum is reached (with probability one), but without actually knowing when the optimum has been reached [19].

The first important results in the area relevant to our work appeared in the 1960s and 70s, namely the optimisation of rational functions using interval arithmetic by Moore and Yang [20], which was then generalised to Lipschitz-continuous functions by Piyavskii [1]. An application of this approach is that it makes possible to use efficient discrete algorithms such as branch-and-bound for continuous optimisation [21].

One of the important and immediate applications of optimisation is *regression*, broadly construed. It means finding some parameters for a model so that a target error (loss) function is minimised. This connection is so intuitive and obvious that it is rather surprising that it is not expressed more emphatically in the literature. This broad formulation of regression captures not just conventional regression problems (linear regression, polynomial regression, etc.) but virtually all machine learning algorithms that are sometimes referred to as 'curve fitting' [22]. Principled theoretical connections between optimisation and regression are, rather surprisingly, few and far between [23].

The inspiration for our new approach to global search and regression is in earlier work on 'searchability' [3], [24], concerning the construction of algorithms ('selection functions') for finding elements in compact spaces satisfying a (computable) predicate. Finite sets are trivially compact, and so are trivially searchable. However, certain infinite sets are also searchable using a constructive interpretation of Tychonoff's theorem, which states that the product space of any set of compact spaces is itself compact. The infinite product of a set X is given by the function space $\mathbb{N} \to X$, whose elements are infinitary sequences of elements of X. These infinitary sequences are therefore, in a certain sense, searchable; which is somewhat surprising. This development is particularly interesting in the context of constructive real numbers, as the computable elements of compact intervals of $\mathbb{R}$ can be represented as infinitary sequences of digits taken from a finite set D. In this work, a constructive Tychonoff-style theorem is utilised to search these representation spaces $\mathbb{N} \to D$ of constructive real numbers relative to certain explicit continuity conditions.

One shortcoming of prior work, which we remedy, is the way in which it is formalised in AGDA. Certain key results require the suspension of the termination checker of the proof assistant, and rely on extrinsic, meta-theoretical, 'pen-and-paper' proofs. We solve this problem by enhancing definitions and theorem statements with explicit assumptions of continuity throughout.

The connection between regression and searchable types is not only mathematical but also conceptual. Regressors can be seen as a quantitative counterpart of Hilbert's indefinite choice operator (ϵ). Searchers are the computable versions of this operator, and for a predicate p, $\epsilon(p)$ produces a witness such that $p(\epsilon(p))$ is either true or false depending on whether the witness exists or not. Compare this with a regressor that given a model function M produces parameters such that the tuned model is $M(\mathsf{reg}(M))$. Such a model is usually neither totally true nor totally false, but it is an approximation of our standard of truth, the oracle, as measured by the loss function. This correspondence between regression and logic hints at a further connection with Peirce's concept of *abduction* ("inference to the best explanation") which, looking farther afield, may lead to new and exciting frameworks for quantitative logical modelling in area such as machine learning.

VII. Conclusion

Our goal was to revisit the problem of global optimisation from the point of view of constructive reals. We believe this is a profitable endeavour, which can shed new light on regression analysis, which we can now formulate as an instance of global search. This further leads us towards new theorems stating convergence properties of regression in the general setting of using a *distorted* oracle as a basis for computing the unknown parameters. *Interpolated* distorted oracles, which are created from sample data are of obvious practical relevance in many applications.

To achieve this, we needed to revisit and improve some prior work on *searchable types*, the signed-digit representation of the unit interval and the interval object specification of the same. The theoretical treatment is fully formalised in the safe mode of AGDA, without using the axiom K.

The proofs in constructive type theory have computable content, but the underlying algorithms are inefficient in the extreme. Thus, in order to show the practical potential of our methodology we implement global search for continuous functions on closed intervals using Boehm's *constructive reals* JAVA library. The data representations and algorithms are more practically sophisticated versions of those extracted from our proofs, though we provide them without a formalisation of their correctness.

Our ultimate aim is that this paper will provide a small but convincing step for the field of regression analysis, with its broad applications to areas such as machine learning, towards a direction where the precision of the model can be estimated in a mathematically sound way.

Acknowledgement. We thank Martín Escardó for helping improve the mathematical content and presentation; and Jon Rowe for suggestions and ideas in the early stages of this research. Todd Waugh Ambridge thanks the School of Computer Science's Agda Club for their feedback and fellowship. Dan R. Ghica acknowledges the support of UK EPSRC Grant EP/V001612/1.

REFERENCES

[1] S. Piyavskii, "An algorithm for finding the absolute extremum of a function," *USSR Computational Mathematics and Mathematical Physics*, vol. 12, no. 4, pp. 57–67, 1972. [Online]. Available: https://doi.org/10.1016/0041-5553(72)90115-2

[2] R. B. Kearfott, "An interval branch and bound algorithm for bound constrained optimization problems," *Journal of Global Optimization*, vol. 2, pp. 259–280, 1992. [Online]. Available: https://doi.org/10.1007/BF00171829

[3] M. Escardó, "Infinite sets that admit fast exhaustive search," in *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, July 2007, pp. 443–452. [Online]. Available: https://doi.org/10.1109/LICS.2007.25

[4] M. Escardó, "Compact, totally separated and well-ordered types in univalent mathematics," 2019, TYPES. [Online]. Available: https://www.cs.bham.ac.uk/~mhe/papers/compact-ordinals-Types-2019-abstract.pdf

[5] A. Pinkus, "Weierstrass and approximation theory," *Journal of Approximation Theory*, vol. 107, no. 1, pp. 1 – 66, 2000. [Online]. Available: https://doi.org/10.1006/jath.2000.3508

[6] H.-J. Boehm, "Small-data computing: correct calculator arithmetic," *Communications of the ACM*, vol. 60, no. 8, pp. 44–49, 2017. [Online]. Available: https://doi.org/10.1145/2911981

[7] P. Martin-Löf and G. Sambin, *Intuitionistic type theory*. Bibliopolis Naples, 1984, vol. 9.

[8] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, Institute for Advanced Study, 2013. [Online]. Available: https://homotopytypetheory.org/book

[9] M. Escardó, "TypeTopology. Various new theorems in univalent mathematics written in Agda," 2010-21. [Online]. Available: https://www.cs.bham.ac.uk/~mhe/agda-new/

[10] M. Escardó and T. Waugh Ambridge, "Codistance," 2020. [Online]. Available: https://www.cs.bham.ac.uk/~mhe/agda-new/Codistance.html

[11] M. Escardó, "CountableTychonoff," 2011. [Online]. Available: https://www.cs.bham.ac.uk/~mhe/agda-new/CountableTychonoff.html

[12] M. Escardó and A. Simpson, "A universal characterization of the closed euclidean interval." *Proceedings - Symposium on Logic in Computer Science*, pp. 115–125, 05 2001. [Online]. Available: https://doi.org/10.1109/LICS.2001.932488

[13] A. B. Booij, "Analysis in univalent type theory," Ph.D. dissertation, University of Birmingham, 2020. [Online]. Available: https://abooij.blogspot.com/p/phd-thesis.html

[14] P. Di Gianantonio, "A functional approach to computability on real numbers," *Bulletin-European Association For Theoretical Computer Science*, vol. 50, pp. 518–518, 1993. [Online]. Available: https://users.dimi.uniud.it/~pietro.digianantonio/papers/

[15] M. Escardó, "Real number computation in Haskell with real numbers represented as infinite sequences of digits," 2011. [Online]. Available: https://www.cs.bham.ac.uk/~mhe/papers/fun2011.lhs

[16] C. A. Floudas and C. E. Gounaris, "A review of recent advances in global optimization," *Journal of Global Optimization*, vol. 45, no. 1, p. 3, 2009. [Online]. Available: https://doi.org/10.1007/s10898-008-9332-8

[17] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta numerica*, vol. 13, pp. 271–369, 2004. [Online]. Available: https://doi.org/10.1017/S0962492904000194

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[19] F. J. Solis and R. J. B. Wets, "Minimization by random search techniques," *Math. Oper. Res.*, vol. 6, no. 1, pp. 19–30, Feb. 1981. [Online]. Available: https://doi.org/10.1287/moor.6.1.19

[20] R. Moore and C. Yang, "Interval analysis," Space Division Report LMSD285875, Lockheed Missiles and Space Co, 1959.

[21] S. Skelboe, "Computation of rational interval functions," *BIT Numerical Mathematics*, vol. 14, no. 1, pp. 87–95, 1974. [Online]. Available: https://doi.org/10.1007/BF01933121

[22] J. Pearl, "To build truly intelligent machines, teach them cause and effect," *Quanta Magazine (15 May 2018)*, 2018. [Online]. Available: https://cacm.acm.org/news/227877

[23] J. Calliess, "Lipschitz Optimisation for Lipschitz Interpolation," *CoRR*, vol. abs/1702.08898, 2017. [Online]. Available: https://arxiv.org/abs/1702.08898

[24] M. Escardó, "Exhaustible sets in higher-type computation," *Logical Methods in Computer Science*, vol. 4, no. 3, 2008. [Online]. Available: http://doi.org/10.2168/LMCS-4(3:3)2008