Mathematical foundations of functional programming with real numbers

Martín Escardó   MGS 2003   (written 27-28 Mar)
University of Birmingham   Midlands Graduate School
(corrected 15th Apr)
(& 16th Apr)

1. Introduction

We consider (exact) computations with real numbers.

We use domain theory and in particular the interval domain.

References:

(i) T. Streicher. Mathematical foundations of functional programming. Darmstadt University. Maths dept. 2002, 2003.
http://www.mathematik.tu-darmstadt.de/~streicher

(ii) M. H. Escardó http://www.cs.bham.ac.uk/~mhe

(a) Introduction to exact numerical computation. Notes for a tutorial course at ISSAC 2000.

(b) PCF extended with real numbers: a domain-theoretic approach to higher-order exact real number computation. PhD thesis, Imperial College, 1996.

(c) PCF extended with real numbers. Theoretical Computer Science, vol 162, pp. 79-115, 1996.

(iii) M.H Escardó & T. Streicher. Induction and recursion on the partial real line with applications to Real PCF. Theoretical Computer Science, vol 210 n.1, pp. 121-157, 1999.

We present an extension of the prototipical functional programming language PCF (see reference (i)) with a base type for real numbers, interpreted as the interval domain discussed in Section 2. Several such extensions have been considered, see e.g. references (ii)(b) and (ii)(c).

The extension considered here is new and presented for the first time in these notes. It represents a substantial simplification, from both intuitive and mathematical points of view, of earlier extensions.

From the point of view of these notes, the interval domain and more generally domain theory constitute mathematical tools for the "denotational" semantics of functional programming with real numbers.

From another point of view, not explored in these notes, the interval domain and domain theory provide an "algorithmic framework" for analysis, orthogonal to functional programming with the reals. The reader is referred to

(iv) Edalat's work available from his web page at Imperial College, London.

We start with the interval domain as a model of real-number computation, then move to recursive definitions of real functions, considering the bisection algorithm for root finding as a paradigmatic example. After this, we introduce the extension of the language PCF with real number alluded above. We finish by briefly discussing its expres

Historical disclaimer No historical notes are given in these notes. See the references for plenty of more references and a bit of history.

2. The interval domain

$I\mathbb{R}$ = non-empty, closed and bounded real intervals.

$$x = [\underline{x}, \overline{x}] = \{ r \in \mathbb{R} \mid \underline{x} \leq r \leq \overline{x} \}$$

$$\underline{x} = \inf x \qquad \overline{x} = \sup x$$

$$x \sqsubseteq y \quad \text{iff} \quad x \supseteq y$$
$$\text{iff} \quad \underline{x} \leq \underline{y} \;\&\; \overline{y} \leq \overline{x}$$

Is a continuous domain: (see below)

$$x \ll y \quad \text{iff} \quad x^{\circ} \supseteq y$$
$$\text{iff} \quad \underline{x} < \underline{y} \;\&\; \overline{y} < \overline{x}$$

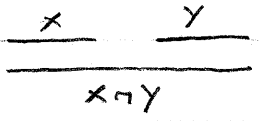
Directed and non-empty upper bounded joins exist:

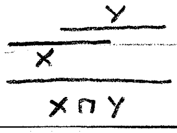
$$\bigsqcup_i x_i = \bigcap_i x_i$$
$$= \left[\sup_i \underline{x}_i \,,\, \inf_i \overline{x}_i \right]$$

Non-empty meets exist: (lower bounded only, because we don't have $\bot$ yet.)

$$\bigsqcap_i x_i = \text{smallest interval containing } \bigcup_i x_i$$
$$= \left[\inf_i \underline{x}_i \,,\, \sup_i \overline{x}_i \right]$$

If x and y are upper bounded, then $x \sqcap y = x \cup y$.





Maximal elements

$[x, x] = \{x\}$

"Total real numbers".

other elements

"Partial real numbers".

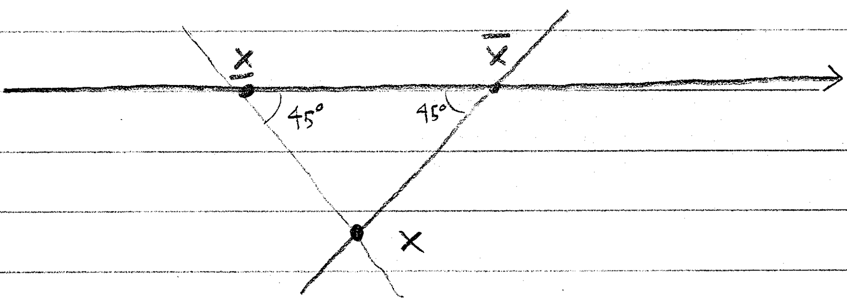
$$\begin{tikzcd} \mathbb{R} \arrow[r, hook] & I\mathbb{R} \\ x \arrow[r, mapsto] & \{x\} \end{tikzcd}$$

If you know topology:

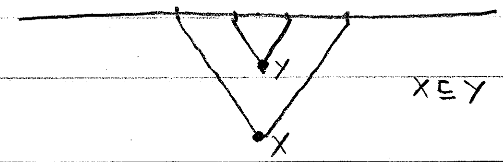
Endowing $\mathbb{R}$ with its usual (Euclidean) topology and $I\mathbb{R}$ with the Scott topology, this becomes a topological embedding.

This observation allows many proofs to be considerably simplified.

Graphical representation in the plane:

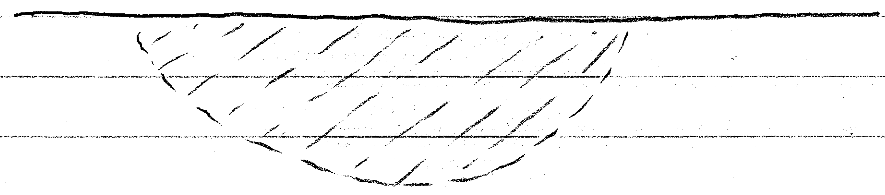


Bijection between intervals in the x-axis and points in the plane below the x-axis. Order becomes containment of triangles:

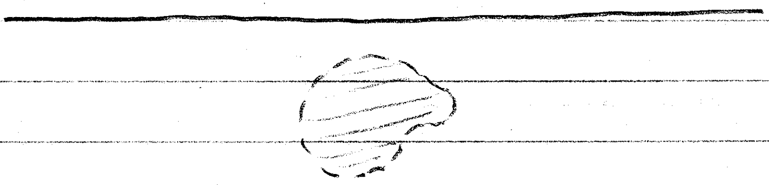


If you know topology (or just real analysis in this case actually):

Scott-open set = open set of the Euclidean half plane which is also upper



Lawson-open set = open set of the Euclidean half plane with no restriction.



Recall from (or learn later in) the domain-theory course:

A function $f: D \to E$ of dcpos is Scott continuous iff it is monotone and $f(\bigsqcup_i x_i) = \bigsqcup_i f(x_i)$

for every directed family $\{x_i\}$.

<u>Lemma</u> (to be used much later)
If the dcpos D and E are continuous then the function $f: D \to E$ is Scott continuous iff it is monotone and

$$y \ll f(x) \text{ implies } y \leq f(x') \text{ for some } x' \ll x.$$

"Finitary parts of the output <u>depend only</u> on finitary parts of the input."

This remark is explained as follows.

Recall (or learn later) that, by definition:

(1) A set $S \subseteq D$ is directed iff it is non-empty and for every x and y in S there is z in S with $x \sqsubseteq z$ and $y \sqsubseteq z$.

(2) $x \ll y$ iff for every directed set $S \subseteq D$ with $y \sqsubseteq \bigsqcup S$ there is some $s \in S$ with $x \sqsubseteq s$.

Think of S as an "abstract" computation (of its join). Its elements are the partial outputs which approximate the final result. The defining condition says that any two partial outputs eventually get superseded by a third finer output. (C.f. also the Church-Rosser property in the λ-calculus course.)

With this, $x \ll y$ can be interpreted as saying that any computation of y or something better than y eventually outputs x or something better than x. In other words, x is an unavoidable step in any computation of y (or something better than y).

The continuity axiom for a continuous dcpo says that the unavoidable parts are not only unavoidable but also enough, and, moreover, they form an abstract computation:

$$x = \bigsqcup^{\uparrow} \{ b \mid b \ll x \}.$$

The set $\{ b \mid b \ll x \}$ of unavoidable steps in computations of x is a computation of x.

For more details, you are referred to the domain-theory course. We now return to continuous functions.

We say that a function $g : \mathrm{I}\mathbb{R} \longrightarrow \mathrm{I}\mathbb{R}$ is <u>total</u> if it maps total numbers to total numbers. This is the case iff there is a (necessarily unique) function $f : \mathbb{R} \longrightarrow \mathbb{R}$ s.t.

$$\begin{tikzcd}
\mathbb{R} \arrow[r, "f"] \arrow[d, hook] & \mathbb{R} \arrow[d, hook] \\
\mathrm{I}\mathbb{R} \arrow[r, "g"'] & \mathrm{I}\mathbb{R},
\end{tikzcd}$$

where the down arrows are the singleton embedding $x \longmapsto \{x\}$ discussed above.

<u>Proposition</u> In this case, if g is Scott continuous then f is continuous (in the usual sense of real analysis).

<u>Proof</u> A topological proof simply observes that the singleton map is a topological embedding. A more direct proof is not hard and is left as an exercise. $\square$

conversely,

<u>Proposition</u> For any given continuous map $f : \mathbb{R} \longrightarrow \mathbb{R}$ there is at least one Scott continuous map $g : \mathrm{I}\mathbb{R} \longrightarrow \mathrm{I}\mathbb{R}$ extending f in the sense of the above diagram.

<u>Construction</u> (The canonical (or maximal) extension)
Let $g(x)$ be the direct image of the interval x via f.
Then g is well defined because continuous functions map (connected sets to connected sets and compact sets to compact sets and hence) closed and bounded intervals to closed and bounded intervals.
And, as it easy to check, direct-image formation always preserves directed joins. (In this last step, continuity of f is not used.) $\square$

To see that there are actually plenty of continuous extensions for a continuous function besides the canonical one, consider the following

<u>Construction</u> Let $\varepsilon > 0$. Define $id_\varepsilon : I\mathbb{R} \longrightarrow I\mathbb{R}$ by

$$id_\varepsilon(x) = [\underline{x} - \varepsilon . dx \ , \ \overline{x} + \varepsilon . dx]$$

where

$$dx = \overline{x} - \underline{x} .$$

Then (exercise!) id_ε is continuous and extends the identity function of $\mathbb{R}$. Hence for any continuous extension $g : I\mathbb{R} \longrightarrow I\mathbb{R}$ of a continuous function $f : \mathbb{R} \longrightarrow \mathbb{R}$, the composites $g \circ id_\varepsilon$ and $id_\varepsilon \circ g$ are also extensions of f. □

<u>Exercise</u> Order maps $I\mathbb{R} \longrightarrow I\mathbb{R}$ "pointwise" i.e $g \sqsubseteq h$ iff for all $x \in I\mathbb{R}$, $g(x) \sqsubseteq h(x)$. Show that the canonical extension is the greatest monotone extension and hence the greatest continuous extension.

<u>Proposition</u> Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a continuous map and $g : I\mathbb{R} \rightarrow I\mathbb{R}$ be its canonical extension.

(i) If f is increasing then $g(x) = [f(\underline{x}), f(\overline{x})]$.

(ii) If f is decreasing then $g(x) = [f(\overline{x}), f(\underline{x})]$.

In general, however, it is very difficult to calculate the canonical extension. (Try a polynomial of degree 3 or higher.)

Functions of several variables (sketched)

For notational simplicity, we consider two variables, leaving the general case to the reader's imagination.

$$\begin{tikzcd} \mathbb{R} \times \mathbb{R} \arrow[r, "f"] \arrow[d, hook] & \mathbb{R} \arrow[d, hook] \\ I\mathbb{R} \times I\mathbb{R} \arrow[r, "g"'] & I\mathbb{R} \end{tikzcd}$$

If g is total in the evident sense and it is continuous, then its restriction f is continuous.

Conversely, any continuous f has a canonical extension g given by

$$g(x,y) = \{ f(r,s) \mid r \in x \text{ and } s \in y \}.$$

The reason this is well-defined is again the fact that rectangles $x \times y \subseteq \mathbb{R} \times \mathbb{R}$ are compact and connected and that continuous maps preserve such types of sets, which in $\mathbb{R}$ are the closed and bounded intervals.

This gives, in particular, canonical extensions of the binary operations $+$, $-$ and $\times$. Addition and subtraction are given by

$$x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}],$$
$$x - y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}].$$

Multiplication is more complicated (boring exercise, not recommended).

Adding a bottom element to the interval domain

There are two reasons for adding an artificial least element $\bot$ (called bottom) to $I\mathbb{R}$, which are not entirely unrelated:

(1) Coping with singularities.

(2) Solving functional equations called "recursive definitions".

Such recursive definitions often give rise to functions with singularities, which in computational terms manifest themselves as infinitely long (temporally speaking) computations that never output.

Concretely, it is convenient to take $\bot = (-\infty, \infty)$ so that the order is still reverse inclusion of sets. Then $\underline{\bot} = -\infty$ and $\overline{\bot} = \infty$. However, $\bot$ often requires a special treatment, unfortunately.

N.B. Possible solution: work with $I[-\infty, \infty]$!

The partial real line

$$\mathcal{R} = (I\mathbb{R})_\bot$$

Lemma $\mathcal{R}$ is a countably based continuous Scott domain.
The intervals with rational endpoints form a countable basis.

Proposition For every dense set $A \subseteq \mathbb{R}$ and every continuous map $f : A \to \mathbb{R}$ there is a Scott continuous extension $g : \mathcal{R} \to \mathcal{R}$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, hook] & \mathbb{R} \arrow[dd, hook] \\
\mathbb{R} \arrow[d, hook] & \\
\mathcal{R} \arrow[r, "g"'] & \mathcal{R}
\end{tikzcd}$$

Proof The continuous Scott domains under the Scott topology are injective topological spaces over dense subspace embeddings. See Gierz et al. Continuous lattices and domains, CUP, 2003. Henceforth referred to as the "new compendium". $\square$

In practice, we'll explicitly construct g for $\overbrace{\text{given } f}^{\text{in a case by case fashion}}$. Here is our first example.

Exercise. Let $f : \mathbb{R}\setminus\{0\} \longrightarrow \mathbb{R}$ be given by $f(x) = 1/x$. Define $g : \mathcal{R} \longrightarrow \mathcal{R}$ by

$$g(x) = \begin{cases} \bot & \text{if } 0 \in x \quad \text{(this includes the case } x=\bot) \\ [1/\underline{x}, 1/\overline{x}] & \text{if } (x \neq \bot \text{ and}) \quad \overline{x} < 0 \\ [1/\overline{x}, 1/\underline{x}] & \text{if } (x \neq \bot \text{ and}) \quad 0 < \underline{x} \end{cases}$$

Then g is a Scott continuous extension of the continuous function f. □

The next four examples should clarify the role of bottom values and arguments.

Exercise Let $f : \mathbb{R}\setminus\{0\} \longrightarrow \mathbb{R}$ be given by $f(x) = -1$ for $x<0$ and $f(x) = 1$ for $x>0$. Then $g,h : \mathcal{R} \to \mathcal{R}$ defined by

$$g(x) = \begin{cases} \{-1\} & \text{if } \overline{x} < 0 \\ [-1,1] & \text{if } 0 \in x \neq \bot \\ \{1\} & \text{if } 0 < \underline{x} \\ \bot & \text{if } x = \bot \end{cases}$$

Actually, we can have $[-1,1]$ even if $x=\bot$

and

$$h(x) = \begin{cases} \{1\} & \text{if } \overline{x} < 0 \\ \bot & \text{if } 0 \in x \\ \{1\} & \text{if } 0 < \underline{x} \end{cases}$$

are Scott continuous extensions of the continuous function f. □

Exercise
$$f : \mathbb{R}\setminus\{0\} \longrightarrow \mathbb{R}, \quad x \longmapsto \sin(1/x)$$

$$g : \mathcal{R} \longrightarrow \mathcal{R}, \quad x \longmapsto \begin{cases} \bot & \text{if } x = \bot \\ [-1,1] & \text{if } 0 \in x \neq \bot \\ f[x] & \text{otherwise.} \end{cases}$$

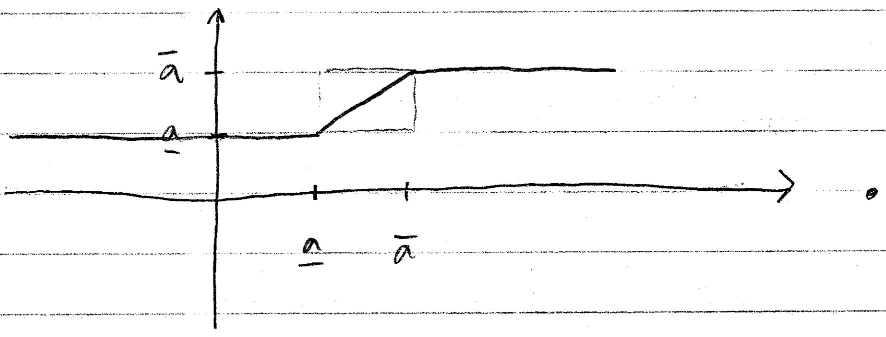
□

Exercise (Important later) Let $a \in \mathbb{IR}$ and define $f : \mathbb{R} \to \mathbb{R}$ by

$$f(x) = \max(\underline{a}, \min(x, \overline{a}))$$

$$= \min(\max(\underline{a}, x), \overline{a}))$$

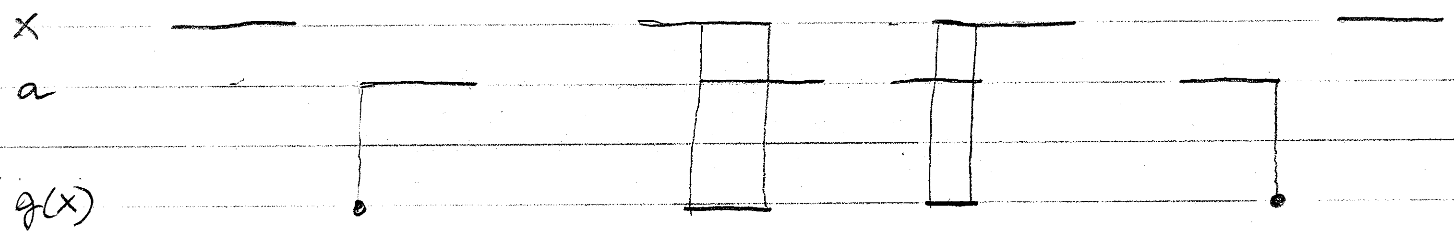
$$= \begin{cases} \underline{a} & \text{if } x \le \underline{a}, \\ x & \text{if } \underline{a} \le x \le \overline{a}, \\ \overline{a} & \text{if } \overline{a} \le x \end{cases}$$

$=$ 

Then f has a greatest continuous extension $g : \mathcal{R} \to \mathcal{R}$ given by

$$g(x) = \begin{cases} \{\underline{a}\} & \text{if } \overline{x} \le \underline{a}, \\ a \sqcup x & \text{if } a \text{ and } x \text{ are bounded above}^{*}, \\ \{\overline{a}\} & \text{if } \overline{a} \le \underline{x}. \end{cases}$$

The three cases are geometrically four:



Notice that $g(\perp) = a$ rather than $\perp$. It is this fact that will be important. $\square$

* I.e. overlap as intervals — recall that $a \sqcup x = a \cap x$.

__Exercise__ Ponder about continuous extensions $g : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ of continuous functions $f : A \to \mathcal{R}$ with $A \subseteq \mathbb{R} \times \mathbb{R}$ dense.

We finish this section by remarking that the canonical extension is hard to achieve in practice.

Consider $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = x^2$. The canonical extension $g : I\mathbb{R} \to I\mathbb{R}$ doesn't coincide with the extension $h : I\mathbb{R} \to I\mathbb{R}$ obtained from the canonical extension $m : I\mathbb{R} \times I\mathbb{R} \to I\mathbb{R}$ of multiplication by $h(x) = m(x, x)$. In fact

$$\begin{aligned} g([-1,1]) &= \{ r^2 \mid r \in [-1,1] \} \\ &= [0,1] \\ &\neq [-1,1] \\ &= \{ rs \mid r, s \in [-1,1] \} \\ &= h([-1,1]) \end{aligned}$$

In fact, more generally, the canonical extension of a polynomial is not given by replacing real operations by interval operations. This phenomenon is well-known in the interval-analysis literature and is a source of many difficulties.

__Exercise__ Let $g : \mathcal{R} \to \mathcal{R}$ be a total Scott continuous and define $\bar{g} : \mathcal{R} \to \mathcal{R}$ by

$$\bar{g}(x) = \bigsqcup \{ \textstyle\prod_{s \in S} g(s) \mid S \subseteq I\mathbb{R} \text{ is finite and } \sqcap S \sqsubseteq x \}.$$

Show that $\bar{g}$ is Scott continuous and agrees with g at total numbers and that $\bar{g}$ is the greatest continuous function pointwise above g. $\square$

3. Recursive definitions

We consider the bisection algorithm as an example — for more examples see references (iii) or (ii)(b).

Let $a, b \in \mathbb{R}$ with $a \leq b$ and $f : \mathbb{R} \to \mathbb{R}$ be a strictly increasing continuous function with $f(a) \leq 0 \leq f(b)$. By the intermediate-value theorem, there is a unique $x \in [a, b]$ with $f(x) = 0$. The following well-known imperative algorithm computes it. More precisely

$$\{x\} = \bigsqcup_i^{\uparrow} [a_i, b_i] = \bigcap_i [a_i, b_i].$$

```
bisect (f, a, b)
{
    $a_0 = a$ , $b_0 = b$;
    for every integer $i \geq 0$, in ascending order, do
        {
            $c = (a_i + b_i)/2$;          // midpoint

            if  $f(c) < 0$
                then  $a_{i+1} = c$;  $b_{i+1} = b_i$;
                else  $a_{i+1} = a_i$  $b_{i+1} = c$;

            output $[a_i, b_i]$;
        }
}
```

We emphasize that this algorithm computes a root of f in the interval $[a, b]$ under the above assumptions on a, b and f. If they are not met, I don't know (and don't care) what the algorithm does. Notice that, under the assumptions, the algorithm converges very quickly to the root (although, under additional assumptions, there are faster algorithms).

Our next goal is to transform the above imperative algorithm into a functional one, using a so-called "recursive definition". Mathematically, we will have an equation with a functional variable, and we have to solve the equation to see what function the equation "recursively" defines, if any.

We'll proceed in a series of attempts, until we eventually get what we want. Each attempt introduces and analyses a new concept.

Let $(\mathbb{R} \to \mathbb{R})$ denote the set of continuous maps $\mathbb{R} \to \mathbb{R}$. The objective is to recursively define a partial function

$$\text{bisect} : (\mathbb{R} \to \mathbb{R}) \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$$

such that if $f \in (\mathbb{R} \to \mathbb{R})$ is strictly increasing in the interval $[a,b]$ and $f(a) \leq 0 \leq f(b)$ then

(i) $\text{bisect}(f,a,b) \in [a,b]$,

(ii) $f(\text{bisect}(f,a,b)) = 0$, and

(iii) if $f(x) = 0$ for $x \in [a,b]$ then $x = \text{bisect}(f,a,b)$.

We use the variable F to range over partial* functions $(\mathbb{R} \to \mathbb{R}) \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. We write an equation for F and hope that the solutions are functions bisect satisfying the above specification.

$$\begin{aligned} F(f,a,b) = {} & \text{if } f\left(\tfrac{a+b}{2}\right) < 0 \\ & \text{then } F(f,(a+b)/2,\, b) \\ & \text{else } F(f,\, a,(a+b)/2). \end{aligned}$$

---

* In the sense that they are defined for a subset of $(\mathbb{R} \to \mathbb{R}) \times \mathbb{R} \times \mathbb{R}$.

It is easy to check that any function bisect satisfying the above specification also satisfies the equation (exercise!).

However, any constant function also satisfies it. Moreover, if we interpret the equality sign as Kleene equality (i.e, either both sides are undefined or else both are defined and the same), as we should, then the totally undefined function also satisfies the equation in F. Hence the above functional equation (aka recursive definition) fails to characterize the functions bisect satisfying the above specification.

Essentially, what happens is that we forgot to output the approximations to the root of the function f. At first sight, the notion of 'outputing while we compute' is alien to the mathematical notion of function. However, in this case a mathematical notion IS available.

Define output : $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ by

$$\text{output}(a, b, x) = \max(a, \min(x, b)).$$

We don't require that $a \le b$, but if this is the case then

$$\begin{aligned} \text{output}(a, b, x) &= \min(\max(a, x), b) \\ &= \begin{cases} a & \text{if } x \le a, \\ x & \text{if } a \le x \le b, \\ b & \text{if } b \le x. \end{cases} \end{aligned}$$

The intended intuition is that output(a,b,x) forces x to be between a and b when $a \le b$, doing nothing to x when it already is.

We now write the following functional equation:

$$F(f,a,b) = \text{output}\,\Big(a, b, \text{ if } f\left(\tfrac{a+b}{2}\right) < 0$$
$$\qquad\qquad\qquad\qquad \text{then } F(f, (a+b)/2, b)$$
$$\qquad\qquad\qquad\qquad \text{else } F(f, a, (a+b)/2)$$
$$\Big).$$

Now the constant functions don't satisfy the equation anymore, but the totally undefined function still does. But we are in the right track.

It is time to move to the interval domain and domain theory more generally.

We shall consider (Scott continuous) functions bisect : $(\mathcal{R}\to\mathcal{R})\times\mathcal{R}\times\mathcal{R}\to\mathcal{R}$ subject to a suitable specification discussed shortly. We anticipate the outcome by saying that the (continuous) solutions of the equation displayed at the top of this page, with the involved real functions replaced by interval extensions and F ranging over the domain $((\mathcal{R}\to\mathcal{R})\times\mathcal{R}\times\mathcal{R}\to\mathcal{R})$, are functions bisect satisfying the specification.

We first invoke some material from the domain-theory course.

<u>Lemma</u> If D and E are continuous Scott domains, then so are

(i) The cartesian product $D\times E$ under the coordinatewise order.

(ii) The set $(D\to E)$ of continuous maps under the pointwise order.

Hence $((\mathcal{R}\to\mathcal{R})\times\mathcal{R}\times\mathcal{R}\to\mathcal{R})$ is a continuous Scott domain. (We'll soon climb up to the next level of nesting of "function spaces"!)

Totality

Partial functions in domain theory are modelled by total functions that have partial values (including bottom, but not only bottom). In fact, we have seen some examples such as $x \mapsto 1/x$ and $x \mapsto \sin(1/x)$.

Nevertheless, one is still interested in the distinction between partiality and totality, and one often uses a hereditary notion. We introduce them for the particular cases we are interested in. Here are the definitions:

$x \in \mathcal{R}$ is total iff it is maximal iff it is a singleton interval.

A function $f \in (\mathcal{R} \to \mathcal{R})$ is total iff it maps total elements to total elements.

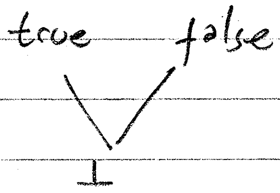
Remark For $(\mathcal{R} \to \mathcal{R})$ totality and maximality are unrelated:

(1) The functions id_ε defined above are total but not maximal.

(2) The canonical extension $g : \mathcal{R} \to \mathcal{R}$ of $f : \mathbb{R}\setminus\{0\} \to \mathbb{R}$ defined by $f(x) = 1/x$ is maximal but not total.

In fact, it is sort of an accident that totality and maximality agree for $\mathcal{R}$. $\square$

The partial booleans

$\mathcal{B} =$ 

(a continuous Scott domain)

The total truth values are the maximal ones, namely true and false.

A function $\mathcal{R}^n \to \mathcal{B}$ is total if it maps tuples of total reals to total boolean

__Annoying proposition__ Any continuous total function $\mathcal{R}^n \to \mathcal{B}$ is constant on tuples of total numbers.

__Topological proof__ The total tuples with the relative Scott topology form a connected topological space, continuous maps preserve connectedness, but the total truth values under the relative Scott topology form a (discrete and hence totally) disconnected space. □

__Exercise__ Find a more direct proof.

In particular, no function $\mathcal{R} \times \mathcal{R} \to \mathcal{B}$ s.t.

$$(x,y) \longmapsto \begin{cases} \text{true} & \text{if } x \text{ and } y \text{ are total and } x<y, \\ \text{false} & \text{if } x \text{ and } y \text{ are total and } x \geq y, \end{cases}$$

can be continuous. This is partially overcome as follows.

Let $S = \{(x,y) \in \mathbb{R}\times\mathbb{R} \mid x \neq y\}$. Then the function $S \to \mathbb{B}$, where $\mathbb{B} = \{\text{true}, \text{false}\} \subseteq \mathcal{B}$, defined by the rule

$$(x,y) \longmapsto \begin{cases} \text{true} & \text{if } x<y, \\ \text{false} & \text{if } x>y, \end{cases}$$

is continuous and hence has a canonical (maximal) extension $\mathcal{R}\times\mathcal{R} \to \mathcal{B}$. One calculates that it is given by

$$(x,y) \longmapsto \begin{cases} \text{true} & \text{if } \overline{x} < \underline{y}, \\ \text{false} & \text{if } \overline{y} < \underline{x}, \\ \bot & \text{otherwise.} \end{cases}$$

(Exercise!)

__Notation:__ $x <_\bot y$

The parallel conditional overcomes the deficiencies of the partial less-than relation just discussed.

$$\text{pif} - \text{then} - \text{else} - : \mathcal{B} \times \mathcal{R} \times \mathcal{R} \longrightarrow \mathcal{R}$$

$$\text{pif } p \text{ then } x \text{ else } y = \begin{cases} x & \text{if } p = \text{true}, \\ y & \text{if } p = \text{false}, \\ x \sqcap y & \text{if } p = \bot. \end{cases}$$

This function is continuous (exercise). Exercise: Formulate this as the canonical extension of the evident conditional map $\mathbb{B} \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$.

Example Although $(x, y) \mapsto (x <_\bot y)$ is partial, it can be used in conjunction with the parallel conditional to define total functions, e.g. $\mathcal{R} \longrightarrow \mathcal{R}$ defined by

$$x \longmapsto \text{pif } x <_\bot 0 \text{ then } -x \text{ else } x$$

is a continuous extension of the absolute-value function $x \mapsto |x| : \mathbb{R} \to \mathbb{R}$. Here '$0$' is a shorthand for $\{0\} = [0,0]$ and '$-$' denotes the canonical extension of the negation function $x \mapsto -x : \mathbb{R} \longrightarrow \mathbb{R}$. $\square$

From now on, we'll often denote a canonical domain-theoretic extension of a real-analysis function by the same symbol. In particular, we have a canonical extension output $: \mathcal{R} \times \mathcal{R} \times \mathcal{R} \longrightarrow \mathcal{R}$ of the continuous function output $: \mathbb{R} \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ discussed earlier.

Exercise Let $a, b \in \mathbb{R}$ with $a \le b$ and $x \in \mathcal{R}$. Then

$$\text{output}(a, b, x) = \text{output}(\{a\}, \{b\}, x) = [a, b] \sqcup x.$$

In particular, $\text{output}(a, b, \bot) = [a, b]$. $\square$

We are now ready to return to the bisection problem.

<u>Definition</u> A (restricted) <u>root functional</u> is a continuous map

$$\mathrm{root} : (\mathcal{R} \to \mathcal{R}) \times \mathcal{R} \times \mathcal{R} \longrightarrow \mathcal{R}$$

such that for all $a, b \in \mathbb{R}$ with $a \le b$,

for every continuous function $f : \mathbb{R} \to \mathbb{R}$ which is strictly increasing on $[a,b]$ and has $f(a) \le 0 \le f(b)$, so that there exists a unique $x \in [a,b]$ with $f(x) = 0$,

for every continuous extension $g : \mathcal{R} \to \mathcal{R}$ of f (canonical or not)

we have that

$$\mathrm{root}(g, \{a\}, \{b\}) = \{x\}. \qquad \square$$

<u>Theorem</u> The continuous solutions in $F \in ((\mathcal{R} \to \mathcal{R}) \times \mathcal{R} \times \mathcal{R} \to \mathcal{R})$ to the functional equation

$$\begin{aligned} F(f,a,b) = \mathrm{output}(a, b, \mathrm{pif}\ & f\left(\tfrac{a+b}{2}\right) <_\bot 0 \\ & \mathrm{then}\ F(f, (a+b)/2, b) \\ & \mathrm{else}\ F(f, a, (a+b)/2)) \end{aligned}$$

are root functionals.

<u>Proof outline to be elaborated below</u> The least continuous solution is a root functional and any solution above (in the domain-theoretic order) a root functional is also a root functional. The second step is easy and hence is left as an exercise. We sketch the first in more detail so that the details left can be regarded as exercises. $\square$

We need the following from the domain-theory course.

__Lemma__ Let D be a dcpo with $\bot$. Any continuous endomap $f : D \longrightarrow D$ has a least fixed point, i.e an element $x \in D$ with $f(x) = x$ and $x \sqsubseteq y$ for any $y \in D$ with $f(y) = y$.

__Proof sketch__ $x = \bigsqcup_n f^n(\bot)$. $\square$

We take $D = ((\mathcal{R} \to \mathcal{R}) \times \mathcal{R} \times \mathcal{R} \to \mathcal{R})$. Define $\Phi : D \to D$ by, for all $F \in D$,

$$\Phi(F) = (f, a, b) \mapsto$$
$$\text{output}(a, b, \text{pif } f(\tfrac{a+b}{2}) <_\bot 0$$
$$\text{then } F(f, (a+b)/2, b)$$
$$\text{else } F(f, a, (a+b)/2)).$$

N.B. We are using the "mapsto" notation $x \mapsto e$ which is the semantic analogue of the syntactical λ-notation of the λ-calculus.

This function Φ is continuous, because all functions which are definable from continuous functions using the mapsto calculus are continuous. This follows from the universal properties of cartesian products and function spaces c.f. Streicher's notes (i).

Thus, the equation of the theorem amounts to the fixed-point equation

$$F = \Phi(F)$$

and we wish to prove that the least solution $\bigsqcup_n \Phi^n(\bot)$ is a root functional. We don't do this in detail in this version of these notes for lack of time. Define $F_n = \Phi^n(\bot)$ and $[a_n, b_n] = F_n(f, a, b)$.

We now claim that the sequence $[a_n, b_n]$ is that printed by the imperative algorithm given at the beginning of this section. The slight complication in the proof is that we may be "unlucky" to hit the root of the function, in which case the "powers" of the parallel conditional have to be invoked. We leave the easy, but perhaps laborious, details to the reader.

Summing up, by a <u>recursive definition</u> it is meant an equation

$$F = \Phi(F),$$

where Φ is continuous but not necessarily given explicitly*. The entity it defines is $F = \bigsqcup_n \Phi^n(\bot)$, the least fixed point of Φ.

But why do we take the <u>least</u> fixed point? Mathematically speaking, because it is a canonical one. But there is a much better, computational explanation, contained in a theorem known as "Computational Adequacy". Briefly, we take this fixed point because it is the one "computed by unfolding the definition successively". The full answer is one of the topics of the next section.

* As in e.g. the theorem.

4. PCF extended with real numbers

We begin with PCF and immediately refer the reader to the notes by Streicher (i). The sections relevant to us are 1 and 2, to begin with, and the reader will be alerted when 3 becomes necessary. No other section is necessary, but the readers are invited to take their time to carefully study the whole set of notes.

We summarize Section 1 of (i) rather briefly.

PCF has a base type for natural numbers interpreted as

$$\mathcal{N} = \mathbb{N}_\perp = \begin{tikzcd}[column sep=small] 0 & 1 & 2 & 3 & \cdots \\ & & \perp \arrow[ull, no head] \arrow[ul, no head] \arrow[u, no head] \arrow[ur, no head] & & \end{tikzcd}$$

in the "Scott model". It may have a base type for booleans, interpreted as $\mathcal{B} = \mathbb{B}_\perp$, and it usually has. Streicher has chosen to omit it, but we reintroduce it.

It has successor, predecessor and test-for-zero functions on the natural numbers and a conditional.

It has recursion, via the least-fixed-point operator, λ-abstraction and function application.

All types are obtained from base types by iterating a function-type constructor (arrow).

PCF has operational and denotational semantics, which coincide at base types ("computational adequacy").

Our task is to extend PCF with a base type for real numbers following the above programme. We call the resulting language MGS.

We call the base type of reals ρ or "real" (in typewriter face, without the quotes, which unfortunately is not available in the printer used to produce this document). It is interpreted, in the intended domain-theoretic model, as the partial real line $\mathcal{R}$.

Our first subtask is to decide which primitive functions on the reals (suitably extended to the partial reals) we should/ought/might/could have. A natural choice would be the (canonical extensions of) the four basic operations $+, -, \cdot, /$, the output function and the less-than relation (as in the previous section). Moreover, the previous section also suggests that we include the parallel conditional (although this is debatable, see the discussion in the concluding section). In fact, this is what the author, as a programmer, would choose. However, the author is now writting as someone who is expected to prove properties of the language, and hence tries to make the language as simple as possible. Hence we take a subset of the above operations as primitive. This is reasonable, because it will turn out to be the case that the full set is definable from the restricted subset. This will be a formidable programming exercise, however, and in practice one would actually include the full set of operations as primitive.

The subset we choose consists of

(1) The parallel conditional (but see concluding section).

(2) The less-than test $(-) <_\perp p : \mathcal{R} \to \mathcal{B}$ with p rational,

(3) The canonical extensions $\mathcal{R} \to \mathcal{R}$ of the functions $p + (-) : \mathbb{R} \to \mathbb{R}$ and $p \cdot (-) : \mathbb{R} \to \mathbb{R}$ with p rational again.

(4) For each $b \in \mathbb{IR}$ with rational endpoints a function $\text{output}_b : \mathcal{R} \to \mathcal{R}$ related to the output function of the previous section, which we now discuss.

<u>Lemma</u> For any $b \in \mathcal{R}$ there is a unique continuous function

$$\text{output}_b : \mathcal{R} \to \mathcal{R}$$

s.t.

$$\text{output}_b(x) = b \sqcup x \quad \text{if } b \text{ and } x \text{ are bounded above.}$$

<u>Proof</u> (exercise) For $b = \bot$ this is clear. For $b \in \mathbb{IR}$, show that any such monotone function satisfies

$$\text{output}_b(x) = \begin{cases} \{\underline{b}\} & \text{if } \overline{x} < \underline{b}, \\ \{\overline{b}\} & \text{if } \overline{b} < \underline{x}. \end{cases}$$

Then observe that if b and x are not bounded above then either $\overline{x} < \underline{b}$ or $\overline{b} < \underline{x}$. Hence there is exactly one such monotone function. To conclude, show that it is continuous. □

<u>Lemma</u> Define the <u>oriented-join operation</u> $(-) \overrightarrow{\sqcup} (-) : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ by

$$x \overrightarrow{\sqcup} y = \text{output}_x(y).$$

Then $(\mathcal{R}, \overrightarrow{\sqcup}, \bot)$ is a monoid, i.e,

$$\bot \overrightarrow{\sqcup} x = x \overrightarrow{\sqcup} \bot = x$$

and

$$(x \overrightarrow{\sqcup} y) \overrightarrow{\sqcup} z = x \overrightarrow{\sqcup} (y \overrightarrow{\sqcup} z).$$

<u>Proof.</u> There is an elegant lattice theoretic proof. However, a geometric proof is more appropriate and actually simpler. Exercise: work it out — that is, draw some pictures (based on pictures given in Section 2). □

Lemma For $p \in \mathbb{R}$ and $b, x \in \mathcal{R}$,

$$p + (b \overrightarrow{\sqcup} x) = (p+b) \overrightarrow{\sqcup} (p+x)$$

and

$$p\,(b \overrightarrow{\sqcup} x) = pb \overrightarrow{\sqcup} px.$$

Proof Again geometrical. □

The above two lemmas are used in order to construct a (computationally adequate) operational semantics for our language. We first officially introduce the language.

MGS is PCF extended with a base type ρ or "real" and the following term-formation rules (where we omit typing-context information which should be reconstructed by the reader based on Streicher's carefully written notes).

(1) If $L : \text{bool}$, $M, N : \text{real}$ are terms then so is

$$(\text{pif } L \text{ then } M \text{ else } N) : \text{real}.$$

(2) If $M : \text{real}$ is a term and p is a rational number then we have a term

$$(M < p) : \text{bool}.$$

(3) If p is a rational number and $M : \text{real}$ is a term then so are

$$(p + M) : \text{real},$$
$$(pM) : \text{real}.$$

(4) If $b \in \mathbb{IR}$ has rational endpoints and $M : \text{real}$ is a term then so is

$$(\text{output}_b\ M) : \text{real}.$$

The denotational (or mathematical) semantics of MGS has already been explained. our next subtask is to provide an operational (or computational) semantics. A reading of Streicher's notes (i) reveals that there are two different, but equivalent styles, based on "big-step" or "small-step" computation rules. The big-step style rules attempt to compute the "final" (total) value of a term of ground type. However, computations on the reals are infinite and hence don't have "final" values. We are thus forced to use the small-step style. We write $M \longrightarrow N$ rather than $M \rhd N$, for historical reasons.

Before defining the computation rules, we define the operational meaning of terms of base type:

<u>Lemma</u> For a closed term M of base type respectively nat, bool and real, the following sets are either empty or directed:

$$\begin{array}{ll}
(1) & \{ n \in \mathbb{N} \mid M \longrightarrow^* n \} \subseteq \mathcal{N}, \\
(2) & \{ p \in \mathbb{B} \mid M \longrightarrow^* p \} \subseteq \mathcal{B}, \\
(3) & \{ b \in I\mathbb{R} \mid M \longrightarrow^* \text{output}_b\, M' \text{ for some } M' \} \subseteq \mathcal{R}.
\end{array}$$

(In fact, the first two are either empty or singletons).

<u>Proof</u> It cannot be provided until we know the definition of the small-step relation $\longrightarrow$ on terms, but in any case it will be left as an exercise. $\square$

<u>Definition</u> With the same "respectivity" as in this lemma, we define: (N.B. $\sqcup \emptyset = \bot$)

$$\begin{array}{ll}
(1) & [M] = \bigsqcup \{ n \in \mathbb{N} \mid M \longrightarrow^* n \}, \\
(2) & [M] = \bigsqcup \{ p \in \mathbb{B} \mid M \longrightarrow^* p \}, \\
(3) & [M] = \bigsqcup \{ b \in I\mathbb{R} \mid M \longrightarrow^* \text{output}_b\, M' \text{ for some } M' \}.
\end{array}$$

The small-step rules are based on lemmas formulated above, and it is intended that $[\![M]\!] = [M]$ for closed terms M of base type. However, this will require a far from obvious argument.

<u>The accumulation rule</u> (based on associativity of oriented joins)

$$\text{output}_b\,(\text{output}_c\ M) \longrightarrow \text{output}_{b \vec{\sqcup} c}\ M.$$

The calculation $b \vec{\sqcup} c$ is on rational intervals and hence directly computable. (N.B. We could have used dyadic rather than rational intervals without loss of generality and with considerable gain of efficiency. But we ignore efficiency for the moment. See the concluding section.)

<u>The arithmetic rules</u> (based on the lemma following associativity)

$$p + (\text{output}_b\ M) \longrightarrow \text{output}_{p+b}\ (p+M),$$

$$p\ (\text{output}_b\ M) \longrightarrow \text{output}_{pb}\ (pM).$$

Again the computations $p+b$ and pb are on rational numbers. The idea, as above, is to try to get a term of the form $\text{output}_c\ N'$ out of a term N. The first two of the following have a complementary effect* (cf. "introduction" versus "elimination" rules in type theory and natural deduction).

<u>The comparison rules</u> (based on the definition of $<_\perp$)

$$(\text{output}_b\ M < p) \longrightarrow \text{true} \quad \text{if } \overline{b} < p,$$

$$(\text{output}_b\ M < p) \longrightarrow \text{false} \quad \text{if } p < \underline{b}.$$

* Wrt the above.

The conditional rules (based on the definition of the parallel conditional)

$$\text{pif true then } M \text{ else } N \longrightarrow M,$$
$$\text{pif false then } M \text{ else } N \longrightarrow N,$$

$$\text{pif } L \text{ then output}_b\, M \text{ else output}_c\, N$$
$$\longrightarrow \text{output}_{b \sqcap c}\,(\text{pif } L \text{ then output}_b\, M \text{ else output}_c\, N).$$

Yet again, the computation $b \sqcap c$ is on rational intervals. Notice that we are not able to get rid of the prefices output_b and output_c of the terms M and N. (Exercise: Think about that.)

Finally, we need rules that prepare the ground for the application of the above rules.

The preparation rules

$$\text{output}_b\, M \longrightarrow \text{output}_b\, M' \quad \text{if } M \longrightarrow M',$$

$$p + M \longrightarrow p + M' \quad \text{if } M \longrightarrow M',$$

$$pM \longrightarrow pM' \quad \text{if } M \longrightarrow M',$$

$$(M < p) \longrightarrow (M' < p) \quad \text{if } M \longrightarrow M',$$

$$\text{pif } L \text{ then } M \text{ else } N \longrightarrow \text{pif } L' \text{ then } M \text{ else } N \quad \text{if } L \longrightarrow L',$$

$$\text{pif } L \text{ then } M \text{ else } N \longrightarrow \text{pif } L \text{ then } M' \text{ else } N \quad \text{if } M \longrightarrow M',$$

$$\text{pif } L \text{ then } M \text{ else } N \longrightarrow \text{pif } L \text{ then } M \text{ else } N' \quad \text{if } N \longrightarrow N'.$$

The last three rules have the annoying property that there is no canonical way of applying them. Of course, we can systematically "schedule" them in a "fair" way, but there is no canonical fair scheduler. This is not just annoying — it is also inefficient (see the concluding section). But it works, and for the moment we ignore efficiency issues. In any case, the adjective "parallel" to the conditional is based on this annoying fact.

Our final subtask is to prove that the denotational (mathematical) and operational (computational) meanings of programs (closed terms of ground type) M agree:

$$[\![M]\!] = [M]. \qquad \text{(computational adequacy.)}$$

One half, namely $[M] \sqsubseteq [\![M]\!]$, known as <u>soundness</u>, is easy. By design:

<u>Lemma</u> If $M \to M'$ then $[\![M]\!] = [\![M']\!]$. $\square$

Hence if $M \to^* M'$ then $[\![M]\!] = [\![M']\!]$ by straightforward induction. We consider M: real and leave the cases M: nat and M: bool to the reader as yet another exercise. By definition of join and of $[M]$, $[M] \sqsubseteq [\![M]\!]$ iff $b \sqsubseteq [\![M]\!]$ for every b with $M \to^* \mathrm{output}_b\, M'$ for some M'. For any b and M' with $M \to^* \mathrm{output}_b\, M'$ we have, by the above lemma, $[\![M]\!] = [\![\mathrm{output}_b\, M']\!] = \mathrm{output}_b\, [\![M']\!] \sqsupseteq \mathrm{output}_b(\bot) = b$ using (monotonicity of output_b and) the defining property of output_b. This proves:

<u>Proposition</u> Soundness holds. $\square$

The difficult task is to establish <u>completeness</u>, i.e., $[\![M]\!] \sqsubseteq [M]$. We now refer the reader to section 3 of Streicher's notes (a). We extend his proof from the language PCF to the language MGS.

Of course, we recall the method of proof in detail as we extend it. Before doing this, we elucidate the concrete meaning of completeness from a computational point of view, formulating a lemma which is needed later.

<u>Lemma</u> For a closed term $M : \text{real}$,

$$[\![M]\!] \sqsubseteq [M] \text{ iff for every } y \ll [\![M]\!] \text{ with } y \neq \bot$$
$$\text{there is some } b \text{ with } y \sqsubseteq b \text{ and}$$
$$M \longrightarrow^* \text{output}_b\, M' \text{ for some } M'.$$

<u>Proof</u> This follows by definition of $\ll$ and of $[M]$, using the fact that the defining join of $[M]$ is either empty (which accounts for the case $y = \bot$) or else directed, which has already been formulated as a lemma. □

In other words, M $\overset{\text{eventually}}{\text{outputs}}$ any approximation of $[\![M]\!]$.

In order to prove that $[\![M]\!] \sqsubseteq [M]$ for M a closed term of base type, we proceed, as in Streicher's notes, by induction on types. The difficulty, as always in difficult inductive proofs, is to find a suitable induction hypothesis.

<u>Definition</u> We define a family of relations $R_\sigma \subseteq D_\sigma \times (\text{closed terms})_\sigma$ indexed by types σ by induction as follows:

(1) For a base (or ground) type γ,
$$d \; R_\gamma \; M \text{ iff } d \sqsubseteq [M].$$

(2) For a function type $\sigma \to \tau$ we define
$$f \; R_{\sigma\to\tau} \; F \text{ iff } f(d) \; R_\tau \; FM \text{ whenever } d \; R_\sigma \; M.$$

Completeness follows if we are able to prove that $[\![M]\!] \; R \; M$ for every closed term M — of ground type – but we need to prove this for all types.

In fact, in order to get the desired conclusion, we need to prove a bit more. This is formulated as Lemma 3.3 in Streicher's notes.

Basically, to prove this lemma, we need to take care of application, λ-abstraction, the fixed-point operator and the primitive operations for each base type. Fortunately, application and λ-abstraction don't need to be reworked. Unfortunately, the fixed-point operator, as presented there, does. Of course, the new primitive operations also do. We thus (re)work what we need, in sufficient, but certainly not complete, details.

(1) $d' \sqsubseteq d \ R \ M$ implies $d' \ R \ M$.

Immediate for base types and straightforward for function types (inductive proof). As in Streicher's notes.

(2) $\bot \ R \ M$.

Ditto.

(3) $\{ d \mid d \ R \ M \}$ is closed under directed joins.

Ditto. So far it doesn't matter what the base types and primitive operations are.

<u>Definition</u> By induction on types, define:

$$M \underset{\sim}{\sqsubseteq}_\gamma N \text{ iff } [M] \sqsubseteq [N],$$

$$F \underset{\sim}{\sqsubseteq}_{\sigma \to \tau} G \text{ iff } FM \underset{\sim}{\sqsubseteq}_\tau GM \text{ for every } M : \sigma \text{ closed.}$$

(4) $d \mathrel{R} M \lessapprox M'$ implies $d \mathrel{R} M'$.

Again a straightforward proof.

(5) $M(YM) \lessapprox YM$.

This time Streicher's proof doesn't work, as it relies on the fact that the only base type is "nat". This time I take this as a debt rather than an exercise, but I promise that I am in posession of a proof. (cf. my thesis.)

(6) $f \mathrel{R} F$ implies $(\mathrm{fix}\ f) \mathrel{R} (Y\ F)$

Here fix and Y are the semantical and syntactical fixed-point operators respectively. Given (5), Streicher's proof works.

(7) $[\![Y]\!] \mathrel{R} Y$

Immediate consequence of the above, recalling that $[\![Y]\!] = \mathrm{fix}$ by definition.

Now we have to take care of the primitive real functions. All of them work in essentially the same way. We do $p + (-)$ as a representative example.

Lemma $[\![p+M]\!] \mathrel{R} p+M$ if $[\![M]\!] \mathrel{R} M$.

Proof By definition of R_{real} and $[M]$ for $M : \mathrm{real}$, $[\![M]\!] \mathrel{R} M$ amounts to $[\![M]\!] \sqsubseteq [M]$, and what we want is $[\![p+M]\!] \sqsubseteq [p+M]$. Let $y \ll [\![p+M]\!] = p + [\![M]\!]$. By continuity of the domain $\mathcal{R}$ and by Scott continuity of the function $p+(-) : \mathcal{R} \to \mathcal{R}$, using the fact that

the defining join of $[\![M]\!]$ is either empty or directed, if $y \neq \bot$ then there is $x \ll [\![M]\!]$ with $y \sqsubseteq p + x$, using a lemma characterizing continuity of functions of continuous domains.
By a lemma (unfortunately with no label) stated earlier in this section, there are b and M' s.t. $x \sqsubseteq b$ and $M \rightarrow^* \text{output}_b\, M'$.
Hence $p + M \rightarrow^* \text{output}_{p+b}\, (p + M')$ by the preparation and arithmetical rules. But $y \sqsubseteq p + x \sqsubseteq p + b \sqsubseteq [\![\text{output}_{p+b}\, (p+M')]\!]$
Hence the same lemma without label, in the other direction, gives what we want. $\square$

This proves:

Theorem completeness holds.

Hence we have computational adequacy.

Example of application of adequacy

We have proved, semantically, that a certain functional finds roots of certain functions. The recursive definition immediately (see section below) gives rise to a program.
By adequacy, this program computes the roots of such functions.
(Exercise work out the (routine) details.)

5. Expressivity of the programming language

<u>Programming exercise</u> Show that the canonical extensions of the operations $+, -, \cdot, /$ are definable in the language. Show that $(-) <_\perp (-) : \mathcal{R}\times\mathcal{R} \to \mathcal{R}$ as defined in section 3 is definable in the language. Show that the canonical extension output: $\mathcal{R}\times\mathcal{R}\times\mathcal{R} \longrightarrow \mathcal{R}$ of the function output: $\mathbb{R}\times\mathbb{R}\times\mathbb{R} \to \mathbb{R}$ defined in section 3 is definable in the language.

(c.f. references (ii)(b) and (iii)). $\square$

As a syntactical notion of computability, one can take definability in the language. But there is also a semantical definition of computability cf. references (ii)(b) and (i). Syntactical computability easily implies semantical computability. The converse, when it holds is known as (Turing-) universality. Our language is universal at first-order types, as some programming together with the arguments of reference (ii)(b) show. The same programming and the same reference (or alternatively reference (iii)), show that the language extended with a primitive for a certain (semantically computable) existential quantifier is universal at all types.

6. Concluding remarks

Missing. I've run out of time.

I may type and/or expand these notes in the future.