

Domains for recursion

Domain theory is a comparatively young subject having started about 25 years ago. It has connection with various other topics. It provides many examples of categories, all of a similar nature. In fact, some people seem to behave as though it is just a part of **category theory**. Domains provide a way of getting at topological notions without having to do any topology. In this sense it provides many refinements of the category **Top** of topological spaces.

Part of the novelty of domain theory is that it gives a way of handling spaces of values where there may be awkward values around (such as ‘undefined’, ‘non-terminating’, ‘empty’, . . . , and so on). Because of this domain theory can provide models of systems where such awkward values arise quite naturally, such as higher order λ -calculi or the untyped λ -calculus.

Another central feature (although not exactly the novelty that some people think) is that it codifies certain methods of finding fixed points of appropriate operators. It is this which leads to methods of solving certain higher order equations.

We will use this feature to make sense of ‘the solution or solutions of a recursive specification’. In fact, this is one of the easiest introductions to the whole topic. (It is interesting that one of the standard surveys of the topic makes almost no mention of this simple application.)

Contents

1	Introduction	1
	Exercises	5
2	Flat domains	6
	Exercises	7
3	Recursion operator	8
	Exercises	11
4	Monotonicity	11
	Exercises	14
5	Domains	15
	Exercises	19
	Some solutions	20

1 Introduction

We know that some recursive specifications, the more commonplace ones, do produce a unique function, and that function is total. However, some specifications are not like this, and for these we must ask several questions.

- Is there any function, perhaps a partial function, which meets the specification?
- Is there a total function which meets the specification?
- Is there a canonical solution, that is a function which meets the specification and can be regarded as ‘special’ in some sense?

In these notes we see how we can make sense of these questions. We then find the answers to the questions are: Yes; Not necessarily; Yes. Furthermore, it is the way we answer the third question that is the content of these notes.

To begin let’s look at a few examples of recursive specifications. Some of these we have seen elsewhere, and some not. Each one is chosen because it has, or might have, hidden demons.

1.1 EXAMPLES. (a) For a set \mathbb{S} , function $k : \mathbb{S}'$, and subset $\mathbb{B} \subseteq \mathbb{S}$, let

$$f : \mathbb{S} \longrightarrow \mathbb{S}$$

be given by

$$f s = \begin{cases} f s^- & \text{if } s \notin \mathbb{B} \\ s & \text{if } s \in \mathbb{B} \end{cases}$$

where $s^- = ks$ (for $s \in \mathbb{S}$).

The semantics of a while loop has this form.

(b) A simple modification of the specification of (a) makes it look more devious. Using the same data $\mathbb{S}, k, \mathbb{B}$ we can ask for a function $f : \mathbb{S}'$ where

$$f s = \begin{cases} f^2 s^- & \text{if } s \notin \mathbb{B} \\ s & \text{if } s \in \mathbb{B} \end{cases}$$

(for each $s \in \mathbb{S}$). As in (a) we take $s^- = ks$.

At this stage it is not at all clear what the effect of the ‘ f^2 ’ will be.

(c) There are other obvious modifications we could make to the specification of (a). Here is a particular case of a slight extension to (b). We require

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

where

$$f x = \begin{cases} f^2(x + 11) & \text{if } x \leq 100 \\ x - 10 & \text{if } 100 < x \end{cases}$$

(for $x \in \mathbb{N}$).

(d) We tend to think of specifications, and recursive specifications in particular, as taking place over ‘discrete’ sets. In fact, this need not be the case. In (a) and (b) we put no conditions at all on the set \mathbb{S} . Let’s look at a concrete example.

Using the reals \mathbb{R} we can ask for a function

$$f : \mathbb{R} \longrightarrow \mathbb{R}$$

where

$$fx = \begin{cases} 1 + f^2(x - 1) & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$$

(for $x \in \mathbb{R}$). ■

It turns out that many recursions can be massaged into one particular form. This is by no means the most general form but it is useful as an introduction to the methods we develop here.

1.2 DEFINITION. Given the data functions

$$g : \mathbb{S} \longrightarrow \mathbb{T} \quad h : \mathbb{S} \longrightarrow \mathbb{T}' \quad k : \mathbb{S} \longrightarrow \mathbb{S}$$

and $\mathbb{B} \subseteq \mathbb{S}$ let

$$f : \mathbb{S} \longrightarrow \mathbb{T}$$

be given by

$$fs = \begin{cases} hs(fs^-) & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases}$$

(for each $s \in \mathbb{S}$). Here $s^- = ks$. We refer to this specification as the SPEC. ■

This SPEC can turn out to be rather more complicated than it looks. The data functions can be high order and there may be hidden parameters.

We are going to develop a method of analysing this and many other specifications. In the first instance we will concentrate on this particular one, but almost everything we do will generalize to many other cases.

To make a start let us take the SPEC at face value and try to evaluate the function f (whatever that might be). Thus, for the time being we take the naive view that there is just one function f . Of course, we might run into trouble at some point, but that will indicate where the problems lie.

Given $s \in \mathbb{S}$ how might we calculate fs ? After a while we see that the evaluation can be done in two phases; an unwinding followed by a rewinding.

(Unwinding) Given $s \in \mathbb{S}$ let

$$s_0 = s \quad t_0 = fs_0$$

so that s_0 is the supplied input and t_0 is the required output. By repeated passes through SPEC we find that

$$\begin{array}{llll} t_0 & = & fs_0 = hs_0t_1 & \text{where } s_1 = ks_0 \text{ and} \\ t_1 & = & fs_1 = hs_1t_2 & \text{where } s_2 = ks_1 \text{ and} \\ t_2 & = & fs_2 = hs_2t_3 & \text{where } s_3 = ks_2 \text{ and} \\ & & \vdots & \\ t_i & = & fs_i = hs_it_{i+1} & \text{where } s_{i+1} = ks_i \text{ and} \\ t_{i+1} & = & \dots & \end{array}$$

where this unwinding continues until $s \in \mathbb{B}$.

In short we set

$$s_0 = s \quad s_{i+1} = ks_i$$

and as each s_i is generated we test whether or not $s_i \in \mathbb{B}$.

(Rewind) Once we find the smallest index $*$ with $s_* \in \mathbb{B}$ we have

$$t_* = fs_* = gs_*$$

and then we can work back up the evaluation template generated in the unwinding phase to produce

$$\dots, t_i, \dots, t_2, t_1, t_0$$

and so obtain the required value t_0 .

This description immediately pinpoints the problem. In general there is no reason why the unwinding phase should terminate. In other words the generated sequence

$$s = s_0, s_1, s_2, \dots$$

may never get into \mathbb{B} . To get round this problem we look for a *partial* function f which meets the specification.

Let $\mathbb{D} \subseteq \mathbb{S}$ be given by

$$s \in \mathbb{D} \iff (\exists i)[s_i \in \mathbb{B}]$$

(for $s \in \mathbb{S}$). For each $s \in \mathbb{D}$ let $|s|$ be the least i with $s_i \in \mathbb{B}$. We may call this the descent or unwinding **length** of s . We also use a nominal length $|s| = \omega$ to indicate $s \notin \mathbb{D}$. Thus

$$s \in \mathbb{D} \iff |s| < \omega \quad s \notin \mathbb{D} \iff |s| = \omega$$

connects the two notions.

With this we see that the above analysis leads to the following.

1.3 THEOREM. *There is a total function*

$$f_D : \mathbb{D} \longrightarrow \mathbb{T}$$

which meets SPEC. This function is given by

$$fs = (hs_0 \circ \dots \circ hs_{l-1} \circ g)s_l$$

where $l = |s|$ is the length of the input $s \in \mathbb{D}$.

Furthermore, if

$$f_E : \mathbb{E} \longrightarrow \mathbb{T}$$

is any other total function which meets SPEC, then $\mathbb{D} \subseteq \mathbb{E}$ and $f_D = f_E|_{\mathbb{D}}$.

We have enough material from above to give a formal proof of this result, but we won't bother. This is because such a proof is a bit messy, and there is a much cleaner version once we have developed a few more ideas which enables us to consider the problem in a slightly different setting.

We can go through a similar analysis for almost any recursion we may meet and prove a similar result. In other words each such specification does have a canonical solution, and every other solution is an extension of this. We will set up this idea in a more general context which has connections with techniques used in other areas.

The canonical solution is the unique smallest solution (as measured by the domain of definition). You may wonder why we don't try for a larger solution. The reason is that, in general, there is no larger solution that can be described as 'canonical'. The specification (d) of Examples 1.1 illustrates this.

Exercises

1 Consider the specifications (a) of Examples 1.1. This is a simplified version of the SPEC of Definition 1.2.

Give an explicit description of the function generated by the U/R process. To do this you may find the '*k*-length' $|s|$ of the input $s \in \mathbb{S}$ useful.

2 Consider the specifications (b) of Examples 1.1. This is *not* an instance of the SPEC of Definition 1.2.

(a) Why not?

(b) Can you think of an extension of SPEC which would cover this case?

(c) Nevertheless show that a modified version of the U/R process will produce a solution of the specification.

(d) Give an explicit description of this solution.

(e) Can you think of a slight modification to this specification which would require a complete rethink of your evaluation process?

3 Consider the specification (c) of Examples 1.1. Show that the obvious modification of the U/R process produces a unique function and this is total. You should give an explicit description of this function.

4 Consider the specification (d) of Example 1.1.

(a) Show that each of the total functions given by

$$(i) \quad fx = x \qquad (ii) \quad fx = \lfloor x \rfloor \qquad (iii) \quad fx = \begin{cases} x & \text{if } x \in \mathbb{Z} \\ \lfloor x \rfloor + 1/2 & \text{if } x \notin \mathbb{Z} \end{cases}$$

(for $x \in \mathbb{R}$) meets the specification.

(b) Show that a modified U/R process does produce a function which meets the specification, but this function is not total. How is this function related to the functions (i, ii, iii) of (a)?

5 Consider a while loop

while B do Q od

in a programming language. Here B is a boolean expression and Q is an instruction. Let \mathbb{S} be the space of states for this loop.

Write down a recursive specification of the semantics of this loop (in terms of the semantics of Q).

How does this specification compare with SPEC?

2 Flat domains

When we handle functions we usually think of *total* functions, that is we work in the category **Set** of sets and total functions. However, as we have just seen, sometimes it is more natural to deal with *partial* functions. We could work in a more extensive category with a larger collection of arrows, but we won't. For what we do here there is a more useful trick.

How can we handle a partial function

$$f : \mathbb{S} \longrightarrow \mathbb{T}$$

which need not be defined for each input $s \in \mathbb{S}$. Somehow we need to get at the domain of definition \mathbb{D} of f . To do this we adjoin to \mathbb{T} a tag \perp

$$\mathbb{T}_\perp = \mathbb{T} \cup \{\perp\}$$

which we think of as a nominal value 'undefined'. We extend the total function

$$f : \mathbb{D} \longrightarrow \mathbb{T}$$

to a total function

$$f_\perp : \mathbb{S} \longrightarrow \mathbb{T}_\perp$$

by

$$f_\perp s = \begin{cases} fs & \text{if } s \in \mathbb{D} \\ \perp & \text{if } s \notin \mathbb{D} \end{cases}$$

(for $s \in \mathbb{S}$). In other words we extend f to the whole of \mathbb{S} by taking the value \perp where f is undefined. We now work with f_\perp in place of f .

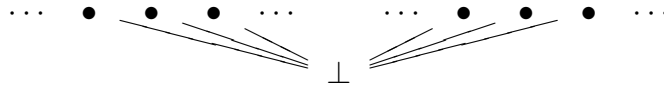
In this way we can gather all the partial function from \mathbb{S} to \mathbb{T} together into one type $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ and not worry too much about domains of definition.

We will return to this shortly. Before that we observe that \mathbb{T}_\perp has some useful structure.

The set \mathbb{T}_\perp is partially ordered by

$$u \leq t \iff u = t \text{ or } u = \perp$$

(for $t, u \in \mathbb{T}_\perp$). In other words we think of \mathbb{T} as a discrete set and place the tag \perp below the whole of \mathbb{T}



This produces a **flat domain** by **lifting the discrete domain** (or set) \mathbb{T} . All three highlighted phrases have technical meanings. We won't define them here but we will give a good impression of the meanings. However, you should be aware that the use here of the word 'domain' is not the same as in the phrase 'domain of definition'. (I am not certain who thought it was a good idea to have yet another meaning of the word. You can always expect this kind of nonsense from some people, but if this was the person I think it was then he should have known better.)

Recall that for posets S, T the appropriate morphisms

$$S \longrightarrow T$$

are the monotone functions. Two such morphisms can be compared by the pointwise comparison to produce another poset. (Thus for two **Pos**-objects S, T the hom-set $\mathbf{Pos}[S, T]$ is **enriched** as a poset.)

Here we are concerned with functions

$$\mathbb{S} \longrightarrow \mathbb{T}_\perp$$

where the source is discrete and the target is flat. For two such function f, g the pointwise comparison gives

$$f \leq g \iff (\forall s \in \mathbb{S})[fs \leq gs] \iff (\forall s \in \mathbb{S})[fs = gs \text{ or } fs = \perp]$$

(where the central condition uses the comparison on \mathbb{T}_\perp). This furnishes $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ as a poset which is a domain (in the technical sense) but is certainly not flat.

Notice that the comparison on $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ is nothing more than the comparison by extension of the corresponding partial functions.

Domain theory is a comparatively recent development, no more than 25 years old. It is the study of various categories of domains with appropriate morphisms. Everything we do here can be set in this more general context, but that would obscure its connections with recursion. (It is clear that already many workers in the field do not understand this connection.)

Exercises

6 Let

$$\mathbb{S} = \{a, b\} \quad \mathbb{T} = \{p, q\}$$

where a, b, p, q are distinct. Draw a picture of the poset $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ of all partial functions from \mathbb{S} to \mathbb{T} .

7 A While program operates in the context

$$X(1) : \mathbb{X}(1), \dots X(l) : \mathbb{X}(l)$$

that is it manipulates the registers $X(1), \dots X(l)$ which must hold values from the indicated sets or be empty.

- (a) Write down the state space of this program.
- (b) Re-do Exercise 5.

3 Recursion operator

The crucial trick is to view SPEC in a different way. We don't think of it as a requirement to be met (an equation to be solved) but as a way of converting one function into another.

3.1 DEFINITION. For SPEC the associated operator

$$\Gamma : (\mathbb{S} \rightarrow \mathbb{T}_\perp) \longrightarrow (\mathbb{S} \rightarrow \mathbb{T}_\perp)$$

is given by

$$\Gamma f s = \begin{cases} h s (f s^-) & \text{if } s \notin \mathbb{B} \\ g s & \text{if } s \in \mathbb{B} \end{cases}$$

for each $f \in (\mathbb{S} \rightarrow \mathbb{T}_\perp)$ and $s \in \mathbb{S}$. Here $s^- = k s$. ■

Of course, this operator Γ is determined by the given data g, h, k and \mathbb{B} , and these are assumed fixed.

Using Γ the problem posed by SPEC can be rephrased in a succinct manner. SPEC is asking for a fixed point of Γ , a function f such that $\Gamma f = f$. This observation together with the partial orderings hanging around more or less tell us how to solve the problem.

Before we get into that there is a minor problem that we should clear up. Given

$$f : \mathbb{S} \longrightarrow \mathbb{T}_\perp \quad s \in \mathbb{S} - \mathbb{B}$$

we have

$$\Gamma f s = h s t \text{ where } t = f s^- \text{ where } s^- = k s$$

using the given function h, k . But observe that $t = f s^-$ could be \perp , and then

$$\Gamma f s = h s \perp$$

which is what? The original data function

$$h : \mathbb{S} \longrightarrow \mathbb{T} \longrightarrow \mathbb{T}$$

needs to be extended to a function

$$h : \mathbb{S} \longrightarrow \mathbb{T}_\perp \longrightarrow \mathbb{T}_\perp$$

by selecting a value

$$hs\perp$$

for each $s \in \mathbb{S}$. The obvious way to do this is to let

$$hs\perp = \perp$$

and this is what we will do. There are other possibilities, but shortly you will see why this is the correct choice.

The SPEC give us an operator Γ , and to solve the specification it is sufficient to find a fixed point of Γ . To do this we observe that Γ has a couple of nice properties and then use a quite general technique. We go into the details of this in the next section. To conclude this section let's see how the Unwinding/Rewinding technique finds a fixed point.

Theorem 1.3 gives a description of the partial function we are interested in. We extend that to a total function by adding the nominal value \perp where necessary. We need to rephrase that description in a more convenient form.

Each $s \in \mathbb{S}$ has a length $|s|$ which we take as a natural number or ω . Thus the domain of definition of the original function is given by

$$s \in \mathbb{D} \iff |s| < \omega \iff (\exists i)[s_i \in \mathbb{B}]$$

using the descent sequence $(s_i \mid i < \omega)$ generated from s (by iterating k). Similarly

$$|s| = 0 \iff s \in \mathbb{B}$$

extracts the given subset \mathbb{B} of \mathbb{S} . By construction we have

$$|s| = 1 + |s^-|$$

for each s with $|s| \neq 0$ (using the convention that $1 + \omega = \omega$). For s with finite length the descent sequence of s continues until it bottoms out by meeting a member of \mathbb{B} . Using the length $|s|$ we see that

$$\bar{s} = s_{|s|}$$

is this bottoming out point. Notice also that $s = s_0$ and $s^- = ks = s_1$ bottom out at the same point, since

$$s = s_0, s_1, \dots, s_{|s|} \quad ks = s_1, \dots, s_{|s|}$$

are the descent chains of s and $s^- = ks$, respectively. In other words, we have

$$\overline{s^-} = \bar{s}$$

when both sides make sense.

Consider any $s \in \mathbb{S}$ with $0 < |s| < \omega$, and let $l = |s|$ be its length. Consider any component s_i of the descent chain with index $i < l$ (so that $s_i \neq \bar{s}$). Using the data function h we obtain a 1-placed function

$$hs_i : \mathbb{T} \longrightarrow \mathbb{T}$$

on \mathbb{T} . We can compose these functions to obtain a compound function

$$h[s] = hs_0 \circ \cdots \circ hs_{l-1}$$

on \mathbb{T} . Notice that

$$h[s] = hs \circ h[s^-]$$

when both sides make sense.

As a particular case of these construction, if $|s| = 1$ then

$$\bar{s} = s^- \quad h[s] = hs$$

hold.

Using this notation we have a function

$$f : \mathbb{S} \longrightarrow \mathbb{T}_\perp$$

given by

$$fs = \begin{cases} (h[s] \circ g)\bar{s} & \text{if } 0 < |s| < \omega \\ gs & \text{if } |s| = 0 \\ \perp & \text{if } |s| = \omega \end{cases}$$

for each $s \in \mathbb{S}$. This is precisely the function given by the U/R process.

3.2 THEOREM. *The function f generated by the unwinding/rewinding process applied to SPEC as given above is a fixed point of the associated operator Γ .*

Proof. For each $s \in \mathbb{S}$ we have

$$\Gamma fs = \begin{cases} hs(fs^-) & \text{if } |s| \neq 0 \\ gs & \text{if } |s| = 0 \end{cases} \quad fs^- = \begin{cases} (h[s^-] \circ g)\bar{s}^- & \text{if } 0 < |s^-| < \omega \\ gs & \text{if } |s^-| = 0 \\ \perp & \text{if } |s^-| = \omega \end{cases}$$

so that, since $\bar{s}^- = \bar{s}$ and $|s| = 1 + |s^-|$, we have

$$\Gamma fs = \begin{cases} (hs \circ h[s^-] \circ g)\bar{s} & \text{if } 1 < |s| < \omega \\ (hs \circ g)s & \text{if } |s| = 1 \\ hs\perp & \text{if } |s| = \omega \\ gs & \text{if } |s| = 0 \end{cases} = \begin{cases} (h[s] \circ g)\bar{s} & \text{if } 1 < |s| < \omega \\ (h[s] \circ g)s & \text{if } |s| = 1 \\ \perp & \text{if } |s| = \omega \\ gs & \text{if } |s| = 0 \end{cases} = fs$$

as required. Observe how the choice of $hs\perp$ is used. ■

It turns out that the unwinding/rewinding process is sufficient to solve any recursive specification and find a fixed point of the associated operator. In fact, when extended (in length and number of phases) the process will produce a fixed point of any suitable operator. Of course, the proof of Theorem 3.2 is a bit messy because it gets involved with some irrelevant details. The whole process is much smoother if we work in a more general situation using the properties of the operator which makes everything happen.

Exercises

8 Using the three data functions

$$\mathbb{P} \xrightarrow{g} \mathbb{T} \qquad \mathbb{N}, \mathbb{P}, \mathbb{T} \xrightarrow{h} \mathbb{T} \qquad \mathbb{N}, \mathbb{P} \xrightarrow{k} \mathbb{P}$$

suppose

$$\mathbb{N}, \mathbb{P} \xrightarrow{f} \mathbb{T}$$

is specified by a body recursion, that is

$$f(0, p) = gp \qquad f(x', p) = h(x, p, f(x, k(x, p)))$$

for $x \in \mathbb{N}, p \in \mathbb{P}$.

Show that this specification can be rephrased in the form of Definition 1.2 in at least two essentially different ways.

Which is the ‘better’ version?

4 Monotonicity

We know that $(\mathbb{S} \rightarrow \mathbb{T}_\perp)$ is a poset. The SPEC gives an operator Γ of type $(\mathbb{S} \rightarrow \mathbb{T}_\perp)'$, and there is a standard property of functions between posets which we should always look for.

4.1 LEMMA. *The associated operator Γ of SPEC is monotone.*

Proof. Consider a pair of functions

$$f_1, f_2 : \mathbb{S} \longrightarrow \mathbb{T}_\perp$$

with $f_1 \leq f_2$, that is

$$f_1 s = f_2 s \quad \text{or} \quad f_1 s = \perp$$

for each $s \in \mathbb{S}$. We must show that $\Gamma f_1 \leq \Gamma f_2$, that is

$$\Gamma f_1 s = \Gamma f_2 s \quad \text{or} \quad \Gamma f_1 s = \perp$$

for each $s \in \mathbb{S}$.

Consider any $s \in \mathbb{S}$. If $s \in \mathbb{B}$ then

$$\Gamma f_1 s = gs = \Gamma f_2 s$$

and we are done. Otherwise

$$\Gamma f_1 s = hst_1 \qquad \Gamma f_2 s = hst_2$$

where

$$t_1 = f_1 s^- \quad t_2 = f_2 s^-$$

are the relevant components. Since $f_1 \leq f_2$ there are two cases; either we have $t_1 = t_2 = t$ (say) or $t_1 = \perp$. In the first case we have

$$\Gamma f_1 s = h s t = \Gamma f_2 s$$

and we are done. In the second case we have

$$\Gamma f_1 s = h s \perp = \perp$$

and again we are done. ■

We are going to see that it is this monotonicity which goes a long way towards helping us find fixed points of Γ without having to get inside the thing and come out all covered in muck.

We have already observed that $(\mathbb{S} \rightarrow \mathbb{T}_\perp)$ is a poset. We now observe that it has a bottom. The function \perp given by

$$\perp s = \perp$$

for each $s \in \mathbb{S}$. (Here we have deliberately made the two symbols ‘ \perp ’ and ‘ \perp ’ slightly different, whereas often the same symbol is used.) Since $\perp \leq f$ for each $f \in (\mathbb{S} \rightarrow \mathbb{T}_\perp)$ we have $\Gamma \perp \leq \Gamma f$ by the monotonicity of Γ , and now we can hit this comparison again and again.

4.2 DEFINITION. For the SPEC with associated operator Γ set

$$f_i = \Gamma^i \perp$$

that is

$$f_0 = \perp \quad f_{i+1} = \Gamma f_i$$

for each $i < \omega$. ■

Since $\perp \leq f_1$ a simple induction gives

$$\perp = f_0 \leq f_1 \leq f_2 \leq \cdots \leq f_i \leq \cdots \quad (i < \omega)$$

and this ascending chain has a rather stable structure.

4.3 LEMMA. For each $s \in \mathbb{S}$ the ascending chain of values

$$\perp = f_0 s \leq f_1 s \leq f_2 s \leq \cdots \leq f_i s \leq \cdots$$

is eventually constant.

Proof. It could be that each value $f_i s$ is \perp , but then we have a constant chain. Otherwise there is some index j with $f_j s \neq \perp$, and then $f_j s = f_{j+1} s$, and hence $f_j s = f_i s$ for all $i \geq j$. ■

This observation enables us to put an upper bound on the generated chain of functions.

4.4 DEFINITION. Using the chain f_\bullet generated in Definition 4.2 we set

$$f_\omega s = \text{eventual value of } f_i s$$

as i increases. ■

Almost trivially we have $f_i \leq f_\omega$ for each $i < \omega$. A slightly longer argument shows that f_ω is the least upper bound of the f_i . (We make this notion precise in the next section.) There is also a far more important property of f_ω .

4.5 THEOREM. *The function f_ω is the least fixed point of Γ .*

Proof. We have to show two things; that f_ω is a fixed point, and it lies below each other fixed point. The proof of the second of these is slightly shorter.

Let f be any fixed point of Γ . We show

$$f_i \leq f$$

by induction on i . The base case, $i = 0$, is trivial (since $f_0 = \perp$). For the induction step, $i \mapsto i + 1$, since Γ is monotone the induction hypothesis gives

$$f_{i+1} = \Gamma f_i \leq \Gamma f = f$$

since f is a fixed point of Γ . Finally we have $f_\omega \leq f$ since f_ω is the least upper bound of the chain.

It remains to show that f_ω is a fixed point of Γ . To this end consider any $s \in \mathbb{S}$. We have

$$\Gamma f_\omega s = \begin{cases} hst & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases}$$

where $t = f_\omega s^-$ in the top clause. But

$$t = f_i s^-$$

for all sufficiently large i , and hence

$$\Gamma f_\omega s = \begin{cases} hs(f_i s^-) & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases} = \Gamma f_i s = f_{i+1} s \leq f_\omega s$$

for each such i . In fact, $f_\omega s = f_i s$ for all sufficiently large i , and hence

$$\Gamma f_\omega s = \Gamma f_i s = f_{i+1} s = f_\omega s$$

for all sufficiently large i .

This range of i may depend on s , but there is always enough of them to give

$$\Gamma f_{\omega} s = f_{\omega} s$$

and hence

$$\Gamma f_{\omega} = f_{\omega}$$

as required. ■

This result and its proof contains two important pieces of information. The minor piece gives a refinement of Theorem 3.2.

4.6 THEOREM. *For the SPEC the unwinding/rewinding algorithm produces the least fixed point of the associated operator Γ .*

The major piece of information is the insight.

Notice that the proof of Theorem 4.5 make no reference to the origins of Γ as a recursion operator. It uses only certain properties of Γ . Certainly the monotonicity is important, but a more subtle property is that certain generated sequences are eventually constant. This suggest there is a more general family of operators we might work with. To formulate the appropriate notion we have to do a little bit more work to get rid of the clumsy ‘eventually constant generated sequence’.

Exercises

9 Consider the specifications (a) of Examples 1.1.

- (a) Describe the operator $\Gamma : \mathbb{S} \longrightarrow \mathbb{S}_{\perp}$ associated with this specification.
- (b) Describe the functions

$$f_r = \Gamma^r \perp$$

for $r < \omega$.

10 Consider the specifications (b) of Examples 1.1. This is *not* an instance of the SPEC of Definition 1.2, and so Definition 3.1 does not apply. However, the idea can be modified.

(a) Show that the specification does give an Γ operator on an appropriate poset (formed using lifting). You should take some care to say what kind of functions Γ operates on, and remember the specification should ask for the fixed points of Γ .

- (b) Show that this operator Γ is monotone.
- (c) Describe the functions

$$f_r = \Gamma^r \perp$$

for $r < \omega$.

11 Consider the specification (c) of Examples 1.1. By Exercise 3 we know this has a unique solution and this solution is total.

- (a) Convert this specification into an operator Γ on a poset.
- (b) Show that Γ is monotone.
- (c) Describe the functions

$$f_r = \Gamma^r \perp$$

for $r < \omega$. (You might want to calculate f_r for a few values of r (say $r \leq 4$).

(d) Show that there is some smallish r such that f_r is the unique fixed point of Γ .

12 Consider the specification (d) of Example 1.1.

- (a) Rephrase this as an operator on an appropriate domain.
- (b) Re-do part (a) of Exercise 4.

13 Suggest a way of converting a while loop into a monotone operator Γ on a suitable domain.

What do the approximations $\Gamma^r \perp$ correspond to?

5 Domains

What are the features used in the proof of Theorem 4.5? There are two, one easy to pin down and one less so. Both are concerned with the operator Γ .

- Γ acts on a poset and is monotone.
- Γ interacts with ‘eventually constant sequences’ in a nice way.

The first of these is easy to put in a more general setting. The second is not so straight forward to handle, and we will see how this is done shortly.

Roughly speaking a domain is a poset which is ‘sufficiently complete’, and it is this second property which needs to be looked at in more detail.

Given a subset X of a poset (A, \leq) , and element $a \in A$ is an **upper bound** of X (in A) if $x \leq a$ for each $x \in X$. An element a is a **least upper bound** or **supremum** of X (in A) if it is an upper bound and lies below each upper bound of X . Trivially (since the comparison is antisymmetric) each set X can have at most one least upper bound. We write $\bigvee X$ for the supremum (when it exists).

There are various kinds of domains some of which are good for some jobs but not for others. There is no universally accepted definition of the ‘best’ notion of a domain, but every domain does satisfy the following.

5.1 DEFINITION. A **proto-domain** is a poset (A, \leq) which is ω -complete. That is, each ascending ω -chain

$$a_0 \leq a_1 \leq a_2 \leq \dots \leq a_i \leq \dots \quad (i < \omega)$$

of elements in A has a least upper bound. ■

Sometimes, in fact for most uses, a domain is required to have a bottom \perp , but every now and then this can be a nuisance. Thus this property has been omitted. Proto-domains are usually called by other names, such as ω -ccpo which stands for ω -chain-complete partial order.

The ω -completeness is enough for what we do here (and for many other uses). However, once we start a more general analysis we see that we really need a stronger completeness property.

5.2 DEFINITION. For a poset A a subset X is **directed** if it is non empty, and for each $x, y \in X$ there is some $z \in X$ with $x, y \leq z$.

A poset A is **directed complete** if each directed subset has a supremum. ■

Trivially, each ω -chain through a poset is directed, so each directed complete poset is a proto-domain. In fact, directed completeness is often taken as the least requirement on a domain. It is an easy exercise to show that a poset is directed complete exactly when it is chain complete, where here a chain may have any ordinal length.

(At this point we need a health warning. In some of the earlier literature – and some current literature – directed complete posets were referred to as *complete* posets when, of course, they need not be anything like complete. Also some people have taken it into their heads to write \sqsubseteq for the comparison on a domain. It is not clear why, but it seems to make them happy.)

Each proto-domain is a poset, and there is a standard way of comparing posets by monotone functions. However, these functions are not good enough for domain purposes, for we need to take into account the completeness properties.

5.3 DEFINITION. A monotone function

$$A \xrightarrow{f} B$$

between proto-domains is **proto-continuous** if

$$f\left(\bigvee X\right) = \bigvee (fx_i \mid i < \omega)$$

for each ascending chain

$$X = (x_i \mid i < \omega)$$

of elements of A .

A monotone function f , as above, between directed complete posets is **continuous** if

$$f\left(\bigvee X\right) = \bigvee (fx \mid x \in X)$$

holds for each directed subset X of A . ■

We need to do a little bit of work to check that these definitions make sense. Firstly, a monotone function f between proto-domains will convert an ascending chain X in the source into an ascending chain $f[X]$ in the target, so we can take the supremum of this chain of target values. Similarly, a monotone function between directed complete poset will convert a directed subset into a directed subset, so again we can take the supremum in the target.

Sometimes these kinds of continuity are referred to as **Scott-continuity** (after one of the founders of domain theory, Dana Scott). There seems little point in this when there is no other kind of continuity around to confuse the issue.

(As you should have spotted, there are some topological notions lurking behind all this. Sometimes different topologies are needed on the same carrier, and that can lead to different notions of continuity.)

It is easy to check that both these kinds of monotone functions are closed under composition. Thus we obtain a pair of categories (in fact subcategories of **Pos**).

5.4 DEFINITION. Let **Dpo** be the category of directed complete posets and continuous functions. ■

There is also a category of proto-domains and proto-continuous functions. However, we haven't given that a name because it doesn't need one. Any sensible person will realize that if we have to use categorical notions, then that category is the wrong thing to be looking at.

As you can see, the basic ideas of domain theory are rather simple. Furthermore, one of the most important results is very easy to prove, and we can do that now.

We need a bit of a preamble.

Consider a proto-continuous function

$$f : A \longrightarrow A$$

on a proto-domain. We say an element $a \in A$ is **inflated** by f if $a \leq fa$. The element a is a **fixed point** of f if $fa = a$ (so that a is simultaneously inflated and deflated). Notice that the bottom (if it exists) is inflated by every proto-continuous functions. The next result is most often applied to that element.

Suppose the element a is inflated by f . Then

$$a \leq fa$$

and hence

$$a \leq fa \leq f^2a$$

by monotonicity. By iterating f we obtain an ascending chain

$$(f^i a \mid i < \omega)$$

of elements of A , and this must have a supremum.

5.5 THEOREM. *As above let f be a proto-continuous function on a proto-domain A , and suppose the element a is inflated by f . Then the supremum*

$$b = \bigvee (f^i a \mid i < \omega)$$

is the least fixed point of f above a .

Proof. A use of the proto-continuity gives

$$fb = f \left(\bigvee (f^i a \mid i < \omega) \right) = \bigvee (f^{i+1} a \mid i < \omega) = b$$

to show that b is a fixed point of f .

Consider any other fixed point c above a . We have $a \leq c$ so that

$$a \leq fa \leq fc = c$$

and hence

$$f^i a \leq c$$

follows by induction on i . Thus $b \leq c$, as required. ■

That was a doddle wasn't it? What is more pleasing is that this result and proof is almost the whole of the content of Theorem 4.5. The only thing to sort out is how the current result handles 'eventually constant sequences'.

Consider the situation of Theorem 4.5. We have a discrete domain (that is, set) \mathbb{S} and a flat domain \mathbb{T}_\perp where we can now read 'domain' in the technical sense. Certainly both of these are proto-domains (and in fact directed complete). Observe that every monotone function

$$f : \mathbb{S} \longrightarrow \mathbb{T}_\perp$$

is proto-continuous and, in fact, continuous. This is because a directed subset of \mathbb{T}_\perp has no more than two elements.

Let $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ be the poset of monotone (and hence continuous) functions from the indicated source to the indicated target. We need to check that this poset has enough domain-like properties. To do this we go back to Lemma 4.3.

5.6 LEMMA. *Let*

$$f_0 \leq f_1 \leq f_2 \leq \cdots \leq f_i \leq \cdots \quad (i, \omega)$$

be an ascending ω -chain of members of $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$. Then for each $s \in \mathbb{S}$ the ascending chain of values

$$\perp = f_0 s \leq f_1 s \leq f_2 s \leq \cdots \leq f_i s \leq \cdots$$

is eventually constant.

The proof of this is identical to that of Lemma 4.3, for there we did not use the fact that the f_i were obtained in a certain way.

5.7 COROLLARY. *The poset $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ is a proto-domain.*

Proof. For each ascending chain $(f_i | i < \omega)$ we may set

$$f_\omega = \left(\bigvee (f_i | i < \omega) \right) s = \text{eventual value of } (f_i s | i < \omega)$$

for each $s \in \mathbb{S}$ to obtain the required supremum. ■

Clearly, we are going to apply Theorem 5.5 to the proto-domain

$$A = [\mathbb{S} \rightarrow \mathbb{T}_\perp]$$

and the function Γ on A . To do that we need the following.

5.8 LEMMA. *The associated operator Γ of SPEC is proto-continuous.*

Proof. By Lemma 4.1 this function Γ is monotone. The proof of the required continuity property is embedded in the proof of Theorem 4.5. Let $(f_i | i < \omega)$ be an ascending chain of members of $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ with supremum f_ω . Then for each $s \in \mathbb{S}$ we have

$$\Gamma f_\omega s = \Gamma f_i s$$

for all sufficiently large i (where the required largeness depends on s). Thus

$$\left(\Gamma \left(\bigvee (f_i | i < \omega) \right) \right) s = \Gamma f_\omega s = \bigvee (\Gamma f_i s | i < \omega) = \left(\bigvee (\Gamma f_i | i < \omega) \right) s$$

and hence

$$\Gamma \left(\bigvee (f_i | i < \omega) \right) = \bigvee (\Gamma f_i | i < \omega)$$

as required. ■

Bit of end chat

Exercises

14 Show that the poset

$$\mathbf{Dpo}[S, T]$$

of continuous maps between two objects of **Dpo** is itself and object of **Dpo**.

15 (a) Show that the associated proto-domain $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ of SPEC is, in fact, directed complete.

(a) Show that the associated operator Γ of SPEC is continuous.

Some solutions

For section 1

1 As in the U/R process, we consider the iterates

$$s_i = k^i s$$

of k on s and extract a subset $\mathbb{D} \subseteq \mathbb{S}$ by

$$s \in \mathbb{D} \iff (\exists i)[s_i \in \mathbb{B}]$$

for $s \in \mathbb{S}$. For $s \in \mathbb{D}$ the length $|s|$ of s is generated by

$$|s| = 0 \iff s \in \mathbb{B} \quad |s| = 1 + |ks| \quad \text{if } s \in \mathbb{D} - \mathbb{B} \quad |s| = \omega \iff s \notin \mathbb{D}$$

(for $s \in \mathbb{S}$). Thus we have

$$\begin{aligned} s \notin \mathbb{D} &\iff |s| = \omega \\ s \in \mathbb{D} &\iff |s| < \omega \\ s \in \mathbb{B} &\iff |s| = 0 \end{aligned}$$

and we can work entirely with $|\cdot|$.

It isn't too difficult to show that

$$fs = \begin{cases} k^{|s|}s & \text{if } |s| < \omega \\ \perp & \text{if } |s| = \omega \end{cases}$$

is the function produced by the U/R process. The proof of this is essentially the same as part (d) of the next solution, but easier. ■

2 (a) The nested f (as in f^2s^-) is the problem. To rephrase this as an instance of SPEC we would need a function h such that

$$hst = ft$$

for certain s, t . But this function h should be independent of f , and it isn't.

(b) We can allow the use of a function h which can consume a function f as an input, not just values fs of the function.

(c) For the first phase we can still unwind by setting

$$s_0 = s \quad s_{i+1} = ks_i$$

that is

$$s_i = k^i s$$

for $i < \omega$. As before, this unwinding continues until $s_i \in \mathbb{B}$. After that we track back through

$$\begin{aligned} t_0 &= fs_0 = f(fs_1) = ft_1 \\ t_1 &= fs_1 = f(fs_2) = ft_2 \\ t_2 &= fs_2 = f(fs_3) = ft_3 \\ &\vdots \end{aligned}$$

to find that this specification has the same solution as that of Exercise 1.

(d) Carrying over the notation of Solution 1 we show that the function given by

$$fs = \begin{cases} k^{|s|}s & \text{if } |s| < \omega \\ \perp & \text{if } |s| = \omega \end{cases}$$

is the function generated by the modified U/R process.

Let f be the function generated by this process. We know that the domain of definition of f is \mathbb{D} , the set of those $s \in \mathbb{S}$ with $|s| < \omega$. Thus it suffices to show

$$|s| < \omega \implies fs = k^{|s|}s$$

(for $s \in \mathbb{S}$), and to do this we proceed by induction on $|s|$. More precisely, we show that

$$[m] \quad (\forall s \in \mathbb{S})[|s| \leq m \implies fs = k^{|s|}s \in \mathbb{B}]$$

holds for all $m \in \mathbb{N}$, and we proceed by induction on m .

For the base case, $m = 0$, we have $|s| = 0$, so that $s \in \mathbb{B}$ and then

$$fs = s = k^{|s|}s = s \in \mathbb{B}$$

as required.

For the induction step, $m \mapsto m + 1$, we have

$$|s| = 1 + |ks|$$

where we may apply the induction hypothesis to $s^- = ks$. Thus

$$fs^- = k^{|ks|}(ks) = k^{1+|ks|}s = k^{|s|}s \in \mathbb{B}$$

for this s . But the specification gives

$$fs = f(fs^-) = f(k^{|s|}s) = k^{|s|}s$$

where the final equality holds since $k^{|s|}s \in \mathbb{B}$ (by the second part of the induction hypothesis).

(e) Consider the specification

$$fs = \begin{cases} f^2s^- & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases}$$

where $g : \mathbb{S} \longrightarrow \mathbb{S}$ is some known function. The first unwinding phase is just as before, but then things can get much wilder. As an example suppose we find that $|s| = 3$. Thus we generate a template

$$\begin{aligned} f s_0 &= f(f s_1) \\ f s_1 &= f(f s_2) \\ f s_2 &= f(f s_3) \end{aligned}$$

and then

$$f s_3 = g s_3 = (g \circ k^3) s = s^+ \quad (\text{say})$$

by the base clause of the specification. We now require

$$f s_2 = f(f s_3) = f s^{++}$$

which forces us to open another unwinding phase, and we have no idea how long that will take. Once that is over to produce a value

$$f s_2 = s^{++}$$

say, we require

$$f s_1 = f(f s_2) = f s^{+++}$$

which requires yet another unwinding phase.

In general this process will require many nestings of unwindings. It will produce a function

$$f s = (g \circ k^{e(l)} \circ g \circ k^{e(l-1)} \circ g \cdots g \circ k^{e(1)} \circ g \circ k^{e(0)}) s$$

for s in the appropriate domain of definition. It is not at all clear how many exponents $e(0), \dots, e(l)$ will be generated nor what the size of these might be.

All this can be indexed by an ordinal, but one that is larger than ω .

However, it isn't always this bad, as the next exercise shows. ■

3 A rather tedious argument shows that the obvious evaluation process does produce a total function, that is the evaluation process does terminate for each input. A better way of organizing this is given in Exercise 11.

Here we show that

$$f x = \begin{cases} x - 10 & \text{if } 100 < x \\ 91 & \text{if } x \leq 100 \end{cases}$$

(for $x \in \mathbb{N}$) is the unique solution of the specification. This can be done quite quickly.

For $x > 100$ the specification immediately gives

$$f x = x - 10$$

as required. In particular, $f 101 = 91$.

Now consider $90 \leq x \leq 100$. We have $101 \leq x + 11$ and hence

$$fx = f^2(x + 11) = f(x + 1)$$

using the two clauses of the specification. In particular,

$$fx = f(x + 1) = f(x + 2) = \dots = f101 = 91$$

for these x . Notice that we have $fx = 91$ for all $90 \leq x \leq 101$.

Finally, consider $x < 90$. There is a unique r with

$$90 \leq x + 11(r + 1) \leq 101$$

and then

$$fx = f^{2(r+1)}(x + 11(r + 1))$$

by repeated use of the top clause of the specification. but now

$$fx = f^{2r+1}(f(x + 11(r + 1)))f^{2r+1}91 = 91$$

by the calculations above. ■

4 (a) This kind of verification can be done better with a certain idea to be introduced later. Thus we postpone this part of the solution.

(b) For an input $x \in \mathbb{R}$ the U phase will generate

$$x, x - 1, x - 2, x - 3, \dots, x - r, \dots$$

and this will continue until stage $r \in \mathbb{N}$ with $x - r = 0$. Thus if $x \in \mathbb{R} - \mathbb{N}$ then the unwinding will continue for ever. If $x \in \mathbb{N}$ then this phase will bottom out will produce a function

$$f : \mathbb{N} \longrightarrow \mathbb{R}$$

when combined with a modified rewinding phase.

By repeated use of the specification we have

$$fx = r + f^{2r}(x - r)$$

for each $x \in \mathbb{R}$ and $r \in \mathbb{N}$ provided none of $x, x - 1, x - 2, \dots, x - r - 1$ is 0. Thus, for $x \in \mathbb{N}$ we have

$$fx = xr + f^{2x}0 = x$$

since $f0 = 0$.

This shows that the partial function

$$f : \mathbb{R} \longrightarrow \mathbb{R}_\perp$$

given by

$$fx = \begin{cases} x & \text{if } x \in \mathbb{N} \\ \perp & \text{if } x \in \mathbb{R} - \mathbb{N} \end{cases}$$

meets the specification.

Each other function which meets the specification (including (i, ii, iii)) is an extension of this particular solution. ■

5 Let P be the given while loop. Let Q be the semantic function of the component instruction Q and let \mathcal{P} be that of P . Both of these operate on states σ taken from the associated state space \mathbb{S} . Let $\mathbb{B} \subseteq \mathbb{S}$ be the set of states for which the boolean condition B is false. Then

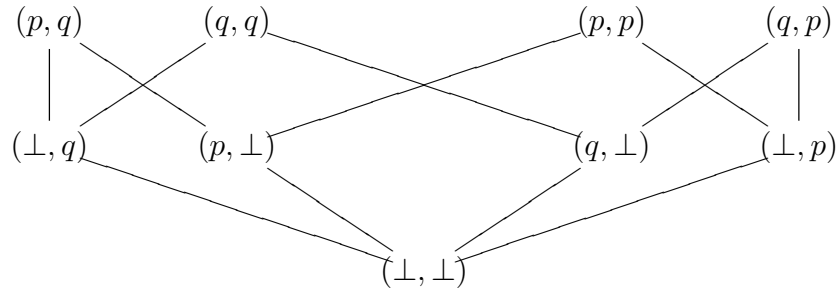
$$\mathcal{P}\sigma = \begin{cases} \mathcal{P}\sigma^- & \text{if } \sigma \notin \mathbb{B} \\ \sigma & \text{if } \sigma \in \mathbb{B} \end{cases}$$

where $\sigma^- = Q\sigma$ (for $\sigma \in \mathbb{S}$) is the recursive specification of \mathcal{P} .

Or is it? The program semantics must handle ‘undefined’ values or ‘empty’ registers. This requires a slight modification of the description above. Once this is done the while loop behaves more or less like the simplified U/R process associated with part (a) of Examples 1.1. ■

For section 2

6 We write (r, s) for the function f with $fa = r$ and $fb = s$. Thus we can think of $[\mathbb{S} \rightarrow \mathbb{T}_\perp]$ as a set of ordered pairs.



The poset is a kind of crown with a bottom. ■

7 (a) Naively we think of a state as a tuple

$$\sigma = (\sigma(1), \dots, \sigma(l))$$

where $\sigma(i) \in \mathbb{X}(i)$ for each $1 \leq i \leq l$. This suggests that

$$\mathbb{X}_1 \times \dots \times \mathbb{X}_l$$

is the state space. However, a register may be empty, so we must allow \perp as a possible value. In other words, a state is a tuple, as above, but where each component may be \perp . Thus

$$\mathbb{S} = \mathbb{X}(1)_\perp \times \dots \times \mathbb{X}(l)_\perp$$

is the state space. This is a product of flat domains but is not itself flat.

(b) The boolean expression B , the component instruction Q , and the while loop P produce functions

$$\mathbb{S} \xrightarrow{\mathcal{B}} \{\text{true}, \text{false}\}_\perp \quad \mathbb{S} \xrightarrow{\mathcal{Q}} \mathbb{S} \quad \mathbb{S} \xrightarrow{\mathcal{P}} \mathbb{S}$$

respectively. Notice how the lifted booleans are used here. With these the specification of \mathcal{P} is

$$\mathcal{P}\sigma = \begin{cases} \mathcal{P}\sigma^- & \text{if } \mathcal{B}\sigma = \text{true} \\ \sigma & \text{if } \mathcal{B}\sigma = \text{false} \\ (\perp, \dots, \perp) & \text{if } \mathcal{B}\sigma = \perp \end{cases}$$

where $\sigma^- = \mathcal{Q}\sigma$ (for $\sigma \in \mathbb{S}$).

Actually, this is still not quite correct. We probably want the program to abort if $\mathcal{B}\sigma = \perp$ is reached. This can be built into the semantics, but need a more sophisticated analysis. ■

For section 3

8 For the first way we hide the parameters p . Let

$$\mathbb{G} = [\mathbb{P} \rightarrow \mathbb{T}]$$

and let

$$G : \mathbb{N} \longrightarrow \mathbb{G} \quad H : \mathbb{N} \longrightarrow \mathbb{G}'$$

be given by

$$Gxp = gp \quad Hx'\phi p = h(x, p, \phi p^+)$$

for $x \in \mathbb{N}, p \in \mathbb{P}, \phi \in \mathbb{G}$ and where $p^+ = k(x, p)$. The value $H0\phi p$ doesn't matter. With these we have

$$fx = \begin{cases} Hx(fx^-) & \text{if } x \neq 0 \\ Gx & \text{if } x = 0 \end{cases}$$

where $x^- = x - 1$. To verify this merely evaluate $f0$ and fx' at p .

For the second way we use the pair $s = (x, p)$ as the recursion variable. With

$$\mathbb{S} = \mathbb{N} \times \mathbb{P}$$

let

$$G : \mathbb{S} \longrightarrow \mathbb{T} \quad H : \mathbb{S} \longrightarrow \mathbb{T}' \quad K : \mathbb{S} \longrightarrow \mathbb{S}$$

be such that

$$G(x, p) = gp \quad H(x', p)t = h(x, p, t) \quad K(x', p) = (x, k(x, p))$$

and let $\mathbb{B} \subseteq \mathbb{S}$ be given by

$$(x, p) \in \mathbb{B} \iff x = 0$$

for $x \in \mathbb{N}, p \in \mathbb{P}, t \in \mathbb{T}$. The other values of the function don't matter. With these we have

$$f_s = \begin{cases} Hs(fs^-) & \text{if } x \notin \mathbb{B} \\ Gs & \text{if } s \in \mathbb{B} \end{cases}$$

where $s^- = Ks$. To verify this merely evaluate $f(0, p)$ and $f(x', p)$.

The first method has certain advantages for theoretical work. However, it uses functions as inputs to other functions. The second method is better for practical evaluation of the f . ■

For section 4

9 (a) The operator

$$\Gamma \in (\mathbb{S} \rightarrow \mathbb{S}_\perp)'$$

is given by

$$\Gamma f_s = \begin{cases} fs^- & \text{if } s \notin \mathbb{B} \\ s & \text{if } s \in \mathbb{B} \end{cases}$$

for $f : (\mathbb{S} \rightarrow \mathbb{S}_\perp), s \in \mathbb{S}$. As usual we set $s^- = ks$ (for $s \in \mathbb{S}$).

(b) We use the length $|s|$ of an input $s \in \mathbb{S}$ as described in Solution 1. We show

$$f_r s = \begin{cases} k^{|s|} s & \text{if } |s| < r \\ \perp & \text{if } r \leq |s| \end{cases}$$

for $r < \omega$ and $s \in \mathbb{S}$. We proceed by induction on r . The base case, $r = 0$, is immediate since $0 \leq |s|$ for every $s \in \mathbb{S}$. For the induction step, $r \mapsto r + 1$, we have

$$f_{r+1} s = \Gamma f_r s = \begin{cases} f_r(ks) & \text{if } s \notin \mathbb{B} \\ s & \text{if } s \in \mathbb{B} \end{cases} = \begin{cases} k^{|ks|}(ks) & \text{if } |s| \neq 0 \text{ and } |ks| < r \\ \perp & \text{if } |s| \neq 0 \text{ and } r \leq |ks| \\ s & \text{if } |s| = 0 \end{cases}$$

using the induction hypothesis for the second equality. But now we remember that

$$0 < |s| < \omega \implies |s| = 1 + |ks| \quad |s| = \omega \iff |ks| = \omega$$

so that

$$f_{r+1} s = \begin{cases} k^{1+|ks|} s & \text{if } |s| \neq 0 \text{ and } |ks| < r \\ \perp & \text{if } |s| \neq 0 \text{ and } r \leq |ks| \\ s & \text{if } |s| = 0 \end{cases} = \begin{cases} k^{|s|} s & \text{if } 0 < |s| < r + 1 \\ \perp & \text{if } r + 1 \leq |s| \\ s & \text{if } |s| = 0 \end{cases}$$

which gives the required result (by combining the top and bottom clauses). ■

10 (a) As always, we must deal with partial functions, and we do this by allowing the required function f to output \perp (to indicate ‘undefined’). In this particular specification we have a compound f^2s which, if $fs = \perp$, unravels to $f\perp$. In other words, in this case we must allow f to consume \perp as well. Thus we can not look for a function $f : \mathbb{S} \longrightarrow \mathbb{S}_\perp$. We must look for a function

$$f : \mathbb{S}_\perp \longrightarrow \mathbb{S}_\perp$$

where both source and target are lifted. The crucial trick is not to deal with all functions on \mathbb{S}_\perp but only the monotone functions, that is those functions f such that

$$ft \leq fs$$

for all $s, t \in \mathbb{S}_\perp$ with $t \leq s$. In other words, either $f\perp = \perp$ or $fs = f\perp$ for all $s \in \mathbb{S}_\perp$.

For convenience we let

$$\mathbb{S}_\perp' = [\mathbb{S}_\perp \rightarrow \mathbb{S}_\perp]$$

be the set of all monotone functions on \mathbb{S}_\perp . You should take care to distinguish this from

$$\mathbb{S}'_\perp = [\mathbb{S} \rightarrow \mathbb{S}]_\perp$$

which is not the same domain. Similarly

$$\mathbb{S}_\perp'' = [\mathbb{S}_\perp' \rightarrow \mathbb{S}_\perp'] = [[\mathbb{S}_\perp \rightarrow \mathbb{S}_\perp] \rightarrow [\mathbb{S}_\perp \rightarrow \mathbb{S}_\perp]]$$

and it is this domain that houses the required operator Γ . Thus we define Γ by

$$\Gamma fs = \begin{cases} f^2s^- & \text{if } s \in \mathbb{S} - \mathbb{B} \\ s & \text{if } s \in \mathbb{B} \\ \perp & \text{if } s = \perp \end{cases}$$

(for each $f \in \mathbb{S}_\perp'$, $s \in \mathbb{S}_\perp$). As usual we have $s^- = ks$.

(b) Consider a pair of monotone functions

$$f_1, f_2 : \mathbb{S}_\perp \longrightarrow \mathbb{S}_\perp$$

with $f_1 \leq f_2$. Thus

$$f_1s \leq f_2s$$

for each $s \in \mathbb{S}$. We must show that $\Gamma f_1 \leq \Gamma f_2$, that is

$$\Gamma f_1s \leq \Gamma f_2s$$

for each $s \in \mathbb{S}$. Looking at Γ we see that only an input $s \in \mathbb{S} - \mathbb{B}$ might cause a problem. Thus we must show that

$$f_1(f_1s^-) \leq f_2(f_2s^-)$$

for such an input s . But

$$f_1 s^- \leq f_2 s^-$$

since $f_1 \leq f_2$, and hence

$$f_1(f_1 s^-) \leq f_1(f_2 s^-) \leq f_2(f_2 s^-)$$

by the monotonicity of f_1 and a second use of the comparison $f_1 \leq f_2$.

(c) We have the length $|s|$ of $s \in \mathbb{S}$, and it is convenient to let $|\perp| = \omega$. Just as in Solution 9, we find that

$$f_r s = \begin{cases} k^{|s|} s & \text{if } |s| < r \\ \perp & \text{if } r \leq |s| \end{cases}$$

for $r < \omega$ and $s \in \mathbb{S}_\perp$. The crucial observation is that for $|s| < \omega$ we have $|k^{|s|} s| = 0$, and then the previous proof is easily modified. ■

11 (a) As explained in Solution 11 we convert the specification into a operator Γ on \mathbb{N}_\perp'' where \mathbb{N}_\perp' is the poset of monotone functions on \mathbb{N}_\perp . Thus we set

$$\Gamma f x = \begin{cases} f^2(x + 11) & \text{if } x \leq 100 \\ x - 10 & \text{if } 100 < x \\ \perp & \text{if } x = \perp \end{cases}$$

for $f \in \mathbb{N}_\perp'$, $x \in \mathbb{N}_\perp$. (Do not confuse the size comparison on \mathbb{N} with the domain comparison on \mathbb{N}_\perp . It is the size comparison that is used in the clauses above.)

(b) This is almost the same as the verification of Solution 10(b).

(c) We have

$$f_0 x = \perp$$

for all $x \in \mathbb{N}_\perp$, and a simple calculation give

$$f_1 x = \begin{cases} x - 10 & \text{if } 101 \leq x \\ \perp & \text{otherwise} \end{cases}$$

(for $x \in \mathbb{N}_\perp$). We show there is a descending chain

$$l_0 > l_1 > \dots > l_r > \dots$$

of integers such that

$$f_{r+1} x = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } l_r \leq x \leq 101 \\ \perp & \text{otherwise} \end{cases}$$

(for $r < \omega$, $x \in \mathbb{N}_\perp$). In particular, eventually we have $l_r \leq 0$ and then f_{r+1} is the required maximal function.

We already have $l_0 = 101$, so it suffices to see what happen on a pass through Γ .

Consider a function f given by

$$fx = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } l \leq x \leq 101 \\ \perp & \text{otherwise} \end{cases}$$

for some l . We then have

$$\Gamma fx = \begin{cases} x - 10 & \text{if } 101 \leq x \\ f^2(x + 11) & \text{if } x \leq 100 \\ \perp & \text{if } x = \perp \end{cases} = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } x = 101 \\ f(x + 1) & \text{if } [*] \\ f91 & \text{if } l \leq x + 11 \leq 101 \\ \perp & \text{otherwise} \end{cases}$$

by a use of the description of f . The missing condition $[*]$ on x is

$$x \leq 100 \text{ and } 102 \leq x + 11$$

which is just

$$91 \leq x \leq 100$$

and hence

$$\Gamma fx = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } x = 101 \\ f(x + 1) & \text{if } 91 \leq x \leq 100 \\ f91 & \text{if } l - 11 \leq x \leq 90 \\ \perp & \text{otherwise} \end{cases}$$

which we must simplify further. First of all

$$f91 = \begin{cases} 91 & \text{if } l \leq 91 \\ \perp & \text{otherwise} \end{cases}$$

and this gives the value Γfx for $l - 11 \leq x \leq 90$. Similarly we have

$$f(x + 1) = \begin{cases} x - 9 & \text{if } 102 \leq x + 1 \\ 91 & \text{if } l \leq x + 1 \leq 101 \\ \perp & \text{otherwise} \end{cases} = \begin{cases} x - 9 & \text{if } 101 \leq x \\ 91 & \text{if } l - 1 \leq x \leq 100 \\ \perp & \text{otherwise} \end{cases}$$

and this is used only for $91 \leq x \leq 100$. In particular, the top clause is never applicable. Putting these together we have

$$\Gamma fx = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } x = 101 \\ 91 & \text{if } [**] \\ 91 & \text{if } [***] \\ \perp & \text{otherwise} \end{cases}$$

where the two conditions $[**]$ and $[***]$ are

$$91 \leq x \leq 101 \text{ and } l - 1 \leq x \quad l - 11 \leq x \leq 90 \text{ and } l \leq 91$$

respectively. Thus is we set

$$l^- = \begin{cases} l - 1 & \text{if } 91 < l \\ l - 11 & \text{if } l \leq 90 \end{cases}$$

and then

$$\Gamma f x = \begin{cases} x - 10 & \text{if } 102 \leq x \\ 91 & \text{if } l^- \leq x \leq 101 \\ \perp & \text{otherwise} \end{cases}$$

for each $x \in \mathbb{N}_\perp$, to give the required result.

(d) Starting from $l_0 = 101$ we may set

$$l_{r+1} = l_r^-$$

for each $r < \omega$. Then $l_{10} = 91$ and

$$l_{10+i} = 91 - 11i$$

so that $l_{18} = 3$ and $l_{19} < 0$. This shows that f_{20} is the fixed point of Γ . ■

12 (a) As explained in Solution 11 we convert the specification into an operator $\Gamma : \mathbb{R}_\perp''$ given by

$$\Gamma f x = \begin{cases} 1 + f^2(x - 1) & \text{if } x \in \mathbb{R} - \{0\} \\ 0 & \text{if } x = 0 \\ \perp & \text{if } x = \perp \end{cases}$$

for $f \in \mathbb{R}_\perp'$, $x \in \mathbb{R}_\perp$.

(b) It suffices to show that each of the functions (i, ii, iii) when canonically extended to a function in \mathbb{R}_\perp' (by $f\perp = \perp$) gives a fixed point of Γ . In fact, we can combine the three verifications into one more general result.

Each $x \in \mathbb{R}$ can be written

$$x = [x] + \langle x \rangle$$

where $x = [x]$ is the integer floor of x and $\langle x \rangle$ is the rational part. Thus

$$x = [x] \in \mathbb{Z} \quad 0 \leq \langle x \rangle < 1$$

for each $x \in \mathbb{R}$. Consider any function $\ell \in [0, 1)'$ with $\ell 0 = 0$ and $\ell^2 = \ell$. We show that the function f given by

$$f x = [x] + \ell \langle x \rangle$$

for $x \in \mathbb{R}$ and $f\perp = \perp$ is a fixed point of Γ . The functions

$$(i) \quad \ell y = y \quad (ii) \quad \ell y = 0 \quad (iii) \quad \ell y = 1/2$$

for $y \in [0, 1)$ give the three particular cases.

Consider this ℓ -induced function. Using the conditions on ℓ , for each $x \in \mathbb{R}$ we have

$$\lfloor fx \rfloor = \lfloor x \rfloor \quad \langle fx \rangle = \ell \langle x \rangle$$

and hence

$$f^2x = \lfloor fx \rfloor + \ell \langle fx \rangle = \lfloor x \rfloor + \ell^2 \langle x \rangle = \lfloor x \rfloor + \ell \langle x \rangle = fx$$

holds. In particular

$$f^2(x-1) = f(x-1) = \lfloor x-1 \rfloor + \ell \langle x-1 \rangle = \lfloor x \rfloor - 1 + \ell \langle x \rangle$$

so that

$$1 + f^2(x-1) = fx$$

holds for each $x \in \mathbb{R}$. Notice also that $f0 = 0$ and $f\perp = \perp$. Thus we have

$$\Gamma fx = \left\{ \begin{array}{ll} 1 + f^2(x-1) & \text{if } x \in \mathbb{R} - \{0\} \\ 0 & \text{if } x = 0 \\ \perp & \text{if } x = \perp \end{array} \right\} = \left\{ \begin{array}{ll} fx & \text{if } x \in \mathbb{R} - \{0\} \\ 0 & \text{if } x = 0 \\ \perp & \text{if } x = \perp \end{array} \right\} = fx$$

for each $x \in \mathbb{R}_\perp$, as required. ■

13 We use the analysis of Solution 7. The state space \mathbb{S} is a product of flat domains and the associated operator

$$\Gamma : \mathbb{S}' \longrightarrow \mathbb{S}'$$

is given by

$$\Gamma \mathcal{P}\sigma = \left\{ \begin{array}{ll} \mathcal{P}\sigma^- & \text{if } \mathcal{B}\sigma = \text{true} \\ \sigma & \text{if } \mathcal{B}\sigma = \text{false} \\ \hat{\perp} & \text{if } \mathcal{B}\sigma = \perp \end{array} \right.$$

for each $\mathcal{P} \in \mathbb{S}'$, $\sigma \in \mathbb{S}$. As before we have $\sigma^- = \mathcal{Q}\sigma$. Here we have written $\hat{\perp}$ for the constant state (\perp, \dots, \perp) . This is the bottom element of \mathbb{S} .

Using the function $\perp \in \mathbb{S}'$ where

$$\perp\sigma = \hat{\perp}$$

for each state σ , we find that

$$\Gamma^r \perp\sigma$$

is the result of running the program P from state σ and allowing no more than r passes through the loop. ■

For section 5

14 Given the two domains S, T let F be a directed family of continuous functions $f : S \longrightarrow T$. For each $s \in S$ the subset

$$\{fs \mid f \in F\}$$

of T is directed. Thus we may set

$$gs = \bigvee \{fs \mid f \in F\}$$

to produce a function $g : S \longrightarrow T$. We must show that g is continuous and is the supremum of F .

For $r \leq s$ from S we have

$$gr = \bigvee \{fr \mid f \in F\} \leq \bigvee \{fs \mid f \in F\} = gs$$

to show that g is monotone.

Consider directed subset X of S . Then

$$g(\bigvee X) = \bigvee \{f(\bigvee X) \mid f \in F\} = \bigvee \{fx \mid f \in F, x \in X\} = \bigvee \{gx \mid x \in X\}$$

to give the required continuity. The first equality merely unravels the definition of g . The second equality first uses the given continuity of each f and then combine the resulting supremum of suprema into a single supremum. The third equality rephrases this doubly indexed supremum by attacking those components with the same index x .

To show that g is the supremum of F first observe that g is certainly above each $f \in F$. Consider any other upper bound h of F . For each $s \in S$ we have $fs \leq hs$, and hence $gs \leq hs$ (since gs is defined as a supremum). Since this holds for each s we have $g \leq h$, as required. ■

15 (a) This is an immediate consequence of Exercise 14.

(b) Consider any directed family F of monotone function $f : \mathbb{S} \longrightarrow \mathbb{T}_\perp$. We investigate what $\bigvee F$ might be.

Consider any $s \in \mathbb{S}$. Then either $fs = \perp$ for all $f \in F$ or there is some $f \in F$ with $fs \in \mathbb{T}$. Suppose the second case holds. Since F is directed, for each $g \in F$ there is some $h \in F$ with $f, g \leq h$, and hence

$$gs \leq hs = fs$$

(since \mathbb{T} is discrete). This show that the set of values

$$\{fs \mid f \in F\}$$

is either $\{\perp\}$ or $\{\perp, t\}$ or $\{t\}$ for some $t \in \mathbb{T}$. Thus we have

$$(\bigvee F)_s = fs$$

for at least one $f \in F$ and, in fact, for a cofinal subset of such f .

We can now argue as in the proof of Theorem 4.5 and Lemma 5.8.

For a directed family $F \subseteq [\mathbb{S} \rightarrow \mathbb{T}_\perp]$ and $s \in \mathbb{S}$ we have

$$\Gamma \left(\bigvee F \right)_s = \begin{cases} hst & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases}$$

where $t = (\bigvee f)s^-$ in the top clause. But

$$t = fs^-$$

for a cofinal family of members f of F . Hence, for any such f we have

$$\begin{aligned} \Gamma \left(\bigvee F \right)_s &= \begin{cases} hs(fs^-) & \text{if } s \notin \mathbb{B} \\ gs & \text{if } s \in \mathbb{B} \end{cases} \\ &= \Gamma fs \leq \bigvee \{ \Gamma fs \mid f \in F \} = \left(\bigvee \{ \Gamma f \mid f \in F \} \right)_s \end{aligned}$$

where the last equality uses the definition of the supremum operation on the domain. This gives

$$\Gamma \left(\bigvee F \right) \leq \bigvee \{ \Gamma f \mid f \in F \}$$

and the converse comparison is immediate. ■