# Interval Analysis Without Intervals

Paul Taylor

20 February 2006

**Abstract**

We argue that Dedekind completeness and the Heine–Borel property should be seen as part of the "algebraic" structure of the real line, along with the usual arithmetic operations and relations. Dedekind cuts provide a uniform and natural way of formulating differentiation, integration and limits. They and these examples also generalise to intervals. Together with the arithmetic order, cuts enjoy proof-theoretic introduction and elimination rules similar to those for lambda abstraction and application. This system completely axiomatises computably continuous functions on the real line.

We show how this calculus (of "single" points) can be translated formally into Interval Analysis, interpreting the arithmetic operations a la Moore, and compactness as optimisation under constraints. Notice that interval computation is the conclusion and not the starting point.

This calculus for the real line is part of a more general recursive axiomatisation of general topology called Abstract Stone Duality.

## 1 Introduction

Reliable computation gives the authority of a theorem to the results of numerical computations, by setting interval bounds on each step of the calculation. This may be done either within the fixed precision of machine arithmetic, or by allowing this to extend to as many digits as may be required. In either form, this discipline usually fits into the bigger picture of continuous-valued computation by adapting techniques from *numerical* analysis. It often does this by giving them "double vision", *i.e.* computing upper and lower bounds where standard methods would just take whatever rounded value the machine arithmetic provides.

The phrase "without intervals" in the title of this paper signifies that it is part of a programme whose long term motivation is to connect directly with *theoretical* analysis. In this, the basic entities are *single, exact, real* numbers. We begin by introducing a calculus for these, called *Abstract Stone Duality* (ASD), together with some examples to show that this is adequate for *analysis* and not merely *arithmetic*. We then define a formal translation that turns the ASD calculus into one involving machine-representable *intervals*.

The theoretical benefit of (the availability of) this translation is that it eliminates the double vision of interval analysis, in which interval analogues of single-valued concepts (arithmetic, trigonometry, differentiation, Banach spaces ...) have to be devised *ad hoc*, one at a time, in a fashion that gets increasingly estranged from any theoretical roots. Real analysis can be developed within ASD in a single-valued style that is very similar to the traditional one, and then *translated* into intervals. Intervals are therefore the outcome and not the starting point of the method.

This paper advocates some very heretical views about the foundations of real and interval analysis. You can either burn me at the stake for blasphemy, or open your mind to the very simple and natural arguments that I give here. It is true that these benefit from the enlightenment that was provided in the 20th century by proof theory, lambda calculus, category theory, domain theory, denotational semantics and the theory of continuous lattices. These disciplines are now practised in computer science departments, but most of the mathematics that appears here was already familiar in the 19th century.

The widely made assertion that mathematics is founded on set theory blatantly disregards the historical evidence. This is particularly so in the case of the aspects of "infinitesimal calculus" that underlie the vast majority of numerical techniques. The Fourier representation of reasonably well behaved functions, for example, was developed by Dirichlet and others in the *early* 19th century. It was only Cantor's *subsequent* discovery that some extremely unpleasant "functions" could also be represented pointwise by trigonometric series that led him to the invention of set theory. Along with this came fantastical infinities, on which subject he felt moved to offer advice to the Vatican [Dau79]. Besides generating well known paradoxes during its history, set theory is computationally meaningless.

Euclidean geometry was axiomatised using natural ideas such as lines and circles, so topology should also treat such things as open and compact subspaces as its primitives. Then, of course, we must not fall into the trap of referring to a *set* or *collection* [Fef77] of open subspaces: Marshall Stone advised us "always topologize", so these too form a *space*, whose (non-Hausdorff) topology is now called after Dana Scott. Carrying out this plan, we obtain an axiomatisation of locally compact spaces that, since it was never polluted by set-theoretic ingredients, is inherently computable [G]. Curiously, the description of open and closed subspaces using continuous functions, rather than as set-theoretic complements, leads to a very strong duality between them that pervades the theory. It also makes the arguments look more similar to classical ones than to intuitionistic but set-based theories [J].

## 2 Natural axioms for the real line

We shall concentrate on the part of the theory that applies to the real line. The arguments that *equality* of arbitrary real numbers can never the proved computationally have been well rehearsed. *Inequality* and the *strict* arithmetical ordering of numbers, on the other hand, can be witnessed by computing them sufficiently precisely.

Therefore, besides the usual arithmetic operations $+$, $-$, $\times$ and $\div$,

- *real* numbers admit the observable relations $<$, $>$ and $\neq$ but not $\geq$, $\leq$ or $=$,
- whereas *integers* and *rational* numbers admit all six of them.

Topologically, this is because $\mathbb{R}$ is *Hausdorff* but not *discrete*, whereas $\mathbb{N}$ and $\mathbb{Q}$ have both properties.

Notice that there is an asymmetry between things that are classically negatives of one another. Indeed, logical negation ($\neg$) will not be allowed in our calculus. It does permit conjunction ($\wedge$) and disjunction ($\vee$) of properties, since these may be understood in terms of running processes serially or in parallel. In this setting,

- "truth" of a proposition ($\top$) amounts to *termination* of a program, which is *observable*,
- and "falsity" ($\bot$) to *divergence*, which is not.

Negation (interchanging $\top$ with $\bot$) is forbidden computationally. This is because it would solve the *halting problem*, which Alan Turing taught us was impossible [Tur35]. An excellent introduction to the kind of logic conflated with topology and computation that we're using was given in [Vic88], though this point of view has moved on somewhat since that book was written.

Since we intend to manipulate logical expressions in the same way as arithmetical ones, we need to give a name to their type: $\Sigma$. Topologically, this is called the *Sierpiński space* and (classically) it looks like $\begin{pmatrix}\odot\\ \bullet\end{pmatrix}$, with one open point $\top$ and one closed one $\bot$. To C, C++ and Java programmers, $\Sigma$ is known as `void`, whereas it is called `unit` in ML. Functional programs of this type can either terminate or diverge: they do not `return` any "value". It is therefore imperative not to confuse $\Sigma$ with `bool`, which has two different positive values.

Existential quantification over natural, rational or real numbers is allowed. For $\exists_{\mathbb{N}}$, we can set off parallel processes $\phi 0$, $\phi 1$, ... on successive days, and need only observe that *one* of them terminates. Any observable or open property that holds of a *real* number also holds of some nearby rational, and $\mathbb{Q}$ is enumerable, so $\exists_{\mathbb{Q}}$ and $\exists_{\mathbb{R}}$ are allowed.

Universal quantification over infinite sets may seem to be computationally impossible. But $\mathbb{R}$ is not a set. In fact, as we shall explain later, we *can* universally quantify over any *compact* subspace, such as the closed interval $[0,1]$. However, annoyingly for analysis, we *cannot* write $\forall n$ or $\forall\epsilon > 0$, because $\mathbb{N}$ and $(0,\infty) \subset \mathbb{R}$ are not compact. Instead, $n$ and $\epsilon$ may be *free* variables or parameters in an argument [J].

The integers *as numbers* will not play any role in this paper. We do, however, need them for (primitive) *recursive* definitions, for example to index the terms of power series. For this reason, we only define "logician's arithmetic" for them: zero and successor.

Since we intend to perform a syntactic translation of the calculus, it is useful to summarise the operations and variable conventions in a table:

| | | | arguments: | $\mathbb{N}$ | $\mathbb{R}$ | $\mathbb{N}\&\Sigma$ | $\mathbb{R}\&\Sigma$ | $\mathbb{N}\&?$ | $\Sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| results: | $\mathbb{N}$ | $k,n,m$ | $0$ | $\mathsf{succ}$ | | | | $\mathsf{rec}$ | $\mathsf{the}$ |
| | $\mathbb{R}$ | $a,b,x,y,z$ | $0,1$ | $n$ | $+-\times\div$ | | | $\mathsf{rec}$ | $\mathsf{cut}$ |
| | $\Sigma$ | $\delta,\upsilon,\phi,\psi$ | $\top,\bot$ | $=\leq\geq<>\neq$ | $<>\neq$ | $\exists n$ | $\exists x:\mathbb{R}, \forall x:[a,b]$ | $\mathsf{rec}$ | $\land,\lor$ |

Notice that there is no operation that turns real numbers into integers, and nor can there be, since $\mathbb{R}$ is *connected* [J]. However, the *Archimedean axiom* says (parametrically in $x:\mathbb{R}$) that $\exists n{:}\mathbb{Z}.\, n-1 < x < n+1$.

The general theory of topology of which this is a part (ASD) includes function-spaces of the form $\Sigma^X$, whose terms denote open subspaces of $X$. They are written in the notation of the $\lambda$-calculus rather than set theory, so we write $\lambda x{:}X.\,\phi$ instead of $\{x \in X \mid \phi(x)\}$, and $\phi a$ in place of $a \in U$. There is also a way of constructing (some) subspaces, but we shall make very little use of this or $\lambda$-notation in this paper. The reason why we *can* avoid these parts of the calculus is that they *strongly normalise* [A, B].

Any expression in this calculus denotes a continuous function that can be computed by a parallel machine. That is, in principle: one objective of this paper is to give an (incomplete) explanation of how to do it. Conversely, any totally defined function that is continuous and computable on the reals may be expressed in the calculus [G].

It is important to appreciate, however, that, when we say that we may only define *continuous* functions, these may nevertheless be morphisms amongst many different objects in our category. In particular, many of the "discontinuous" functions that are traditionally defined on the reals may still be expressed in our calculus. We just have to use other types besides $\mathbb{R}$ itself. For example:

- The space of *ascending reals*, which is classically the set of real numbers together with $\pm\infty$ and the Scott topology, allows us to define *lower semicontinuous* functions.
- The *interval domain* [Sco70, EL04] provides a continuous meaning to some common kinds of discontinuity, such as that of the sign function. It also presents the Lipschitz property as a generalisation of differentiability that is enjoyed by any (definable, continuous) function.

Like $\Sigma$, these spaces carry non-Hausdorff topologies, and they are familiar in domain theory. But we concentrate on $\mathbb{R}$ because it is more familiar, important and challenging.

Two operations in the final column of the table may be unfamiliar. These define integers *by description* and real numbers as *Dedekind cuts*. Given that we are looking for numerical results (of either kind), these operations are how we derive benefit from logical computations. A *description* is a predicate with an integer argument that is (provably) uniquely satisfied: we obtain the integer in question by manipulating the description into the form "$\mathsf{the}\, n.\, n = 5$". We may obtain *general* from *primitive* recursion using definition by description [A, D]. We devote the next three sections to Dedekind cuts.

# 3 Dedekind cuts and intervals

How can we use the "observable logic" of the previous section to specify a real number $x$? The only symbols that make logical sub-formulae directly from a number are

$$d < x, \qquad a \neq x \qquad \text{and} \qquad x < u,$$

but we may discount $a \neq x$, as it is the same as $x < a \vee a < x$. The others state *lower* (<u>d</u>ownwards) and *<u>u</u>pper* bounds on the unknown value $x$.

Richard Dedekind [Ded72] used this idea to *define* the real numbers from the rationals: the *Dedekind cut* $(D, U)$ corresponding to a real number $x$ consists of the sets $D$ and $U$ of *all possible* lower and upper bounds. He required them to *partition* the rationals, so $D \cup U = \mathbb{Q}$, but then each rational has two different representations. So subsequent authors have modified the definition, omitting the rational number in question, if that's what the cut defines. Then $D$ and $U$ are *open* subspaces, which we represent as *predicates* $\delta$ and $\upsilon$, where

$$\delta d \text{ means } d < x \qquad \text{and} \qquad \upsilon u \text{ means } x < u.$$

These predicates are expressed in the language of the previous section, which only uses the *strict* inequalities $<$ and $>$.

Now, the point is to derive $x$ from $\delta$ and $\upsilon$, so we have to characterise those pairs of open subspaces that can be cuts. Of course, they are *disjoint*, extend respectively towards $-\infty$ and $+\infty$ and "touch" in the middle, like this:

$$-\infty \quad \leftarrow \delta \rightarrow \qquad\qquad d \qquad x \qquad u \qquad\qquad \leftarrow \upsilon \rightarrow \quad +\infty$$

Formally, we write these conditions as

$$\begin{array}{rcl@{\qquad}rcl}
(d < e) \wedge \delta e & \Rightarrow & \delta d & \upsilon t \wedge (t < u) & \Rightarrow & \upsilon u \\
\exists e.\, (d < e) \wedge \delta e & \Leftarrow & \delta d & \exists t.\, \upsilon t \wedge (t < u) & \Leftarrow & \upsilon u \\
\exists d.\, \delta d & \Leftrightarrow & \top & \exists u.\, \upsilon u & \Leftrightarrow & \top \\
(\delta d \wedge \upsilon u) & \Rightarrow & (d < u) & (\delta d \vee \upsilon u) & \Leftarrow & (d < u),
\end{array}$$

where the $\Leftarrow$ conditions are how we express the idea of an open subset of *rationals* in an order-theoretic way; these conditions hold automatically for predicates on the *reals*.

Beware that $\Rightarrow$ is not another logical connective that we forgot to include in the table in the previous section: it is not observable, for the same reason as $\neg$ is not. We write $\phi \Rightarrow \phi$ to mean that *whenever* $\phi$ is observed, *so* is $\psi$. We *prove* such implications *mathematically*, but cannot *observe* them *computationally*.

There are rather a lot of things here to prove, before we can claim that the pair $[\delta, \upsilon]$ defines a real number! In practice, however, the first four conditions are trivial consequences of the transitivity and interpolation properties of $<$, whilst the seventh (disjointness) holds so long as there is some intuitive sense in which there is *something* in between $\delta$ and $\upsilon$.

For this reason, it is useful to consider *pseudo-cuts* that satisfy (at least) these five out of eight conditions. In particular, if (just) $\exists u.\, \upsilon u$ fails, so there is no upper bound, the pseudo-cut defines $+\infty$, and similarly failure of $\exists d.\, \delta d$ indicates $-\infty$. For some purposes it is also useful to drop the disjointness requirement, so that the empty interval ($\delta d \equiv \upsilon u \equiv \top$) becomes the top element of a *lattice*.

The final condition is called *locatedness*, and is often stated as

$$\epsilon > 0 \;\Rightarrow\; \exists du.\, \delta d \wedge \upsilon u \wedge (u - d < \epsilon).$$

It expresses the idea that, whilst we can only work with approximations, they can be made arbitrarily precise, *i.e.* to within $\epsilon \equiv 10^{-k}$, where $k$ is the number of digits required.

We see the meaning of a pseudo-cut more clearly when we consider its complement, which is *convex closed subspace* of $\mathbb{R}$, *i.e.* an *interval* that is closed but possibly unbounded. So we seem to have arrived on familiar territory. Indeed, it's an easy but valuable exercise to check that any interval $[e, t]$ defines a pseudo-cut $[\delta, \upsilon]$ by

$$\delta d \equiv (d < e) \qquad \text{and} \qquad \upsilon u \equiv (t < u),$$

which is why we have written square brackets around the pair $[\delta, \upsilon]$. Conversely, in classical analysis, we may recover $[e, t]$ from $[\delta, \upsilon]$ by defining

$$e \equiv \mathsf{sup}\,\{d \mid \delta d\} \qquad \text{and} \qquad t \equiv \mathsf{inf}\,\{u \mid \upsilon u\}.$$

However, $\mathsf{sup}$ and $\mathsf{inf}$ were not defined in the previous section, and indeed they *do not exist* in our or any competing brand of *constructive* analysis, such as those of Brouwer [TvD88] or Bishop [Bis67].

Now, we want to do computation and not philosophy, so as our first example of a Dedekind cut, let's try to *compute* the supremum of a subset $K \subset \mathbb{R}$. We define

$$\delta' d \equiv \exists x \in K.\, d < x \qquad \text{and} \qquad \upsilon' u \equiv \forall x \in K.\, x < u,$$

and can show that they satisfy all eight conditions for a cut, so long as $K$ is bounded and non-empty *and the quantifiers* $\exists x \in K$ *and* $\forall x \in K$ *are meaningful* [J].

Of course, we've just fallen into the set-theoretic trap again. To pull ourselves out, we have to forget about the "set" $K$, and instead give meanings to these "quantifiers". Notice that they are applied to open sets or predicates, which we represent as functions of type $\Sigma^{\mathbb{R}}$, and yield truth values, of type $\Sigma$. Hence the quantifiers themselves are function*als* of type $\Sigma^{\Sigma^{\mathbb{R}}}$ or, if you prefer, $(\mathbb{R} \to \Sigma) \to \Sigma$. In [J] we explain how $\forall x \in K$ captures the properties of an abstract *compact* subspace $K$.

One situation where the quantifiers are easily defined is when $K$ has just two (not necessarily distinct) elements, $a, b : \mathbb{R}$, and then $\forall \equiv \wedge$ and $\exists \equiv \vee$. These may be used to define $\mathsf{max}(a, b)$, $\mathsf{min}(a, b)$ and $|a| \equiv \mathsf{max}(a, -a)$ as Dedekind cuts [I]. Other functions may then be defined *piecewise*, so long as the pieces agree at their changeover points.

Returning to the interval $[e, t]$ above, with $e \le t$, the calculus of the previous section allows us to define

$$\begin{aligned} \forall x \in [e, t].\, \phi x &\equiv \forall y{:}[0, 1].\, \phi\big(ey + t(1 - y)\big) \\ \exists x \in [e, t].\, \phi x &\equiv \phi e \vee \phi t \vee \exists x{:}\mathbb{R}.\, (e < x < t) \wedge \phi x, \end{aligned}$$

where the second may be simplified if either $e = t$ or $e < t$. It's another useful exercise to show that our definition of $\mathsf{sup}[e, t]$ as a Dedekind cut does yield $t$, as we would expect.

Similarly, if the pseudo-cut $[\delta, \upsilon]$ is bounded, *i.e.* we have $\delta d_0$ and $\upsilon u_0$ for some $d_0, u_0$, we may also define

$$\forall x \in [\delta, \upsilon].\, \phi x \equiv \forall x \in [d_0, u_0].\, \delta x \vee \phi x \vee \upsilon x,$$

but $\delta$ and $\upsilon$ do not supply enough information to define the other quantifier. Hardly surprisingly, $\exists x \in K$ tells us what's *inside* $K$, whereas the pseudo-cut $[\delta, \upsilon]$ was defined in terms of what lies *outside* it, namely its lower and upper bounds. This is why we cannot locate the endpoints.

Indeed, we have shown that an interval (*i.e.* a convex closed subspace) $K$ has endpoints *if and only if* we can say what's inside it, by defining $\exists x \in K$, *cf.* the definition of an interval as a subset of $\mathbb{R}$. In ASD, a subspace with $\exists$ is called *overt*. The theorem in interval analysis that the direct image of an interval (with endpoints) is another such follows from the fact that overt subspaces behave in a formally similar way to compact ones. That is,

- the direct image of an overt subspace under a continuous map is overt (*cf.* compact);
- any open subspace of an overt space is overt (*cf.* closed);
- any overt subspace of a discrete space is open (*cf.* Hausdorff).

Constructive analysts, like classical ones, have their own collection of bizarre counterexamples. These are usually based on famous unsolved conjectures such as that of Goldbach ($2n = p +$

$q$), about which no sane person cares. Since we can prove classically that all pseudo-cuts have endpoints, and "just want to do something simple", why don't we just *define* an "interval" to have endpoints, *i.e.* as $[e, t]$?

One reason is that such a definition would be "analysis with double vision" again, on which we have already made a conceptual advance: The examples in the next section illustrate that the definition that is more general with respect to constructive logic, *i.e.* the weakening of the notion of Dedekind cut, arises entirely naturally in traditional analysis.

Requiring endpoints is also mathematically wrong.

The calculus that we introduced in the previous section was intended to reflect what we understand about both the topology on the real line and computing with real numbers. In particular, we only make assertions that are *computationally observable.* The principal example of something that is *not* observable is the halting problem, *i.e.* the situation when a program runs forever, failing to terminate.

Consider any program you please. Maybe it tests the Goldbach conjecture. Maybe it prevents a nuclear missile from being fired this second. Let $a_n \equiv 1$ if the program is still running at time $n$, but 0 if it has terminated. What is $\mathsf{inf}\, a_n$? What are the endpoints of the interval $\bigcap[-a_n, +a_n]$? What is $\sum 2^{-n} a_n$? In my second example, can you afford the price of knowing the answer?

Abbas Edelat [EL04] has demonstrated that the *interval domain* [Sco70] is a useful theoretical and computational tool for understanding convergence. It links the metrical Banach fixed point theorem in analysis with the order-theoretic one that underlies the semantics of higher-type recursive programs. In a *domain* we must be able to compute least upper bounds of ascending sequences, which we can quite do simply for pseudo-cuts using (set-theoretic unions of open subsets or) the existential quantifier. This corresponds to *intersection* of the complementary closed subspaces. It is not difficult to see that *any* pseudo-cut may be expressed as such an intersection of a nested sequence of closed intervals with rational endpoints. In other words, rational intervals provide a *basis* for the interval domain.

There is actually a constructive sense in which any pseudo-cut $[\delta, \upsilon]$ does have endpoints, namely $\delta$ and $\upsilon$. However, these are not *real* numbers, *i.e.* members of $\mathbb{R}$. They belong to the *non-Hausdorff* spaces of *ascending* and *descending* reals respectively, for which topological continuity is known as *semi-continuity.*

The way in which classical logic wrecks the system is this: It is one thing to expect that the "simple" mathematical questions that we happen to ask will probably always be decidable. It is quite another matter to introduce an *axiom* into the system that guarantees this in all circumstances. This is the difference between paying your debts and giving away your credit card.

Finally, you may be worried that pseudo-cuts make computation with intervals vastly more complicated than it was. It would, in fact, be possible to represent Dedekind cuts and intervals as pairs $\langle\delta, \upsilon\rangle$ of $\lambda$-terms with higher types, but we shall not do so. We shall show how expressions that involve them may be translated into a form that uses *intervals with rational or machine-representable endpoints.* Now we have three kinds of intervals, which are quite different things, related in the way that rational, real and ascending or descending real numbers are. In particular, we can compute directly with rational numbers and rational intervals, but only indirectly with real numbers and general intervals.

# 4 Cuts and intervals in analysis

Dedekind cuts, if they feature at all, are usually dropped very quickly in favour of limits in the analysis textbooks. Being numbers, limits are considered to be serious scientific computation, whereas logical predicates are idle philosophical speculation. We, on the other hand, will calculate with both arithmetical and logical formulae on an equal footing.

In this section we examine three of the most important examples of limits. Our purpose is to demonstrate that Dedekind cuts provide a uniform and straightforward way of formulating these concepts. Also, when we weaken cuts to intervals, the generalisations are themselves familiar ideas (albeit footnotes) in analysis. Recall that we just have to say when $d$ and $u$ are strict lower and upper bounds of the unknown value $x$.

Historically,the first theorem of interval-valued computation was proved by Archimedes. He gave the famous estimate $3\frac{10}{71} < \pi < 3\frac{1}{7}$ for the area of a circle by inscribing and circumscribing regular 96-gons. As is usual in ancient Greek mathematics, the generalisation to $3 \cdot 2^n$-gons is implicit, for lack of suitable notation (it was even necessary for Archimedes to invent ways of writing large and fractional numbers). It is also accompanied by a Euclidean proof that relates the area to the circumference ("$\pi r^2 = \frac{1}{2} r(2\pi r)$"), by bounding them both with polygons [Dij87, Chapter VI].

In this and modern accounts of Riemann integration, it is usual but unnecessary to slice the inner and outer approximants in the same way. We just have to define $\int_a^b f x \, \mathsf{d}x$ by the cut $[\delta, \upsilon]$, where $\delta d$ and $\upsilon u$ hold when

$$\exists f^- f^+.\ d < \textstyle\sum_a^b f^- x \, \mathsf{d}x \ \wedge\ \forall x \in [a, b].\, f^- x < f x < f^+ x \ \wedge\ \textstyle\sum_a^b f^+ x \, \mathsf{d}x < u,$$

in which $f^-$ and $f^+$ are functions whose integrals $\sum_a^b f^\pm x \, \mathsf{d}x$ can be computed in some elementary way. For Archimedes and Riemann, these *elementarily integrable functions* were polygons and piecewise constant functions respectively, whose integrals are the sums of areas of triangles or rectangles. Piecewise constant "functions" involve steps, so for us they are interval- rather than real-valued maps, but this is a red herring: (continuous) piecewise *linear* ("sawtooth") functions would do just as well.

A more important issue for both foundations and computation is the meaning of $\exists f^- f^+$. Just as $\exists_{\mathbb{R}}$ is equivalent to $\exists_{\mathbb{Q}}$, these quantifiers involve a *search* of the *encodings* of elementarily integrable functions $f^-$ and $f^+$. This encoding usually consists of a finite *list* of argument–value pairs, where the arguments *partition* the interval $[a, b]$. (Notice for future reference that such partitions of intervals have been used in computational analysis since ancient times.)

Now, the function $f : [a, b] \to \mathbb{R}$ is *integrable* so long as the $[\delta, \upsilon]$ that results from this process is a Dedekind cut. As we commented in the previous section, it is easy to show that it is a bounded pseudo-cut or interval. The most difficult question is whether there are inner and outer approximations $d$ and $u$ with $u - d < (b - a)\epsilon$ for arbitrarily small $\epsilon > 0$. This follows from the fact that all real-valued functions that are definable in our calculus are *uniformly* continuous [J]. That is, that the upper and lower rectangular slices are within $\epsilon$ of one another vertically so long as their width is less than a suitable $\delta > 0$ that depends on $\epsilon$ (with apologies for the clash of notation).

Textbook accounts of Riemann integration that are anxious to get hold of numbers as soon as possible define the integral as a limit of sums that are defined as functions of the width of the slices. Then one must check that different methods of slicing yield the same final result. For us, none of this is necessary: the uniqueness question is resolved by showing that *any* outer estimate exceeds *any* inner one, irrespectively of the slicing.

We barely have to make any change at all to this definition of integral to allow $f$ to be an *interval*-valued function, $x \mapsto [\delta_x, \upsilon_x]$, such as the piecewise constant (step) functions that we thought might be problematic. We just

$$\text{replace} \quad f^- x < f x < f^+ x \quad \text{by} \quad \delta_x(f^- x) \wedge \upsilon_x(f^+ x).$$

Such a function has an interval-valued integral, unless it is itself is single-valued *almost everywhere*, in a sense that I leave you to formulate.

Turning to our second example, limits of sequences, the various brands of constructive analysis are at pains to emphasise that Dedekind cuts are more general than Cauchy sequences, because subsets are more general than sequences. The classical definition is that

$$\forall \epsilon > 0.\, \exists n_\epsilon.\, \forall n, m \geq n_\epsilon.\, |a_n - a_m| < \epsilon,$$

but constructive analysts require the *modulus of convergence* $n_\epsilon$ and its inverse $\epsilon_n$ to be specified *as functions* (such as $\epsilon_n \equiv \frac{1}{n}$ or $2^{-n}$) that satisfy $\epsilon_n \to 0$. Then the sequence is Cauchy if

$$n, m : \mathbb{N},\ \epsilon > 0 \ \vdash\ n < n_\epsilon \ \vee\ m < n_\epsilon \ \vee\ |a_n - a_m| < \epsilon,$$

or, more simply, $$|a_n - a_m| < \epsilon_{\mathsf{min}(n,m)}.$$

Notice that the second and third definitions eliminate the (inner) quantifiers, and so are admissible as formulae in our calculus, whereas the classical one isn't.

Even though they are both more complicated to define and more restricted in their logical generality than Dedekind cuts, Cauchy sequences are usually preferred in theoretical and automated approaches to constructive analysis. This is because their advocates claim that cuts involve an *impredicative* notion of subset. In fact, our cuts are much less general than the subsets or predicates of constructive set- and type theories, and are therefore (almost) interdefinable with Cauchy sequences.

We see why the explicit modulus of convergence $\epsilon_n$ of a Cauchy sequence is needed as soon as we write down the corresponding Dedekind cut,

$$\delta d \ \equiv\ \exists n{:}\mathbb{N}.\ (d < a_n - \epsilon_n) \qquad \upsilon u \ \equiv\ \exists n{:}\mathbb{N}.\ (a_n + \epsilon_n < u).$$

If you try to do the same thing with the classical definition, you will find that it requires *all* of the (infinitely many) terms to be known, so there is no computational meaning. The explicit modulus of convergence gives a guarantee in advance of the future behaviour of the sequence. From this it is provable, though not computationally observable, that $\forall m > n.\ d < a_n - \epsilon_n < a_m$.

From the way in which we use the terms $(a_n)$ of the sequence here, we see that they may as well be intervals $[a_n - \epsilon_n, a_n + \epsilon_n]$ themselves. Therefore the precision to which we need to compute $a_n$ is prescribed by its position in the sequence. Instead of isolated numbers, the sequence becomes a nest of intervals, whose limit is their intersection, which is the directed join in the interval domain.

For the crucial locatedness property of the cut $[\delta, \upsilon]$, we need $\epsilon_n \to 0$, without which we obtain an interval as the limit, instead of a point. Classically, *any* sequence is "interval-Cauchy" in this sense, with $\epsilon_n \equiv \mathsf{sup}\,\{|a_m - a_{m'}| \mid n < m < m'\}$, and the endpoints of its interval-limit are — you guessed it — the lim inf and lim sup. Constructively, since we have to specify $\epsilon_n$ "in advance", but not tightly, the lim inf and lim sup have weaker properties, and it is doubtful whether they are of much use any more.

A more formal proof that $[\delta, \upsilon]$ is a cut, in the case where $\epsilon_n \equiv 2^{-n}$ is given in [J].

Because of the way in which we envisage using Dedekind cuts for computation in our calculus, it will not be necessary to translate them "back" into Cauchy sequences. However, in order to output the results of computations to the user, we are required to turn a cut into an arbitrarily narrow interval with decimal endpoints, for which we just need the metrical form of the locatedness property. This amounts to the same thing as a Cauchy sequence, with $\epsilon_n \equiv 10^{-n}$, except that $n$ takes a particular value (the number of digits requested by the user), rather than being the subscript variable of a sequence. Dependant Choice is needed to generate a sequence.

Our third example is the *derivative* of a function. Numerical differentiation — as the name itself should warn us — is notoriously unstable, whilst other constructive authors have found that the best definition of a *differentiable function* is as a *pair*, combining the function with its derivative. Even classical accounts that (go on to) treat partial differentiation, *i.e.* of functions of vectors, recognise that the two-term Taylor series

$$f(x+h) \ =\ f(x) + hf'(x) + o(h)$$

is a better formulation than the limit of the quotient $\frac{\delta f}{\delta x}$ as $\delta x \to 0$.

In order to give a definition using Dedekind cuts, we must therefore bound both the derivative and the original function,

$$e_0 < f(x) < t_0 \quad \text{and} \quad e_1 < f'(x) < t_1.$$

We can express this using bounds on the function over some interval around $x$,

$$\exists\delta > 0.\, \forall h{:}[0,\delta]. \qquad \begin{aligned} & e_1 + e_1 h < f(x+h) < t_0 + t_1 h \\ \wedge \quad & e_1 - t_1 h < f(x-h) < t_0 - e_1 h, \end{aligned}$$

which confines the function to a region in the shape of a *bow tie* [EL04].

This formula

- defines a Dedekind cut in $(e_0, t_0)$ since $f : \mathbb{R} \to \mathbb{R}$ was assumed to be a function;
- is a Dedekind cut in $(e_1, t_1)$ if $f$ is *differentiable* at $x$; but
- is a bounded interval in $(e_1, t_1)$ if $f$ is *Lipschitz* at $x$.

In principle, it is possible for an interval-valued function to have a single-valued derivative, although not in a very interesting way. By the "fundamental theorem of calculus", integration and differentiation are inverse, except that the integral needs to be given an initial value. A single-valued integrand may therefore be recovered as the derivative of an interval-valued integral if the initial value is itself an interval.

Uncertainty in *both* the function and its derivative is, recall, an important principle in physics, called after Heisenberg. In this case, the function is position and its derivative is velocity or momentum, for which the two-dimensional pseudo-cut is a rectangle whose area is at least Planck's constant.

We can set up the fundamental theorem of calculus as an isomorphism $C^1 \cong \mathbb{R} \times C^0$ that involves the spaces $C^1$ of (continuously) differentiable functions and $C^0$ of (continuous) functions. Alternatively, instead of *restricting* the derivatives to being single-valued, we may *generalise* the integrands to those interval-valued functions (such as steps) that have single-valued integrals. Repeating this, we obtain the Dirac delta-function $\delta(x)$ as the interval-valued second derivative of the absolute value function, $|x|$. However, we cannot re-integrate that, as we cannot distinguish between $\delta(x)$ and $2\delta(x)$ as intervals.

Notice that *arbitrary* interval-valued functions have interval-valued derivatives, and "calculating" them amounts to writing down the formula above. If we can evaluate this formula, even to give an order-of-magnitude Lipschitz bound, we have a way of estimating what (additional) precision is required of an input to yield the desired precision of the output. This gives some quantitative basis to the $\epsilon$–$\delta$ definition of continuity.

## 5 Type theory for Dedekind cuts

Now that we have shown that Dedekind cuts provide a versatile way of expressing the familiar ideas in Real Analysis, we devote the rest of the paper to a *syntactic* analysis of the structure that we defined in Section 2. This treats Dedekind completeness and the Heine–Borel property as additional "algebraic operations".

The methods that we shall use are quite basic ones in Proof Theory and Type Theory. That is to say, from the same disciplines that have previously chosen to define the real line in terms of Cauchy sequences. The key proof-theoretic observation is that Dedekind completeness is the *introduction* rule corresponding to the arithmetic order relations as *elimination* rules.

This syntactic symmetry was originally formulated by Gerhard Gentzen [Gen35]. For each connective and quantifier there is

- an *introduction rule* whose *conclusion* is a formula that uses the new symbol,
- an *elimination rule* whose main *premise* uses it,
- a *$\beta$-rule* that says how to simplify a proof that contains an introduction rule followed by the corresponding elimination rule, and
- an *$\eta$-rule* that simplifies things the other way round.

For example, the introduction and elimination rules for $P \Rightarrow Q$ are

$$\dfrac{\begin{matrix}[P]\\ \vdots \\ Q\end{matrix}}{P \Rightarrow Q} \qquad\qquad \dfrac{P \qquad P \Rightarrow Q}{Q}$$

where $[P]$ means that the formula $P$ is *discharged*, *i.e.* it is no longer an assumption on which $P \Rightarrow Q$ depends.

Outside Proof Theory, proofs are not usually regarded as mathematical objects themselves, and one proof of a theorem is as good as another. However, there is a very important formal analogy, known as the *Curry–Howard isomorphism*, between proofs and programs. Under this analogy, $P \Rightarrow Q$ corresponds to the *function-space* $P \to Q$ or $Q^P$, and *proofs* of $P \Rightarrow Q$ to *functions* $P \to Q$.

Since we *do* distinguish between functions, and manipulate them as objects, we need a notation for them, called the $\lambda$-*calculus*. We define or *abstract* a function using its introduction rule, and *apply* it using the elimination rule:

$$\dfrac{\begin{matrix}[x : P]\\ \vdots \\ f(x) : Q\end{matrix}}{\lambda x.\, f(x) : P \Rightarrow Q} \qquad\qquad \dfrac{a : P \qquad f : P \Rightarrow Q}{f(a) : Q}$$

where $x$ is a *bound variable*. The $\beta$- and $\eta$-rules simplify things as follows:

$$\big(\lambda x.\, f(x)\big)(a) \;=\; f(a) \qquad \text{and} \qquad \lambda x.\, f(x) \;=\; f.$$

The $\beta$-rule involves *substitution* of the expression $a$ for the variable $x$, and may be seen as *computation*. Indeed, *functional programming* is precisely the development of this idea into a practical tool. This can nowadays compete with *imperative* programming languages like C in all but the most intensive numerical or communications applications, and is far superior for conceptual applications such as compilers.

There are certain important technical caveats about variables and substitution, which we shall overlook in this rhetorical treatment. You can find them in any textbook on the subject; for example, the proof theory and $\lambda$-calculus that we need is amply covered by [GLT89] in an elementary way.

In *certain* calculi of logic or types, repeated use of the $\beta$-rules eventually leads to a *normal form*. In the case of $\lambda$-terms, this looks like

$$\lambda x_1 x_2 \ldots x_n.\, y(a_1)(a_2)\cdots(a_m),$$

where $y$ is a variable that may be free or one of the $x_1 x_2 \ldots x_n$, and $a_1 \ldots a_m$ are also normal forms. In proof theory, $\beta$-rules are known (confusingly for us) as *cuts* (they "cut" out substitutions), and normalisation is called *cut-elimination* or the *Hauptsatz* (because it was the "main theorem" of Gentzen's paper).

It is now a standard procedure to present the connectives of any syntactic calculus as families of introduction, elimination, $\beta$- and $\eta$-rules. Moreover, Bill Lawvere observed (in the case of the quantifiers) that such families amount to an *adjunction* between functors between categories. For a comprehensive categorical treatment, see [Tay99].

The rules for the various connectives do not have exactly the same pattern — to expect them to do so would be to make the same mistake that Peacock, Boole and others made in the 19th century, compelling logic to obey the axioms of arithmetic. Indeed, the *patterns* are fundamental, the connectives being merely the names that we give to them. For example, implication and function-abstraction involve discharged hypotheses and bound variables. Conjunction has two premises in its introduction rule and *two* elimination rules, whilst disjunction combines both of these complications. The introduction rule for comprehension (subset abstraction) has one "substantive" premise (an element of the ambient set) and one logical one (the predicate that determines membership of the subset).

Before we apply these ideas to Dedekind cuts, let's formulate the rules for *definition by description*.[1] We say that a predicate $\phi n$ with an integral argument is a *description* if we can prove that it can be satisfied by exactly one number:

$$\frac{\begin{array}{c}[n:\mathbb{N}]\\ \vdots \\ \phi n : \Sigma\end{array} \qquad \exists n.\, \phi n \Leftrightarrow \top \qquad \begin{array}{c}[n, m:\mathbb{N},\ \phi n \Leftrightarrow \phi m \Leftrightarrow \top]\\ \vdots \\ n = m\end{array}}{(\mathsf{the}\, n.\, \phi n) : \mathbb{N}}$$

This allows us to *introduce* $(\mathsf{the}\, n.\, \phi n)$ as a number. Conversely, if this is a validly formed expression, we may use this premise to *eliminate* the description operator and deduce certain things about $\phi$:

$$\frac{(\mathsf{the}\, n.\, \phi n) : \mathbb{N}}{\exists n.\, \phi n \Leftrightarrow \top} \qquad\qquad \frac{(\mathsf{the}\, n.\, \phi n) : \mathbb{N} \qquad \phi m_1 \Leftrightarrow \phi m_2 \Leftrightarrow \top}{m_1 = m_2}$$

The two $\beta$-rules say that $(\mathsf{the}\, n.\, \phi n)$ deserves its name.

$$\frac{(\mathsf{the}\, n.\, \phi n) : \mathbb{N}}{\phi(\mathsf{the}\, n.\, \phi n) \Leftrightarrow \top} \qquad\qquad \frac{(\mathsf{the}\, n.\, \phi n) : \mathbb{N} \qquad \phi m}{(\mathsf{the}\, n.\, \phi n) = m}$$

As is commonly the case, the $\eta$-rule is the trivial way of making a description, namely to provide the actual number, $m$, and define $\phi n$ as $n = m$:

$$(\mathsf{the}\, n.\, n = m) = m.$$

It's easy to overlook the role of *equality* $(=)$ in this, but it is very important:

$$\text{equality} : \text{description} \quad :: \quad \text{application} : \text{abstraction}.$$

The purpose of the foregoing lesson in proof theory was to observe that the real line enjoys the same analogy,

$$\text{order} : \text{Dedekind completeness}.$$

So here are the rules. We may use a Dedekind cut $[\delta, \upsilon]$ to *introduce* a real number,

$$\frac{\begin{array}{c}[d:\mathbb{R}]\\ \vdots \\ \delta d : \Sigma\end{array} \qquad \begin{array}{c}[u:\mathbb{R}]\\ \vdots \\ \upsilon u : \Sigma\end{array} \qquad \text{axioms of Section 3}}{(\mathsf{cut}\, du.\, \delta d \wedge \upsilon u) : \mathbb{R},}$$

which we may read as, "the unique real number for which $d$ and $u$ are bounds exactly when $\delta d \wedge \upsilon u$". Conversely, if this is a validly formed expression, we may use this premise to *eliminate* the cut operator, and deduce certain things about $\delta$ and $\upsilon$, namely the axioms of Section 3.

The $\beta$-rule says that $(\mathsf{cut}\, du.\, \delta d \wedge \upsilon u)$ obeys the order relations that $\delta$ and $\upsilon$ specify:

$$e < (\mathsf{cut}\, du.\, \delta d \wedge \upsilon u) < t \qquad \Longleftrightarrow \qquad \delta e \wedge \upsilon t.$$

Notice that, as for the $\lambda$-calculus and descriptions, this simply substitutes part of the context for the bound variables. The $\eta$-rule says that any real number $a$ defines a Dedekind cut in the obvious way:

$$\delta d \ \equiv\ (d < a), \qquad \text{and} \qquad \upsilon u \ \equiv\ (a < u).$$

[1] See [A] for the relationship amongst descriptions, sobriety and general recursion and a brief historical survey. I am grateful to Claus-Peter Wirth for bringing [Pea97, §22] to my attention, in which Giuseppe Peano sets out the rules for descriptions in a way that differs very little from that here.

Returning to the general theory, we have said that it *sometimes* provides a normal form. In calculi that allow primitive recursion, the proof ranges from being delicate in simple cases [GLT89, Chapter 6], *via* being the subject of *thèses d'état*, to being impossible. However, if we consider individual connectives instead of the whole system, some useful simplification can still be made.

Recall that, if there is a normal form, its outermost connective or last rule is an *introduction* rule. In our examples, this is

- introduction of $P \Rightarrow Q$,
- $\lambda$-abstraction,
- definition of $(\mathsf{the}\, n.\, \phi n)$ by description, or
- definition of a real number by a Dedekind cut.

We don't need a full normalisation theorem to obtain this — just the relavant $\eta$-rule. So, without loss of generality, we may assume that

- any function $f : P \to Q$ is of the form $\lambda x{:}P.\ fx$, where $fx : Q$ is of base type,
- in particular, any predicate $\phi : \Sigma^{X_1 \times \cdots \times X_n}$ in ASD is of the form $\lambda x_1 \ldots x_n.\ \phi x_1 \ldots x_n$, where $\phi x_1 \ldots x_n$ is of the one base type, $\Sigma$;
- any integral expression $a : \mathbb{N}$ is of the form $\mathsf{the}\, n.\ \phi n$, where $\phi n \equiv (n = a) : \Sigma$; and
- any real-valued expression $a : \mathbb{R}$ is of the form $(\mathsf{cut}\, du.\ \delta d \wedge \upsilon u)$, where $\delta d \equiv (d < a)$ and $\upsilon u \equiv (a < u) : \Sigma$.

Therefore, *we only need to handle logical terms*, those of type $\Sigma$. Numerical terms are represented by logical ones enclosed in description or Dedekind cut operations.

So long as we avoid those connectives (recursion and $\lambda$-application) that can make terms longer, we can normalise everything else. In particular, definition of integers and real numbers by (the introduction rules for) descriptions and Dedekind cuts can be eliminated from expressions by simple and terminating syntactic translations such as

$$\psi(\mathsf{the}\, n.\ \phi n) \qquad \Longleftrightarrow \qquad \exists n.\ \phi n \wedge \psi n.$$

So, whereas analysts have traditionally manipulated real *numbers* or, where necessary, sets of them, and have treated *logical* formulae in a very casual way, we have found that the logic is the central arena of computation, and the other types are side shows.

Before we proceed to the actual syntactic translation, we need to consider what happens if we manipulate formulae that purport to be descriptions or Dedekind cuts, but actually fail the relevant premises. Notice that these premises involve implications, so they are not computationally observable, *i.e.* they cannot be verified by a "run-time system". This simply has to go on with the computation *somehow*, on the implicit assumption that the expressions have been correctly formed.

Of course, we have already discussed the failure of Dedekind's axioms in Section 3 — we obtain intervals instead of numbers, and Ramon Moore defined the corresponding generalisation of arithmetic. The nonsense to which ill formed descriptions lead was amply discussed by Bertrand Russell in [RW13, Introduction, Chapter III(1)].

## 6 The translation

In this section we shall describe the translation that replaces

- *continuous* variables and terms (written with *italic* letters) that denote real numbers or vectors, about which we reason using pure mathematics,
- by *cellular* variables and terms (in $\mathsf{sans\ serif}$) whose values are intervals, cubes, *etc.* with machine-representable coordinates, with which we may compute directly.

As you see, we have switched notation from the separate bounds $d$ and $u$ considered earlier to a machine-representable interval $\mathsf{a} \equiv [d, u]$, where $\underline{\mathsf{a}} \equiv d$ and $\overline{\mathsf{a}} \equiv u$. The cellular notation can be generalised straightforwardly to $\mathbb{R}^{\mathbf{n}}$, in which the cells could be close-packed spheres [CS88] instead of cubes.

We shall view the cell $\mathsf{a}$ through Euclid's eyes, not Cantor's. It is either an open or a closed subspace, according to context — since it is a regular figure, we may pass back and forth between the interior $(\mathsf{a})$ and the closure $[\mathsf{a}]$. In particular,

$$\begin{array}{rlrl} x \in \mathsf{a} & \text{means} \quad x \in (\mathsf{a}) & \text{or} & \underline{\mathsf{a}} < x < \overline{\mathsf{a}} \\ \text{but} \quad \forall x \in \mathsf{a} & \text{means} \quad \forall x \in [\mathsf{a}] & \text{or} & \forall x \in [\underline{\mathsf{a}}, \overline{\mathsf{a}}], \end{array}$$

whilst $\exists x \in \mathsf{a}$ means both $\exists x \in (\mathsf{a})$ and $\exists x \in [\mathsf{a}]$, as these are equivalent, so long as $\underline{\mathsf{a}} < \overline{\mathsf{a}}$.

The interval analogues of the relations $\in$, $<$, $>$ and $\neq$ are respectively

$$\begin{array}{rcl} \mathsf{a} \Subset \mathsf{b} & \equiv & [\mathsf{a}] \subset (\mathsf{b}) \quad \text{or} \quad (\underline{\mathsf{b}} < \underline{\mathsf{a}}) \wedge (\overline{\mathsf{a}} < \overline{\mathsf{b}}) \\ \mathsf{a} \otimes \mathsf{b} & \equiv & \overline{\mathsf{a}} < \underline{\mathsf{b}} \quad \equiv \quad \mathsf{b} \otimes \mathsf{a} \\ \mathsf{a} \pitchfork \mathsf{b} & \equiv & [\mathsf{a}] \cap [\mathsf{b}] = \emptyset \quad \text{or} \quad (\overline{\mathsf{a}} < \underline{\mathsf{b}}) \vee (\overline{\mathsf{b}} < \underline{\mathsf{a}}), \end{array}$$

which are all computationally observable relations. Notice that $\mathsf{x} \Subset \mathsf{w}$ means that $\mathsf{x}$ is "strictly" or *well contained* in $\mathsf{w}$; the symbol is borrowed from the *way below* relation $\ll$ for continuous lattices [GHK+80].

We shall also need ordinary (reflexive) containment,

$$\mathsf{a} \subset \mathsf{b} \;\equiv\; \underline{\mathsf{b}} \leq \underline{\mathsf{a}} \leq \overline{\mathsf{a}} \leq \overline{\mathsf{b}},$$

although, like $\leq$, it only occurs in the theory, not the computation. The intersection, $\mathsf{a} \cap \mathsf{b}$, will be needed for the translation of $\wedge$, the arithmetic operations and repeated variables. It satisfies

$$x \in \mathsf{a} \cap \mathsf{b} \iff x \in \mathsf{a} \wedge x \in \mathsf{b},$$

where the former is defined by

$$\underline{\mathsf{a}}, \underline{\mathsf{b}} \leq \max(\underline{\mathsf{a}}, \underline{\mathsf{b}}) \equiv \underline{\mathsf{a} \cap \mathsf{b}} \quad < x < \quad \overline{\mathsf{a} \cap \mathsf{b}} \equiv \min(\overline{\mathsf{a}}, \overline{\mathsf{b}}) \leq \overline{\mathsf{a}}, \overline{\mathsf{b}}.$$

Finally, the translation uses $\exists \mathsf{x}$. This indicates quantification or search over the possible machine representations of intervals, just as we had it previously over rational numbers and elementarily integrable numbers. However, since these representations are essentially integers, the space of them is not compact, so we do not have $\forall \mathsf{x}$.

The fundamental principle of interval analysis is that Moore's operations compute the direct images of intervals under functions — exactly, in the case of single arithmetic operations, and by overestimation in more complex cases. Indeed, their definitions are commonly given in this form, although we prefer to see this as a *theorem* that is deduced from *purely arithmetical* definitions.

As we did for integrals and limits, we calculate the direct image $\{fx \mid x \in [a, b]\}$ of an interval $[a, b]$ under a function $f$ by characterising its lower and upper bounds,

$$d < \{fx \mid x \in [a, b]\} < u \qquad \text{iff} \qquad \forall x \in [a, b].\, (d < fx < u).$$

In the notation that we have just introduced, this is

$$\{fx \mid x \in \mathsf{a}\} \Subset \mathsf{b} \qquad \text{iff} \qquad \forall x \in \mathsf{x}.\, fx \in \mathsf{b}.$$

Notice that, for general $f$, this uses $\forall$, which we consider in the next section. When $f$ is a single arithmetical operation or some other simple expression, it is sufficient to replace the continuous variable $x$ by the interval $[a, b]$, and the usual arithmetic operations by their interval counterparts. We write

$$d < \{fx \mid x \in [a, b]\} < u \;\equiv\; \forall\!\!\!\forall x \in [a, b].\, (d < fx < u) \;\equiv\; d < (\forall\!\!\!\forall x \in [a, b].\, fx) < u$$

or $$\{fx \mid x \in \mathsf{a}\} \Subset \mathsf{b} \quad \equiv \quad \forall\!\!\!\forall x \in \mathsf{a}.\, (fx \in \mathsf{b}) \quad \equiv \quad (\forall\!\!\!\forall x \in \mathsf{a}.\, fx) \Subset \mathsf{b}$$

in these cases, to exploit the analogy with the universal quantifier. Notice that we define $\mathop{\forall\!\!\!\mid}$ for both arithmetic and logical expressions, where $\mathop{\forall\!\!\!\mid}$ gives the (Moore) direct image in the arithmetical case.

Whilst the notation $\mathop{\forall\!\!\!\mid} x \in \mathsf{x}$ was chosen to look like a quantifier, it is actually the *syntactic* translation that substitutes $\mathsf{x}$ for $x$ and Moore's interval operations $\oplus\ominus\otimes\oslash\Subset\olessthan\ogreaterthan\pitchfork$ for their singleton counterparts $+-\times\div\in<>\neq$. It is therefore accompanied by the same caveats that apply to substitution in the $\lambda$-calculus, in particular concerning its interaction with variable-binding operations such as $\exists$. We may substitute for a free continuous variable $y$ under $\mathop{\forall\!\!\!\mid} x \in \mathsf{x}$, so long as we also apply $\mathop{\forall\!\!\!\mid}$ to the substituand.

It is well known that the Moore interpretations of the *operations* of arithmetic do not obey their usual *laws*, in particular distributivity. There are various techniques for manipulating arithmetic expressions (such as trying to reduce the number of occurrences of each variable) in order to narrow the overestimation of the image. For example, the cognoscenti will already have noticed that it would be quite hopeless to compute the derivative of a function using the formula that we gave in Section 4. We therefore understand that terms of the ASD calculus must be transformed in this way *before* undergoing the translation that we are considering here.

The failure of the arithmetical laws also means that, when we reason with expressions, we are not allowed to apply the rules of arithmetic or logic *within* the scope of the "quantifier" $\mathop{\forall\!\!\!\mid}$. We can only use the clauses of its definition, together with the main induction hypothesis of the proof. On the other hand, *outside* $\mathop{\forall\!\!\!\mid}$ we may use all of the logical and arithmetical equivalences in the usual way, although we shall point them out, as they are the axioms of ASD.

Now we can state the *central proposition* that underlies the meaning of the translation: For each free variable $x$ of a logical formula $\phi$,

$$\phi x \;\Rightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \mathop{\forall\!\!\!\mid} x' \in \mathsf{x}.\, \phi x' \;\Rightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall x' \in \mathsf{x}.\, \phi x' \;\Rightarrow\; \phi x.$$

The proof, which proceeds by induction on the syntactic structure of $\phi$, is given in Section 8. Here we shall discuss its consequences.

We focus on how to compute (an $\epsilon$-approximation to) the value $f(a)$ of a function $f : \mathbb{R} \to \mathbb{R}$ at (a $\delta$-approximation to) an argument $a : \mathbb{R}$. As we said in the previous section, we treat the output, *i.e.* real number $f(a)$, as a Dedekind cut. The central proposition concerns the input, $a$. Using various notations for the input and output intervals, it says that

$$\begin{array}{rcl} f(a) &=& \mathsf{cut}\, et.\, \exists du.\, (d < a < u) \wedge \mathop{\forall\!\!\!\mid} x \in [d,u].\, (e < fx < t) \\ &\equiv& \mathsf{cut}\,\mathsf{y}.\, \exists \mathsf{x}.\, a \in \mathsf{x} \wedge \mathop{\forall\!\!\!\mid} x \in \mathsf{x}.\, fx \in \mathsf{y} \\ &\equiv& \mathsf{cut}\,[y \pm \epsilon].\, \exists \delta > 0.\, \mathop{\forall\!\!\!\mid} x \in [a \pm \delta].\, |fx - y| < \epsilon. \end{array}$$

The last of these is essentially the statement of continuity of $f$ at $a$,

$$a : \mathbb{R},\ \epsilon > 0 \ \vdash\ \exists \delta > 0.\, \forall x \in [a \pm \delta].\, |fx - fa| < \epsilon,$$

which is provable in ASD [J], as of course was our intention.

Now let's substitute this formula into a similar one for $g(b)$.

$$\begin{array}{rcl} g\big(f(a)\big) &=& \mathsf{cut}\,\mathsf{z}.\, \exists \mathsf{y}.\, \mathsf{cut}\,\mathsf{y}'.\, \big(\exists \mathsf{x}.\, a \in \mathsf{x} \wedge \mathop{\forall\!\!\!\mid} x \in \mathsf{x}.\, fx \in \mathsf{y}'\big) \in \mathsf{y} \wedge \mathop{\forall\!\!\!\mid} y \in \mathsf{y}.\, gy \in \mathsf{z} \\ &=& \mathsf{cut}\,\mathsf{z}.\, \big(\exists \mathsf{y}.\, \exists \mathsf{x}.\, a \in \mathsf{x} \wedge \mathop{\forall\!\!\!\mid} x \in \mathsf{x}.\, fx \in \mathsf{y}\big) \wedge \mathop{\forall\!\!\!\mid} y \in \mathsf{y}.\, gy \in \mathsf{z} \end{array}$$

We have used the $\beta$-rule for cuts in the new form, *i.e.* where the bound variable $\mathsf{y}'$ is a rational interval — notice that this swallows part of the context, namely "$\in \mathsf{y}$", treating it in the same way that the $\beta$-rule of the $\lambda$-calculus does the argument of a function.

The corresponding rule for composition of functions in differential calculus is known as the *chain rule*. In our case, we obtain a chain of conjuncts. But notice also the "flow of information" from the argument $a$ to its interval $\mathsf{x}$, to the Moore image $\mathsf{y}$ of $f(\mathsf{x})$, to the Moore image $\mathsf{z}$ of $f(\mathsf{y})$. It goes in this direction because, when we ask for a witness of $\exists\mathsf{y}$, the obvious choice is (slightly wider than) the Moore image $f(\mathsf{x})$.

We have a similar result for a function of two variables,

$$f(a,b) \;=\; \mathsf{cut}\,\mathsf{z}.\,\exists\mathsf{x}\mathsf{y}.\, a \in \mathsf{x} \land b \in \mathsf{y} \land \mathbb{\forall} x \in \mathsf{x}.\, \mathbb{\forall} y \in \mathsf{y}.\, f(x,y) \in \mathsf{z}.$$

If, however, we wish to substitute the same argument in both positions, we have to form the *intersection* of the input intervals:

$$\begin{aligned} f(a,a) &= \mathsf{cut}\,\mathsf{z}.\,\exists\mathsf{x}_1\mathsf{x}_2.\, a \in \mathsf{x}_1 \land a \in \mathsf{x}_2 \land \mathbb{\forall} x \in \mathsf{x}_1.\, \mathbb{\forall} y \in \mathsf{x}_2.\, f(x,y) \in \mathsf{z} \\ &= \mathsf{cut}\,\mathsf{z}.\,\exists\mathsf{x}_1\mathsf{x}_2.\, a \in \mathsf{x}_1 \cap \mathsf{x}_2 \land \mathbb{\forall} x \in \mathsf{x}_1.\, \mathbb{\forall} y \in \mathsf{x}_2.\, f(x,y) \in \mathsf{z} \\ &= \mathsf{cut}\,\mathsf{z}.\,\exists\mathsf{x}.\, a \in \mathsf{x} \land \mathbb{\forall} x \in \mathsf{x}.\, \mathbb{\forall} y \in \mathsf{x}.\, f(x,y) \in \mathsf{z}. \end{aligned}$$

The last equality depends on proving $\Rightarrow$ and $\Leftarrow$ for the logical sub-formulae. For $\Leftarrow$, we just put $\mathsf{x}_1 \equiv \mathsf{x}_2 \equiv \mathsf{x}$. Similarly for $\Rightarrow$, we put $\mathsf{x} \equiv \mathsf{x}_1 \cap \mathsf{x}_2$ and rely on monotonicity of $\mathbb{\forall}$.

Using these transformations, we may simplify complicated arithmetic expressions that involve Dedekind cuts as well as the usual four operations. Moreover, despite the symmetry of the logical connectives, the way in which the cellular variables occur begins to suggest an order of execution. Not surprisingly, this is the usual one, from variables to expressions.

The consequence of this is that the bound variable of the Dedekind cut construction is treated as a receptacle into which we deposit the cellular value that results from the computation. The user may, however, have specified a particular output precision, $\epsilon$. If the cellular value that we obtain is wider than this, we have to pass the demand for greater precision down to our own arguments, and then repeat the computation.

That, however, is still arithmetic. We wanted to do analysis, and have shown that for this we need the quantifiers $\exists x \in \mathbb{R}$ and $\forall x \in [0,1]$. These *bind* the continuous variable $x$, whereas the central proposition itself only treats it as a *free* variable. In fact, the result can easily be adapted to eliminate $x$ when it is bound by an existential quantifier:

$$\begin{aligned} \exists x.\, \phi x &\Leftrightarrow \exists x.\, \exists\mathsf{x}.\, x \in \mathsf{x} \land \mathbb{\forall} x' \in \mathsf{x}.\, \phi x' \\ &\Leftrightarrow \exists\mathsf{x}.\, (\exists x.\, x \in \mathsf{x}) \land \mathbb{\forall} x' \in \mathsf{x}.\, \phi x' \\ &\Leftrightarrow \exists\mathsf{x}.\, \mathbb{\forall} x' \in \mathsf{x}.\, \phi x'. \end{aligned}$$

The sub-formula $\exists x.\, x \in \mathsf{x}$ is redundant (true) because the interval $\mathsf{x}$ is required to be non-trivial. Notice that we have used the logical axioms that

$$\exists u.\, \exists v.\, \psi uv \;\Leftrightarrow\; \exists v.\, \exists u.\, \psi uv \quad \text{and} \quad \exists u.\, \sigma \land \psi u \;\Leftrightarrow\; \sigma \land \exists u.\, \psi u,$$

where the variables $u$ and $v$ may be $n$, $x$ or $\mathsf{x}$.

Observe that, when we bind the variable like this, we lose control of it and the information that flows through it. In particular, in the formula $\exists x.\, \phi x \land \psi x$, we have no idea of whether $\phi$ is the producer and $\psi$ the consumer of this information or *vice versa*. Indeed, they may conduct a lengthy negotiation over it.

We cannot eliminate the universal quantifier quite so easily: as we shall see, it requires the Heine–Borel theorem.

# 7 The Heine–Borel property

For all the careful consideration that we have put into motivating and proving the translation in the previous section, at the symbolic level it is no more than a change of typeface. That is, until we come to $\forall$, which cannot be applied to cellular variables. In this case, we have to provide an *algorithm* that computes $\forall x \in \mathsf{a}$, using $\forall\!\!\!\!|\, x \in \mathsf{a}'$ and recursion. However, whereas we offer here *one* non-trivial way doing something, other people may have *other* ways of doing the same thing more efficiently: the problem is known as *optimisation under constraints*.

Applying the same idea as we used for $\exists$, we need to simplify the expression

$$\forall x \in \mathsf{a}.\, \phi x \;\equiv\; \forall x \in \mathsf{a}.\, \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!\!|\, x' \in \mathsf{x}.\, \phi x'.$$

Classically, this says that, in the closed bounded interval $\mathsf{a}$, each point $x$ lies in some open interval $\mathsf{x}$ for which $\forall\!\!\!\!|\, x' \in \mathsf{x}.\, \phi x'$ holds, so the $\mathsf{x}$s provide an *open cover* of $\mathsf{a}$.

Now, the *Heine–Borel property* says that there is a *finite* sub-cover.

We shall show how this contributes to our translation before we ask what kind of "set" of intervals we mean, or whether Heine–Borel is a theorem or an axiom.

In our simplest application, just one open interval is needed. If the collection of intervals is totally ordered by inclusion, the finite sub-cover has a greatest member. This happens for the predicates $\phi(x, \mathsf{y})$ on intervals that arise in the translation, where we have

$$\mathsf{y}_1 \subset \mathsf{y}_2 \quad \vdash \quad \phi(x, \mathsf{y}_1) \;\Leftarrow\; \phi(x, \mathsf{y}_2),$$

and then $\qquad \forall x \in \mathsf{a}.\, \exists \mathsf{y}.\, y \in \mathsf{y} \wedge \phi(x, \mathsf{y}) \quad \Longleftrightarrow \quad \exists \mathsf{y}.\, y \in \mathsf{y} \wedge \forall x \in \mathsf{a}.\, \phi(x, \mathsf{y}).$

Another common case of this is $\phi_n(x)$ with $\phi_n(x) \Rightarrow \phi_{n+1}(x)$, when

$$\forall x \in \mathsf{a}.\, \exists n.\, \phi_n(x) \quad \Longleftrightarrow \quad \exists n.\, \forall x \in \mathsf{a}.\, \phi_n(x).$$

In analysis, the situations in which we can interchange quantifiers in this way are called *uniform* — continuity, convergence, differentiability. Uniformity is needed for Riemann integration, for integrating or differentiating power series term-by-term, for interchanging integrals, and so on. It is a pity that the textbooks are uniformly so casual in their use of logical formulae, where formal use of the quantifiers would make this issue so much clearer for students. For the proof in ASD that every function is uniformly continuous on any compact domain, see [J].

In general, the sub-formula that $\forall$ encloses is built up using $\wedge$, $\vee$ and $\exists$. Of course, $\forall$ commutes with $\wedge$, whilst we may encode $\vee$ as a special case of $\exists$. Incorporating the problem that arose from the translation, we may then state the Heine–Borel property in the form

$$\forall x \in [0,1].\, \exists n.\, \phi n x \quad \Longleftrightarrow \quad \exists k.\, \bigwedge_{j=0}^{2^k-1} \exists n.\, \forall\!\!\!\!|\, x \in [j \cdot 2^{-k},\ (j+1) \cdot 2^{-k}].\, \phi_n x,$$

using a sufficiently fine binary subdivision of the interval.

How can we read this as a program? We compute

$$\theta[a,b] \;\equiv\; \forall x \in [a,b].\, \exists n.\, \phi_n x$$

recursively in terms of subintervals. The Heine–Borel property says that, when these get smaller than $2^{-k}(b-a)$, we may use $\forall\!\!\!\!|$ instead of deeper recursion. So,

$$\theta[a,b] \quad \Longleftrightarrow \quad \exists k.\ \big(\exists n.\, \forall\!\!\!\!|\, x \in [a,\ a + 2^{-k}(b-a)].\, \phi_n x\big) \ \wedge\ \theta[a + 2^{-k}(b-a),\ b].$$

The idea is to bite off as much of the left-hand end of the interval as we can chew, and then eat the rest in the same way. In each bite, we taste each of the disjuncts on the menu in turn, the point being that the whole meal will consist of several *different* courses.

This procedure raises a number of computational questions that are beyond the scope of this paper.

- We have just interpreted $\wedge$ and $\exists$ in a *sequential* way, whereas in logic they are commutative.
- At what point do we give up on the $n$th disjunct or the $k$th subdivision and try the next one instead?

We would like, notwithstanding the halting problem, to have some kind of *negative* information that can tell us when the current sub-goal is hopeless. Obviously we can't expect the *exact* logical negation, since we can't test for $a \geq b$ or $a = b$, but if we know that $a > b$ then there's no point in trying to prove $a < b$.

We can make a simpler but extremely important short cut if we don't have to subdivide at all, *i.e.* if

$$\forall x \in \mathsf{a}. \phi x \qquad \Longleftrightarrow \qquad \forall\!\!\!\forall x \in \mathsf{a}.\ \phi x.$$

This happens when either

- the interval $\mathsf{a}$ can be made arbitrarily narrow, as was the case for the intervals that arose in the translation, but not when we specify two endpoints; or
- the formula $\phi x$ is sufficiently simple, and in particular the variable $x$ only occurs once in it (in a sense that needs to be formalised).

If we can rely on this short cut, we may use $\forall x \in [a, b].\ \phi x$ as a sub-formula that requires the open subspace $\phi$ to contain an interval $[a, b]$ *connecting* two points that have particular properties. This idiom is crucial to the proof of the intermediate value theorem in [J]. It may be enough to do this when $\phi$ is itself an open interval $(a', b')$, so we want to simplify the quantified sub-formula to $(a' < a) \wedge (b < b')$.

Having gone as far as we can with the computational interpretation, let us tidy up some theoretical loose ends. First, the program appears to make $\forall$ logically redundant. Have we not proved the Heine–Borel *theorem*?

In fact, no. As [I] explains, there are several models of the rest of our calculus (based on domain theory, recursive set theory and other systems) in which the Heine–Borel property fails. Indeed, for exactly this reason, Errett Bishop developed constructive analysis using totally bounded closed subspaces, without ever mentioning compactness in the Heine–Borel sense [Bis67].

The impact of this on our program is that we use the Heine–Borel property to prove *termination*, *i.e.* that the $\exists k$ side really is observable when the $\forall x$ side is.

In ASD, therefore, this property must be an *axiom* for the real line, if we consider this as a base type. If instead we *construct* $\mathbb{R}$ *à la* Dedekind from $\mathbb{Q}$, the property is an instance of a more fundamental axiom of ASD, called the monadicity principle [I].

We also need to explain what we mean by "sets" in situations such as an open cover and its finite sub-cover, since we are not talking about open or compact subspaces of $\mathbb{R}$. As always in computation, we may resort to numerical encodings, using $\mathbb{N}$ and its RE subsets for our "sets". A less clumsy way of doing the same thing, which is standard in functional and logic programming, is to replace numbers and their successors by binary *trees* and pairing. Then we hang a list of trees along the hereditarily rightmost branch of a tree.

But ASD is a theory of general topology, and one of its aims is to eliminate *ad hoc* codings such as this. These "sets" are *topological spaces* of a special kind, namely those that are *discrete* (having $=$) and *overt* (with $\exists$). It can be shown that these form a model of the *naïve set theory* that we teach to undergraduates, including a *finite* powerset. Specifically, they admit products, disjoint sums, equalisers (subsets defined by equations), quotients by equivalence relations [C] and free commutative monoids (sets of lists) [E]. This structure was introduced by André Joyal in 1973 as a way of proving Gödel's incompleteness theorems categorically, but unfortunately neither this work nor much else on the subject has ever actually appeared in print.

Finally, how do I justify the claim that any computably continuous function is representable in my calculus?

On the face of it, you are allowed to choose any of the thousands of extant programming languages as your notion of "computability". Indeed you can, if it has the right features, and omits the wrong ones: its *denotational semantics* provides a translation of your program into *domain theory*, which may in turn be formulated in ASD. For example, Gordon Plotkin considered a simply typed $\lambda$-calculus with conditional execution, recursion and a specified order of evaluation, which may be extended with "parallel or" and a sort of existential quantifier [Plo77]. The domain-theoretic interpretation of a program of type `nat` is a continuous function into the space $\mathbb{N}_\perp$, which is like the Sierpiński space $\Sigma$ (with a closed point $\perp$), except that it has an open point for each integer. Plotkin showed that the program terminates iff its *denotation* is a number, and not $\perp$.

Since the same theory can be set up in ASD, we may understand "computable functions" to be terms of the ASD calculus (with all of its types, not just $\mathbb{N}$ and $\mathbb{R}$). Now suppose that you have a continuous function $f : \mathbb{R} \to \mathbb{R}$ that is defined in whatever mathematical world you prefer (maybe the classical one), along with a program $P(d, q, u)$ in your choice of programming language that accepts rational numbers $d, q, u$ as arguments and terminates iff $d < f(q) < u$. Let $\phi : \mathbb{Q}^3 \to \Sigma$ be the denotational semantics of $P$ in ASD. Now, if you've given me consistent data, this $\phi$ may be used to define a morphism $g : \mathbb{R} \to \mathbb{R}$ in the full ASD calculus, whose interpretation in your mathematical world agrees with $f$ [G]. The final piece in the jig-saw is that the structure that we described in Section 2 has exactly one model in ASD, up to unique isomorphism [I].

# 8 Proof of the central proposition

This final section is an "appendix" that, in a strictly logical development, would be inserted in Section 6. We prove the *central proposition*, that

$$\phi x \qquad \Longleftrightarrow \qquad \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \mathbb{\forall} x \in \mathsf{x}.\, \phi x,$$

by structural induction on $\phi$, starting from the leaves, *i.e.* occurrences of $x$. If you prefer, this is the same as induction on the depth of the expression tree for $\phi$. We have to consider each of the connectives in the table in Section 2.

Since $\mathbb{\forall}$ leaves the logical operations untouched, it is not surprising that the most difficult cases are those involving the *arithmetical* connectives. The key property is known as *roundedness* or *interpolation*. It should be proved for each Moore operation $\circledast \equiv \oplus, \ominus, \otimes, \oslash$ (corresponding to the ordinary $\star \equiv +, -, \times, \div$) in turn, although we shall take it as known. It says that, for intervals $\mathsf{a}$, $\mathsf{b}$ and $\mathsf{w}$ (that we may assume to have machine-representable endpoints),

$$\mathsf{a} \circledast \mathsf{b} \Subset \mathsf{w} \qquad \Longleftrightarrow \qquad \exists \mathsf{yz}.\, \mathsf{a} \Subset \mathsf{y} \wedge \mathsf{b} \Subset \mathsf{z} \wedge \mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w}.$$

It will be convenient to include the case where $\mathsf{a}$ and $\mathsf{b}$ are singletons, *i.e.*

$$a \star b \in \mathsf{w} \qquad \Longleftrightarrow \qquad \exists \mathsf{yz}.\, a \in \mathsf{y} \wedge b \in \mathsf{z} \wedge \mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w},$$

and also that where $\mathsf{w}$ is infinite.

In fact, this lemma is the concrete manifestation of the continuous lattices with the Scott topology [GHK+80] that underly the theory of (non-Hausdorff) locally compact spaces, and hence (the classical interpretation of) ASD.

The induction step for each arithmetic operation $\star$ in an expression proceeds from the induction hypothesis for $a \in \mathsf{y}$ and $b \in \mathsf{z}$ to the result for $a \star b \in \mathsf{w}$. The hypothesis for $a \in \mathsf{y}$ and the corresponding cases in the definition of $\mathbb{\forall}$ give

$$\begin{aligned} a \in \mathsf{y} \quad &\Leftrightarrow \quad \exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge \mathbb{\forall} x \in \mathsf{x}_1.\, (a \in \mathsf{y}) \\ &\Leftrightarrow \quad \exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge (\mathbb{\forall} x \in \mathsf{x}_1.\, a) \Subset \mathsf{y} \\ &\equiv \quad \exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge \mathsf{a}_1 \Subset \mathsf{y}, \end{aligned}$$

where $\mathsf{a}_1$ is a convenient abbreviation for $\mathbb{\forall} x \in \mathsf{x}_1.\, a$, the subscript being inherited from that on $\mathsf{x}_1$. Notice that if $\mathsf{x} \subset \mathsf{x}_1$ then $\mathsf{a} \supset \mathsf{a}_1$.

Putting all of these lemmas together, along with those for manipulating $\exists$ and $\wedge$, we have

$$a \star b \in \mathsf{w} \quad \Leftrightarrow \quad \exists \mathsf{yz}.\, a \in \mathsf{y} \wedge b \in \mathsf{z} \wedge \mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w}$$

$$
\begin{aligned}
&\Leftrightarrow\quad \exists \mathsf{yz}.\,(\exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge \mathsf{a}_1 \Subset \mathsf{y}) \\
&\qquad\qquad \wedge (\exists \mathsf{x}_2.\, x \in \mathsf{x}_2 \wedge \mathsf{b}_2 \Subset \mathsf{z}) \wedge (\mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w}) \\
&\Leftrightarrow\quad \exists \mathsf{x}_1\mathsf{x}_2.\, x \in \mathsf{x}_1 \cap \mathsf{x}_2 \\
&\qquad\qquad \wedge \exists \mathsf{yz}.\,(\mathsf{a}_1 \Subset \mathsf{y}) \wedge (\mathsf{b}_2 \Subset \mathsf{z}) \wedge (\mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w}) \\
&\Leftrightarrow\quad \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \exists \mathsf{yz}.\,(\mathsf{a} \Subset \mathsf{y}) \wedge (\mathsf{b} \Subset \mathsf{z}) \wedge (\mathsf{y} \circledast \mathsf{z} \Subset \mathsf{w}) && \text{monot} \\
&\Leftrightarrow\quad \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \mathsf{a} \circledast \mathsf{b} \Subset \mathsf{w}.
\end{aligned}
$$

The bases cases of the induction for arithmetic expressions are of course the constants (0 and 1) and the variables ($x$ and some other $y$). The cases $c \equiv 0, 1, y$ are all the same:

$$c \in \mathsf{w} \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge c \in \mathsf{w} \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, c \in \mathsf{w},$$

whilst

$$x \in \mathsf{w} \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \mathsf{x} \Subset \mathsf{w} \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, x \in \mathsf{w},$$

where $\underline{\mathsf{w}} < \underline{\mathsf{x}} < x < \overline{\mathsf{x}} < \overline{\mathsf{w}}$.

This establishes the inductive claim when $\phi x$ is of the form $a \in \mathsf{w}$, if $a$ is a purely arithmetic expression. We shall consider recursion and Dedekind cuts after we've dealt with logical formulae. The trick of allowing $\mathsf{w} \equiv [-\infty, b]$ above now lets us deduce the result for $<$ from that for $\in$,

$$a < b \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\,(a < b) \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge (\forall\!\!\!|\, x \in \mathsf{x}.\, a) \olessthan (\forall\!\!\!|\, x \in \mathsf{x}.\, b),$$

and likewise for $>$ and $\neq$.

The induction cases for the logical connectives are easy and similar. For each of them, the three lines of argument below apply the induction hypothesis to the subformulae $\phi x$ and $\psi x$, the logical rules for $\exists$, $\wedge$ and $\vee$, and the relevant cases in the definition of $\forall\!\!\!|$. For the constants, any non-trivial interval will do for $\mathsf{x}$. The one tricky case is $\wedge$, where we need the intersection $\mathsf{x}_1 \cap \mathsf{x}_2$.

$$
\begin{array}{lcll}
\top & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, \top & \\
\bot & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, \bot & \\
\phi x \wedge \psi x & \Leftrightarrow & (\exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge \forall\!\!\!|\, x \in \mathsf{x}_1.\, \phi x) \wedge (\exists \mathsf{x}_2.\, x \in \mathsf{x}_2 \wedge \forall\!\!\!|\, x \in \mathsf{x}_2.\, \psi x) & \\
 & \Leftrightarrow & \exists \mathsf{x}_1\mathsf{x}_2.\, x \in \mathsf{x}_1 \cap \mathsf{x}_2 \wedge (\forall\!\!\!|\, x \in \mathsf{x}_1.\, \phi x) \wedge (\forall\!\!\!|\, x \in \mathsf{x}_2.\, \psi x) & \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge (\forall\!\!\!|\, x \in \mathsf{x}.\, \phi x) \wedge (\forall\!\!\!|\, x \in \mathsf{x}.\, \psi x) & \text{monot} \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}_1.\,(\phi x \wedge \psi x) & \\
\phi x \vee \psi x & \Leftrightarrow & (\exists \mathsf{x}_1.\, x \in \mathsf{x}_1 \wedge \forall\!\!\!|\, x \in \mathsf{x}_1.\, \phi x) \vee (\exists \mathsf{x}_2.\, x \in \mathsf{x}_2 \wedge \forall\!\!\!|\, x \in \mathsf{x}_2.\, \psi x) & \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \big((\forall\!\!\!|\, x \in \mathsf{x}_1.\, \phi x) \vee (\forall\!\!\!|\, x \in \mathsf{x}_2.\, \psi x)\big) & \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}_1.\,(\phi x \vee \psi x) & \\
\exists u.\, \phi u x & \Leftrightarrow & \exists u.\, \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, \phi u x & \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \exists u.\, \forall\!\!\!|\, x \in \mathsf{x}.\, \phi u x & \\
 & \Leftrightarrow & \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, \exists u.\, \phi u x &
\end{array}
$$

We assume that $\forall$ has been eliminated from $\phi$ in favour of the recursive program that we gave in the previous section.

In general, we don't just have logical formulae with arithmetical sub-formulae, but also form new real numbers by specifying them as Dedekind cuts. This means that we have another arithmetical case,

$$(\mathsf{cut}\,\mathsf{y}.\,\theta x \mathsf{y}) \in \mathsf{w} \;\Leftrightarrow\; \theta x \mathsf{w} \;\Leftrightarrow\; \exists \mathsf{x}.\, x \in \mathsf{x} \wedge \forall\!\!\!|\, x \in \mathsf{x}.\, \theta x \mathsf{w},$$

using the cut-$\beta$ rule and the induction hypothesis. This case of the induction is then valid so long as we define

$$(\forall\!\!\!|\, x \in \mathsf{x}.\, \mathsf{cut}\,\mathsf{y}.\,\theta x \mathsf{y}) \Subset \mathsf{w} \;\equiv\; \forall\!\!\!|\, x \in \mathsf{x}.\,\big((\mathsf{cut}\,\mathsf{y}.\,\theta x \mathsf{y}) \in \mathsf{w}\big) \;\equiv\; \forall\!\!\!|\, x \in \mathsf{x}.\, \theta x \mathsf{w}.$$

There are further cases for recursively defined arithmetical and logical expressions. These are similar to the arguments in [E, §2].

## References


[Bis67] Errett Bishop. *Foundations of Constructive Analysis*. Higher Mathematics. McGraw–Hill, 1967.

[CS88] John Horton Conway and Neil Sloane. *Sphere Packings, Lattices and Groups*. Number 290 in Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, 1988.

[Dau79] Joseph Warren Dauben. *Georg Cantor: his Mathematics and Philosophy of the Infinite*. Harvard University Press, 1979.

[Ded72] Richard Dedekind. *Stetigkeit und irrationale Zahlen*. Braunschweig, 1872. Reprinted in *Gesammelte mathematische Werke*, pp. 315–334 and (in English) in *Essays on the theory of numbers*, Dover, 1963.

[Dij87] E. J. Dijksterhuis. *Archimedes*. Princeton University Press, 1987. Written in Dutch in 1938–44; translated by C. Dikshoorn.

[EL04] Abbas Edalat and André Lieutier. Domain theory and differential calculus. *Mathematical Structures in Computer Science*, 14:771–802, 2004.

[Fef77] Solomon Feferman. Categorical foundations and foundations of category theory. In Robert Butts and Jaakko Hintikka, editors, *Logic, Foundations of Mathematics and Computability Theory*, pages 149–169. Reidel, 1977.

[Gen35] Gerhard Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935. English translation in M. E. Szabo, *The Collected Papers of Gerhard Gentzen*, North-Holland, 1969, pp. 68–131.

[GHK+80] Gerhard Gierz, Karl Heinrich Hoffmann, Klaus Keimel, Jimmie Lawson, Michael Mislove, and Dana Scott. *A Compendium of Continuous Lattices*. Springer-Verlag, 1980. Second edition, *Continuous Lattices and Domains*, published by Cambridge University Press, 2003.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[Pea97] Giuseppe Peano. Studii di logica matematica. *Atti della Reale Accademia di Torino*, 32:565–583, 1897. Reprinted in Peano, *Opere Scelte*, Cremonese, 1953, vol. 2, pp. 201–217, and (in English) in H.C. Kennedy, *Selected Works of Giuseppe Peano*, Toronto University Press, 1973, pp 190–205.

[Plo77] Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[RW13] Bertrand Russell and Alfred North Whitehead. *Principia Mathematica*. Cambridge University Press, 1910–13. Second edition, 1927; paperback edition to ∗56, 1962.

[Sco70] Dana Scott. Outline of a mathematical theory of computation. In *4th Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970. Superseded by Oxford technical report PRG-2.

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Number 59 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.

[Tur35] Alan Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (2)*, 42:230–265, 1935.

[TvD88] Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in Mathematics, an Introduction*. Number 121 and 123 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1988.

[Vic88] Steven Vickers. *Topology Via Logic*, volume 5 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1988.

The papers on abstract Stone duality may be obtained from

`www.cs.man.ac.uk/~pt/ASD`


[A] Paul Taylor, Sober spaces and continuations. *Theory and Applications of Categories*, 10(12):248–299, 2002.

[B] Paul Taylor, Subspaces in abstract Stone duality. *Theory and Applications of Categories*, 10(13):300–366, 2002.

[C] Paul Taylor, Geometric and higher order logic using abstract Stone duality. *Theory and Applications of Categories*, 7(15):284–338, 2000.

[D] Paul Taylor, Non-Artin gluing in recursion theory and lifting in abstract Stone duality. 2000.

[E] Paul Taylor, Inside every model of Abstract Stone Duality lies an Arithmetic Universe. *Electronic Notes in Theoretical Computer Science* **416**, Elsevier, 2005.

[F] Paul Taylor, Scott domains in abstract Stone duality. March 2002.

[G–] Paul Taylor, Local compactness and the Baire category theorem in abstract Stone duality. *Electronic Notes in Theoretical Computer Science* **69**, Elsevier, 2003.

[G] Paul Taylor, Computably based locally compact spaces. *Logical Methods in Computer Science*, 2005, to appear.

[H-] Paul Taylor, An elementary theory of the category of locally compact locales. APPSEM Workshop, Nottingham, March 2003.

[H] Paul Taylor, An elementary theory of various categories of spaces and locales. November 2004.

[I] Andrej Bauer and Paul Taylor, The Dedekind reals in abstract Stone duality. Computability and Complexity in Analysis, Kyoto, August 2005.

[J] Paul Taylor, A $\lambda$-calculus for real analysis. Computability and Complexity in Analysis, Kyoto, August 2005.