数理解析研究所講究録
454 巻 1982 年 229-377

LECTURES
ON A
MATHEMATICAL THEORY
OF
COMPUTATION

by

Dana S. Scott

University of Oxford
Mathematical Institute
24-29 St. Giles
Oxford OX1 3LB

Michaelmas Term 1980

Preliminary Version

Completed November 1980

Revised May 1981

TABLE OF CONTENTS

INTRODUCTION

These notes were written in conjunction with the lectures delivered by me for the Semantics of Programming Languages sequence during Michaelmas Term 1980 at Oxford. I started writing around the first week of October and finished at the end of November. The purpose of the course was to provide the foundations needed for the method of denotational semantics; in particular, I wanted to make the connections with recursive function theory more definite and to show explicit, effectively given solutions to domain equations. Roughly, these chapters cover the first half of the book of J.E. Stoy. I plan soon to expand the notes into a book by adding additional chapters on other theoretical topics that time did not permit me to cover in one eight-week term.

When I started writing Lecture I in October, I did not know what the later lectures would contain: I could see no further ahead than part of Lecture III in the beginning. The lectures had to be typed in advance of the class meetings, however, so there was at the time of composition no opportunity for second thoughts of any major proportions: I had to write the text straight through. As a consequence there are many remarks I would like to transpose and many additional points of explanation I see are needed; further worked examples and easier exercises are also required. During the spring, after receiving many helpful comments, I was able to introduce a few changes in the text and make some necessary corrections. However, a complete retyping was impossible. Nevertheless, this preliminary version of the book seems to provide a quite detailed introduction and is sufficient to exhibit the scope of the approach and several applications.

The idea of using neighbourhood systems to give set-theoretical representations of domains had been in the back of

my mind for some time in connection with specific examples. But the thought that a systematic development along these lines might be easier to follow than the more abstract lattice-theoretic and topological approach used by myself and others in many publications only came to me during the IFIP Working Group 2.2 meeting in Copenhagen in mid-June 1980. I gave a brief public presentation at ICALP '80 in Holland in mid-July.

One large mistake I have made is to de-emphasize partial orderings too much, since at the right point the concepts and the language are in fact helpful. The basic plan is that, instead of axiomatizing the theory using partial orderings, the necessary facts come out as *theorems*. For a neighbourhood system $\mathcal{D}$, the set of elements $|\mathcal{D}|$, which consists of filters, *is* naturally partially ordered. And approximable mappings naturally *preserve* the ordering. And so on. The advantage I see from the point of view of exposition is that properties can be brought out one at a time instead of having to put them down all in advance of any experience with the ideas. My own feeling after writing these chapters is that the plan has worked out far better than I could have dared to hope. I was especially glad that I could generate so many *exercises*, and I hope eventually to provide many more. One interesting place at which partial orderings prove their usefulness is in visualizing domains. As it stands now the text does not contain enough in the way of pictures. This will have to be remedied in a future version. Undoubtedly to include enough explanation, several of the lectures will have to be sub-divided into separate chapters.

One major improvement is needed: to bring Exercise 2.22 into the main text. I did not know in advance how often I would need this result for giving (easy) set-theoretical characterizations of domains and structure on them. This will be an easy repair, but it will cause quite a bit of rewriting. Clearly

much more has to be said about the interplay between elements and neighbourhoods. In particular, the character of the elements of a domain, like the power set of a set, has not been sufficiently illustrated, and quite a bit of expansion on this topic is also needed.

Finally I have to explain that I had no time whatsoever to put in references and a bibliography. The ideas I have used have occurred to many, many people who have worked on domains, and I do not wish to claim originality. I *am* claiming some advantage to my style of representation, but I fully realize that a published version will have to have detailed historical references and notes at the ends of the lectures. Needless to say I should very much appreciate any advice or criticism from readers of this preliminary version.

I would like to give a warm word of thanks to the many people who have already commented on the preliminary text both at Oxford and in Boston, where I gave lectures. Very special thanks are due to Steve Comer and Steve Brookes, who spent many hours proof reading the typescripts. The biggest word of thanks, however, is reserved for Elsie Hinkes who, under very considerable pressure, did a wonderful job of typing.

Dana S. Scott
Merton College
Oxford

May 1981

LECTURE I

DOMAINS GIVEN BY NEIGHBOURHOODS

Often an object (or element) can be determined by a selection of its properties. Often it is also the case that it is easier (more convenient, more elementary) to think of these properties than it is to think of the elements themselves. Let us term the properties under consideration *neighbourhoods*, the family of those allowed a *neighbourhood system*. Generally, the collection of these neighbourhoods is, for one reason or another, somewhat restricted; that is, a completely arbitrary property may not be allowable as a neighbourhood. Therefore, the elements determined by selections of neighbourhoods may not be as separable into the discrete objects common to the classical view of set theory. This is particularly true in working with infinite objects: it is hard to specify an infinite element completely. The theory of elements to be studied here, then, is going to permit *partial* elements as well as *total* elements, and each neighbourhood system will define a *domain* of such elements.

Since we may wish to use a neighbourhood system to introduce elements not previously investigated, the neighbourhoods do not have to be regarded as sets of the as-yet-to-be-defined objects. We can take a non-empty set Δ of *tokens* (or "traces") that function as "parts" of elements - or even as parts of "descriptions" of elements. Then a neighbourhood is a subset $X \subseteq \Delta$ containing all those tokens that provide sufficient information when taken together to "approximate" a possible element up to a certain "degree". All these words in inverted commas are vague, and in any case we shall have at the start only a *qualitative* theory of "degree of approximation". A token should be considered as a very "rough" representative of an element, and a neighbourhood should be regarded as "smoothing out" irrelevant details by grouping together *all* those representatives sharing some common feature. One neighbourhood, then,

may be only a very incomplete specification of an (ideal) element; fuller specifications can be secured by taking "convergent" sequences of neighbourhoods. Even then convergence need not be to a total element.

Let us call the family of allowed neighbourhoods $\mathcal{D}$; it is a family of subsets of the set Δ. An obvious first question is: when are two neighbourhoods $X, Y \in \mathcal{D}$ neighbourhoods of the "same" element? This question of course generalizes to a (finite) sequence of neighbourhoods. This property we will call the *consistency* of the sequence of neighbourhoods. By definition this will mean that the given neighbourhoods all contain a common neighbourhood in $\mathcal{D}$. That is, for X, Y to be consistent, there must be a $Z \in \mathcal{D}$ with $Z \subseteq X$ and $Z \subseteq Y$. This is not a very informative definition, but it has something of the flavour of a notion of consistency insofar as it can be expressed within $\mathcal{D}$. When consistency holds it seems reasonable enough at first glance to say that the *intersection* $X \cap Y$ is also an approximation to this common element. If this is reasonable, then $X \cap Y$ should also be regarded as a neighbourhood. This assumption has many consequences, but as a preliminary theory of approximation we will find it quite workable with many natural instances. Taking intersections just means taking more and more properties of the element and putting them together "conjunctively." It is something we do all the time. We therefore accept the idea for the present for giving our first principal definition.

DEFINITION 1.1. A family $\mathcal{D}$ of subsets of a given set Δ is called a *neighbourhood system* (over Δ) iff it is a non-empty family closed under the intersection of finite consistent sequences of neighbourhoods. That is to say, $\mathcal{D}$ must fulfill these two conditions:

(i) $\Delta \in \mathcal{D}$;

(ii) whenever $X, Y, Z \in \mathcal{D}$ and $Z \subseteq X \cap Y$, then $X \cap Y \in \mathcal{D}$. □

We remark that by convention Δ corresponds to the intersection of an *empty* sequence of neighbourhoods; in particular,

$$\bigcap_{i<n} X_i = \Delta, \text{ if } n = 0;$$
$$= \left(\bigcap_{i<n-1} X_i\right) \cap X_{n-1}, \text{ if } n > 0.$$

Of course, from (ii), we can extend the intersection property to any finite sequence. Consequently, we can say $X_0, \ldots, X_{n-1}$ is consistent in $\mathcal{D}$ iff

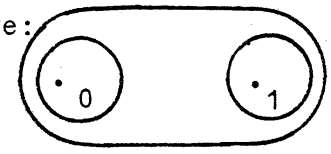
$$\bigcap_{i<n} X_i \in \mathcal{D}.$$

Some examples will help us understand the notions better.

EXAMPLE 1.2. Let $\Delta = \{0,1\}$ and let

$$\mathcal{D} = \{ \{0,1\}, \{0\}, \{1\} \}$$

In pictures we have:

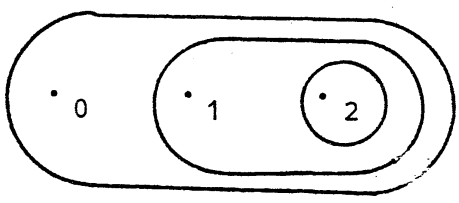


The intention is that 0 and 1 can be completely specified and that they can be identified with the total elements. As we shall see, there is only one partial element: either we give no information (the neighbourhood $\{0,1\}$), or we decide between 0 and 1 (by giving $\{0\}$ or $\{1\}$). □

EXAMPLE 1.3. Let $\Delta = \{0,1,2\}$ and let

$$\mathcal{D} = \{ \{0,1,2\}, \{1,2\}, \{2\} \}$$

In pictures we have:



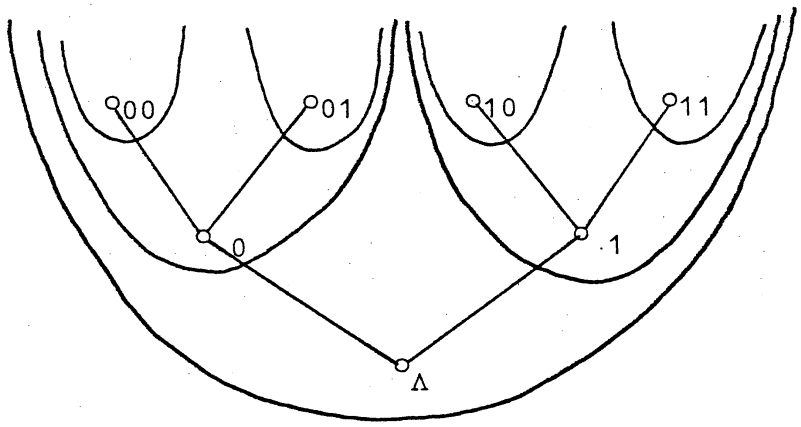
Instead of stepping to the total element (here represented by 2) in one big step, the passage is divided into two steps. (Note 0 and 1 cannot be taken as representing total elements.) This example is not very interesting because the direction of approximation in unique. We need an example with some choice. □

EXAMPLE 1.4 Let

$$\Delta = \{\Lambda, 0, 1, 00, 01, 10, 11\}$$

$$\mathcal{D} = \{\Delta, \{0,00,01\}, \{1,10,11\}, \{00\}, \{01\}, \{10\}, \{11\}\}$$

Or more understandably in pictures:



The tokens are finite sequences of 0's and 1's (up to length 2) with Λ the empty sequence; they form - in the picture - the binary tree with the sequences as the nodes. The neighbourhoods are the subtrees of all nodes above a given node. Obviously this can be generalized to sequences of any length (and to trees less regular than the binary tree). The total elements of the example correspond to the top nodes 00, 01, 10, 11 and the lower nodes to the partial elements. When we are not at a top node we have only partially determined a sequence, and the branching indicates that we have some choice as to how the sequence can be extended. □

It should be noted that, in these three examples, the reason that we have a neighbourhood system is a simple consequence of a

very special circumstance: in these systems two neighbourhoods are either disjoint or one is included in the other. This arrangement of neighbourhoods is by no means necessary.

EXAMPLE 1.5. Let $\Delta = \{0,1,2,3\}$ and let $\mathcal{D}$ be the family of all non-empty subsets of Δ.

This system is a direct generalization of Example 1.2., which was special owing to the small number of tokens. (The other examples were special by virtue of the choice of neighbourhoods.) The verification that the present $\mathcal{D}$ is a neighbourhood system rests on nothing more than the remark that sets are consistent in $\mathcal{D}$ iff they have a non-empty intersection. Clearly the arrangement of neighbourhoods in $\mathcal{D}$ can be as varied as a four-element set will allow; if Δ were made larger, the possible combinations of neighbourhoods could be made as complex as you wish. □

Having some idea now of the variety of neighbourhood systems, we have to discuss what it is they do. As stressed before, the tokens do not have to correspond directly to the elements; but where, we ask, do the elements come from? One obvious suggestion for determining an element is to produce a sequence of "better and better" neighbourhoods:

$$X_0 \supseteq X_1 \supseteq \cdots \supseteq X_n \supseteq \cdots$$

Trivially, any finite initial segment of this sequence is consistent, and so each X_n is a partial approximation to the "limit". If $\mathcal{D}$ were always to be taken as *finite*, of course, there would be no point in discussing limits since any such sequence would eventually be constant. The elements in the finite case would therefore be completely represented by neighbourhoods with the *minimal* neighbourhoods corresponding to the total elements. But there are many reasons to go beyond the finite (though perhaps not too far beyond).

Suppose $\langle Y_n \rangle_{n=0}^{\infty}$ is another "convergent" sequence with

$Y_{n+1} \subseteq Y_n$ for all indices: when do the two sequences of neighbourhoods determine the *same* limit? The two sequences can surely be different; for example, $\langle Y_n \rangle_{n=0}^{\infty}$ could be a subsequence of $\langle X_n \rangle_{n=0}^{\infty}$, say, $Y_n = X_{2n}$. Still we would want to say that the same limit is obtained. Without being given any further structure on the neighbourhoods, a simple answer is just to say that each sequence goes "equally deep" as the other:

for each m there is an n with $X_n \subseteq Y_m$, and

for each n there is an m with $Y_m \subseteq X_n$.

This definition obviously puts sequences into equivalence classes, and so elements could be identified with these. But such a definition is clumsy for two reasons: it is always tiresome to work with equivalence classes, and there is no reason to think that simple infinite sequences are adequate for determining elements without some rather drastic assumptions on $\mathcal{D}$. Nevertheless, the idea is suggestive; we just have to find some construct to represent elements in a unique way and to phrase it in a general enough manner.

Start with $\langle X_n \rangle_{n=0}^{\infty}$ again, which "converges" as before. Think of all the other sequences equivalent to this one in the sense just defined. We can define the class of all terms of all such sequences very easily as being the family:

$$x = \{Z \in \mathcal{D} \mid X_n \subseteq Z \text{ for some } n\}.$$

It is easy to prove that if we form the analogous class for $\langle Y_n \rangle_{n=0}^{\infty}$, then the two families are *equal* if and only if the sequences are *equivalent*. Thus, we seem justified in letting x represent the limit of $\langle X_n \rangle_{n=0}^{\infty}$. All we have to do now is to remark on what sort of class x is as a subfamily of $\mathcal{D}$; what we abstract from the construction, however, will be just a bit more general than taking those x that result from sequences.

DEFINITION 1.6. The (ideal) *elements* of a neighbourhood system $\mathcal{D}$ are those subfamilies $x \subseteq \mathcal{D}$ where:

(i) $\Delta \in x$;

(ii) $X, Y \in x$ always implies $X \cap Y \in x$; and

(iii) whenever $X \in x$ and $X \subseteq Y \in \mathcal{D}$, then $Y \in x$.

The *domain* of all such elements is written as $|\mathcal{D}|$. □

The idea of 1.6 is a well-known mathematical device: the families x satisfying (i) - (iii) are usually called *filters*. Most frequently the emphasis is put on the *maximal* filters, and these would be our *total* elements; however, in general, the proof that maximal filters exist is non-constructive, so for our purposes it is better not to neglect the partial filters. When maximal filters can be found, well and good, but we do not have to insist on them. Note that the generality of 1.6 is achieved by not requiring that there is a sequence of neighbourhoods that "generates" the filter x. (See Exercise 1.22.)

We have often said that neighbourhoods determine partial elements by themselves; we now make this remark precise.

DEFINITION 1.7. For $X \in \mathcal{D}$, the *principal filter* determined by X is defined by:

$$\uparrow X = \{Y \in \mathcal{D} \mid X \subseteq Y\}$$

The principal filters form what we shall call the *finite elements* of the domain $|\mathcal{D}|$. □

It is obvious that the correspondence between X and $\uparrow X$ is one-one and inclusion *reversing*, in the sense that

$$X \subseteq Y \quad \text{iff} \quad \uparrow Y \subseteq \uparrow X$$

for all $X, Y \in \mathcal{D}$. But, except in very special cases, there is much more to $|\mathcal{D}|$ than just the finite elements. Much of our investigation will be concerned with finding out how much more. The finite elements are, in a certain sense, "dense" in $|\mathcal{D}|$, however, because it is also obvious from the definitions that for each $x \in |\mathcal{D}|$

$$x = \bigcup \{\uparrow X \mid X \in x\}.$$

That is, every element is a certain type of "limit" of finite elements. (This statement is made more precise in Exercise 1.21)

We note that we have now had several occasions to use inclusion relationships between elements; this is an important relationship, and we give it a special name.

DEFINITION 1.8. For $x, y \in |\mathcal{D}|$, we say that x *approximates* y iff $x \sqsubseteq y$. The element that approximates all others, $\{\Delta\}$, is called $\perp$ (read: *bottom*); it is the "least defined" element, or the "most partial" element. Elements maximal with respect to the approximation relation are called *total* elements. □

EXAMPLES 1.2 - 1.5 (Revisited). The examples as given were all finite, so any explicitly given filter x is principal, the element is finite, the minimal $X \in x$ tells us all we need to know. In such simple situations there is essentially no difference between elements and neighbourhoods -- except for the reversal of the order as noted. This (necessary) reversal should not, however, become a matter of confusion: the smaller the neighbourhood has become, the more it has "converged", and so the better defined the element has become. In the approximation relation the "poorer" elements are placed below the "better" with the total up at the top. This will become clearer in discussing "infinite" elements.

Example 1.3 will be generalized in Exercise 1.1 Let us here generalize first 1.4. We let

$$\Delta = \Sigma^* \ ,$$

where $\Sigma = \{0,1\}$ and Σ^* means the set of all finite sequences of 0's and 1's, with Λ being the empty sequence. We write $\sigma\tau$ for the *concatenation* or *juxtaposition* of two sequences $\sigma, \tau \in \Sigma^*$. Define

$$\mathcal{B} = \{\sigma\Sigma^* | \sigma \in \Sigma^*\}, \text{ where}$$

$$\sigma X = \{\sigma\tau | \tau \in X\} \ ,$$

for an arbitrary set $X \subseteq \Sigma^*$. In other words, a neighbourhood in $\mathcal{B}$ consists of all *extensions* of a given sequence σ. (Refer back to the finite version of 1.4.) We use the letter "$\mathcal{B}$" to remind us of "binary", and this is an example we shall refer to many times. The proof (if it is not obvious) that $\mathcal{B}$ is a neighbourhood system should be done as an exercise.

What do we find in $|\mathcal{B}|$? Of course $\perp = \{\Delta\} \in |\mathcal{B}|$. For any $x \in |\mathcal{B}|$ and $\sigma \in \Sigma^*$ define

$$\sigma x = \{Y | \sigma X \subseteq Y \text{ some } X \in x\}.$$

Again there is an exercise here to show $\sigma x \in |\mathcal{B}|$. In particular $\sigma\perp \in |\mathcal{B}|$ for all $\sigma \in \Sigma^*$, and these are just the finite elements. The minimal element of $\sigma\perp$ is $\sigma\Delta$. Note that $\sigma_0\perp \sqsubseteq \sigma_1\perp$ if and only if σ_0 is an *initial segment* of the sequence σ_1.

If now $x \in |\mathcal{B}|$ is any explicitly given element (that is, if we know for any $X \in \mathcal{B}$ whether or not $X \in x$), we have but to work out from these definitions that

$$x = \bigcup_{n=0}^{\infty} \sigma_n\perp ,$$

where the $\sigma_n \in \Sigma^*$ and each σ_n is an initial segment of the next σ_{n+1}. In general, in any domain, an element is *uniquely determined by its finite approximations*, and we are just making this explicit in $|\mathcal{B}|$. When we have complete knowledge of x, then there are two cases: either the approximations $\sigma_n\perp$ become constant from some point on (where $n \geqslant n_0$), or not. In the first case x is finite and equal to $\sigma_{n_0}\perp$; in the second case x is *infinite* and the σ_n fill out an infinite (one-way) sequence.

The generalization of 1.5 to the infinite case where

$$\Delta = \mathbb{N} = \{0,1,2,3, \ldots, n, \ldots\}$$

can be made in more than one way: for instance either we use as neighbourhoods *all* non-empty subsets of Δ or just those omitting but a finite number of integers. And, as will become apparent, there are other choices giving domains of quite different characters. □

Many constructions (choices of $\mathcal{D}$) lead to the "same" domain; "sameness" is an important notion and it is to be defined in terms of "isomorphism", which in turn is to be defined in terms of approximation preserving correspondences.

DEFINITION 1.9. Two neighbourhood systems $\mathcal{D}_0$ and $\mathcal{D}_1$ determine *isomorphic domains* iff there is a one-one correspondence between $|\mathcal{D}_0|$ and $|\mathcal{D}_1|$ which preserves inclusion between the elements of the domains. In symbols we write $\mathcal{D}_0 \cong \mathcal{D}_1$. □

It is certain that the property of 1.9 is necessary, but it may not be so clear that it is sufficient. We shall in fact prove in the next lecture that an isomorphism between domains always maps finite elements to finite elements, so it always results from a one-one inclusion-preserving correspondence between neighbourhoods. This is surely as strong as could be hoped. This general result is not needed to see that particular domains *are* isomorphic.

In some of the examples tokens corresponded to total elements and in some to partial elements; it is not difficult to see (ex post facto) that every domain can be presented with tokens exactly corresponding to partial elements.

THEOREM 1.10. Given any neighbourhood system $\mathcal{D}$, define for $X \in \mathcal{D}$

$$[X] = \{x \in |\mathcal{D}| \mid X \in x\}.$$

The subsets $[X] \subseteq |\mathcal{D}|$ for $X \in \mathcal{D}$ form a neighbourhood system over $|\mathcal{D}|$ which determines a domain isomorphic to $|\mathcal{D}|$.

Proof: We note first that

(1) $[\Delta] = |\mathcal{D}|$

Next note that

(2) X, Y are consistent in $\mathcal{D}$ iff $[X] \cap [Y] \neq \emptyset$;

and that for $X, Y \in \mathcal{D}$

(3) $[X] \cap [Y] = [X \cap Y]$ if $X \cap Y \in \mathcal{D}$.

Inasmuch as

(4) $\uparrow X \in [X]$ for all $X \in \mathcal{D}$,

it easily follows that $\mathcal{D}$ and the family

$$\{[X] \mid X \in \mathcal{D}\}$$

are in a one-one, inclusion-preserving correspondence. Thus, we can induce the desired one-one correspondence between the elements of the two systems. □

The import of 1.10 is that the original tokens in Δ can be replaced by the elements of $|\mathcal{D}|$. This process replaces the neighbourhood $X \subseteq \Delta$ by the subset $[X] \subseteq |\mathcal{D}|$. As the passage is inclusion preserving, the domain has not really changed, only its presentation. Though of some theoretical charm, the theorem is not of much use since we still have to get $\mathcal{D}$ from somewhere. It does emphasize, though, that the rôle of the tokens is simply to keep the inclusions (and intersections) of neighbourhoods sorted out. It is *not* always true that the tokens can be identified with the total elements.

The last theorem in this lecture is a result on *closure properties* of a domain with respect to set-theoretical operations which have interesting meanings with respect to approximation.

THEOREM 1.11. If $\mathcal{D}$ is a neighbourhood system and $x_n \in |\mathcal{D}|$ for $n = 0,1,2,\ldots$, then

(i) $\bigcap_{n=0}^{\infty} x_n \in |\mathcal{D}|$; and

(ii) $\bigcup_{n=0}^{\infty} x_n \in |\mathcal{D}|$, provided

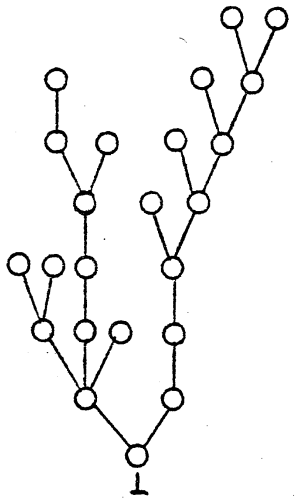
$$x_0 \subseteq x_1 \subseteq x_2 \subseteq \cdots \subseteq x_n \subseteq x_{n+1} \subseteq \cdots .$$

Proof: The conditions of 1.6 have to be checked. For the case of intersection, all of 1.6(i) - (iii) are quite obvious. For the case of union, only 1.6(ii) gives pause and it requires the proviso. If X and Y belong to the union, then $X \in x_n$, say, and $Y \in x_m$. But, either $n \leqslant m$ or $m \leqslant n$, and if $k = \max(n,m)$, then $X, Y \in x_k$. Since $X_k \in |\mathcal{D}|$, we have $X \cap Y \in x_k$; thus, $X \cap Y$ belongs to the union. This proves (ii). □

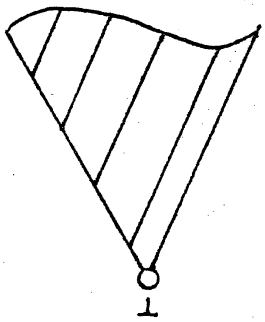
In words, the intersection is the best element that is at the same time an approximation to all of the elements x_n; the intersection is exactly what is common to all the given elements. The union on the other hand is just what the (increas-

ing sequence of the) x_n approximates; the union combines contributions from all the x_n into a "better" element -- but no more than that.

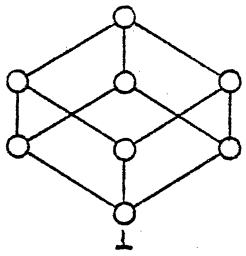
In thinking about domains a rough diagram of the partial-ordering relation $\sqsubseteq$ between elements is often helpful. The picture of 1.4 is an example where the nodes represent the elements. Any finite tree growing up from a root node would also be an example. Indeed, any finite partially ordered set with least element would be an example. (Here no distinction between tokens and elements is necessary.) A lattice diagram is also illustrated.



A TREE



A ROUGH PICTURE



A LATTICE

The root node is the element $\perp$ of $|\mathcal{D}|$; there need be no top node $\top$. *Approximation* is represented by a passage from a lower node to a higher node along the rising lines. The system $\mathcal{D}$ of neighbourhoods is the collection of sets each of which is all

the nodes above a given node. For *infinite* examples, however, care must be given to introduce *limit* nodes. The first few exercises should be provided with pictures to illustrate the structure.

EXERCISES

EXERCISE 1.12. Let $\Delta = \mathbb{N} = \{0,1,2,\ldots,n,\ldots\}$ be the set of non-negative integers. Use as neighbourhoods final segments:

$$\{m \in \mathbb{N} \mid m \geq n\}$$

for $n \in \mathbb{N}$. Verify that this is a neighbourhood system. What are the total elements? What are the finite elements? Draw a picture of the approximation relation in this domain. (Hint: there is only one limit element.)

EXERCISE 1.13. Verify all the assertions made about the system $\mathcal{B}$ defined as the infinite generalization of Example 1.4. Draw a picture similar to that given in the text which includes nodes for all $\sigma \in \Sigma^*$. Show the neighbourhoods, how the approximation relation behaves, and where the total elements lie. (The picture is closely related to the "binary tree", but has to have limit nodes all along the top.)

EXERCISE 1.14. Let $\Delta = \mathbb{N}$ and let $\mathcal{D}$ be the family of finite non-empty subsets of Δ plus the set Δ. Show that this is a neighbourhood system. What are the total elements? What are the finite elements? Draw a picture.

EXERCISE 1.15. Construct non-isomorphic infinite domains where all elements are finite but where there are no infinite chains $\langle x_n \rangle_{n=0}^{\infty}$ of elements with $x_n \sqsubseteq x_{n+1}$ but $x_n \neq x_{n+1}$ for all n.

EXERCISE 1.16. Let $\Delta = \mathbb{N}$ and let $\mathcal{D}$ be the family of *cofinite* subsets of $\mathbb{N}$. Show that $|\mathcal{D}|$ is isomorphic to the partially ordered set of *all* subsets of $\mathbb{N}$ under inclusion. Construct some other neighbourhood systems where $\mathcal{D}$ is closed under finite intersection. What happens to the total elements in such systems?

EXERCISE 1.17. Let $\Delta = \mathbb{R}$ be the real line. Let $\mathcal{D}$ be the set of non-empty open intervals with rational end points plus the set Δ. Show that this is a neighbourhood system. For any real $t \in \mathbb{R}$, show that

$$\{ X \in \mathcal{D} \mid t \in X \}$$

is a filter. Is it always total? What are the total elements of $|\mathcal{D}|$? (Hint: When t is rational consider all intervals with t as a right-hand end point.)

EXERCISE 1.18. Let $\mathcal{D}$ be a neighbourhood system. Call a subset $C \subseteq \mathcal{D}$ *consistent* iff every finite subset of C is consistent in $\mathcal{D}$. Give an example where C is a subset with more than two elements, every pair of neighbourhoods in C is consistent, but C is *not* consistent. Show that if C is consistent, then there is a *least* filter $x \in |\mathcal{D}|$ with $C \subseteq x$. Show generally that the *intersection* of any non-empty collection of filters is again a filter.

EXERCISE 1.19. Define a *positive neighbourhood system* to be a family $\mathcal{D}$ where (ii) of 1.1 is replaced by

(ii') whenever $X, Y \in \mathcal{D}$, then $X \cap Y \neq \emptyset$ iff $X \cap Y \in \mathcal{D}$.

Prove that a positive neighbourhood system is indeed a neighbourhood system in the sense of the earlier definition. Give an example of a neighbourhood system that is *not* positive. (Hint: (suggested by C.A.R. Hoare). Let $\Delta = \mathbb{N} \times \mathbb{N}$, in the plane. Let $\mathcal{D}$ be the family of subsets $X \subseteq \mathbb{N} \times \mathbb{N}$ where all but a finite number of places the *vertical* sections of X are the whole of $\mathbb{N}$ but at the other places the sections are finite and nonempty. Smaller examples are of course possible.)

EXERCISE 1.20. Let $\mathcal{D}$ be any neighbourhood system over a set Δ. Let $\Delta' = \mathcal{D}$ and define

$$\mathcal{D}' = \{\downarrow X \mid X \in \mathcal{D}\}$$

where

$$\downarrow X = \{Y \in \mathcal{D} \mid Y \subseteq X\}.$$

Show that $\mathcal{D}'$ is a positive neighbourhood system and that $|\mathcal{D}|$ and $|\mathcal{D}'|$ are isomorphic. Note that for $\mathcal{D}'$ finite elements and tokens are in a one-one correspondence.

EXERCISE 1.21. Work out in greater detail the proof of 1.10. Remark that the neighbourhood system over $|\mathcal{D}|$ so constructed is positive, thereby obtaining in a different way the same kind of conclusion as in 1.20. Show also that the system over $|\mathcal{D}|$ is *complete* in the sense that every filter is fixed by a *unique* member of the underlying set. (A filter is *fixed by a point* iff it is the filter of *all* neighbourhoods containing that point.) Remark that a complete system is one where tokens and (partial) elements can always be identified (under a suitable one-one correspondence). Show also that consistency of a set $\{X_i \mid i<n\}$ of neighbourhoods in $\mathcal{D}$ is equivalent to saying

$$\bigcap_{i<n} [\, X_i \,] \neq \emptyset$$

EXERCISE 1.22. (For topologists). Show that the neighbourhoods $[X]$ for $X \in \mathcal{D}$ make $|\mathcal{D}|$ into a topological space where the open subsets $\mathcal{U} \subseteq |\mathcal{D}|$ can be characterized by the following two conditions:

(i) whenever $x \in \mathcal{U}$ and $x \subseteq y \in |\mathcal{D}|$, then $y \in \mathcal{U}$; and

(ii) whenever $x \in \mathcal{U}$, then $\uparrow X \in \mathcal{U}$ for some $X \in x$.

Prove also that the inclusion relation on $|\mathcal{D}|$ can be defined topologically as:

(iii) $x \subseteq y$ iff for all open $\mathcal{U} \subseteq |\mathcal{D}|$, if $x \in \mathcal{U}$ then $y \in \mathcal{U}$.

Is $|\mathcal{D}|$ ever a Hausdorff space? Show that if $\langle x_n \rangle_{n=0}^{\infty}$ is a sequence of elements of $|\mathcal{D}|$ with $x_n \subseteq x_{n+1}$ for all n, then

$$\bigcup_{n=0}^{\infty} x_n$$

is not only in $|\mathcal{D}|$ but is a topological limit point of the sequence. Show that any element x is a limit point of the set $\{\uparrow X | X \in x\}$. Are there other limit points?

EXERCISE 1.23. Suppose that the neighbourhood system $\mathcal{D}$ is countable, say,

$$\mathcal{D} = \{X_0, X_1, X_2, \ldots, X_n, \ldots\}$$

Suppose further that the property of consistency of finite sequences of neighbourhoods is decidable (or "effectively known"). Then the following sequence is well defined:

$$Y_0 = X_0$$

$$Y_{n+1} = X_{n+1}, \text{ if this set is consistent with } Y_0, Y_1, \ldots, Y_n;$$

$$= Y_n, \text{ if not.}$$

Show that $\{Y_0, Y_1, \ldots, Y_n, \ldots\}$ is a total element of $|\mathcal{D}|$. (Hint: Show first that $Y_0, Y_1, \ldots, Y_{n-1}$ is consistent for all n.) In such a system show that all filters can be determined by sequences.

EXERCISE 1.24. (For set theorists). Prove, using the Axiom of Choice, or some equivalent principle, that in every domain a partial element can always be extended to a total element. Is this assertion equivalent to the Axiom of Choice? (Hint: Remember to prove that the union of every (transfinite) chain of filters is again a filter.)

EXERCISE 1.25. (For set theorists). Let Δ be any well-ordered set (ordinal). (Even small ordinals like $\omega.3$ or ω^5 are interesting.) Let $\mathcal{D}$ be the family of non-empty *final* segments of Δ. What is $|\mathcal{D}|$? Are all elements finite? Is every approximation to a finite element finite?

EXERCISE 1.26. (For algebraists). Let A be a commutative ring with unit. Let Δ be the set of finite subsets $F \subseteq A$. Define

$$I(F) = \{G \in \Delta \mid F \subseteq \text{ the ideal generated by } G\}$$

Prove that the sets of the form I(F) form a neighbourhood system, and that the corresponding domain is isomorphic to the set of ring-theoretic ideals of A partially ordered by inclusion. What would happen if we excluded from Δ all F with $I(F) = I(\{1\})$, where 1 is the unit of A?

EXERCISE 1.27. Further closure properties of domains can be proved for bounded sets. We say $X \subseteq |\mathcal{D}|$ is *bounded* iff for some $y \in |\mathcal{D}|$ we have $x \subseteq y$ for all $x \in X$. This y is called an *upper bound*. We let

$$\bigsqcup X = \bigcap \{y \in |\mathcal{D}| \mid x \subseteq y \text{ all } x \in X\}.$$

Prove that if X is bounded, then $\bigsqcup X$ is the *least upper bound* for X in $|\mathcal{D}|$. Prove also: if $U, V \in \mathcal{D}$ are neighbourhoods, then $\{U, V\}$ is consistent in $\mathcal{D}$ iff $\{\uparrow U, \uparrow V\}$ is bounded in $|\mathcal{D}|$. (That is, boundedness is for elements what consistency is for neighbourhoods.) Prove finally with the aid of 1.18 that $X \subseteq |\mathcal{D}|$ is bounded iff every finite subset of X is bounded.

LECTURE II

MAPPINGS BETWEEN DOMAINS

The elements of a domain are regarded as being specified by approximations: the neighbourhoods. With the idea of approximation as the dominant notion, therefore, it is natural to look for a concept of mapping (transformation of domains) that in some suitable sense preserves the spirit of the approximations. In a theory of computability, where the (finite) approximations to the elements are all we can ever know at one time, the only mappings that can be computed are those that proceed by approximation, somehow passing from the neighbourhoods of one domain over to the neighbourhoods of the other.

Suppose $X \in \mathcal{D}_0$ is given - it is an approximation to certain elements of $|\mathcal{D}_0|$. (More precisely $\uparrow X$ is the approximation *in* the domain, but it is easier to speak of the neighbourhood X.) What can be said about the approximations of the images of these elements under the mapping we will call f? If X is not a very sharp approximation, then not very much can be said about the image in the other domain $|\mathcal{D}_1|$. Trivially, of course, we can say that Δ_1 is an approximation - because it approximates everything in its domain. Suppose, however, that we could say more. Suppose we could say that both Y and Y' approximate the image of X. If the mapping f is coherent, then it is reasonable to suppose that such a statement would imply that Y and Y' are *consistent* in $\mathcal{D}_1$. But if this is so, then since the two neighbourhoods are meant to cluster around the same images, we can feel some confidence in saying that $Y \cap Y'$ approximates these images., In other words to specify f we do not supply a *unique* image of X, but we say which of the $Y \in \mathcal{D}_0$ approximate the (ideal) image. To make this idea work a *monotonicity condition* is also needed since we are trying to express the idea that "*if* we give at least X as an approximate input to f, *then* we can expect at least Y as output." Thus,

a mapping is taken as a kind of relation between neighbourhoods.

DEFINITION 2.1. An *approximable mapping* $f: \mathcal{D}_0 \to \mathcal{D}_1$ between domains is a binary relation $f \subseteq \mathcal{D}_0 \times \mathcal{D}_1$ between neighbourhoods such that

(i) $\Delta_0 f \Delta_1$;

(ii) $X f Y$ and $X f Y'$ always imply $X f (Y \cap Y')$;

(iii) $X f Y$, $X' \subseteq X$, and $Y \subseteq Y'$ always imply $X' f Y'$. □

Condition (i) we have already discussed; in a sense it means "ask me no questions and I shall tell you no lies." In other words "zero input can expect at least zero output." The other conditions are compatible with having

$$f = \{ < X , \Delta_1 > \mid X \in \mathcal{D}_0 \};$$

that is, f might be the least informative relation and nothing more. But if it is more, then (ii) is, as we explained, a consistency condition. To explain monotonicity in (iii), suppose a mapping relationship is already known, $X f Y$, say. If we *improve* the accuracy of X to $X' \subseteq X$ and if we *degrade* the accuracy of Y to $Y' \supseteq Y$, then we can still assert $X' f Y'$ since this relationship is *less informative* than the former relationship, which was already known. Thus, we see that conditions (i) - (iii) are all reasonably argued as necessary.

One indication that the conditions of 2.1 are sufficient for a definition is that they are exactly what we need to show that f as a neighbourhood relation determines an equivalent elementwise mapping from $|\mathcal{D}_0|$ into $|\mathcal{D}_1|$. (Owing to the equivalence, we use the same symbol f for both.)

PROPOSITION 2.2. Given neighbourhood systems $\mathcal{D}_0$ and $\mathcal{D}_1$, an approximable mapping $f: \mathcal{D}_0 \to \mathcal{D}_1$ always determines a function $f: |\mathcal{D}_0| \to |\mathcal{D}_1|$ between domains by virtue of the formula:

(i) $f(x) = \{Y \in \mathcal{D}_1 \mid \exists X \in x.\ X f Y\}$

for all $x \in |\mathcal{D}_0|$. Conversely, this function uniquely determines

the original relation by the equivalence:

(ii) $X\, f\, Y$ iff $Y \in f(\uparrow X)$

for all $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. Approximable functions are always *monotone* in the following sense:

(iii) $x \subseteq y$ always implies $f(x) \subseteq f(y)$,

for $x, y \in |\mathcal{D}_0|$; moreover two approximable functions $f : \mathcal{D}_0 \to \mathcal{D}_1$ and $g : \mathcal{D}_0 \to \mathcal{D}_1$ are identical as relations iff

(iv) $f(x) = g(x)$, for all $x \in |\mathcal{D}_0|$.

Proof: The argument that formula (i) always gives us $f(x) \in |\mathcal{D}_1|$ when $x \in |\mathcal{D}_0|$ can be safely left to the reader. Note, however, that all the conditions of 2.1 are required to show this. As for (ii), the implication from left to right follows directly from (i) because $X \in \uparrow X$. In the other direction $Y \in f(\uparrow X)$ means that $Z\, f\, Y$ holds for some $Z \in \uparrow X$. But from $X \subseteq Z$ it follows that $X\, f\, Y$, as we wished.

To prove monotonicity, assume $x \subseteq y$. Now $X \in x$ and $X\, f\, Y$ always imply $X \in y$ and $X\, f\, Y$. This means $Y \in f(x)$ always implies $Y \in f(y)$; that is, $f(x) \subseteq f(y)$.

Finally, to check that (iv) means $f = g$ as relations, all that has to be remarked that this follows from formulae (i) and (ii). □

Note that the right-hand side of (ii) can be written:

$$\uparrow Y \subseteq f(\uparrow X),$$

which can be read as saying that the partial element determined by the neighbourhood Y approximates the function value at the element determined by X. This precise relationship of course fits the informal discussion of mapping given earlier. Indeed whenever $x \in [X]$ and $X\, f\, Y$ hold, then $f(x) \in [Y]$ always follows, which is another way to construe the mapping character of f. Some examples of mappings are now called for.

EXAMPLE 2.3. Let T be the neighbourhood system of the two-token domain of Example 1.2. To avoid confusion with some other domains, we will call the two total elements of $|T|$, respectively, true and false. There is only one other finite element here, namely $\perp$ = undefined. We often use these elements as indicators of results: true indicates a positive outcome; false, a negative outcome; and $\perp$ indicates that there is not enough information to decide the outcome totally.

Let $\mathcal{B}$ be the system for the binary tree as in the last chapter. What we wish to define is an approximable mapping $f : \mathcal{B} \to T$. The intuitive idea of the mapping we have in mind is that the binary sequence is being read from left to right, and we are counting the number of 0's seen before the first 1 is encountered. We then test the parity of this count; if it is even, the output is true; if not, false. Using a suggestive informal notation with three dots, some results of the function that does the counting and testing can be written as:

$$f\ (0000101\cdots) = \text{true}$$
$$f\ (1101110\cdots) = \text{true}$$
$$f\ (0111011\cdots) = \text{false}$$
$$f\ (0000000\cdots) = \perp\ .$$

The last equation is necessary, because 0000000 as a partial element *cannot* be counted as either even or odd since it can have *inconsistent* extensions:

$$0000000\perp \subseteq 00000001\perp$$
$$0000000\perp \subseteq 00000000001\perp\ .$$

So, as far as f is concerned, a plain string of 0's is *indefinite*. The same answer holds if the 0's go on infinitely.

To be more precise we want

$$f\ (0^n 1\perp) = \text{true} \quad \text{if } n \text{ is even;}$$
$$= \text{false} \quad \text{if } n \text{ is odd.}$$

As a binary relation $f \subseteq \mathcal{B} \times T$ we will have

$X\ f\ Y$ iff $Y \in \perp$ or $X \subseteq 0^n 1\Delta$ for some $n \in \mathbb{N}$ and either n is even and $Y \in$ true or n is odd and $Y \in$ false.

It should be checked that 2.1(i)-(ii) are satisfied. □

EXAMPLE 2.4. Let us briefly describe an approximable mapping $g: \mathcal{B} \to \mathcal{B}$. Informally, g can be said to "read a sequence from left to right and eliminate the first consecutive run of 1's while copying all the other digits as read." We will have

$$g\,(0^n 1^k\, 0\, x) = 0^{n+1} x$$

provided $k > 0$. (Here 1^k means a string of 1's of length k.) However, if 1^∞ is the infinite sequences of 1's, then

$$g\,(1^\infty) = \perp, \text{ and}$$
$$g\,(0^n 1^\infty) = 0^n.$$

This example is instructive, since it shows that a non-trivial mapping can transform a total element into a partial element. □

Aside from our being able to define particular functions outright, we can combine functions in many different ways; the idea of composition is probably the most basic scheme of combination, and there is a technical name for a family of structures with mappings that can be so combined.

THEOREM 2.5. The class of neighbourhood systems and approximable mappings form a *category*, where the *identity mapping* $I_{\mathcal{D}} : \mathcal{D} \to \mathcal{D}$ relates $X, Y \in \mathcal{D}$ as follows:

(i) $X\ I_{\mathcal{D}}\ Y$ iff $X \subseteq Y$.

If $f : \mathcal{D}_0 \to \mathcal{D}_1$ and $g : \mathcal{D}_1 \to \mathcal{D}_2$ are given, then the *composition* $g \circ f : \mathcal{D}_0 \to \mathcal{D}_2$ relates $X \in \mathcal{D}_0$ and $Z \in \mathcal{D}_2$ as follows:

(ii) $X\ g \circ f\ Z$ iff $\exists\, Y \in \mathcal{D}_1.\ X\ f\ Y$ and $Y\ g\ Z$.

Proof: (We may use MacLane [1971] as the standard reference on category theory, but we require hardly more than the basic definitions at this stage.) To check that we have a category, we need to know that the identity and composition maps really are maps in the category and that certain identity and associative laws hold. Now it is obvious that $I_{\mathcal{D}}$ satisfies 2.1 (i)-(iii). Moreover if $f : \mathcal{D}_0 \to \mathcal{D}_1$, all we have to prove is:

$$f \circ I_{\mathcal{D}_0} = I_{\mathcal{D}_1} \circ f = f$$

Checking one of these equations is enough. Thus, for $X \in \mathcal{D}_0$ and $Z \in \mathcal{D}_1$ we find

$$X\, f \circ I_{\mathcal{D}_0}\, Z \text{ iff } \exists Y \in \mathcal{D}_0.\ X \subseteq Y \text{ and } Y\, f\, Z$$
$$\text{iff } X\, f\, Z.$$

So, f and $f \circ I_{\mathcal{D}_0}$ are the same mapping.

Suppose now that $f : \mathcal{D}_0 \to \mathcal{D}_1$ and $g : \mathcal{D}_1 \to \mathcal{D}_2$. We have to verify that $g \circ f$ is an approximable mapping. First off, there is no trouble in seeing that $\Delta_0 g \circ f \Delta_2$ holds. Next, suppose that $X\, g \circ f\, Z$ and $X\, g \circ f\, Z'$ hold. Then we have $X\, f\, Y$ and $Y\, g\, Z$ for some choice of $Y \in \mathcal{D}_1$. Also $X\, f\, Y'$ and $Y'\, g\, Z'$ hold for some choice of $Y' \in \mathcal{D}_1$. By 2.1 (ii) it follows that $X\, f\, (Y \cap Y')$. Since $Y \cap Y' \subseteq Y$, we conclude $(Y \cap Y')\, g\, Z$ by 2.1 (iii); similarly $(Y \cap Y')\, g\, Z'$. Invoking 2.1 (ii) again, we obtain $(Y \cap Y')\, g\, (Z \cap Z')$, and $X\, g \circ f\, (Z \cap Z')$ is proved.

Suppose finally that $X' \subseteq X\, g \circ f\, Z \subseteq Z'$. Now $X\, f\, Y$ and $Y\, g\, Z$ for some $Y \in \mathcal{D}_1$. But then $X'\, f\, Y$ holds; for a similar reason $Y\, g\, Z'$ holds also. Therefore, $X'\, g \circ f\, Z'$ is established, which means that we have checked 2.1 (iii) for $g \circ f$ and have completed the proof that $g \circ f : \mathcal{D}_0 \to \mathcal{D}_1$.

The verification of associativity is a purely logical deduction. Thus suppose that in addition to f and g we have $h : \mathcal{D}_2 \to \mathcal{D}_3$. If $X \in \mathcal{D}_0$ and $W \in \mathcal{D}_3$ we find

$$\begin{aligned} X\, h \circ (g \circ f)\, W \quad &\text{iff } \exists Z \in \mathcal{D}_2.\ X\, g \circ f\, Z \text{ and } Z\, h\, W \\ &\text{iff } \exists Z \in \mathcal{D}_2\ \exists Y \in \mathcal{D}_1.\ X\, f\, Y \text{ and } Y\, g\, Z \text{ and } Z\, h\, W \\ &\text{iff } \exists Y \in \mathcal{D}_1\ \exists Z \in \mathcal{D}_2.\ X\, f\, Y \text{ and } Y\, g\, Z \text{ and } Z\, h\, W \\ &\text{iff } \exists Y \in \mathcal{D}_1.\ X\, f\, Y \text{ and } Y\, (h \circ g)\, W \\ &\text{iff } X\, (h \circ g) \circ f\, W. \end{aligned}$$

So, as relations, $h \circ (g \circ f) = (h \circ g) \circ f$. □

It may seem as though we have, in the definition of composition, written things backwards. But the reason is that when mappings are taken as elementwise functions, then the order is preserved in expressions involving the usual function value notation. We have, for example:

PROPOSITION 2.6. Given $f : \mathcal{D}_0 \to \mathcal{D}_1$ and $g : \mathcal{D}_1 \to \mathcal{D}_2$, the following equations hold:

(i) $I_{\mathcal{D}_0}(x) = x$, and

(ii) $(g \circ f)(x) = g(f(x))$,

for all $x \in |\mathcal{D}_0|$. □

The proof is not troublesome and is left as an exercise. In technical language the result shows that the category defined in Theorem 2.5 is equivalent to a "concrete category" of sets and functions, namely the domains and elementwise transformations of 2.2.

Toward the end of the last lecture (see 1.9) we promised to show that isomorphisms of domains always come from approximable mappings, and this we now do. It means that the category contains all the isomorphisms it should have.

THEOREM 2.7. Every isomorphism between domains results from an approximable mapping between the neighbourhood systems. Moreover, finite elements are always transformed into finite elements.

Proof: Suppose that $f : |\mathcal{D}_0| \to |\mathcal{D}_1|$ is a one-one, inclusion-preserving function defined on elements, where the range of the function is the whole of $|\mathcal{D}_1|$, of course. Taking the hint from 2.2, there is only one way we could define a neighbourhood mapping; namely, we consider the relation $Y \in f(\uparrow X)$ for $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. What has to be shown is that this is an approximable mapping which determines the original function via the formula 2.2 (i).

The first part is easy; indeed, there is a general result that monotone functions on finite elements of one domain to arbitrary elements of another domain always determine approximable mappings (cf. Exercise 2.8). What remains, then, is to show that the relation re-defines the function. This comes down to showing that for $x \in |\mathcal{D}_0|$

$$f(x) = \{Y \in \mathcal{D}_1 \mid \exists X \in x \,.\, Y \in f(\uparrow X)\}.$$

Consider the right-hand side of this equation: it is a filter. (This either can be proved directly or Exercise 2.11 can be used.) Because f is an onto-function, we can call the right-hand side $f(x')$ for some $x' \in |\mathcal{D}_0|$. But since $X \in x$ implies $\uparrow X \subseteq x$ and $f(\uparrow X) \subseteq f(x)$, the right-hand side is included in the left-hand side. In other words $f(x') \subseteq f(x)$. But, since f is an isomorphism $x' \subseteq x$ follows.

In the other direction, if $X \in x$, then $f(\uparrow X) \subseteq f(x')$ holds by definition, so $\uparrow X \subseteq x'$. This implies $X \in x'$; and, as X is arbitrary, $x \subseteq x'$ follows. So $x = x'$, and $f(x) = f(x')$ as desired.

Finally, consider any finite element $\uparrow X \in |\mathcal{D}_0|$ where $X \in \mathcal{D}_0$. What we have to show is that $f(\uparrow X)$ is finite in $|\mathcal{D}_1|$. Because f is an isomorphism, we can associate uniquely to every $Y \in f(\uparrow X)$ an element $y_Y \subseteq \uparrow X$ in $|\mathcal{D}_0|$ where $f(y_Y) = \uparrow Y$. (Just apply the inverse of the function f.) Define

$$z = \bigcup \{y_Y \mid Y \in f(\uparrow X)\}.$$

Because $Y' \subseteq Y$ always implies $y_{Y'} \subseteq y_Y$ and each $y_Y \in |\mathcal{D}_0|$, it is easy to show z is a filter and hence is in $|\mathcal{D}_0|$ also (cf. Exercise 2.11). Because each $y_Y \subseteq \uparrow X$, then $z \subseteq \uparrow X$, too. But each $y_Y \subseteq z$, so $\uparrow Y = f(y_Y) \subseteq f(z)$ and hence $Y \in f(z)$. As this holds for all $Y \in f(\uparrow X)$, the inclusion $f(\uparrow X) \subseteq f(z)$ follows, as well as $\uparrow X \subseteq z$. Therefore, $z = \uparrow X$ and so $X \in z$. But then $X \in y_Y$ for some $Y \in f(\uparrow X)$, by definition of Z. Since $\uparrow X \subseteq y_Y$, we obtain $f(\uparrow X) \subseteq \uparrow Y$. But of course the opposite inclusion is also true from the choice of Y. This means that $f(\uparrow X) = \uparrow Y$ is finite in $|\mathcal{D}_1|$ as claimed. We can apply the same argument to the inverse function; and, thus, the finite elements of $|\mathcal{D}_0|$ and $|\mathcal{D}_1|$ are in a one-one inclusion-preserving correspondence under the isomorphism. □

EXERCISES

EXERCISE 2.8. With reference to the proof of 2.2 show that an approximable mapping is uniquely determined by its elementwise effect on finite elements. Moreover any arbitrary monotone function on finite elements of $|\mathcal{D}_0|$ with values in $|\mathcal{D}_1|$ comes from an approximable $f : \mathcal{D}_0 \to \mathcal{D}_1$.

EXERCISE 2.9. Prove that if $f: \mathcal{D}_0 \to \mathcal{D}_1$ is an approximable mapping, then the elementwise mapping $f: |\mathcal{D}_0| \to |\mathcal{D}_1|$ satisfies the equation

$$f(x) = \bigcup\{f(\uparrow X) \mid X \in x\}$$

for all $x \in |\mathcal{D}_0|$. Conversely, show that every elementwise function satisfying this equation comes from an approximable mapping as defined in 2.2.

EXERCISE 2.10. Carry out the proof of Proposition 2.6; and in addition show that, if $f,g : \mathcal{D}_0 \to \mathcal{D}_1$ are two approximable mappings, there exists $h : \mathcal{D}_0 \to \mathcal{D}_1$ such that

$$h(x) = f(x) \cap g(x)$$

for all $x \in |\mathcal{D}_0|$.

EXERCISE 2.11. Let $\langle I, \leqslant \rangle$ be a non-empty abstract partially ordered set; suppose it is *directed* in the sense that whenever $i, j \in I$, then $i \leqslant k$ and $j \leqslant k$ for some $k \in I$. Suppose that $a: I \to |\mathcal{D}|$ is such that

$$i \leqslant j \text{ implies } a_i \subseteq a_j$$

for all $i, j \in I$. Prove that

$$\bigcup\{a_i \mid i \in I\}$$

is always a filter in $|\mathcal{D}|$. (Note the ways this lemma could be used in the proof of 2.7; but be careful in defining the partially ordered set and do not confuse $\subseteq$ and $\supseteq$.) In words we could say that the domain of filters is *closed under directed unions*. Prove also that if $f : \mathcal{D} \to \mathcal{D}'$ is an approximable mapping, then for any directed union

$$f\left(\bigcup\{a_i \mid i \in I\}\right) = \bigcup\{f(a_i) \mid i \in I\};$$

that is, *approximable mappings always preserve directed unions*. If an elementwise function preserves directed unions, must it come from an approximable mapping? (Hint: Invoke 2.9.)

EXERCISE 2.12. Suppose $\langle I, \leqslant \rangle$ is a directed, partially ordered set and $f_i : \mathcal{D}_0 \to \mathcal{D}_1$ is a family of approximable mappings indexed by $i \in I$, where we assume

$$i \leqslant j \text{ implies } f_i(x) \subseteq f_j(x)$$

for all $i, j \in I$ and all $x \in |\mathcal{D}_0|$. Prove that there is an approximable mapping $g : \mathcal{D}_0 \to \mathcal{D}_1$ where

$$g(x) = \bigcup \{f_i(x) \mid i \in I\}$$

for all $x \in |\mathcal{D}_0|$.

EXERCISE 2.13. (For topologists.) Recall Exercise 1.22 where it was shown that any domain $|\mathcal{D}|$ is a topological space. Prove from Exercise 2.9 that the functions $f : |\mathcal{D}_0| \to |\mathcal{D}_1|$ determined by approximable mappings are exactly *the continuous functions between these spaces*. (Hint: To prove continuity, remark that by 2.9

$$f^{-1}[Y] = \bigcup \{[X] \mid Y \in f(\uparrow X)\};$$

hence, the inverse image of any open set is open. In the other direction, suppose that $f : |\mathcal{D}_0| \to |\mathcal{D}_1|$ is topologically continuous. Argue that for all $x \in |\mathcal{D}_0|$ and all open subsets $U \subseteq |\mathcal{D}_1|$ we have

$$f(x) \in U \text{ iff } \exists X \in x \,.\, f(\uparrow X) \in U.$$

This holds because an open subset of $|\mathcal{D}_0|$ is always a union of basic open subsets of the form $[X']$ for $X \in \mathcal{D}_0$ and because

$$x = \bigcup \{\uparrow X \mid X \in x\}$$

for all $x \in |\mathcal{D}_0|$.)

EXERCISE 2.14. Let $f : |\mathcal{D}_0| \to |\mathcal{D}_1|$ be an isomorphism between domains. Let $\varphi : \mathcal{D}_0 \to \mathcal{D}_1$ be the one-one correspondence between neighbourhoods provided by Theorem 2.7 where

$$f(\uparrow X) = \uparrow \varphi(X)$$

for all $X \in \mathcal{D}_0$. Show that the approximable mapping determined by f is just the relationship $\varphi(X) \subseteq Y$. In addition prove that if $X, X' \in \mathcal{D}_0$ are consistent, then

$$\varphi(X \cap X') = \varphi(X) \cap \varphi(X').$$

Remark that the isomorphisms between domains correspond exactly to the isomorphisms between neighbourhood systems (in the sense of one-one inclusion preserving correspondences).

EXERCISE 2.15. (For topologists). Consider the one-token system with

$$\mathcal{O} = \{ \{0\}, \emptyset \}$$

We can regard $|\mathcal{O}|$ as having just two finite elements $\perp$ (bottom) and $\top$ (top), where $\perp \subseteq \top$. For any system $\mathcal{D}$, show that the open subsets U of $|\mathcal{D}|$ are in a one-one correspondence with the approximable mappings $f : \mathcal{D} \to \mathcal{O}$, where the correspondence is given by the equation

$$U = \{x \in |\mathcal{D}| \mid f(x) = \top\}.$$

What are the open subsets of $|\mathcal{O}|$? of $|T|$? of $|\mathcal{B}|$?

EXERCISE 2.16. In the discussion of $\mathcal{B}$ in Chapter 1 we defined a mapping $x \mapsto \sigma x$ for any given $\sigma \in \Sigma^*$. Is this (elementwise) mapping approximable? Show in addition that the mapping $f : \mathcal{B} \to T$ of 2.3 is uniquely determined among approximable mappings by the equations:

$$\begin{aligned} f\,(1x) &= \text{true}, \\ f\,(01x) &= \text{false}, \text{ and} \\ f\,(00x) &= f(x). \end{aligned}$$

EXERCISE 2.17. Establish in detail that the mapping $g : \mathcal{B} \to \mathcal{B}$ of Exercise 2.4 is approximable. Is it uniquely determined by these equations:

$$\begin{aligned} g(0x) &= 0g(x), \\ g(11x) &= g(1x), \\ g(10x) &= 0x, \\ g(1) &= \perp, \end{aligned}$$

or are some missing?

EXERCISE 2.18. What is the meaning in words of the approximable mapping $h : \mathcal{B} \to \mathcal{B}$, where

$$h(0x) = 00h(x), \text{ and}$$
$$h(1x) = 10h(x),$$

for all elements $x \in |\mathcal{B}|$? Is h an isomorphism? Does there exist a map $k : \mathcal{B} \to \mathcal{B}$ where

$$k \circ h = I_{\mathcal{B}},$$

and is k one-one?

EXERCISE 2.19. Generalize Definition 2.1 in an appropriate way in order to define the concept of *an approximable mapping*

$$f : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2$$

of two variables. (Hint: f can be taken to be a certain kind of ternary relation

$$f \subseteq \mathcal{D}_0 \times \mathcal{D}_1 \times \mathcal{D}_2,$$

where we can write

$$X, Y \; f \; Z$$

for the relationship among neighbourhoods.) What is the corresponding version of Proposition 2.2 for functions of two variables?

EXERCISE 2.20. Discuss again the example of Exercise 1.15 where the domain turns out to be the powerset (set of all subsets) of $\mathbb{N}$. Show how the finite elements can be taken to be the finite subsets of $\mathbb{N}$ and can be identified with the tokens of a suitable neighbourhood system P. (Hint: Define $\uparrow F$ for finite sets $F \subseteq \mathbb{N}$.) Show that both union and intersection ($x \cup y$ and $x \cap y$) are functions on $|P|$ that are approximable in the sense of Exercise 2.19. (The elements of $|P|$ are being identified with arbitrary sets $x \subseteq \mathbb{N}$.) Show also the following transformations approximable:

$$x + 1 = \{n + 1 \mid n \in x\}, \text{ and}$$
$$x - 1 = \{n \mid n + 1 \in x\}.$$

EXERCISE 2.21. The system $\mathcal{B}$ of **2.3** has as its total elements only the infinite sequences. Modify the construction of $\mathcal{B}$ to another neighbourhood system $\mathcal{C}$ which has *both* the finite and infinite sequences as total elements. (Hint: $\mathcal{B} \subseteq \mathcal{C}$.) Show that there is an approximable map $x\,y$ on elements naturally extending ordinary juxtaposition of sequences. (Hint: Write 01001 for a *total* finite sequence and $01001\perp$ for the corresponding finite partial element. Remember to distinguish between Λ (the total empty sequence) and $\perp$ (the undefined sequence). The definition should work out so that if x is an infinite sequence (hence, total), then $x\,y = x$ for all y. What will $x\,y$ equal if x is not total? In other words, the construction possesses a rather strong left-to-right bias.)

EXERCISE 2.22. (For set theorists). We have remarked in Exercise 1.18 and in Exercise 2.11 that any domain $|\mathcal{D}|$, as a family of sets (in fact, a family of subsets of the set $\mathcal{D}$ itself), is closed under the intersection of an arbitrary non-empty sub family and under the union of any directed sub family. For those familiar with the subject matter, the example of the (proper) ideals of a commutative ring (with unit) is also seen to be such a family. What is the abstract situation? Let $\mathbf{C}$ be *any* family of sets with these closure properties. It is to be shown that $\mathbf{C}$ is inclusion-isomorphic to a domain. (Hint: Let Δ be the set of finite sets included in sets in $\mathbf{C}$. For $F \in \Delta$, define its "closure" by the equation:

$$\overline{F} = \bigcap \{X \in \mathbf{C} \mid F \subseteq X\}.$$

Every $\overline{F} \in \mathbf{C}$, and these will prove to be the "finite" elements of $\mathcal{C}$. The neighbourhood system $\mathcal{D}$ over Δ can be taken to be the sets of the form

$$C(F) = \{G \in \Delta \mid F \subseteq \overline{G}\}$$

for $F \in \Delta$. Notice that for all $X \in \mathbf{C}$

$$X = \bigcup \{\overline{F} \mid F \subseteq X \text{ and } F \in \Delta\}.)$$

Check that approximable functions on these families are just those preserving directed unions.

LECTURE III

DOMAIN CONSTRUCTS

Having now seen a number of domains presented through their neighbourhood systems, we need next to introduce general constructs for forming new domains from old. There are an unlimited number of such constructs (technically called *functors*), but we have time only to single out a few of the more important ones. Outstanding among all of them is the notion of product of systems, which in our chosen category has all the expected properties. For the time being in order to simplify notation we assume of the underlying sets Δ_0 and Δ_1 of systems $\mathcal{D}_0$ and $\mathcal{D}_1$ that they are disjoint. There is no loss of generality as $\mathcal{D}_1$ can always be replaced by an isomorphic system disjoint from $\mathcal{D}_0$ in the required sense.

DEFINITION 3.1. Let neighbourhood systems $\mathcal{D}_0$ and $\mathcal{D}_1$ be given over disjoint sets Δ_0 and Δ_1. The *product system* over $\Delta_0 \cup \Delta_1$ is defined by:

$$\mathcal{D}_0 \times \mathcal{D}_1 = \{X \cup Y \mid X \in \mathcal{D}_0 \text{ and } Y \in \mathcal{D}_1\}.$$

For elements $x \in |\mathcal{D}_0|$ and $y \in |\mathcal{D}_1|$ we also define:

$$\langle x,y \rangle = \{X \cup Y \mid X \in x \text{ and } Y \in y\}. \quad \square$$

PROPOSITION 3.2. The construct $\mathcal{D}_0 \times \mathcal{D}_1$ always gives a neighbourhood system where for elements $x,x' \in |\mathcal{D}_0|$ and $y,y' \in |\mathcal{D}_1|$ we have

(i) $\quad \langle x,y \rangle \subseteq \langle x',y' \rangle$ iff $x \subseteq x'$ and $y \subseteq y'$.

Moreover, there is a one-one correspondence between the elements of $|\mathcal{D}_0 \times \mathcal{D}_1|$ and pairs of elements of $|\mathcal{D}_0|$ and $|\mathcal{D}_1|$ since all elements of $|\mathcal{D}_0 \times \mathcal{D}_1|$ are of the form $\langle x,y \rangle$.

Proof: Owing to the disjointness of Δ_0 and Δ_1, we note that for $X, X' \in \mathcal{D}_0$ and $Y, Y' \in \mathcal{D}_1$ we have

(1) $\quad X \cup Y \subseteq X' \cup Y'$ iff $X \subseteq X'$ and $Y \subseteq Y'$.

Thus, $\{X \cup Y, X' \cup Y'\}$ is consistent in $\mathcal{D}_0 \times \mathcal{D}_1$ iff $\{X, X'\}$ is

consistent in $\mathcal{D}_0$ and $\{Y, Y'\}$ is consistent in $\mathcal{D}_1$. In the consistent case we find

(2) $$(X \cup Y) \cap (X' \cup Y') = (X \cap X') \cup (Y \cap Y'),$$

and so $\mathcal{D}_0 \times \mathcal{D}_1$ is closed under consistent intersection. As $\Delta_0 \cup \Delta_1 \in \mathcal{D}_0 \times \mathcal{D}_1$, it is certainly a neighbourhood system.

It is easy to check by the previous calculations that $\langle x,y\rangle \in |\mathcal{D}_0 \times \mathcal{D}_1|$ if $x \in |\mathcal{D}_0|$ and $y \in |\mathcal{D}_1|$. The proof of 3.2(i) follows directly from the definition and (1).

Suppose $z \in |\mathcal{D}_0 \times \mathcal{D}_1|$. Define as a temporary notation:

$$z_0 = \{X \in \mathcal{D}_0 \mid X \cup \Delta_1 \in z\}, \text{ and}$$

$$z_1 = \{Y \in \mathcal{D}_1 \mid \Delta_0 \cup Y \in z\}.$$

Clearly, both $z_0 \in |\mathcal{D}_0|$ and $z_1 \in |\mathcal{D}_1|$. In view of the formula

(3) $$(X \cup \Delta_1) \cap (\Delta_0 \cup Y) = X \cup Y,$$

we can calculate that

$$z = \langle z_0, z_1 \rangle.$$

Moreover, if $z = \langle x,y \rangle$, then

$$\langle x,y \rangle_0 = x \text{ and } \langle x,y \rangle_1 = y.$$

The one-one correspondence required is thus established. □

There is more going on in the proof of 3.2 than just a one-one correspondence between elements and pairs. The extra information is best formalized by introducing a notation for mappings.

DEFINITION 3.3. *Projection mappings*

$$p_0 : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_0 \text{ and } p_1 : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_1$$

are defined as relations where

$$(X \cup Y)\ p_0\ X' \text{ iff } X \subseteq X', \text{ and } (X \cup Y)\ p_1\ Y' \text{ iff } Y \subseteq Y'$$

hold for all $X, X' \in \mathcal{D}_0$ and $Y, Y' \in \mathcal{D}_1$. Given $f : \mathcal{D}_2 \to \mathcal{D}_0$ and $g : \mathcal{D}_2 \to \mathcal{D}_1$, the *paired mapping*

$$\langle f,g \rangle : \mathcal{D}_2 \to \mathcal{D}_0 \times \mathcal{D}_1$$

is defined as a relation where

$$Z \langle f,g \rangle (X \cup Y) \text{ iff } Z\ f\ X \text{ and } Z\ g\ Y$$

holds for all $X \in \mathcal{D}_0$, $Y \in \mathcal{D}_1$, and $Z \in \mathcal{D}_2$. □

PROPOSITION 3.4. The mappings p_0, p_1 and $\langle f,g \rangle$ are approximable mappings, provided f and g are, and we have:

(i) $p_0 \circ \langle f, g \rangle = f$ and $p_1 \circ \langle f, g \rangle = g$.

Moreover, for $z \in |\mathcal{D}_0 \times \mathcal{D}_1|$, we have:

(ii) $p_0(z) = z_0$ and $p_1(z) = z_1$,

in the notation of the proof of 3.2. Further if $h : \mathcal{D}_2 \to \mathcal{D}_0 \times \mathcal{D}_1$ is any approximable mapping, then

(iii) $h = \langle p_0 \circ h, p_1 \circ h \rangle$.

Moreover, for all $w \in |\mathcal{D}_2|$, we have:

(iv) $\langle f,g \rangle (w) = \langle f(w), g(w) \rangle$,

where again on the right-hand side the notation of the proof of 3.2 is used. □

The proof of this result is left as an exercise. Note the consequence that there is a one-one correspondence between pairs of approximable mappings $f : \mathcal{D}_2 \to \mathcal{D}_0$ and $g : \mathcal{D}_2 \to \mathcal{D}_1$ and mappings $h : \mathcal{D}_2 \to \mathcal{D}_0 \times \mathcal{D}_1$. It is clear that we generalize all this to products

$$\mathcal{D}_0 \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1}$$

of several systems.

The product construct also neatly explains functions of several variables. In Exercise 2.19 we used the informal notation

$$f : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2$$

and suggested regarding f as a ternary relation

$$X, Y \; f \; Z .$$

But now with $\mathcal{D}_0 \times \mathcal{D}_1$ given an independent meaning, all we have to do is to regard f as a binary relation with

$$(X \cup Y) \; f \; Z$$

equivalent to the old relationship. We can also employ an element-wise notation as in $f(\langle x, y \rangle)$, which can more easily be written $f(x, y)$. Similar remarks apply to functions of more than two arguments.

We have discussed several times what it means for a function f(x) to come from an approximable mapping. It is interesting to ask the analogous question for functions of several arguments.

THEOREM 3.5. An elementwise function

$$f : |\mathcal{D}_0 \times \mathcal{D}_1| \to |\mathcal{D}_2|$$

of two arguments comes from an approximable mapping iff for each fixed $a \in |\mathcal{D}_0|$ and each fixed $b \in |\mathcal{D}_1|$ the transformations

$$x \mapsto f(x, b) \text{ and } y \mapsto f(a, y)$$

come from approximable mappings of one argument.

Proof: As this is the first time we have had to deal with constants in functions, a lemma is useful.

LEMMA 3.6. Given $b \in |\mathcal{D}_1|$, the constant function

$$b : |\mathcal{D}_0| \to |\mathcal{D}_1|$$

where $b(x) = b$ for all $x \in |\mathcal{D}_0|$, comes from the approximable mapping such that

$$X \; b \; Y, \text{ iff } Y \in b,$$

for all $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. □

There is no real confusion here in using "b" both for function and value. Returning, then, to the proof of 3.5, we see that the reason that $x \mapsto f(x, b)$ comes from an approximable mapping is that the mapping in question is the composition of two approximable mappings, namely $f \circ \langle I_{\mathcal{D}_0}, b \rangle$. Clearly we can interchange the rôles of $\mathcal{D}_0$ and $\mathcal{D}_1$ to get at $y \mapsto f(a, y)$.

Conversely, assume that both these functions come from approximable mappings no matter the choice of a and b. Clearly the mapping to determine f is the relation from $X \cup Y$ to Z where

$$Z \in f(\Uparrow X, \Uparrow Y) = f(\Uparrow(X \cup Y)).$$

To prove that this determines f we calculate by the formula of Exercise 2.9:

$$\begin{aligned} f(x,y) &= \bigcup\{f(\uparrow X, y) \mid X \in x\} \\ &= \bigcup\{\bigcup\{f(\uparrow X, \uparrow Y) \mid Y \in y\} \mid X \in x\} \\ &= \bigcup\{f(\uparrow X, \uparrow Y) \mid X \in x \text{ and } Y \in y\} \\ &= \bigcup\{f(\uparrow(X \cup Y)) \mid (X \cup Y) \in \langle x,y\rangle\}. \end{aligned}$$

And, again by 2.9, this is what was needed. □

Said more informally, a function of several arguments is approximable in all the variables *jointly* if it is approximable in each of the variables *separately*.

The type of argument used in 3.5 in the first half of the proof also provides a generalization of 2.6 to functions of several arguments. When we form a function like

$$f(g(x,z,\ldots), h(y,x,\ldots), k(z,w,\ldots),\ldots)$$

from given functions $f, g, h, k, \ldots$; we call the process *substitution*.

PROPOSITION 3.7. The functions of several arguments between domains coming from approximable mappings are closed under substitution.

Proof: An example will establish the method. Suppose there are four variables involved taking values in domains provided by systems $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$. We might have a substitution like:

$$f(g(x_0, x_1), h(x_1, x_2), k(x_3, x_0, x_2)).$$

Here it might be that the values of the functions inside come from quite other systems; for instance,

$$k : \mathcal{D}_3 \times \mathcal{D}_0 \times \mathcal{D}_2 \to \mathcal{D}_4$$

might be possible. By using projections

$$p_i : \mathcal{D}_0 \times \mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{D}_3 \to \mathcal{D}_i,$$

where $i < 4$, we can assure that we have several functions all on the same product; thus,

$$k \circ \langle p_3, p_0, p_2\rangle : \mathcal{D}_0 \times \mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{D}_3 \to \mathcal{D}_4.$$

Now no matter on what domains f is defined, the following composition makes sense:

$f \circ \langle g \circ \langle p_0, p_1 \rangle, h \circ \langle p_1, p_2 \rangle, k \circ \langle p_3, p_0, p_2 \rangle\rangle$;
and in fact this is the desired function. Writing it this way makes it clear that the function comes from an approximable mapping: we apply 3.3 (generalized, of course, to products with several terms) to construe the parts between brackets < and > as approximable mappings, and then by this trick the composition $\circ$ is the ordinary composition of 2.6. □

It has to be admitted that there is a slight point overlooked in forming products like $\mathcal{D} \times \mathcal{D}$ with two identical domains. This is discussed in Exercise 3.14, invoking explicit isomorphisms.

The construct that makes the whole theory of domains work so smoothly is the function - space construct: it is possible to regard functions as *objects* which form a domain. Look back at Definition 2.1 and compare it with the original definition of element in 1.6. There are obvious formal similarities, except that filters are sets of neighbourhoods and mappings are sets of pairs of neighbourhoods (relations). But as we saw in 1.10 it is possible to turn the filters into tokens *via* a simple definition of neighbourhood. We apply the same kind of definition to the mappings.

DEFINITION 3.8. Given neighbourhood systems $\mathcal{D}_0$ and $\mathcal{D}_1$, the *function space* $(\mathcal{D}_0 \to \mathcal{D}_1)$ is the system whose set of tokens is the set of approximable mappings of Definition 2.1 and whose neighbourhoods are finite non-empty intersections of sets of the form

$$[X, Y] = \{f : \mathcal{D}_0 \to \mathcal{D}_1 \mid X \, f \, Y\},$$

where $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. □

We have been calling our mappings "approximable" for a long time now without saying exactly how they can be approximated! Definition 3.8 supplies the missing key, because once a domain has been defined, then the general theory gives an explicit meaning to the word approximation. We still have to verify, however, that the mappings do correspond to the elements of the domain.

PROPOSITION 3.9. Let neighbourhoods $X_i \in \mathcal{D}_0$ and $Y_i \in \mathcal{D}_1$ be given for $i < n$. Then the set of $[X_i, Y_i]$ for $i < n$ is consistent in $(\mathcal{D}_0 \to \mathcal{D}_1)$ iff the following condition holds:

(i) whenever $I \subseteq \{0, 1, \ldots, n-1\}$ and $\{X_i \mid i \in I\}$ is consistent in $\mathcal{D}_0$, then $\{Y_i \mid i \in I\}$ must be consistent in $\mathcal{D}_1$.

Moreover, when consistency holds, the least approximable mapping f_0 belonging to the intersection of the $[X_i, Y_i]$ is defined by:

(ii) $X f_0 Y$ iff $\bigcap \{Y_i \mid X \subseteq X_i\} \subseteq Y$

for $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$.

Proof: Suppose the $[X_i, Y_i]$ are consistent in $(\mathcal{D}_0 \to \mathcal{D}_1)$. Since the function space is being defined outright as a positive system, consistency means

$$f \in \bigcap \{[X_i, Y_i] \mid i < n\}$$

for some $f : \mathcal{D}_0 \to \mathcal{D}_1$. Now, with f in hand, let us check condition (i). Suppose $\{X_i \mid i \in I\}$ is consistent. This means

$$x \in \bigcap \{[X_i] \mid i \in I\}$$

for some $x \in |\mathcal{D}_0|$. Suppose $i \in I$, so $x \in [X_i]$. Since $X_i f Y_i$ holds, $f(x) \in [Y_i]$. This means, therefore, that

$$f(x) \in \bigcap \{[Y_i] \mid i \in I\},$$

and so $\{Y_i \mid i \in I\}$ is consistent.

For the converse, suppose (i) is the case. We take (ii) as the definition of a mapping and remark that for an arbitrary $X \in \mathcal{D}_0$, the set $\{X_i \mid X \subseteq X_i\}$ is automatically consistent in $\mathcal{D}_0$. By our assumption, the set $\{Y_i \mid X \subseteq X_i\}$ is therefore consistent. This means that

$$\bigcap \{Y_i \mid X \subseteq Y_i\} \in \mathcal{D}_1.$$

(Keep in mind that i is restricted to those $i < n$, and there are only finitely many neighbourhoods being considered here.) It is thus almost immediate that the relation f_0 defined by (ii) satisfies conditions of 2.1 and so is an approximable mapping $f_0 : \mathcal{D}_0 \to \mathcal{D}_1$. By construction

$$X_i f_0 Y_i$$

holds trivially for all $i < n$; therefore,

$$f_0 \in \bigcap \{[X_i, Y_i] \mid i < n\}$$

and the desired consistency is established.

Finally suppose that f is any mapping in the neighbourhood under discussion; this means $X_i \, f \, Y_i$ holds for all $i < n$. Suppose $X \, f_0 \, Y$ holds. We have for $X \subseteq X_i, X \, f \, Y_i$; so

$$X \, f \, \bigcap \{Y_i \mid X \subseteq X_i\} \subseteq Y.$$

Thus, $X \, f \, Y$ follows; hence, as relations, $f_0 \subseteq f$. In other words f_0 is the minimal element of the neighbourhood. □

We note that, as a consequence of what we have just proved, when the neighbourhood is consistent, then

$$\bigcap \{[X_i, Y_i] \mid i < n\} \subseteq [X, Y]$$

is exactly equivalent to

$$\bigcap \{Y_i \mid X \subseteq X_i\} \subseteq Y.$$

Note also that a single neighbourhood $[X_0, Y_0]$ is always consistent since it contains the *constant mapping* k where

$$X \, k \, Y \quad \text{iff} \quad Y_0 \subseteq Y,$$

for all $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. Some other simple observations about these neighbourhoods are just translations of the conditions of Definition 2.1:

$$[\Delta_0, \Delta_1] = |\mathcal{D}_0 \to \mathcal{D}_1|;$$

$$[X, Y] \cap [X, Y'] = [X, Y \cap Y']; \text{ and}$$

$$X' \subseteq X \text{ and } Y \subseteq Y' \text{ imply } [X, Y] \subseteq [X', Y'],$$

for all $X, X' \in \mathcal{D}_0$ and $Y, Y' \in \mathcal{D}_1$. We are now ready to prove the main result about the construct.

THEOREM 3.10. Given neighbourhood systems $\mathcal{D}_0$ and $\mathcal{D}_1$, the function space system $(\mathcal{D}_0 \to \mathcal{D}_1)$ is complete in the sense that every filter in $|\mathcal{D}_0 \to \mathcal{D}_1|$ is fixed by a unique approximable mapping.

Proof: Let $f : \mathcal{D}_0 \to \mathcal{D}_1$ be an approximable mapping. By the very definition of $(\mathcal{D}_0 \to \mathcal{D}_1)$ it determines a filter by the definition:

$$\hat{f} = \{F \in (\mathcal{D}_0 \to \mathcal{D}_1) \mid f \in F\}.$$

Trivially $[X,Y] \in \hat{f}$ iff $f \in [X,Y]$ iff $X\, f\, Y$; so this filter uniquely determines the relation f. What we have to show is that every filter in $|\mathcal{D}_0 \to \mathcal{D}_1|$ is of this form.

Suppose $\varphi \in |\mathcal{D}_0 \to \mathcal{D}_1|$ is any filter. A relation can be defined at once by

$$X\, \hat{\varphi}\, Y \quad \text{iff} \quad [X,Y] \in \varphi.$$

In view of the remarks we made just before stating this theorem, there is no problem in showing that $\hat{\varphi}$ is an approximable mapping. Since the neighbourhoods of the function space are in any case finite intersections of sets like $[X,Y]$, those $[X,Y] \in \varphi$ generate φ. This means that $\hat{\hat{\varphi}} = \varphi$. By definition $\hat{\hat{f}} = f$, so there is a one-one correspondence between mappings and filters. (This correspondence is obviously inclusion preserving, too.) □

We now know just about everything about $|\mathcal{D}_0 \to \mathcal{D}_1|$ as a domain: the elements correspond isomorphically to the approximable mappings; the finite elements are explained completely by 3.9; and we have seen how to calculate with neighbourhoods. The final step is to relate the function space to other domains by appropriate mappings. In doing this we shall freely construe elements of $|\mathcal{D}_0 \to \mathcal{D}_1|$ as approximable mappings in view of 3.10.

THEOREM 3.11. Given neighbourhood systems $\mathcal{D}_1$ and $\mathcal{D}_2$, there is a uniquely determined approximable mapping

$$\text{eval} : (\mathcal{D}_1 \to \mathcal{D}_2) \times \mathcal{D}_1 \to \mathcal{D}_2,$$

where for all $f : \mathcal{D}_1 \to \mathcal{D}_2$ and all $x \in |\mathcal{D}_1|$ we have

(i) $\text{eval}(f, x) = f(x)$.

Proof: For $F \in (\mathcal{D}_1 \to \mathcal{D}_2)$ and $X \in \mathcal{D}_1$ and $Y \in \mathcal{D}_2$ define eval as a relation by:

$$F \cup X \ \text{eval}\ Y \ \text{iff}\ X\, f\, Y \ \text{for all}\ f \in F.$$

Remember that neighbourhoods in the function space are sets of approximable mappings. It is easily checked that this definition makes eval approximable. We now calculate the function values by the formula of 2.2 (i):

$$\text{eval}\ (f, x) = \{Y \in \mathcal{D}_2 \mid \exists F \in (\mathcal{D}_1 \to \mathcal{D}_2)\ \exists X \in x.\ f \in F \text{ and } F\ U\ X\ \text{eval}\ Y$$

Because, again by 2.2 (i), we have

$$f\ (x) = \{Y \in \mathcal{D}_2 \mid \exists X \in x\ .\ X\ f\ Y\},$$

we can see from the definition of eval that $\text{eval}\ (f, x) \subseteq f\ (x)$. Suppose that $Y \in f(x)$. Then $X\ f\ Y$ holds for some $X \in x$. We can write $f \in [X, Y] \in (\mathcal{D}_1 \to \mathcal{D}_2)$ and it is clear that

$$[X, Y]\ U\ X\ \text{eval}\ Y$$

holds by definition. Therefore, $Y \in \text{eval}\ (f, x)$, and so $f(x) \subseteq \text{eval}\ (f, x)$. □

This theorem is essential for our programme: it shows that in taking functions as objects the very basic operation of forming the function value is an approximable mapping. In other words we can treat the expression $f(x)$ not just as a function of x, as we have done from the start, but also as a function of f as well. The result also indicates that there are useful maps defined on domains that themselves are function spaces; we shall meet many more of these. The next theorem provides further examples.

THEOREM 3.12. Given neighbourhood systems $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2$ there is associated with every approximable mapping $g: \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2$ a uniquely determined approximable mapping

$$\text{curry}\ (g) : \mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2)$$

such that for $x \in |\mathcal{D}_0|$ and $y \in |\mathcal{D}_1|$

(i) $\text{curry}\ (g)(x)(y) = g(x,y)$.

Moreover we have these functional equations:

(ii) $\text{eval} \circ \langle \text{curry}\ (g) \circ p_0,\ p_1 \rangle = g$, and

(iii) $\text{curry}\ (\text{eval} \circ \langle h \circ p_0,\ p_1 \rangle) = h$,

where the $p_i : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_i$ are the projection mappings and $h : \mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2)$ is any approximable mapping. This provides an isomorphism between the domains $|\mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2|$ and $|\mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2)|$ and so we can regard

$$\text{curry} : (\mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2) \to (\mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2))$$

as itself being an approximable mapping.

Proof : Given g as indicated, we can define curry (g) as a relation and as an approximable mapping by:

$$X \text{ curry } (g)\ [Y,Z] \quad \text{iff } X \cup Y\ g\ Z \qquad \text{(but see Exer. 5.21)}$$

for all $X \in \mathcal{D}_0$, $Y \in \mathcal{D}_1$, $Z \in \mathcal{D}_2$. This is sufficient because an approximable mapping is intersective in the right-hand neighbourhood, so we know from the above exactly what X curry$(g) \bigcap\{[Y_i, Z_i] \mid i < n\}$ means for all finite intersections. The remark after 3.9 is then helpful in checking that by this definition curry (g) satisfies the monotonicity condition and so is indeed approximable. We now calculate :

$$\begin{aligned}
\text{curry } (g)\ (x)\ (y) &= \{Z \in \mathcal{D}_2 \mid \exists\, Y \in y\ .\ Y \text{ curry } (g)\ (x)\ Z\ \} \\
&= \{Z \in \mathcal{D}_2 \mid \exists\, Y \in y\ \exists X \in x.\ X \text{ curry}(g)[Y,Z]\} \\
&= \{Z \in \mathcal{D}_2 \mid \exists\, Y \in y\ \exists X \in x.\ X \cup Y\ g\ Z\} \\
&= \{Z \in \mathcal{D}_2 \mid \exists\, W \in \langle x,y\rangle.\ W\ g\ Z\} \\
&= g\ (\langle x,y\rangle) = g(x,y).
\end{aligned}$$

This proves (i). We also see, that if we take the left-hand side of (ii) and apply it to a pair $\langle x,y\rangle$, it reduces to $g(x,y)$ by virtue of (i). Thus, the two functions in (ii) are the same.

Turning to (iii), call the left-hand side k. Using (i) again, we find

$$\begin{aligned}
k(x)(y) &= \text{eval} \circ \langle h \circ p_0,\ p_1\rangle\ (\langle x,y\rangle) \\
&= \text{eval}\ (\langle h \circ p_0\ (\langle x,y\rangle,\ p_1(\langle x,y\rangle)\rangle) \\
&= \text{eval}\ (\langle h\ (x)\ , y\rangle) \\
&= h\ (x)(y).
\end{aligned}$$

As this is true for all $y \in |\mathcal{D}_1|$, then $k(x) = h(x)$ follows. As this is true for all $x \in |\mathcal{D}_0|$, then $k = h$ follows, and (iii) is proved.

Taking (ii) and (iii) together, it is clear that the domains $|\mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_2|$ and $|\mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2)|$ are in a one-one correspondence. But from the very definition of curry it is clear that

$$\text{curry}\ (g) \subseteq \text{curry}\ (g') \text{ iff } g \subseteq g'.$$

Hence, curry is an isomorphism, and we can invoke 2.7 to conclude that it comes from an approximable mapping. □

We close this lecture with some order-theoretic properties of function spaces that characterize inclusion and upper bounds of functions in a "pointwise" manner.

THEOREM 3.13. For approximable functions $f,g : \mathcal{D}_0 \to \mathcal{D}_1$ we have

(i) $f \subseteq g$ iff $f(x) \subseteq g(x)$ for all $x \in |\mathcal{D}_0|$.

For subsets $F \subseteq |\mathcal{D}_0 \to \mathcal{D}_1|$ we have

(ii) F is bounded in $|\mathcal{D}_0 \to \mathcal{D}_1|$ iff $\{f(x) \mid f \in F\}$ is bounded in $|\mathcal{D}_1|$ for each $x \in |\mathcal{D}_0|$;

and in that case for all $x \in |\mathcal{D}_0|$:

(iii) $(\bigsqcup F)(x) = \bigsqcup \{f(x) \mid f \in F\}$.

Proof. The implication in (i) from left to right follows because evaluation is monotone in the function as well as the argument. The converse implication is a consequence of 2.2(ii).

For the proof of (ii) and (iii) we see that by (i) if F is bounded, so is every set $\{f(x) \mid f \in F\}$. For the converse direction, it is clear that (iii) defines *some* pointwise mapping; we have only to prove that it is *approximable*. The calculation that $\bigsqcup F$ preserves directed unions (see 2.9 and 2.11) is probably the simplest way to reach the conclusion. □

EXERCISES

EXERCISE 3.14. For the most part we can assume that there is at most a *countable number* of tokens; thus, without loss of generality the underlying sets Δ_i of given systems $\mathcal{D}_i$ could be assumed to be subsets of Σ^* where $\Sigma = \{0,1\}$. (Any denumerable set would do.) Show that the product $\mathcal{D}_0 \times \mathcal{D}_1$ could be defined as the system over the set $0\Delta_0 \cup 1\Delta_1$ where

$$\mathcal{D}_0 \times \mathcal{D}_1 = \{0X \cup 1Y \mid X \in \mathcal{D}_0 \text{ and } Y \in \mathcal{D}_1\}.$$

In other words, the assumption of the disjointness of Δ_0 and Δ_1 is unnecessary. Give, therefore, the revised definition of $\langle x,y \rangle$ for elements, and prove that for a single system $\mathcal{D}$, there exists an approximable mapping

$$\text{diag} : \mathcal{D} \to \mathcal{D} \times \mathcal{D}$$

where $\text{diag}(x) = \langle x,x \rangle$ for all $x \in |\mathcal{D}|$. Also extend the definition to a product of n-factors

$$\mathcal{D}_0 \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1}$$

which will be a system over the set

$$\bigcup_{i<n} 1^i 0\Delta_i \quad .$$

Note that for a 2-termed product we simplify $10\Delta_1$ to $1\Delta_1$.

EXERCISE 3.15. Establish the usual isomorphisms:

(i) $\mathcal{D}_0 \times \mathcal{D}_1 \cong \mathcal{D}_1 \times \mathcal{D}_0$;

(ii) $\mathcal{D}_0 \times (\mathcal{D}_1 \times \mathcal{D}_2) \cong (\mathcal{D}_0 \times \mathcal{D}_1) \times \mathcal{D}_2 \cong \mathcal{D}_0 \times \mathcal{D}_1 \times \mathcal{D}_2$.

How does the product of no factors fit in? Prove also:

(iii) $\mathcal{D}_0 \cong \mathcal{D}'_0$ and $\mathcal{D}_1 \cong \mathcal{D}'_1$ imply $\mathcal{D}_0 \times \mathcal{D}_1 \cong \mathcal{D}'_0 \times \mathcal{D}'_1$.

EXERCISE 3.16. Let $\mathcal{D}$ be a given neighbourhood system over $\Delta \subseteq \Sigma^*$. Define

$$\Delta^\infty = \bigcup_{n=0}^{\infty} 1^n 0 \Delta$$

so that Δ^∞ is split into infinitely many disjoint copies of Δ. Let $\mathcal{D}^\infty$ be the least family of subsets of Σ^* where

(1) $\Delta^\infty \in \mathcal{D}^\infty$, and

(2) whenever $X \in \mathcal{D}$ and $Y \in \mathcal{D}^\infty$, then $0X \cup 1Y \in \mathcal{D}^\infty$.

Show that $\mathcal{D}^\infty$ is a neighbourhood system over Δ^∞. Prove the isomorphism

$$\mathcal{D}^\infty \cong \mathcal{D} \times \mathcal{D}^\infty \ .$$

Show, moreover, that the elements of $|\mathcal{D}^\infty|$ are in a one-one correspondence with arbitrary *infinite sequences* $\langle x_n \rangle_{n=0}^{\infty}$ of elements $x_n \in |\mathcal{D}|$ by using combinations of neighbourhoods

$$0X_0 \cup 10X_1 \cup \cdots \cup 1^n 0X_n \cup \cdots$$

where from some point on all the X_m are equal to Δ.

EXERCISE 3.17. Using the $\mathcal{B}$ and $\mathcal{T}$ of Example 2.3 show there is a one-one approximable mapping

$$f : \mathcal{B} \to \mathcal{T}^\infty$$

and another approximable mapping

$$g : \mathcal{T}^\infty \to \mathcal{B}$$

such that

$$g \circ f = I_{\mathcal{B}} \text{ and } f \circ g \subseteq I_y .$$

Are $\mathcal{B}$ and $\mathcal{T}^\infty$ isomorphic? Are $\mathcal{B}$ and $\mathcal{T} \times \mathcal{B}$ isomorphic?

EXERCISE 3.18. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be neighbourhood systems over Δ_0 and Δ_1, where we again assume that these are subsets of Σ^*. We assume that in addition *no neighbourhood is empty*. Why is this possible without loss of generality? Define the *sum system* by:

$$\mathcal{D}_0 + \mathcal{D}_1 = \{\{\Lambda\} \cup 0\Delta_0 \cup 1\Delta_1\} \cup \{0X \mid X \in \mathcal{D}_0\} \cup \{1Y \mid Y \in \mathcal{D}_1\}.$$

Prove that this is a neighbourhood system over $\{\Lambda\} \cup 0\Delta_0 \cup 1\Delta_1$. (Throwing in $\{\Lambda\}$ was not all that necessary, but note that

$$\mathcal{B} = \mathcal{B} + \mathcal{B} ,$$

and this is an equality of sets not just an isomorphism of systems.) Prove that in general there are mappings

$$in_i : \mathcal{D}_i \to \mathcal{D}_0 + \mathcal{D}_1 \quad \text{and} \quad out_i : \mathcal{D}_0 + \mathcal{D}_1 \to \mathcal{D}_i$$

where $out_i \circ in_i = I_{\mathcal{D}_i}$. Where does the assumption $\emptyset \notin \mathcal{D}_i$ come in here? How can these sums be generalized to n-terms? (Hint: As for products use sets $1^i 0\Delta_i$.) Draw some pictures.

EXERCISE 3.19. Suppose we are given systems and approximable mappings

$$f : \mathcal{D}_0 \to \mathcal{D}_0' \quad \text{and} \quad g : \mathcal{D}_1 \to \mathcal{D}_1' .$$

Prove there are approximable mappings

$$f \times g : \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_0' \times \mathcal{D}_1 \quad \text{and} \quad f + g : \mathcal{D}_0 + \mathcal{D}_1 \to \mathcal{D}_0' + \mathcal{D}_1'$$

such that

(i) $(f \times g)(x,y) = \langle f(x), g(y) \rangle$

for all $x \in |\mathcal{D}_0|$ and $y \in |\mathcal{D}_1|$, and rewrite this as:

(ii) $f \times g = \langle f \circ p_0, g \circ p_1 \rangle$.

In addition prove that

(iii) $out_0 \circ (f + g) \circ in_0 = f$, and

(iv) $out_1 \circ (f + g) \circ in_1 = g$.

Do equations (iii) and (iv) uniquely determine $f + g$?

EXERCISE 3.20. (For category theorists). Show that the result of 3.19 can be used to prove that + and x on the category of domains and approximable maps are indeed functors. Show further that x *is* the categorical product for this category.

EXERCISE 3.21. In the proofs of 3.12 in the definition of curry (g) it is rather cavalierly assumed that the neighbourhood [Y,Z] uniquely determines Y and Z. Show that this is true if $Z \neq \Delta_2$. (Hint: Find explicitly the least of $f \in [Y,Z]$.) Show that if $Z = \Delta_2$ the biconditional stated at the start of the proof is still valid even though Y is not uniquely determined. (Hint: Remember that Δ_1 g Δ_2 must hold.) For arbitrary pairs of neighbourhoods of $(\mathcal{D}_1 \to \mathcal{D}_2)$ is there a simple criterion for identity?

EXERCISE 3.22. Prove that there is an approximable mapping

$$\text{comp}: (\mathcal{D}_1 \to \mathcal{D}_2) \times (\mathcal{D}_0 \to \mathcal{D}_1) \to (\mathcal{D}_0 \to \mathcal{D}_2)$$

where for all $g : \mathcal{D}_1 \to \mathcal{D}_2$ and $f : \mathcal{D}_0 \to \mathcal{D}_1$ we have

$$\text{comp}\ (g\ f) = g \circ f.$$

Show this directly by writing down the neighbourhood relation and by building the mapping up from eval and curry (on suitable domains) using $\circ$ and $\langle , \rangle$. (Hint: Fill in maps in the following sequence of domains:

$$(\mathcal{D}_0 \to \mathcal{D}_1) \times \mathcal{D}_0 \to \mathcal{D}_1$$

$$(\mathcal{D}_1 \to \mathcal{D}_2) \times ((\mathcal{D}_0 \to \mathcal{D}_1) \times \mathcal{D}_0) \to (\mathcal{D}_1 \to \mathcal{D}_2) \times \mathcal{D}_1$$

$$((\mathcal{D}_1 \to \mathcal{D}_2) \times (\mathcal{D}_0 \to \mathcal{D}_1)) \times \mathcal{D}_0 \to (\mathcal{D}_1 \to \mathcal{D}_2) \times \mathcal{D}_1$$

$$((\mathcal{D}_1 \to \mathcal{D}_2) \times (\mathcal{D}_0 \to \mathcal{D}_1)) \times \mathcal{D}_0 \to \mathcal{D}_2$$

$$(\mathcal{D}_1 \to \mathcal{D}_2) \times (\mathcal{D}_0 \to \mathcal{D}_1) \to (\mathcal{D}_0 \to \mathcal{D}_2)$$

The maps are of course not uniquely determined, but the shifting of brackets ought to suggest the right choice.)

EXERCISE 3.23. (For category theorists.) Show that the results of 3.11 and 3.12 prove that the category of domains and approximable mappings is a *cartesian closed category*. (Mac Lane [1971] pp. 95-96 may be consulted for a very brief introduction.) What is the *terminal domain* in this category? What sort of functor is $(\mathcal{D}_0 \to \mathcal{D}_1)$?

EXERCISE 3.24. Establish some more isomorphisms :

(i) $(\mathcal{D}_0 \to (\mathcal{D}_1 \times \mathcal{D}_2)) \cong (\mathcal{D}_0 \to \mathcal{D}_1) \times (\mathcal{D}_0 \to \mathcal{D}_2)$

(ii) $(\mathcal{D}_0 \to \mathcal{D}_1^{\infty}) \cong (\mathcal{D}_0 \to \mathcal{D}_1)^{\infty}$

(iii) $\mathcal{D}_0 \times (\mathcal{D}_1 + \mathcal{D}_2) \cong (\mathcal{D}_0 \times \mathcal{D}_1) + (\mathcal{D}_0 \times \mathcal{D}_2)$

(iv) $(\mathcal{D}_0 + \mathcal{D}_1) \to \mathcal{D}_2 \cong (\mathcal{D}_0 \to \mathcal{D}_2) \times (\mathcal{D}_1 \to \mathcal{D}_2)$

If some of the above are *not* true, perhaps at least some mapping relationships can be established.

EXERCISE 3.25. (For topologists.) Recall from Exercises 1.21 and 2.13 on how to regard a domain $|\mathcal{D}|$ as a topological space. Using 3.10 show that the family of open subsets of $|\mathcal{D}|$ is isomorphic to a domain.

EXERCISE 3.26. Show that for every domain $\mathcal{D}$ there is an approximable mapping

$$\text{cond} : \mathcal{T} \times \mathcal{D} \times \mathcal{D} \to \mathcal{D},$$

called the *conditional operator*, satisfying

(i) cond (true, x, y) = x

(ii) cond (false, x, y) = y

(iii) cond ($\perp$, x, y) = $\perp$.

(Hint: Recalling that $\mathcal{T} = \{\{0\}, \{1\}, \{0,1\}\}$, define cond as a relation by

$$0C \cup 10X \cup 110Y \text{ cond } Z \text{ iff } 0 \in C \text{ and } X \subseteq Z \text{ or } 1 \in C \text{ and } Y \subseteq Z \text{ or } 0,1 \in C \text{ and } \Delta \subseteq Z ,$$

where $C \in \mathcal{T}$ and $X \in \mathcal{D}$ and $Y \in \mathcal{D}$ and where we are using the construction of Exercise 3.14.) Find a similar operator in the domain

$$\mathcal{T} \times \mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_0 + \mathcal{D}_1 .$$

Show also there is an approximable mapping

$$\text{which} : \mathcal{D}_0 + \mathcal{D}_1 \to \mathcal{T}$$

such that for all $x \in |\mathcal{D}_0 + \mathcal{D}_1|$

$$\text{cond}\,(\text{which}(x),\ \text{in}_0(\text{out}_0(x)),\ \text{in}_1(\text{out}_1(x))) = x.$$

EXERCISE 3.27. (For set theorists.) Give another proof that the family of approximable mappings $f : \mathcal{D}_0 \to \mathcal{D}_1$ is isomorphic to a domain by employing the general argument of Exercise 2.22. How does this compare with the proof method of 3.9 and 3.10? Can the general remarks also be employed to show that

$$\text{eval} : (\mathcal{D}_1 \to \mathcal{D}_2) \times \mathcal{D}_1 \to \mathcal{D}_2$$

is approximable without bringing in the neighbourhoods in such an explicit way? (Hint: Use 3.5 and the idea of Exercise 2.12.)

EXERCISE 3.28. In the function space $(\mathcal{D}_0 \to \mathcal{D}_1)$ let

$$\bigcap \{[X_i, Y_i] \mid i < n\}$$

be a (non-empty) neighbourhood. In 3.9 the minimal element of this neighbourhood is characterized as a relation f_0. Show that as an elementwise mapping it can be defined by the formula

$$f_0(x) = \bigsqcup \{\uparrow Y_i \mid x \in [X_i]\},$$

for $x \in |\mathcal{D}_0|$. Try to draw a picture of $|\mathcal{D}_0|$ with neighbourhoods $[X_i]$ and the corresponding values of the function f_0.

LECTURE IV

FIXED POINTS AND RECURSION

Having at this point a large supply of examples of domains (and further constructs of new domains), we now have to consider some other ways of defining functions - other than by explicit compositions of the very basic functions already mentioned. One of the most fruitful techniques is an infinitely *iterated* composition that is at the back of the idea of *recursion*. We will use the process over and over again in these lectures, not only to define new functions but also to define new domains. The heart of the matter lies in the so-called "Fixed-point Theorem":

THEOREM 4.1. For any approximable mapping $f : \mathcal{D} \to \mathcal{D}$ on any domain, there exists a *least* element $x \in |\mathcal{D}|$ where

$$f(x) = x.$$

Proof: Let f^n for $n \in \mathbb{N}$ stand for the n-fold composition of f with itself. That is,

$$f^0 = I_{\mathcal{D}}, \text{ and}$$
$$f^{n+1} = f \circ f^n.$$

Define

$$x = \{X \in \mathcal{D} \mid \Delta f^n X, \text{ for some } n \in \mathbb{N}\}.$$

We see $X \in x$ iff there is a finite sequence $\Delta = X_0, X_1, \ldots, X_n = X$ where $X_i \, f \, X_{i+1}$ holds for all $i < n$. Now since $\Delta \, f \, \Delta$ automatically holds, a sequence for an $X \in x$ can always be extended to a longer sequence just by adding more Δ's on the front.

We want to prove $x \in |\mathcal{D}|$. Clearly $\Delta \in x$, and if $X \subseteq Y$ and $X \in x$, then $Y \in x$. All that remains to be shown is the closure of x under intersection. Note that if

$$U \, f \, V \text{ and } U' \, f \, V'$$

hold and U, U' are consistent in $\mathcal{D}$, then V and V' are consistent and

$$(U \cap U')\ f\ (V \cap V')$$

must hold. Generalizing this to sequences, if

$$\Delta = X_0\ f\ X_1\ f\ \cdots\ f\ X_n = X, \text{ and}$$
$$\Delta = Y_0\ f\ Y_1\ f\ \cdots\ f\ Y_n = Y$$

both hold (and note we have arranged the lengths of the two sequences to be equal), then each pair X_i, Y_i is consistent and we have

$$\Delta = (X_0 \cap Y_0)\ f\ (X_1 \cap Y_1)\ f\ \cdots\ f(X_n \cap Y_n) = X \cap Y.$$

This establishes the desired closure.

We also note that if $X \in x$ and XfY then $Y \in x$. Therefore, $f(x) \subseteq x$ and indeed by its very construction x is the *least* element of $|\mathcal{D}|$ with this property. (Why?) But f is monotone, so $f(f(x)) \subseteq f(x)$; hence, $x = f(x)$. By what we have already said it must be the least such element. □

Because the element we have shown to exist in 4.1 is a least element, it is *unique*. That is, we have associated with each $f : \mathcal{D} \to \mathcal{D}$ a special element $x_f \in |\mathcal{D}|$ determined by the choice of f. A function has therefore been defined mapping the set $|\mathcal{D} \to \mathcal{D}|$ into $|\mathcal{D}|$. The next result shows that this function, or operator on functions, is in fact approximable.

THEOREM 4.2. For any domain $\mathcal{D}$, there is an approximable mapping

$$\mathrm{fix} : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$$

such that if $f : \mathcal{D} \to \mathcal{D}$ is any approximable mapping, then

(i) $\mathrm{fix}\ (f) = f\ (\mathrm{fix}\ (f))$.

Furthermore, if $x \in |\mathcal{D}|$, then

(ii) $f(x) \subseteq x$ implies $\mathrm{fix}(f) \subseteq x$.

And this last property implies that fix is unique. Explicitly we can characterize fix by the equation:

(iii) $\mathrm{fix}\ (f) = \bigcup_{n=0}^{\infty} f^n(\perp)$,

for all $f : \mathcal{D} \to \mathcal{D}$

Proof: Formula (iii) can be put in a more elementary form:

$$\mathrm{fix}\ (f) = \{X \mid \Delta\, f^n X, \text{ for some } n \in \mathbb{N}\}.$$

To show an elementwise mapping approximable we can use the formula of Exercise 2.9, applied to the above as the definition of fix:

$$(*) \qquad \mathrm{fix}\ (f) = \bigcup \{\mathrm{fix}\ (\uparrow F) \mid f \in [F]\},$$

where F ranges over the neighbourhoods of $(\mathcal{D} \to \mathcal{D})$, and where $\uparrow F$ can be considered to be the least element of F as calculated in 3.9.

Now from the definition of fix, it is clear that whenever $f \subseteq g$, then $\mathrm{fix}\,(f) \subseteq \mathrm{fix}\,(g)$, because $f^n \subseteq g^n$. (That is, fix is obviously monotone.) Next, if $f \in F$, then $\uparrow F$ is a (finite) approximation to f; so $\uparrow F \subseteq f$ and $\mathrm{fix}\ (\uparrow F) \subseteq \mathrm{fix}\ (f)$. This means that half of equation (*) already holds by monotonicity. All that is left is to prove the other half.

So suppose $X \in \mathrm{fix}\ (f)$. Then, as we have already remarked, there is a finite sequence of neighbourhoods where

$$\Delta = X_0\ f\ X_1 \cdots X_{n-1}\ f\ X_n = X.$$

Let the function-space neighbourhood be defined as

$$F = \bigcap \{[X_i, X_{i+1}] \mid i < n\},$$

and note that since $f \in [F]$ we have at once consistency. But, by 3.9, $\uparrow F \in [F]$, so the *same* sequence of X_i is sufficient to show that

$$X \in \mathrm{fix}\ (\uparrow F).$$

In other words, if X belongs to the left-hand side of (*), it also belongs to the right-hand side. This completes the proof of (*).

Formula (i) is just a restatement of what we proved in 4.1. And (ii) follows easily, because $f(x) \subseteq x$ implies that $\Delta \in x$ and whenever $X \in x$ and $X\ f\ Y$, then $Y \in x$. Thus, by induction, if $\Delta\, f^n X$, then $X \in x$. So $\mathrm{fix}\ (f) \subseteq x$.

Finally, if $\mathrm{fax} : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ were any other operator satisfying (i) and (ii), we would prove at once that

$$\mathrm{fix}\ (f) \subseteq \mathrm{fax}\ (f) \quad \text{and}$$
$$\mathrm{fax}\ (f) \subseteq \mathrm{fix}\ (f).$$

That is to say, the two operators are identical. □

The reader may have noticed that we used recursion in the proof of 4.1 (we had to define f^n for all $n \in \mathbb{N}$). But 4.1 and 4.2 can be used to justify definitions by recursion on a large number of domains - definitions where the process of iteration is far from being as straightforward. In discussing this point, let us start with some basic examples.

EXAMPLE 4.3. The infinite generalization of our original example 1.2 is the system

$$N = \{\{n\} \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\}$$

The total elements are clearly in a one-one correspondence with the integers in $\mathbb{N}$. We can apply the construction of Exercise 3.16 to obtain a domain

$$F = N^{\infty}.$$

So we already know quite a bit about this domain - but it has a much more familiar presentation.

Let Φ be the set of all *finite partial functions* $\varphi \subseteq \mathbb{N} \times \mathbb{N}$ (that is, finite sets of ordered pairs of integers where, if $(n,m) \in \varphi$ and $(n,m') \in \varphi$, then $m = m'$). Define

$$\uparrow \varphi = \{\psi \in \Phi \mid \varphi \subseteq \psi\}.$$

Consider the neighbourhood system

$$F' = \{\uparrow\varphi \mid \varphi \in \Phi\}.$$

It is an easy exercise to show that F and F' are isomorphic and that the elements of these domains correspond exactly to the (possibly infinite) *partial functions* $\pi \subseteq \mathbb{N} \times \mathbb{N}$. Moreover, the *total* elements just correspond to the *total* functions $\tau : \mathbb{N} \to \mathbb{N}$ ("function" in the ordinary, set-theoretical sense of the word).

Another easy exercise is to show that the domains

$$F \text{ and } (N \to N)$$

by our definitions are *NOT* isomorphic; though the two domains are closely related. We can define a mapping

$$\mathrm{val} : F \times N \to N$$

by the relationship

$$\uparrow\varphi \cup \{n\}\ \mathrm{val}\ \{m\} \text{ iff } (n, m) \in \varphi.$$

(Of course val has to relate other neighbourhoods such as:

$$\uparrow \varphi \cup \mathbb{N}\ \mathrm{val}\ \mathbb{N},$$

but these are all.) It is then simple to prove that if $\pi \in |F|$ is regarded as a partial function $\pi : \mathbb{N} \to \mathbb{N}$ and if for $n \in \mathbb{N}$ we define $\hat{n} \in |N|$ by

$$\hat{n} = \{\{n\}, \mathbb{N}\},$$

then we have

$$\mathrm{val}\ (\pi, \hat{n}) = \widehat{\pi(n)}, \text{ if } \pi \text{ is defined at } n;$$
$$= \{\mathbb{N}\}, \text{ otherwise.}$$

(Remember that $\{\mathbb{N}\} \in |N|$ is the "undefined" element.)
This means that

$$\mathrm{curry}\ (\mathrm{val}) : F \to (N \to N)$$

is a one-one function on elements. (The rather slight trouble with $(N \to N)$ is that it has *more* elements than F.)

So much for the construction of F, we now wish to consider mappings

$$f : F \to F$$

and their uses. Consider the possibility

$$f(\pi)(n) = 0, \text{ if } n = 0;$$
$$= \pi(n-1) + n - 1, \text{ if } n > 0.$$

If π were a total function, then $f(\pi)$ would be total. But if π is partial, and if it is, say, undefined at k, then $f(\pi)$ becomes undefined at $k + 1$. Note that $f(\pi)$ is always defined at 0. Note, too, that f is an approximable mapping because it is completely determined by what it does to finite (partial) functions. Indeed,

$$f(\pi) = \bigcup \{f(\varphi) \mid \varphi \subseteq \pi\},$$

where φ ranges over Φ.

Well, we have proved that every approximable map of a domain into itself has a (least) fixed point. What is the least fixed point of this f? Suppose $\sigma = f(\sigma)$. Then $\sigma(o) = 0$, and

$$\begin{aligned}\sigma(n+1) &= f(\sigma)(n+1)\\ &= \sigma(n) + n.\end{aligned}$$

By induction, then

$$\sigma(n) = \sum_{i<n} i$$

and σ is a total function. (Therefore, f has a *unique* fixed point.)

Actually, we can make the procedure more systematic by defining as fixed points elements of $(N \to N)$ rather than F. In the first place we have $\hat{0} \in |N|$, and from now on we will not distinguish between n and $\hat{n}$. Next we have two mappings:

$$\text{succ, pred} : N \to N$$

where, as approximable mappings we have

$$\begin{aligned}&X \text{ succ } Y \text{ iff } \exists n \in \mathbb{N} .\ n \in X \text{ and } n+1 \in Y,\\ &X \text{ pred } Y \text{ iff } \exists n \in \mathbb{N} .\ n+1 \in X \text{ and } n \in Y,\end{aligned}$$

for all $X, Y \in N$. This is *correct*, but what we mean in more understandable terms is:

$$\begin{aligned}&\text{succ } (n) = n + 1;\\ &\text{pred } (n) = n - 1, \text{ if } n > 0;\\ &\qquad\quad = \bot, \quad \text{if } n = 0.\end{aligned}$$

Here, n has been identified with $\hat{n} \in |N|$ and $\bot = \{\mathbb{N}\} \in |N|$. Moreover, we have a mapping

$$\text{zero} : N \to T$$

which is such that

$$\begin{aligned}\text{zero}(n) &= \text{true}, \text{ if } n = 0;\\ &= \text{false}, \text{ if } n > 0.\end{aligned}$$

The *structured domain*

$$\langle N, 0, \text{succ}, \text{pred}, \text{zero} \rangle$$

can be called "THE domain of integers" for our present theory. We shall meet many other structured domains in the sequel.

Now the iterated summation function σ can be completely characterized - as a map $\sigma : N \to N$ rather than as an element $\sigma \in |F|$ - by the following equation:

$$\sigma(n) = \text{cond}\ (\text{zero}(n), 0, \sigma(\text{pred}(n)) + \text{pred}(n))$$

The only problem is that we have not defined $+ : N \times N \to N$. (A direct definition is left to the reader; general remarks are given later.) But + could be any function of two variables in order to make the point about the form of the definition of σ. Remember

$$\text{cond} : T \times N \times N \to N,$$

as defined in Exercise 3.26. We do not put cond in as part of the structure of N because (as should be clear from 3.26) it is part of the structure of T.

The above equation for σ is properly called a *functional equation*; it will be written as a fixed-point equation in Lecture V when we have the notation for the λ-calculus. □

EXAMPLE 4.4. The domain C of finite or infinite binary sequences mentioned in Exercise 2.21 may be regarded as a generalization of N. This can be made plain by saying how we wish to regard C as a structured domain. To do this we should recall what C is as a neighbourhood system. In the first place

$$B = \{\sigma\Sigma^* \mid \sigma \in \Sigma^*\}$$

where $\Sigma = \{0,1\}$. To form the system C we have

$$C = B \cup \{\{\sigma\} \mid \sigma \in \Sigma^*\}.$$

The total elements of B correspond to *infinite* binary sequences; while the total elements of C to *finite or infinite* sequences. To simplify notation let us write for $\sigma \in \Sigma^*$

$$\sigma = \uparrow\{\sigma\} \qquad \text{(a total element)};$$

$$\sigma\bot = \uparrow\sigma\Sigma^* \qquad \text{(a partial element)}.$$

In other words we identify σ with the corresponding total element in $|C|$.

We wish now to think of C as a structured domain seen as a kind of generalization of N. The empty sequence Λ will play the rôle of $0 \in |N|$; the map succ has two different analogues for C, however. Just as for B we define for $x \in |C|$ and $\sigma \in \Sigma^*$:

$$\sigma x = \{Y \mid \sigma X \subseteq Y \text{ some } X \in x\},$$

where of course now X and Y range over C. It should be checked that $\sigma\tau$ has the right meaning whether we think of $\tau \in \Sigma^*$ or $\tau \in |C|$. The two "successor" mappings we are looking for are

$$x \mapsto 0x \quad \text{and} \quad x \mapsto 1x.$$

All the maps $x \mapsto \sigma x$ can be obtained as compositions of these iterated as many times as needed.

Here are two questions which we now should ask:

What plays the rôle of zero? The answer is not unique, because in C there are several distinctions that have to be made; in fact we will define three maps:

$$\text{empty, zero, one} : C \to T$$

where the three maps take on truth-values to distinguish various kinds of elements in $|C|$ as follows:

What plays the rôle of pred? The mapping will be called tail, and it is characterized by:

$$\text{tail } (0x) = x,$$
$$\text{tail } (1x) = x, \text{ and}$$
$$\text{tail } (\Lambda) = \perp.$$

It is left to the reader to show that tail exists as an approximable mapping.

empty (Λ) = true,
empty (0x) = false,
empty (1x) = false,
zero (Λ) = false
zero (0x) = true
zero (1x) = false
one (Λ) = false
one (0x) = false
one (1x) = true

Again, it is an exercise to show these are approximable. The structured domain is therefore

⟨C, Λ, 0, 1, tail, empty, zero, one⟩

Note that we have changed the meaning of some of the symbols in passing from N to C. Note too that there is a confusion between 0 as an element and 0 as the map $x \mapsto 0x$. There are just too few symbols! In any case this is only an example and not a philosophy of life, so the reader can be expected not to suffer too much.

An example of a definition of an *element* of $|C|$ by a fixed-point equation is:

$$a = 01a.$$

This equation has one and only one solution in $|C|$, the infinite sequence that alternates 0's and 1's. Note that a is also characterized by:

$$a = 0101a.$$

Another element is

$$b = 010b,$$

which is quite different from a.

An example of a *map* in $|C \to C|$ has the characterization

$$d(\Lambda) = \Lambda$$
$$d(0x) = 00d(x), \text{ and}$$
$$d(1x) = 11d(x).$$

We can write:

$$d(x) = \text{cond}\ (\text{empty}\ (x),\ \Lambda,\ \text{cond}\ (\text{zero}(x),\ 00d(\text{tail}(x)),\ 11d(\text{tail}(x))))$$

As we shall see in due course, this can be regarded as a fixed-point definition of d.

An example of a map in $|C \times C \to C|$ was suggested in 2.21. We can write:

$$x\,y = \text{cond}\ (\text{empty}\ (x),\ y,\ \text{cond}\ (\text{zero}\ (x),\ 0(\text{tail}(x)\ y),\ 1\ (\text{tail}\ (x)\ y)))$$

It should be checked that this equation exactly characterizes the intended mapping. □

The examples we have given with N and C are examples of definitions of functions by *recursion*. The literal meaning of "recursion" is "running backwards", and a look at the equations for our examples will show that the functions are characterized by giving their values either *outright* (e.g. at 0 or at Λ) or at *earlier* arguments (e.g. at pred(x) or at tail(x)). The reader should keep in mind that a recursive "definition" is not really a definition in the sense of *explicit definition* but rather is a characterization; a theorem has to be proved to show that such functions exist. Now we have a general definition of domain and a general theorem on fixed points and a general construction of function-space domain; *THEREFORE*, we know that there are solutions to our equations *PROVIDED THAT* the variables range over elements of a domain and that the other, given functions that appear in the equations are already known to be approximable (continuous). This proviso is very important, and we shall remark on it time after time.

But, as is well known, recursion also can be done over *sets* like $\mathbb{N}$, and we should examine now the connection between the familiar kind of recursion and what we are doing over domains. Of course, one simple connection is already provided by the way we regard $\mathbb{N}$ as a subset of N. But there are other useful connections that can be employed in a way that may seem more direct.

DEFINITION 4.5. A structured set $\langle \mathbb{N}, 0, {}^{+} \rangle$, where $0 \in \mathbb{N}$ and ${}^{+}: \mathbb{N} \to \mathbb{N}$ is a unary function, is said to be *a model for Peano's Axioms* if the following conditions are satisfied:

(i) $0 \neq n^{+}$, for all $n \in \mathbb{N}$;

(ii) $n^{+} = m^{+}$ implies $n = m$, for all $n, m \in \mathbb{N}$;

(iii) whenever $x \subseteq \mathbb{N}$ and $0 \in x$ and $x^{+} \subseteq x$, then $x = \mathbb{N}$.

Here $x^{+} = \{n^{+} \mid n \in x\}$. □

Clause (iii) is recognized as the principle of mathematical induction stated in terms of sets. We usually think of $\mathbb{N}$ as being "God given", and (i) - (iii) as known without question. Suppose God, however, decides to withdraw His set of integers and substitute another. We can ask: "Oh! Why did You take from us our beloved numbers? Why must we now live with these strange new beasts?" God will probably reply "Trust Me!" Perhaps we should in view of the theorem:

THEOREM 4.6. All models of Peano's Axioms are isomorphic.

Proof: There are several ways to give the proof, but, for the sake of illustration, an application of the fixed-point theorem is appropriate here. Let $\langle \mathbb{N}, 0, {}^{+} \rangle$ be one model, and let $\langle \mathbb{M}, \square, {}^{\#} \rangle$ be another. Let $\mathbb{N} \times \mathbb{M}$ be the ordinary cartesian product of the two sets and let

$$P(\mathbb{N} \times \mathbb{M})$$

be the powerset (set of all subsets) of $\mathbb{N} \times \mathbb{M}$. As in Exercises 1.15 and 2.20, we regard this set of elements as a domain, whose finite elements are just the finite subsets of the given set $\mathbb{N} \times \mathbb{M}$. The following mapping on $u \subseteq \mathbb{N} \times \mathbb{M}$ is easily proved approximable:

$$u \mapsto \{(0, \square)\} \cup \{(n^{+}, m^{\#}) \mid (n, m) \in u\}.$$

(This assertion should be checked as an exercise.) We thus let r be the (least) fixed point:

$$r = \{(0, \square)\} \cup \{(n^{+}, m^{\#}) \mid (n, m) \in r\}$$

This $r \subseteq \mathbb{N} \times \mathbb{M}$ as a binary relation will turn out to be a one-one correspondence giving the required isomorphism.

First of all we see by construction that

(i) $0\ r\ \Box$;

(ii) $n\ r\ m$ implies $n^+\ r\ m^\#$.

So, if r proves to be a one-one correspondence, it will then be the desired isomorphism. Now, the two sets shown in the equation

$$\{(0,\Box)\} \cap \{(n^+,m^\#) \mid (n,m) \in r\} = \emptyset$$

are disjoint by virtue of axiom 4.5(i). Therefore, 0 in $\mathbb{N}$ corresponds by r to one and only one element of $\mathbb{M}$, namely the element $\Box$. Let $x \subseteq \mathbb{N}$ be the set of all elements of $\mathbb{N}$ corresponding by r to a unique element of $\mathbb{M}$. We have just shown $0 \in x$. Suppose $n \in x$, and let $m \in \mathbb{M}$ be the unique element with $n\ r\ m$. Now $n^+\ r\ m^\#$ holds, so n^+ corresponds to at least one element of $\mathbb{M}$. If $n^+\ r\ k$ also holds, then since $(n^+, k) \neq (0,\Box)$, the fixed-point equation implies

$$n^+ = n_0^+ \text{ and } k = m_0^\#$$

for some $(n_0,m_0) \in r$. By axiom 4.5(ii), $n = n_0$, and, by uniqueness (remember $n \in x$), $m = m_0$; thus, $m^\#$ is the unique correspondent for n^+. We have proved $n^+ \in x$. Therefore, $x^+ \subseteq x$; so by 4.5(iii), $x = \mathbb{N}$ holds. Otherwise said, *every* element in $\mathbb{N}$ corresponds to a unique element of $\mathbb{M}$.

Note that the rôles of $\mathbb{N}$ and $\mathbb{M}$ are completely symmetric, *and* they satisfy the same axioms as structured sets. It follows, then, that every element of $\mathbb{M}$ corresponds to a unique element of $\mathbb{N}$. The proof that r is a one-one correspondence is now complete. □

EXERCISES

EXERCISE 4.7. Formula 4.2(iii) shows how to find the *least* fixed point of $f : \mathcal{D} \to \mathcal{D}$. Suppose on the other hand that $a \in |\mathcal{D}|$ is such that $a \subseteq f(a)$. Will there be a fixed point $x = f(x)$ with $a \subseteq x$?

(Hint: How do we know $\bigcup_{n=0}^{\infty} f^n(a) \in |\mathcal{D}|$?)

EXERCISE 4.8. Suppose $f : \mathcal{D} \to \mathcal{D}$ and $S \subseteq |\mathcal{D}|$ are such that

(i) $\perp \in S$;

(ii) $x \in S$ always implies $f(x) \in S$;

(iii) whenever $\{x_n\}_{n=0}^{\infty} \subseteq S$ and $x_n \subseteq x_{n+1}$ for all n, then $\bigcup_{n=0}^{\infty} x_n \in S$.

Conclude that $fix(f) \in S$. (This could be called the principle of *fixed-point induction*.) Apply the method to a set of the form

$$S = \{x \in |\mathcal{D}| \mid a(x) = b(x)\},$$

where $a, b : \mathcal{D} \to \mathcal{D}$ are approximable, and where we know $a(\perp) = b(\perp)$, and $f \circ a = a \circ f$ and $f \circ b = b \circ f$.

EXERCISE 4.9. Show that there is an approximable operator

$$\Psi : ((\mathcal{D} \to \mathcal{D}) \to \mathcal{D}) \to ((\mathcal{D} \to \mathcal{D}) \to \mathcal{D})$$

such that for $\Theta : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ and $f : \mathcal{D} \to \mathcal{D}$ we have

$$\Psi\ (\Theta)\ (f) = f(\Theta\ (f)).$$

Prove further that $fix : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ is the least fixed point of Ψ.

EXERCISE 4.10. Given a domain $\mathcal{D}$ and an element $a \in |\mathcal{D}|$, construct a domain $\mathcal{D}_a$ where

$$|\mathcal{D}_a| = \{x \in |\mathcal{D}| \mid x \subseteq a\}.$$

Show that if $f : \mathcal{D} \to \mathcal{D}$ is approximable, then f can be *restricted* to an approximable map $f' : \mathcal{D}_{fix(f)} \to \mathcal{D}_{fix(f)}$ where $f'(x) = f(x)$ for all $x \in |\mathcal{D}_{fix(f)}|$.

How many fixed points does f' have in $|\mathcal{D}_{fix(f)}|$?

EXERCISE 4.11. (Suggested by G. Plotkin). We can regard fix as assigning a fixed-point operator to each domain $\mathcal{D}$. Show that fix is uniquely determined by the following general conditions on an assignment $\mathcal{D} \mapsto F_{\mathcal{D}}$:

(i) $F_{\mathcal{D}} : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$;

(ii) $F_{\mathcal{D}}(f) = f(F_{\mathcal{D}}(f))$ for all $f : \mathcal{D} \to \mathcal{D}$;

(iii) whenever $f_0 : \mathcal{D}_0 \to \mathcal{D}_0$ and $f_1 : \mathcal{D}_1 \to \mathcal{D}_1$ are given and $h : \mathcal{D}_0 \to \mathcal{D}_1$ is such that $h(\perp) = \perp$ and $h \circ f_0 = f_1 \circ h$, then

$$h(F_{\mathcal{D}_0}(f_0)) = F_{\mathcal{D}_1}(f_1).$$

(Hint: Apply 4.7 to prove fix satisfies (iii). In the other direction use 4.10.)

EXERCISE 4.12. Need an approximable $f : \mathcal{D} \to \mathcal{D}$ have a *maximum* fixed point? Give an example where there are *many* fixed points.

EXERCISE 4.13. The proof of 4.1 uses the integers, whereas the proof of 4.6 uses 4.1. There is a hint of circularity here! It can be eliminated by the following steps:

(1) If a domain $\mathcal{D}$ has an element a where, for $f : \mathcal{D} \to \mathcal{D}$ the relation $f(a) \subseteq a$ holds, then the least fixed point can be defined by

$$fix(f) = \bigcap \{x \in |\mathcal{D}| \mid f(x) \subseteq x\}.$$

Note that $fix(f) \subseteq a$. (Hint: Remark that by 1.17 the formula gives a well-defined element. Call the element b. Prove that $f(b) \subseteq b$ by showing that $f(b) \subseteq x$ whenever $f(x) \subseteq x$. Then note that $f(f(b)) \subseteq f(b)$ so that $b \subseteq f(b)$ also. Conclude $b = fix(f)$ as least fixed point.)

(2) Remark that this proof uses only the monotonicity property of $f : |\mathcal{D}| \to |\mathcal{D}|$. Remark, too, that (1) can always be applied to power-set domains PA for any set A.

(3) Review the proof of 4.6 and establish by a fixed-point method that for any structured set $\langle Z, z, ^{\bullet} \rangle$ there is a *unique* function $s : \mathbb{N} \to Z$ such that

(i) $s(0) = z$;

(ii) $s(n^{+}) = s(n)^{\bullet}$, for $n \in \mathbb{N}$.

(4) Employ (3) for the proof of 4.1 by identifying $\langle Z, z, ^{\circ} \rangle$.

EXERCISE 4.14. Need a *monotone* function $f : \mathcal{P}A \to \mathcal{P}A$ always have a *maximum* fixed point?

EXERCISE 4.15. (For set theorists.) Let $f : |\mathcal{D}| \to |\mathcal{D}|$ be a monotone function on (the elements of) a domain. Show that f has a *maximal* fixed point (i.e. a fixed point that cannot be extended to a larger fixed point). (Hint: By Zorn's Lemma consider a maximal chain

$$C \subseteq \{x \in |\mathcal{D}| \mid x \subseteq f(x)\},$$

and use 2.11 to remark that $\bigcup C \in |\mathcal{D}|$.) Now argue that f has a least fixed point.

EXERCISE 4.16. (For fixed-point nuts). Show that a monotone function as in 4.15 has an "optimal" fixed point in the sense that it is the greatest fixed point below all the maximal fixed points and at the same time it is the largest fixed point consistent with all other fixed points. *Consistency* for sets of *elements* means having a common upper bound. (Hint: Follow these steps:

(1) Show that any non-empty set S of fixed points has a largest fixed point below by using the formula

$$f(\bigcap S) \subseteq \bigcap S$$

and finding the least fixed point over $\bigcap S$.

(2) Letting a be the fixed point of (1) constructed from the set of maximal fixed points, remark that a is consistent with any other fixed point $x = f(x)$, since x can be extended to a maximal one. Suppose b is consistent with all fixed points, then $b \subseteq y$ if y is maximal. (Why?).)

EXERCISE 4.17. (For algebraists). Suppose $\langle S, 1, \cdot \rangle$ is a semigroup with unit (sometimes called a *monoid*). Remark that $\mathcal{P}S$ is a domain. For $a, b \in S$, what is the least $x \in \mathcal{P}S$ such that

$$x = \{1\} \cup \{a, b\} \cup x \cdot x$$

where in general for $x, y \subseteq S$

$$x \cdot y = \{t \cdot u \mid t \in x \text{ and } u \in y\}$$

Need the fixed point be unique?

EXERCISE 4.18. In Example 4.3 there are many unproved assertions about N and F. These should be checked. In particular, the isomorphism theorem of 4.6 could be proved by constructing a simple domain M from $\mathbb{M}$ in the way N is constructed from $\mathbb{N}$.

EXERCISE 4.19. There are many unproved assertions in Example 4.4! In particular discuss "Peano's Axioms" for $\{0,1\}^*$. Show, moreover, that one : $C \to T$ can be defined from the rest of the structure by a fixed-point equation.

EXERCISE 4.20. For approximable $f, g : \mathcal{D} \to \mathcal{D}$ prove that

$$\text{fix}\,(f \circ g) = f(\text{fix}\,(g \circ f)).$$

EXERCISE 4.21. Show that the less-than-or-equal-to relation $\ell \subseteq \mathbb{N} \times \mathbb{N}$ is uniquely determined by the fixed point equation

$$\ell = \{(n,n) \mid n \in \mathbb{N}\} \cup \{(n,m^+) \mid (n,m) \in \ell\}$$

Consider the structured set $\langle P\mathbb{N}, \mathbb{N}, {}^+ \rangle$ where, as before,

$$x^+ = \{n^+ \mid n \in x\}.$$

What is the unique function $[\cdot] : \mathbb{N} \to P\,\mathbb{N}$ given by 4.13(3)? Prove that the structures $\langle \mathbb{N}, 0, {}^+ \rangle$ and $\langle [m], m, {}^+ \rangle$ are uniquely isomorphic for each $m \in \mathbb{N}$, and connect the isomorphism with ordinary addition of integers. Can the same be done for multiplication? (Hint: Consider the fixed-point equation:

$$n \cdot \mathbb{N} = \{0\} \cup \{n + m \mid m \in n \cdot \mathbb{N}\},$$

where $n \in \mathbb{N}$ is fixed.)

EXERCISE 4.22. Suppose $\mathbb{N}^*$ is a structured set satisfying only axioms (i) and (ii) of 4.5. Must there be a subset $\mathbb{N} \subseteq \mathbb{N}^*$ that satisfies (i), (ii), and (iii)? (Hint: Use a least fixed point in $P\,\mathbb{N}^*$.) (For set theorists): How do we know from the axioms of set theory that there exists such a set $\mathbb{N}^*$?

EXERCISE 4.23. (Suggested by S. Eilenberg). Suppose $f : \mathcal{D} \to \mathcal{D}$ is approximable on a given domain $\mathcal{D}$. Suppose $a_n : \mathcal{D} \to \mathcal{D}$ is a sequence of approximable maps where

(i) $a_0(x) = \perp$, for all $x \in |\mathcal{D}|$

(ii) $a_n \sqsubseteq a_{n+1}$ in $\mathcal{D} \to \mathcal{D}$, for all $n \in \mathbb{N}$;

(iii) $\bigsqcup_{n=0}^{\infty} a_n = I_{\mathcal{D}}$ in $\mathcal{D} \to \mathcal{D}$;

(iv) $a_{n+1} \circ f = a_{n+1} \circ f \circ a_n$, for all $n \in \mathbb{N}$.

Prove that f has a *unique* fixed point. (Hint: Show that if $x = f(x)$, then $a_n(x) \sqsubseteq a_n(fix(f))$ for all $n \in \mathbb{N}$ by induction on n.)

EXERCISE 4.24. (For set theorists). Let $f : A \to B$ and $g : B \to A$ be one-one functions (*into*, not necessarily *onto*!) Prove the Schroeder - Bernstein theorem to the effect that there exists a one-one correspondence $h : A \leftrightarrow B$. (Hint: (Suggested by A. Tarski). By the fixed-point theorem find $X \subseteq A$ where

$$X = (A - g(B)) \cup g(f(X))$$

where $f(X)$ = the image of the set f under the function f. Define $h \subseteq A \times B$ as a union of two restrictions:

$$h = f \restriction X \cup g^{-1} \restriction (A - X).$$

A picture helps.)

EXERCISE 4.25. Perhaps the domains N and C are not exactly analogous? C was based on $\{0,1\}$ as the underlying set of tokens. Construct a system C_1 based on $\{1\}^*$ (= finite strings of 1's) with neighbourhoods:

$$C_1 = \{\{1^m \mid m \geq n\} \mid n \in \mathbb{N}\} \cup \{\{1^n\} \mid n \in \mathbb{N}\}.$$

What structure should be put on C_1 strictly analogous to that on C ($= C_2$)? What kinds of approximable maps relate N, C_1, and C_2? Draw some pictures.

LECTURE V

TYPED λ - CALCULUS

In Examples 4.3 and 4.4, after suitable domains have been constructed, functions are characterized by recursion equations whose form of expression is - basically - a composition or substitution of known functions together with the function to be defined. This method can be made more precise and more easily usable by expanding our notation for functions - particularly by inventing a "temporary" notation for a function as a thing in itself without having to have special letters for functions. The device is called λ - *abstraction*. It is related to ordinary set abstraction (the $\{x \mid \cdots\}$ - notation already much used in these lectures), but we gear the approach to domains and their elements, and especially to function spaces.

At this stage it would not be so helpful to produce a rigorously formal definition of the syntax of the typed λ - calculus; we shall try to suggest what is needed by example. There are so many examples at hand, the less formal discussion ought to be sufficient.

In the first place we should set aside, in the notational store room as it were, a stock of variables

$$x, y, z, w, \ldots \quad .$$

These variables will be required in different "sizes" or "types". Roughly speaking there should be an infinite number of variables to range over the elements of each domain $\mathcal{D}$. We could perhaps write

$$x_0^{\mathcal{D}}, \; x_1^{\mathcal{D}}, \; x_2^{\mathcal{D}}, \; \ldots,$$

but the subscripts to insure an infinity of variables and the superscripts to record the typing of the variables lead to a notation as

tiresome to write as it is to read. We simply agree that we can have as many variables as we need and that they come in all the types.

Strictly speaking we should also introduce type *symbols* and not confuse types with domains. But if the reader will simply keep in mind that *form* in language has always to be kept distinct from *content*, the confusion at the type level will not matter so very much. A point at which the confusion might cause a real confusion concerns *compound types*. Given $\mathcal{D}_0$ and $\mathcal{D}_1$ we can form such compounds as

$$\mathcal{D}_0 + \mathcal{D}_1, \quad \mathcal{D}_0 \times \mathcal{D}_1, \quad \mathcal{D}_0 \to \mathcal{D}_1.$$

What has to be remembered is that a compound domain (neighbourhood system), $\mathcal{D}_0 \times \mathcal{D}_1$ say, does not uniquely determine the "parts" $\mathcal{D}_0$ and $\mathcal{D}_1$. (We could make it do so, but it would cost some effort.) Of course, the *symbol* "$\mathcal{D}_0 \times \mathcal{D}_1$" has well defined parts. The point is that *different* ways of forming a compound domain could lead to the *same* result, meaning that a domain does not let us retrace its exact history of construction. Compound symbols, however, always carry their histories around with them, since otherwise they would not be readable. What we want, of course, are *both* domain symbols *and* domains, the latter being the meanings of the former. Most of the time we can happily pretend that it is only the domains themselves we have to think about.

Besides variables, we will also need certain *constants*. For instance, the symbol 0 (perhaps, better 0^N) denotes a certain element of $|N|$. Similarly, in view of Theorem 4.2, for each domain $\mathcal{D}$ there is a well-determined element $\mathrm{fix}^{\mathcal{D}}$ of the compound type $((\mathcal{D} \to \mathcal{D}) \to \mathcal{D})$ denoting the least fixed-point operator. We have considered any number of similar constants of a great variety of types already (cf. 4.3 and 4.4; cond is an especially good one). We can say that the variables and constants are *atomic terms*, where "atomic" here means non-compound.

To form compound terms, there are several means: for example, if $\tau,\ldots,\sigma$ is a list of already obtained terms (including variables or constants), then we can form an ordered *tuple*

$$\langle \tau, \ldots, \sigma \rangle .$$

We have already done so in 3.1. If the types of $\tau, \ldots, \sigma$ are $\mathcal{D}, \ldots, \mathcal{D}'$, respectively, then the type of the tuple is the product domain

$$\mathcal{D} \times \cdots \times \mathcal{D}' ,$$

because we intend that the tuple denote an element of this domain. (The tuple notation for *functions* as in 3.3 is being forgotten for the time being.)

Next suppose that τ has type $(\mathcal{D}_0 \to \mathcal{D}_1)$ and σ has type $\mathcal{D}_0$, then the usual *function-value notation*

$$\tau\ (\sigma)$$

is a compound term of type $\mathcal{D}_1$. We also use

$$\tau\ (\sigma_0, \ldots, \sigma_{n-1})$$

as an abbreviation of

$$\tau\ (<\sigma_0, \ldots, \sigma_{n-1}>),$$

where, if the types of $\sigma_0, \ldots, \sigma_{n-1}$ are $\mathcal{D}_0, \ldots, \mathcal{D}_{n-1}$, then the type of τ has to be of the form

$$((\mathcal{D}_0 \times \cdots \times \mathcal{D}_{n-1}) \to \mathcal{D}_n)$$

where $\mathcal{D}_n$ is the type of the compound. In this manner, with functions applied to tuples, we have the full facility of substitution into functions of many variables just by iterating the notation.

Having taken into account function *value*, it remains to provide for function *definition*. Suppose that $x_0, \ldots, x_{n-1}$ is a list of distinct variables of types $\mathcal{D}_0, \ldots, \mathcal{D}_{n-1}$. Suppose further that τ is a term - no matter how complicated - of type $\mathcal{D}_n$. Then we can regard τ as defining a function of n - variables of type

$$((\mathcal{D}_0 \times \cdots \times \mathcal{D}_{n-1}) \to \mathcal{D}_n).$$

What we have not done is to reward our regard by, as yet, providing a quick-to-write "name" for that function. This we now do; it is called

$$\lambda x_0, \ldots, x_{n-1}.\ \tau ,$$

where we stress that the x_i must be *distinct* variables and that this

expression denotes the *whole function*. That is why we provide it with a special symbol.

Here is an example of the λ-notation

$$\lambda\, x,y.\ x\,,$$

which is read "lambda ex wye ... (pause) ... ex". If the types of x and y are $\mathcal{D}_0$ and $\mathcal{D}_1$, then the type of the above is

$$((\mathcal{D}_0 \times \mathcal{D}_1) \to \mathcal{D}_0).$$

Indeed, we know this function very well: it is the *first projection function* p_0 of 3.3 and the equation

$$p_0 = \lambda\, x,y.\ x$$

is true, as is the equation

$$p_1 = \lambda\, x,y.\ y.$$

In the notation of 3.3, we also find the true equation

$$\langle f,g\rangle = \lambda w.\langle f(w),\ g(w)\rangle,$$

where on the right-hand side we are using "official" λ-notation for a function of type

$$(\mathcal{D}_2 \to (\mathcal{D}_0 \times \mathcal{D}_1)).$$

The notation on the left is just an *abbreviation* and it should not be confused with the pair (2-tuple) of type

$$((\mathcal{D}_2 \to \mathcal{D}_0) \times (\mathcal{D}_2 \to \mathcal{D}_1)).$$

(Since the two domains just mentioned are isomorphic, the possible confusion is not all that serious. On the other hand, one confusion we will completely overlook is that between 1-tuples <x> and elements x. Strictly speaking they are different, but we shall not bother to make the distinction.)

Here are some other examples of true equations:

$$\text{eval} = \lambda f,x.\ f(x) \qquad \text{(cf. 3.11)}$$

$$\text{curry} = \lambda g\, \lambda x\, \lambda y\ .\ g(x,y) \qquad \text{(cf. 3.12)}$$

The first should be immediately clear; while the second is particularly instructive. What is being illustrated is that the λ-notation can

be *iterated*. The distinction being drawn is between

$$\lambda x_0, x_1, \ldots, x_{n-1} . \tau \quad \text{and} \quad \lambda x_0 \lambda x_1 \cdots \lambda x_{n-1} . \tau .$$

The first has type

$$((\mathcal{D}_0 \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1}) \to \mathcal{D}_n) ;$$

while the second has type

$$(\mathcal{D}_0 \to (\mathcal{D}_1 \to (\cdots (\mathcal{D}_{n-1} \to \mathcal{D}_n) \cdots))).$$

This is related also to the true equation

$$\text{curry}(\lambda x,y.\tau) = \lambda x \lambda y.\tau ,$$

which shows that there are operators relating to the two notations. The first is the *multivariate* form; the second is the *curried* form.

Here is another true equation

$$\text{fix} = \text{fix}\ (\lambda F\ \lambda f.\ f(F(f))),$$

where the fix on the left has type $((\mathcal{D} \to \mathcal{D}) \to \mathcal{D})$ and that on the right type

$$(((((\mathcal{D} \to \mathcal{D}) \to \mathcal{D}) \to ((\mathcal{D} \to \mathcal{D}) \to \mathcal{D})) \to ((\mathcal{D} \to \mathcal{D}) \to \mathcal{D}))$$

This is the content of Exercise 4.9. (This also shows why type superscripts are tiresome.)

The combination

$$\text{fix}\ (\lambda x.\tau)$$

occurs so often, that from time to time we abbreviate it as

$$!x.\tau,$$

but remember it only makes sense if x and τ have the *same* type. For example in 4.3 we could have written

$$\sigma = !f\lambda n.\ \text{cond}\ (\text{zero}(n), 0, f(\text{pred}(n)) + \text{pred}(n))$$

and read this as

> "σ is the least (recursively defined) function f whose value at n is cond $(\cdots)$."

We note that in the so-called "body" of the expression inside the

cond-part the variable f occurs again. That is just the point! This is a *recursive* definition; it is made into an *explicit* definition by invoking the least fixed-point operator.

In a λ-expression, $\lambda x,y,z.\tau$, say, the variables x, y, z are being *bound* in τ; but τ may have other variables that are nowhere bound in τ and these remain *free variables* of the whole expression. Bound variables are dummy variables and may be rewritten by other variables; thus

$$\lambda x.\tau = \lambda y.\tau[y/x]$$

is a true equation *PROVIDED* the variable y does not occur in τ. In the equation the notation $\tau[y/x]$ means the result of *substituting* (rewriting) the variable y for the variable x throughout the term τ. We can also write $\tau[\sigma/x]$ for substituting a whole term σ for a variable in the other term.

We have already spoken of "true equations", but how do we know that these curious equations are meaningful at all? They are, but this is something that has to be proved.

THEOREM 5.1. Every typed λ-term τ defines an approximable function of its free variables.

Proof: We argue by an induction on the complexity of τ; there will only be a few cases to consider since the "syntax" of λ-terms is limited — even though terms can be of any length.

If τ is a variable or a constant there is nothing to prove. We already know that

$$x \longmapsto x \quad \text{and} \quad x \longmapsto k$$

are approximable functions.

Suppose τ has the form

$$\langle \sigma_0, \cdots, \sigma_{n-1} \rangle .$$

Then the σ_i are less complex terms, and so we can assume — as our induction hypothesis — that they define approximable functions of the free variables. Having said this, we just apply the already

proved 3.4 to conclude (after a suitable generalization to the multivariate case) that τ, which takes on tuples as values, also defines an approximable function.

Next, suppose τ has the form

$$\sigma_0 \ (\ \sigma_1 \ ),$$

where we are sure that the types of all the terms match properly. Again we can assume the σ_i to be well behaved. But the values we seek can also be written as

$$\text{eval } (\sigma_0, \sigma_1).$$

Since eval is approximable by 3.11, we just have to invoke an instance of 3.7 to gain the desired conclusion.

Finally, suppose that τ has the form

$$\lambda \ x \ . \ \sigma.$$

By a judicious choice of the order of the variables in σ (including x), we can assume that σ defines an approximable function

$$g : \mathcal{D}_0 \times \cdots \times \mathcal{D}_{n-1} \times \ \mathcal{D}_n \to \mathcal{D}'$$

where $\mathcal{D}'$ is the type of σ, $\mathcal{D}_n$ is the type of x, and $\mathcal{D}_0, \ \ldots, \ \mathcal{D}_{n-1}$ are the types of the remaining free variables of σ. We apply 3.12 and obtain an approximable function

$$\text{curry } (g) : \mathcal{D}_0 \times \cdots \times \mathcal{D}_{n-1} \to (\mathcal{D}_n \to \mathcal{D}').$$

But, this is just exactly the function defined by τ.

We leave as an exercise the more general case of a term τ of the form

$$\lambda x_0, \ \ldots, \ x_{k-1} \cdot \sigma \ ,$$

which has a string of bound variables. □

We can now say more precisely what it means to call $\sigma = \tau$ a "true equation". This means that, if we employ the method of the proof of 5.1, the two terms define the *same function* of the free variables. For example,

$$\lambda x.\tau = \lambda y.\tau\,[y/x]$$

is true, provided y does not occur free in the term τ, since the systematic generation of the function defined by $\lambda x.\tau$ does not depend on what the variable x *looks like* but only on its *position* in the term τ. Some other obviously desirable rules for generating true equations are stated in the exercises. But one rule is so basic that we state it here in full generality.

THEOREM 5.2. For suitably typed λ-terms the following equation is true:

$$(\lambda x_0, \ldots, x_{n-1}.\tau)(\sigma_0, \ldots, \sigma_{n-1}) = \tau[\sigma_0/x_0, \ldots, \sigma_{n-1}/x_{n-1}].$$

Proof: It will be sufficient to carry out the proof for n = 1. The proof proceeds by induction on the complexity of the term τ. In case τ is a *constant* k, the result reads

$$(\lambda x.k)(\sigma) = k,$$

and this is a true equation.

In case τ is a *variable* (in particular, the variable x), the result reads

$$(\lambda x.x)(\sigma) = \sigma,$$

and again this is a true equation.

In case τ is a *tuple* (say, $<\tau_0, \tau_1>$) the result reads

$$(\lambda x.<\tau_0, \tau_1>)(\sigma) = <\tau_0[\sigma/x], \tau_1[\sigma/x]>.$$

This is true, because the left-hand side can be transformed by the true equation

$$(\lambda x.<\tau_0, \tau_1>)(\sigma) = <(\lambda x.\tau_0)(\sigma), (\lambda x.\tau_1)(\sigma)>;$$

and then we apply the inductive assumption for τ_0 and for τ_1.

In case τ is an *application*, we want (supposing the term is $\tau_0(\tau_1)$),

$$(\lambda x.\tau_0(\tau_1))(\sigma) = \tau_0[\sigma/x](\tau_1[\sigma/x]).$$

We can proceed as in the last case, noting that the left-hand side equals

$$\text{eval}\ ((\lambda x.\langle \tau_0, \tau_1 \rangle)(\sigma)).$$

In case τ is an *abstract* (say, $\lambda y.\tau_0$), we want

$$(\lambda x.\ \lambda y.\ \tau_0)(\sigma) = \lambda y.\ \tau_0\ [\sigma/x]$$

PROVIDED the variable y is not free in σ. For this we require the true equation

$$(\lambda x.\lambda y.\tau)(\sigma) = \lambda y.(\lambda x.\tau)(\sigma).$$

We argue for this by letting g be the function of n + 2 free variables defined by τ. Then, by 5.1, the λ-term $\lambda x.\lambda y.\tau$ defines the function curry (curry (g)) of n arguments. We can call this function h for the moment. We can write

$$h(v)(\sigma)(y) = g(v, \sigma, y),$$

where v is a *list* of arguments. But, with an appropriate combinator inv, which applied to g inverts the order of the last two arguments, we can write

$$h(v)(\sigma)(y) = \text{curry}\ (\text{inv}\ (g))(v, y)(\sigma).$$

But, curry (inv(g)) is just the function defined by $(\lambda x.\tau)$. So what we have proved as true is

$$(\lambda x.\lambda y.\tau)(\sigma)(y) = (\lambda x.\tau)(\sigma).$$

But if y is not free in α and

$$\alpha(y) = \beta$$

is true, then so is

$$\alpha = \lambda y.\beta\ .$$

This completes the proof. □

We note that if τ' is the term $\lambda x, y.\tau$, then τ' (x,y) means the same as τ. This gives a convenient way of indicating free variables: we just write $\sigma(x,y)$ - where x, y are *not* free in σ - and this will have the same values as any term τ which does involve the extra free variables x and y. We use this notational device in the next theorem.

PROPOSITION 5.3. The least fixed point of

$$\lambda x, y.\ \langle \tau(x,y),\ \sigma(x,y) \rangle$$

is the pair with coordinates

$$!\,x.\tau(x,!\,y.\sigma(x,y)) \quad \text{and}$$

$$!\,y.\sigma(!\,x.\tau(x,y),y).$$

Proof: (We are assuming that x and y are not free in τ and σ.) The purpose of the fixed-point search is to find the least solution of the *pair* of equations

$$x = \tau(x,y) \quad \text{and} \quad y = \sigma(x,y).$$

In other words, we are generalizing the fixed-point equation from one to two variables - and, of course, we could go much further to any number of variables. To this end, let

$$y_* = !\,y.\sigma(!\,x.\tau(x,y),\ y), \quad \text{and}$$

$$x_* = !\,x.\tau(x,y_*).$$

Then

$$x_* = \tau(x_*, y_*),$$

and

$$y_* = \sigma(!\,x.\tau(x,y_*),\ y_*)$$

$$= \sigma(x_*, y_*).$$

This proves that $\langle x_*, y_* \rangle$ is one fixed-point pair.

Suppose, then, that $\langle x_0, y_0 \rangle$ is the least solution. (Why does a least solution have to exist? Hint: Consider a suitable mapping of type

$$\mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_0 \times \mathcal{D}_1\ ,$$

where $\mathcal{D}_0$ is the type of x and $\mathcal{D}_1$ the type of y.) Then we know

$$x_0 = \tau(x_0, y_0) \text{ and } y_0 = \sigma(x_0, y_0),$$

and also $x_0 \subseteq x_*$ and $y_0 \subseteq y_*$. But from

$$\tau(x_0, y_0) \subseteq x_0,$$

it follows that

$$!\,x.\tau(x,y_0) \subseteq x_0.$$

Consequently

$$\sigma(!x.\tau(x,y_0),y_0) \subseteq \sigma(x_0,y_0) \subseteq y_0.$$

By the fixed-point definition of y_*, we have $y_* \subseteq y_0$, so $y_* = y_0$; whence,

$$x_* = !x.\tau(x,y_*) = !x.\tau(x,y_0) \subseteq x_0.$$

So also $x_* = x_0$. We have the right formula for y_0, and a similar argument gives x_0. □

The purpose of giving the above proof was to illustrate the use of the least fixed-point operator in *proofs*. We have such true principles as:

$$!x.\tau(x) = \tau(!x.\tau(x));$$

and

$$\tau(y) \subseteq y \text{ implies } !x.\tau(x) \subseteq y,$$

provided, of course, that x is not free in τ. These, together with the monotonicity of all the functions, were just the methods used in the above proof. Here is another example.

PROPOSITION 5.4. Let x, y, and $\tau(x,y)$ be of the same type $\mathcal{D}$ and let g be of type $(\mathcal{D} \to \mathcal{D})$, then the equation

$$\lambda x\, !y.\tau(x,y) = !g\, \lambda x.\tau(x,g(x))$$

is true.

Proof: Let f be the function on the left-hand side. We can write

$$f(x) = !y.\tau(x,y) = \tau(x,f(x)).$$

Therefore

$$f = \lambda x.\tau(x,f(x)),$$

and it follows that

$$g_0 = !g.\lambda x.\tau(x,g(x)) \subseteq f.$$

Then we have at once, by definition of g_0,

$$g_0(x) = \tau(x,g_0(x)),$$

for any given x. But by definition of f we find

$$f(x) = !y.\tau(x,y) \subseteq g_0(x).$$

As this holds for all x, then $f \subseteq g_0$ follows. So the equation is true. □

The last proof is instructive as it uses equations and inclusions between *functions*. In particular we have just made use of the principle:

> if $\tau \subseteq \sigma$ holds for all values of x,
> then $\lambda x.\, \tau \subseteq \lambda x.\, \sigma$ holds.

This is another form of Theorem 3.13(i).

TABLE 5.5. In the displayed table we give a summary of uses of the λ-notation to define various *combinators*. We have mentioned some of these equations before, and there are some combinators here we have not mentioned before - their meanings, however, should be clear.

$$p_0 = \lambda x, y.\, x$$

$$p_1 = \lambda x, y.\, y$$

$$\text{pair} = \lambda x \lambda y.\, \langle x, y \rangle$$

$$\text{n-tuple} = \lambda x_0 \lambda x_1 \ldots \lambda x_{n-1} \cdot \langle x_0, x_1, \ldots, x_{n-1} \rangle$$

$$\text{diag} = \lambda x.\, \langle x, x \rangle$$

$$\text{funpair} = \lambda f \lambda g \lambda x.\, \langle f(x), g(x) \rangle$$

$$\text{proj}_i^n = \lambda x_0, x_1, \ldots, x_{n-1} \cdot x_i$$

$$\text{inv}_{i,j}^n = \lambda x_0, \ldots, x_i, \ldots, x_j, \ldots, x_{n-1} \cdot \langle x_0, \ldots, x_j, \ldots, x_i, \ldots, x_{n-1} \rangle$$

$$\text{eval} = \lambda f, x.\, f(x)$$

$$\text{curry} = \lambda g \lambda x \lambda y.\, g(x, y)$$

$$\text{comp} = \lambda g, f \lambda x.\, g(f(x))$$

$$\text{const} = \lambda k \lambda x.\, k$$

$$\text{fix} = \lambda f\, !\, x.\, f(x)$$

A TABLE OF COMBINATORS

It is important to note that since we have not typed the variables, these equations are ambiguous: they only become precise when the types are specified. It follows, therefore, that what we find in the table are *schemes* for combinators; there are actually infinitely many distinct combinators corresponding to any one equation depending on how the variables have types chosen for them. Clearly it is better to imagine this variety of combinators than it is to try to notate them with type superscripts.

One interest of combinators is that it is often possible to write expressions without variables - if enough combinators are used. This is sometimes useful, but it can become clumsy. On the other hand , if the same combination occurs over and over, it is sometimes useful to give it a name. This is what we do with, say, *composition* where

$$\mathsf{comp}\ (g, f) = g \circ f.$$

On the one side we have the prefix notation, and on the other, the more common infix notation. With either notation the variable seen in $\lambda x.g(f(x))$ has been got rid of. The choice between equivalent notations ought to be based on a desire for readability. □

The reader will have noted that there are some combinators not appearing in Table 5.5. The reason is that combinators like cond, succ, pred, zero, 0 cannot be defined in the pure λ-notation but are specific to domains like T and N; we, thus, have to regard them as primitive. But once they are in hand, a very large number of other functions can be defined from these combined with λ-expressions. The next theorem gives an indication of the possibilities.

THEOREM 5.6. For every partial recursive function $h : \mathbb{N} \to \mathbb{N}$, there is a λ-term τ of type $(N \to N)$ such that the only constants occurring in τ are

$$\mathsf{cond},\ \mathsf{succ},\ \mathsf{pred},\ \mathsf{zero},\ 0$$

and where if $h(n) = m$, then

$$\tau(n) = m$$

is true; and if h (n) is undefined, then

$$\tau(n) = \bot$$

is true. The equation $\tau(\bot) = \bot$ is also true.

Proof: We have only formulated the theorem for functions of one variable - but to give the proof, it is convenient to pass through functions of any number of (integer) variables. We shall also have to recall the precise definition of the notion of partial recursive function.

It is also convenient to work with *(very) strict* functions

$$f : N^k \to N.$$

These are functions such that if $n_0, \ldots, n_{k-1} \in |N|$ and $n_i = \bot$ for at least one $i < k$, then

$$f(n_0, \ldots, n_{k-1}) = \bot.$$

It is easy to check that compositions of strict functions are strict. It is also easy to see that any *partial* function

$$g : \mathbb{N}^k \to \mathbb{N}$$

extends to a strict (approximable) function

$$\bar{g} : N^k \to N,$$

which takes the same values as g as long as g is defined; otherwise $\bar{g}$ takes the value $\bot$. What we want to show for *partial recursive* g is that the corresponding $\bar{g}$ is defined by a λ-expression.

In the first place we have to check that *primitive recursive* functions have λ-definitions in this sense. We recall that primitive recursive functions are generated from certain elementary starting functions by multi-variate composition and the scheme of primitive recrusion. The starting functions are the constant function with value zero and the "identity" or "projection" functions. For example, $g(n_0, n_1, n_2) = n_1$ for all $n_0, n_1, n_2 \in \mathbb{N}$ is one of the starting functions. Now we cannot just use the λ-term

$$\lambda x_0, x_1, x_2 \cdot x_1$$

to represent $\bar{g}$, because the function so defined is not strict. But any function in $|N^k \to N|$ can be cut down to a strict function by a simple device. Consider

$$\lambda x.\ \mathrm{cond}\ (\mathrm{zero}(x), x, x)$$

with x of type *N*. This is the strict version of the identity function of one argument. The strict projection function of two arguments can be defined by

$$\lambda x_0, x_1.\ \mathrm{cond}\ (\mathrm{zero}(x_1), x_0, x_0).$$

The one of three arguments by:

$$\lambda x_0, x_1, x_2.\ \mathrm{cond}\ (\mathrm{zero}(x_0), \mathrm{cond}\ (\mathrm{zero}(x_2), x_1, x_1), \mathrm{cond}\ (\mathrm{zero}(x_2), x_1, x_1)).$$

This is not done very elegantly, and the reader can find for himself a general solution based on perhaps a better notation for the required compositions of functions.

As we remarked, strict functions are closed under substitution, and any substitution of a batch of functions into another function can be given by a λ-term, if the various functions can themselves be so defined. It only remains to λ-define functions obtained by primitive recursion. Thus, suppose, for the sake of argument, that

$$f : \mathbb{N} \to \mathbb{N} \quad \text{and} \quad g : \mathbb{N} \to \mathbb{N}$$

are given as total functions with $\bar{f}$ and $\bar{g}$ being λ-definable. From them, we obtain by primitive recursion $h : \mathbb{N} \to \mathbb{N}$ where

$$h(0,m) = f(m),$$

$$h(n+1, m) = g(n,m,h(n,m))$$

for all $n, m \in \mathbb{N}$. The λ-term defining $\bar{h}$ is

$$!\, k\, \lambda x, y.\ \mathrm{cond}\ (\mathrm{zero}(x), \bar{f}(y), \bar{g}(\mathrm{pred}(x), y, k(\mathrm{pred}(x), y))).$$

Here we have had to use the fixed-point operator on a variable k of type $(N^2 \to N)$. The variables x, y are of type *N* and the cond-construction puts the two traditional equations into two clauses of one expression. It is easy to see that the fixed-point function *is* strict and is nothing more than $\bar{h}$.

That completes the representation of primitive recursive functions. To obtain the partial recursive functions, the idea is to use the so-called μ-scheme (least number operator) and, further, to close up under substitution. We need only treat the μ-scheme. Suppose, by way of example, f(n,m) is given as a

primitive recursive function. We then define h (generally, a partial function) by

$$h(m) = \text{the least } n \text{ where } f(n,m) = 0.$$

This is often written

$$h(m) = \mu n.\ f(n,m) = 0.$$

Supposing, as we may, $\bar{f}$ is λ-definable, we introduce first

$$\bar{g} = !\, g\, \lambda x, y.\ \text{cond}\ (\text{zero}\ (\bar{f}(x,y)), x, g\ (\text{succ}(x), y)).$$

Then $\bar{h} = \lambda y.\ \bar{g}(0,y)$. This is easily seen to be strict. Also easy to see is that if h(m) is defined, then $\bar{g}(0,m) = h(m)$. But, if h(m) is not defined, it takes some argument to make sure that the least fixed-point construction forces $\bar{g}(0,m) = \perp$. However, the argument is not very difficult. □

What is *not* said in 5.6 is that every λ-term defines a partial recursive function. This is true (with suitable control over the constants and types in the expression), but the proof requires a full analysis of computability properties of domain constructions. This is the topic of Lecture VII.

It should be remarked that the types of variables needed for the proof of 5.6 never get very high. In fact, types like N, N^k, and $(N^k \to N)$ were the only ones needed (with perhaps T thrown in also).

Recursion on N was the topic of 5.6; further examples of recursion on other domains are included in the exercises.

EXERCISES

EXERCISE 5.7. Find definitions of

$$\lambda x, y.\tau \quad \text{and} \quad \sigma(x,y)$$

which use only λv with one variable and applications only to one argument at a time. Note that use must be made of the combinators p_0, p_1, pair. Generalize the result to functions of many variables.

EXERCISE 5.8. (For combinator nuts.) Table 5.5 was meant to show how combinators could be defined in terms of λ-expressions. Can the tables be turned to show that with enough combinators available, every λ-expression can be defined by combining combinators, using $\sigma(\tau)$ as the only mode of combination?

EXERCISE 5.9. Suppose that $f, g : \mathcal{D} \to \mathcal{D}$ are approximable and $f \circ g = g \circ f$. Show that f and g have a least *common* fixed point $x = f(x) = g(x)$. (Hint: Refer back to Exercise 4.20.) If in addition $f(\bot) = g(\bot)$, show that $\mathrm{fix}(f) = \mathrm{fix}(g)$. In particular will $\mathrm{fix}(f) = \mathrm{fix}(f^2)$? What if we only assume $f \circ g = g^2 \circ f$?

EXERCISE 5.10. Suppose $\mathcal{D}_0$ and $\mathcal{D}_1$ are neighbourhood systems over disjoint sets Δ_0 and Δ_1. Define the *smash product* $\mathcal{D}_0 \otimes \mathcal{D}_1$ with neighbourhoods

$$\{\Delta_0 \cup \Delta_1\} \cup \{X \cup Y \mid X \in \mathcal{D}_0 \setminus \{\Delta_0\} \text{ and } Y \in \mathcal{D}_1 \setminus \{\Delta_1\}\}.$$

Show that this *is* a neighbourhood system. Define $(\mathcal{D}_0 \to_\bot \mathcal{D}_1)$ so that $|\mathcal{D}_0 \to_\bot \mathcal{D}_1|$ consists exactly of the *strict functions*. By introducing appropriate combinators, show that

$$(\mathcal{D}_0 \to_\bot (\mathcal{D}_1 \to_\bot \mathcal{D}_2)) \text{ and } ((\mathcal{D}_0 \otimes \mathcal{D}_1) \to_\bot \mathcal{D}_2)$$

are isomorphic.

EXERCISE 5.11. For any domain $\mathcal{D}$ we may regard $\mathcal{D}^\infty$ as consisting of (bottomless) *stacks* of elements of $\mathcal{D}$. With this image in mind, define appropriate combinators with the obvious meanings:

$$\mathrm{head} : \mathcal{D}^\infty \to \mathcal{D};$$

$$\mathrm{tail} : \mathcal{D}^\infty \to \mathcal{D}^\infty;$$

$$\mathrm{push} : \mathcal{D} \times \mathcal{D}^\infty \to \mathcal{D}^\infty.$$

Using the fixed-point theorem argue that there is a combinator

$$\mathrm{diag} : \mathcal{D} \to \mathcal{D}^\infty$$

where for all $x \in |\mathcal{D}|$ we have

$$\mathrm{diag}(x) = \langle x \rangle_{n=0}^{\infty}.$$

(Hint: Try a recursive definition, say

$$\text{diag}(x) = \text{push}\ (x,\ \text{diag}(x)),$$

but be sure to prove *all* terms of diag(x) equal x.) Also introduce by an appropriate recursion a combinator

$$\text{map} : (\mathcal{D} \to \mathcal{D})^{\infty} \times \mathcal{D} \to \mathcal{D}^{\infty}$$

where for elements of the suitable types:

$$\text{map}\ (\langle f_n \rangle_{n=0}^{\infty},\ x) = \langle f_n(x) \rangle_{n=0}^{\infty}.$$

EXERCISE 5.12. On any domain $\mathcal{D}$ introduce (as a least fixed point) a combinator

$$\text{while} : (\mathcal{D} \to T) \times (\mathcal{D} \to \mathcal{D}) \to (\mathcal{D} \to \mathcal{D})$$

by the recursion

$$\text{while}(p,f)(x) = \text{cond}\ (p\ (x),\ \text{while}\ (p,f)\ (f(x)),\ x).$$

Prove that

$$\text{while}\ (p,\ \text{while}\ (p,f)) = \text{while}\ (p,f).$$

Show how while could have been used to obtain the least number operator mentioned in the proof of 5.6. Generalize the idea to define a combinator

$$\text{find} : \mathcal{D}^{\infty} \times (\mathcal{D} \to T) \to \mathcal{D}$$

with the meaning "find the first term of the sequence (if any) which satisfies the given precicate."

EXERCISE 5.13. Prove the existence of a one-one function $\text{num} : \mathbb{N} \times \mathbb{N} \nrightarrow \mathbb{N}$ such that

$$\text{num}\ (0,0) = 0\ ;$$

$$\text{num}\ (n,m+1) = \text{num}(n+1,m) + 1\ ;$$

$$\text{num}\ (n+1,0) = \text{num}(0,n) + 1.$$

Draw a picture (i.e. an infinite matrix) for the function and find a closed form for its values, if possible. Use the function to prove the isomorphism of the domains

$$P\ \mathbb{N},\ P(\mathbb{N} \times \mathbb{N}),\ P\ \mathbb{N} \times P\ \mathbb{N}.$$

EXERCISE 5.14. Show that there are approximable mappings

$$\text{graph} : (P\mathbb{N} \to P\mathbb{N}) \to P\mathbb{N} \text{ and}$$

$$\text{fun} : P\mathbb{N} \to (P\mathbb{N} \to P\mathbb{N}),$$

where we have

$$\text{fun} \circ \text{graph} = \lambda f.\, f, \text{ and}$$

$$\text{graph} \circ \text{fun} \supseteq \lambda x.\, x.$$

(Hint: Using the notation

$$[n_0, n_1, \ldots, n_k] = \text{num}(n_0, [n_1, \ldots, n_k])$$

two such combinators can be given by formulae

$$\text{fun}(u)(x) = \{m \mid \exists\, n_0, \ldots, n_{k-1} \in x.\, [n_0+1, \ldots, n_{k-1}+1, 0, m] \in u\}$$

$$\text{graph}(f) = \{[n_0+1, \ldots, n_{k-1}+1, 0, m] \mid m \in f(\{n_0, \ldots, n_{k-1}\})\},$$

where k is variable - meaning all finite sequences are to be considered.)

EXERCISE 5.15. (For algebraists.) We can regard $\langle \{0,1\}^*, \Lambda, \cdot \rangle$ as the free semigroup on two generators 0 and 1. The powerset $P\{0,1\}^*$ is taken as a domain as in Exercise 4.17. For "words" $e \in \{0,1\}^*$ define

$$e^* = \{\Lambda, e, e^2, e^3, \ldots, e^n, \ldots\}.$$

Show that the least fixed point of

$$z = \{e\} \cdot z \cup \{e'\}$$

in $P\{0,1\}^*$ is $z = e^* \cdot \{e'\}$. Show further (as suggested by David Park) that the least solution of

$$x = a \cdot x \cup b \cdot y \cup c$$

$$y = b \cdot x \cup a \cdot y \cup d$$

has

$$x = (a \cup b \cdot a^* \cdot b)^* \cdot (c \cup b \cdot a^* \cdot d),$$

where the $\{\cdot\}$ has been dropped off $\{a\}$, $\{b\}$ etc., and where the *-notation has been extended to the whole domain, so that

$$z^* = \Lambda \cup z^* \cdot z.$$

(Hint: Apply 5.3.)

EXERCISE 5.16. Return to the discussion of Example 4.4 and the construction of the domain of finite and infinite binary sequences. Give a fixed-point definition of neg: $C \to C$, where

$$\text{neg}\ (0x) = 1\ \text{neg}\ (x);$$
$$\text{neg}\ (1x) = 0\ \text{neg}\ (x).$$

Prove that neg (neg (x)) = x for all $x \in |C|$. Also define merge : $C \times C \to C$, where for $\varepsilon, \delta \in \{0,1\}$ we have:

$$\text{merge}\ (\varepsilon x, \delta y) = \varepsilon\ \delta\ \text{merge}\ (x,y).$$

(Note: There may be a little trouble with merge (x,y) when x is finite and total and y is infinite - you have to decide what you want in e.g. merge (Λ,y).) Prove that

$$\text{merge}\ (x,x) = d\ (x),$$

in the notation of 4.4. Consider also the infinite non-periodic sequence

$$t = 0\ \text{merge}\ (\text{neg}(t), \text{tail}(t)).$$

Prove that the n^{th} digit of t is the sum mod 2 of the digits of the number n written in the binary scale (a suggestion of J. Lambek). Show also that $t \neq u\,a\,a\,a\,v$ where a is any finite sequence $\neq \Lambda$, and where u is finite.

LECTURE VI

INTRODUCTION TO DOMAIN EQUATIONS

The major reason for introducing the theory of domains is to have a notion of *computability* incorporating both finite and infinite elements. In our many examples already explored we have seen how functions (functionals, operators, combinators) can be defined on domains; owing to the property of approximability (continuity) of these functions, we have also seen how they can be "calculated" by finite approximation. In this lecture further examples of domains will be constructed -- especially domains having infinite elements, which can be introduced in a variety of ways giving rise to interesting structural possibilities. The next lecture then treats a precise notion of computability appropriate to these domains; while the last lecture opens up new methods of domain construction.

EXAMPLE 6.1. Let $\mathcal{D}$ be fixed as a given domain. We are now familiar with a useful construct like $\mathcal{D} \times \mathcal{D}$ whose elements are ordered pairs $\langle x,y \rangle$ of elements x, y of $\mathcal{D}$. The question is: can this construct be iterated? The answer is obviously yes, since $\mathcal{D} \times (\mathcal{D} \times \mathcal{D})$ and $(\mathcal{D} \times \mathcal{D}) \times (\mathcal{D} \times \mathcal{D})$ and so on can be formed with elements $\langle x, \langle y,z \rangle \rangle$ and $\langle\langle u,v \rangle, \langle x,y \rangle\rangle$ and the like. But the real question is: can the construct be iterated *indefinitely?* AND can the results be collected together into a *single domain*? The answer is yes, but it requires a bit of work to get it right. The method to be introduced will be open to many variations, so more than one answer is possible, giving non-isomorphic domains.

In order to collect all the iterates into one large domain we give ourselves first a very big domain inside of which the desired family of neighbourhoods will be found. There are many ways to make this choice, and we are fixing on one that will keep the notation simple. We have often used binary sequences for examples and constructions, but for this example let us use

ternary sequences. Let $\Sigma = \{0,1,2\}$ and let Σ^* be all finite sequences from this three-letter alphabet. We will select subsets of Σ^* for our neighbourhoods. As Σ^* is countably infinite, it is without much loss of generality to assume that $\mathcal{D}$ is a neighbourhood system over Δ where we take $\Delta \subseteq \Sigma^*$. Also without loss of generality we can assume $\emptyset \notin \mathcal{D}$. (Why?) We wish to find another set $\Gamma \subseteq \Sigma^*$ to be the set of tokens for the new domain. After we find it, we will still have to say just which $X \subseteq \Gamma$ are appropriate for the structure we want.

The totality $\{X \mid X \subseteq \Sigma^*\}$ is, as a powerset, isomorphic to the set of elements of a domain: a point we have remarked several times. So, by the Fixed-Point Theorem we know there is a set $\Gamma \subseteq \Sigma^*$ where

$$\Gamma = 0\,\Delta \;\cup\; 1\,\Gamma \cup 2\,\Gamma\,.$$

In fact $\Gamma = \{1,2\}^*\,0\,\Delta$, because we can say:

$$\{1,2\}^* = \{\Lambda\} \;\cup\; 1\{1,2\}^* \cup 2\{1,2\}^*\,.$$

The domain we are looking for will be found as a domain $\mathcal{D}^{\S}$ over Γ. The reason for splitting Γ up, as shown in the equation above, is to ensure that if $X,Y \in \mathcal{D}^{\S}$ are two neighbourhoods in the system $\mathcal{D}^{\S}$, then $1\,X \cup 2\,Y$ has a chance of being also in $\mathcal{D}^{\S}$ because

$$1\,X \;\cup\; 2Y \subseteq \Gamma.$$

This will make $\mathcal{D}^{\S} \times \mathcal{D}^{\S}$ isomorphic to a part of $\mathcal{D}^{\S}$. If we make $\mathcal{D}$ also isomorphic to a part of $\mathcal{D}^{\S}$, then all the iterated products will be contained in $\mathcal{D}^{\S}$.

What is a neighbourhood system? Just a set of sets. But $\mathcal{P}\,\mathcal{P}\,\Sigma^*$ is a domain (as a powerset) and because $\Gamma \subseteq \Sigma^*$, we find

$$\mathcal{D}^{\S} \in \mathcal{P}\,\mathcal{P}\,\Sigma^*$$

as an element. But elements of domains can often be defined by fixed-point equations. Indeed we will introduce $\mathcal{D}^{\S}$ this way:

$$\mathcal{D}^{\S} = \{\Gamma\} \cup \{0X \mid X \in \mathcal{D}\} \cup \{1X \cup 2Y \mid X,Y \in \mathcal{D}^{\S}\}.$$

The reader should stop to think why $\mathcal{D}^{\S}$ can be immediately seen to exist by writing such an equation. Of course another way to describe $\mathcal{D}^{\S}$ is to say it is the least family of sets containing (i) the set Γ, (ii) the sets $0X$ for X in the given system $\mathcal{D}$, and (iii) sets $1\,X \cup 2\,Y$ whenever it already contains X and Y (closure

under a set-forming operation). By saying "least", we mean (iv) nothing else belongs to $\mathcal{D}^{\S}$ except as allowed by (i)-(iii); this makes the truth of the *equation* for $\mathcal{D}^{\S}$ clear. So $\mathcal{D}^{\S}$ exists as a family of sets, but what good is it?

By our construction of Γ, all the sets we put into $\mathcal{D}^{\S}$ are subsets of Γ (why?), so $\mathcal{D}^{\S}$ has a chance of being a system over Γ if we can check the closure under intersection. So suppose $Z \subseteq X \cap Y$ where $Z,X,Y \in \mathcal{D}^{\S}$; we want to show $X \cap Y \in \mathcal{D}^{\S}$. We argue by induction on the number of steps required to put X and Y into $\mathcal{D}^{\S}$ by (i)-(iii). There are several cases.

If $X = \Gamma$ or $Y = \Gamma$, there is nothing to prove, because both sets are subsets of Γ. We note that $\emptyset \notin \mathcal{D}^{\S}$, because (i)-(iii) cannot introduce $\emptyset$ as a member of $\mathcal{D}^{\S}$. So, if $X = 0A$ for $A \in \mathcal{D}$, then Y must have this form also (if it is not Γ), because

$$0A \cap (1B \cup 2C) = \emptyset .$$

(That is, if Y had the form (iii), then $Z = \emptyset$ would be a consequence, which is impossible.) Thus, if $X = 0A$ for $A \in \mathcal{D}$, then $Y = 0B$ for some $B \in \mathcal{D}$. But by the same reasoning $Z = 0C$ for some $C \in \mathcal{D}$ also. But the relationship $0C \subseteq 0A \cap 0B$ is equivalent to $C \subseteq A \cap B$. We see, therefore, that $A \cap B \in \mathcal{D}$, and so

$$X \cap Y = 0A \cap 0B = 0(A \cap B)$$

must belong to $\mathcal{D}^{\S}$.

The final case has X,Y,Z all of the form (iii):

$$X = 1A_1 \cup 2A_2 ,$$

$$Y = 1B_1 \cup 2B_2 , \text{ and}$$

$$Z = 1C_1 \cup 2C_2 .$$

We can think of the A_i and B_i put into $\mathcal{D}^{\S}$ *earlier* and the intersection result as being already established for them. But the relationship $Z \subseteq X \cap Y$ is equivalent to $C_i \subseteq A_i \cap B_i$ for $i = 1,2$. Therefore $A_i \cap B_i \in \mathcal{D}^{\S}$, and so does

$$X \cap Y = (1A_1 \cup 2A_2) \cap (1B_1 \cup 2B_2) = 1(A_1 \cap B_1) \cup 2(A_2 \cap B_2) .$$

We have now seen that $\mathcal{D}^{\S}$ is a neighbourhood system, but why was it constructed that way? The reason is simply this isomorphism (or *domain equation*):

$$\mathcal{D}^{\S} \cong \mathcal{D} + (\mathcal{D}^{\S} \times \mathcal{D}^{\S}) \ ,$$

as can be seen by reference to the equation for $\mathcal{D}^{\S}$ and the definitions of + and ×. What are the elements of $\mathcal{D}^{\S}$? There is always

$$\bot = \{\Gamma\}.$$

Next if $x \in |\mathcal{D}|$ we define

$$x^{\S} = \{\Gamma\} \cup \{0\, X \mid X \in x\}.$$

That gives an isomorphic injection

$$\lambda x.x^{\S} : \mathcal{D} \to \mathcal{D}^{\S}.$$

Then for $x, y \in |\mathcal{D}^{\S}|$ we can define

$$\langle x,y \rangle = \{\Gamma\} \cup \{1X \cup 2\,Y \mid X \in x \text{ and } Y \in y\}.$$

We have another isomorphic injection

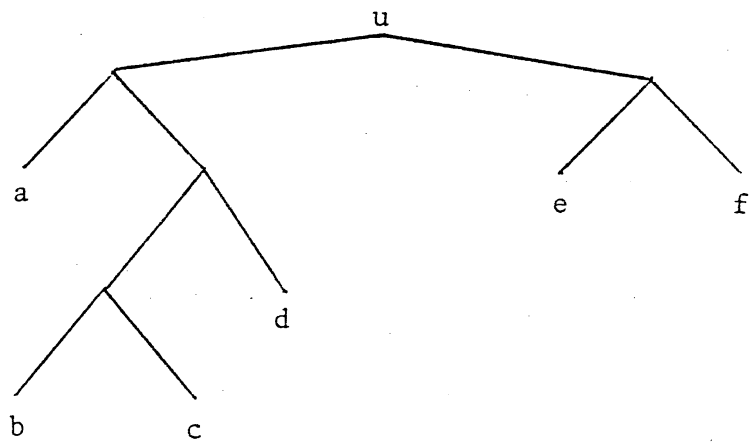
$$\lambda x,y.\langle x,y\rangle : \mathcal{D}^{\S} \times \mathcal{D}^{\S} \to \mathcal{D}^{\S}.$$

Indeed by looking at the neighbourhood definition of $\mathcal{D}^{\S}$ we conclude that the *finite* elements of $\mathcal{D}^{\S}$ are exactly those that are either of the form (i) $\bot$, or (ii) $a^{\S}$, where a is finite in $|\mathcal{D}|$ or (iii) $\langle a,b\rangle$, where a and b are previously obtained finite elements of $|\mathcal{D}^{\S}|$.

Suppose a,..., f are finite in $|\mathcal{D}|$. We can picture the element

$$u = \langle\langle a^{\S}, \langle\langle b^{\S}, c^{\S}\rangle, d^{\S}\rangle\rangle, \langle e^{\S}, f^{\S}\rangle\rangle$$

in $|\mathcal{D}^{\S}|$ as a tree:

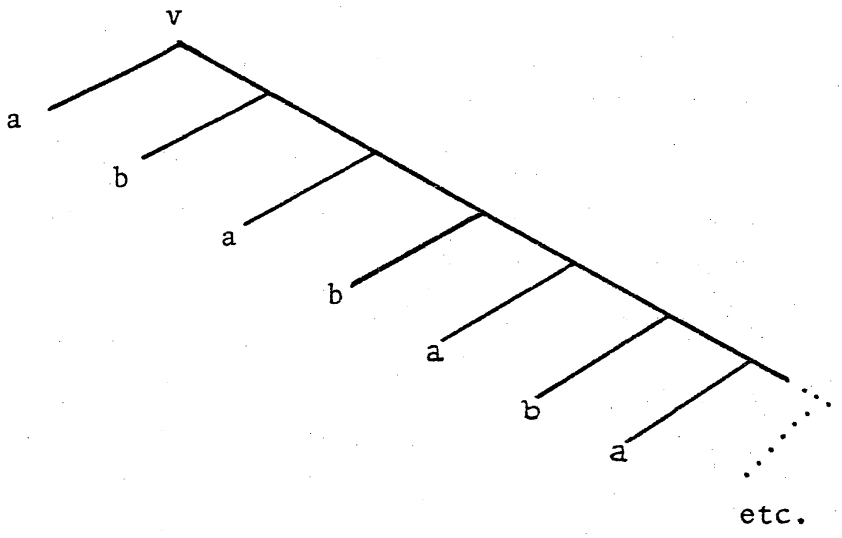


Note that the tree has binary branching with the elements of $|\mathcal{D}|$ at the ends of the branches. Any such tree could be given a notation as an element of $|\mathcal{D}^{\S}|$. The finite elements of $|\mathcal{D}^{\S}|$ correspond exactly to such finite trees.

What of the infinite elements of $|\mathcal{D}^{\S}|$? Are there infinite trees? Let $a, b \in |\mathcal{D}^{\S}|$ be any elements of $|\mathcal{D}^{\S}|$. Since pairing is an approximable mapping, we can solve the fixed-point equation

$$v = \langle a, \langle b, v \rangle \rangle .$$

In pictures we can diagram v roughly as:



The word is "roughly" here, since if a or b were not in the $|\mathcal{D}|$ part of $|\mathcal{D}^{\S}|$, then in the diagram the letters "a" and "b" should be replaced by the corresponding tree diagrams for a and b.

Suppose that a and b are finite. Then we can easily see that the infinite tree v is the limit of the following sequence of finite trees:

$$v_0 = \bot,$$

$$v_{n+1} = \langle a, \langle b, v_n \rangle \rangle \text{ , and}$$

$$v = \bigcup_{n=0}^{\infty} v_n$$

The reader should think how to explain from tree diagrams the approximation relation $v_n \subseteq v$ and more general such relationships.

We could call $\mathcal{D}^\S$ a *tree algebra* over $\mathcal{D}$. There may be others. A general one is a structure of the form

$$<E,\ \mathsf{in}\ ,\ \mathsf{pair}>\ ,$$

where

$$\mathsf{in} : \mathcal{D} \to E\ ,\ \text{and}$$

$$\mathsf{pair} : E \times E \to E\ .$$

The algebra

$$<\mathcal{D}^\S,\ \lambda x.x^\S,\ \lambda x,y.\ <x,y>>\ ,$$

however, is a very special one: it is "minimal" among all tree algebras over $\mathcal{D}$ in a sense we shall have to make precise.

To do this think of how E and $\mathcal{D}^\S$ can differ. In view of the isomorphism that $\mathcal{D}^\S$ satisfies the injection of $\mathcal{D}$ and the pairing are one-one, so no "information" is lost by these mappings. The same may not at all be true of E, but it is reasonable to think that at least we can define an approximable mapping $g : \mathcal{D}^\S \to E$ where

(1) $\quad g\ (\bot) = \bot_E\ ,$

(2) $\quad g\ (x^\S) = \mathsf{in}(x),\ \text{for } x \in |\mathcal{D}|,\ \text{and}$

(3) $\quad g\ (<x,y>) = \mathsf{pair}(g(x),\ g(y)),\ \text{for } x,y \in |\mathcal{D}^\S|.$

By what we said earlier, g will be uniquely determined by (1)-(3), because these equations tell us exactly how to calculate g on all finite elements of $|\mathcal{D}^\S|$. An approximable mapping is always determined by its action on the finite elements. But why does g exist?

It would not be too hard to give an inductive construction of g as a neighbourhood relation, but a fixed-point equation is easier to write down for the same purpose. We need, though, to have the inverse ("predecessor") functions:

$$\text{out} : \mathcal{D}^{\S} \to \mathcal{D}$$

$$\text{proj}_i : \mathcal{D}^{\S} \to \mathcal{D}^{\S}, \text{ for } i = 0,1,$$

where

$$\text{out}(x^{\S}) = x ,$$

$$\text{proj}_0(\langle x,y\rangle) = x, \text{ and}$$

$$\text{proj}_1(\langle x,y\rangle) = y.$$

We also need

$$\text{atom} : \mathcal{D}^{\S} \to T,$$

where

$$\text{atom}(x^{\S}) = \text{true, and}$$

$$\text{atom}(\langle x,y\rangle) = \text{false} .$$

We can then write

$$g(x) = \text{cond}\,(\,\text{atom}(x)\,,\,\text{in}\,(\,\text{out}(x))\,,\,\text{pair}(g(\text{proj}_0(x)),\, g(\,\text{proj}_1(x))))$$

This g exists by fixed-point theory, and it satisfies (1)-(3) by what we know about the structure of $|\mathcal{D}^{\S}|$. As we said, g is unique because the values on finite elements are fixed.

In algebraic language g is a *homomorphism* of tree algebras; and $\mathcal{D}^{\S}$ is called an *initial algebra*, because for any tree algebra E there is a unique homomorphism $g : \mathcal{D}^{\S} \to E$. We note at once that any two initial algebras are isomorphic. For if $\mathcal{D}^{\star}$ were another, there would exist homomorphisms in both directions between $\mathcal{D}^{\S}$ and $\mathcal{D}^{\star}$. But the compositions of homomorphisms are again homomorphisms, and in the case of $\mathcal{D}^{\S}$ if we go from $\mathcal{D}^{\S}$ $\mathcal{D}^{\star}$ and back to $\mathcal{D}^{\S}$, the result must be the identity. The reason is that the identity can be the only homomorphism of an initial algebra into itself. We thus have a precise meaning of the minimal character of $\mathcal{D}^{\S}$. But note it still took a construction to show that the domain $\mathcal{D}^{\S}$ *exists*. □

EXAMPLE 6.2. Our staple examples $\mathcal{B}$ and $\mathcal{C}$ satisfy "domain equations" in the form of isomorphisms as we have previously seen. Indeed

$$\mathcal{B} \cong \mathcal{B} + \mathcal{B}, \text{ and}$$

$$\mathcal{C} \cong \{\{\Lambda\}\} + \mathcal{C} + \mathcal{C} ,$$

where if we liked we could construct both systems over $\{0,1\}^*$ and have :

$$\mathcal{B} = \{\{0,1\}^*\} \cup \{0\,X \mid X \in \mathcal{B}\} \cup \{1\,X \mid X \in \mathcal{B}\}, \quad \text{and}$$

$$\mathcal{C} = \{\{0,1\}^*\} \cup \{\{\Lambda\}\} \cup \{0\,X \mid X \in \mathcal{C}\} \cup \{1\,X \mid X \in \mathcal{C}\}.$$

We leave to the exercises the explanations of what kinds of algebras $\mathcal{B}$ and $\mathcal{C}$ are and why they are initial. Here we want to propose a simple, yet interesting generalization of $\mathcal{B}$.

Consider this domain equation :

$$A \cong A^n + A^n ,$$

where A^n atands for the n-fold cartesian power of A. We can, with the aid of some encoding solve this equation as a neighbourhood system over $\{0,1\}^*$ as follows:

$$A = \{\{0,1\}^*\} \cup \bigcup_{i=0,1} \{\, i \bigcup_{j<n} 1^j 0 X_j \mid X_j \in A \text{ all } j<n\} .$$

For instance, if n = 3, then a typical neighbourhood in A is something like

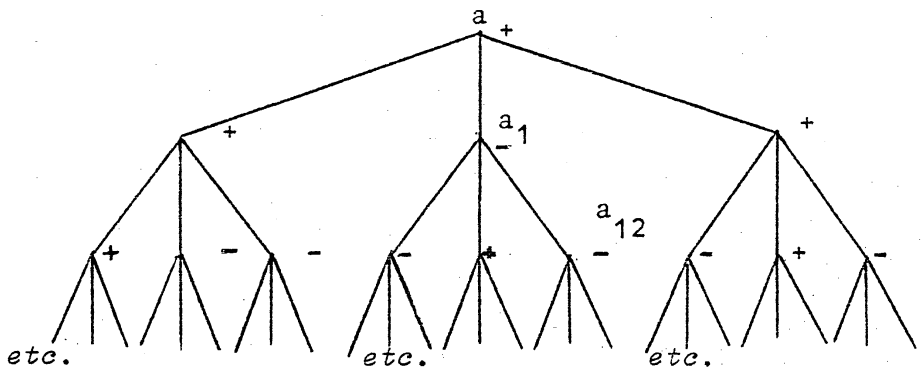
$$00X_0 \cup 010X_1 \cup 0110X_2 ,$$

where $X_0, X_1, X_2 \in A$. The first '0' could also be a '1' in front of each of the terms.

In words, an element of A (other than $\perp$) is an n-tuple of elements of A: but there are two separate copies of these, the left one and the right one. We can write for $a \in |A|$

$$a = \pm\langle a_0, a_1, \ldots, a_{n-1}\rangle,$$

where + is chosen if a is on the right, and − if on the left. As a tree diagram a might look like this for n = 3 :



That is, a is an infinite ternary tree with + or − labels at each node. If each node (subtree) is truly infinite, the element is *total*; if ⊥ is ever encountered, it is only *partial*; if every branch ends with ⊥, the tree is a *finite* element of |A|.

What can be done with such trees? Let $\sigma \in \{0,1,\ldots,n-1\}^*$ be a finite sequence of "digits" each less than n. We let $\Sigma = \{0,1,\ldots,n-1\}$. We can define for $a \in |A|$ the operation $\sigma \mapsto a\sigma$ by recursion on σ:

$$a\,\Lambda = a \text{ , and}$$

$$a\,i\,\sigma = (a_i)\,\sigma \text{ .}$$

The $a\sigma$ are just the *subtrees* of a with σ as a *selector*. We also have a map

$$pos : A \to T$$

where

$$pos(+\langle a_0, a_1, \ldots, a_{n-1}\rangle) \quad \text{true, and}$$

$$pos(-\langle a_0, a_1, \ldots, a_{n-1}\rangle) = \text{false.}$$

We say that a (total) tree a is *eventually periodic* iff the set $\{a\sigma \mid \sigma \in \Sigma^*\}$ is finite. The result is that the "language"

$$L_a = \{\sigma \in \Sigma^* \mid pos\,(a\sigma) = \text{true}\}$$

corresponding to an eventually periodic tree is always a *regular event* of automata theory, and every such language has this form. In fact, a just represents the initial state of an automaton, and $a\sigma$ represents the state after "reading" a tape σ. □

In order to formulate more generally the idea of a domain equation and initial algebra, we must introduce a small amount of the terminology of category theory. To be as specific as possible, think of systems $\mathcal{D}$ over sets $\Delta \subseteq \Sigma^*$ with $\Sigma = \{0,1\}$, say. They form quite an interesting category with respect to the approximable maps $f : \mathcal{D} \to \mathcal{D}'$. Recall that to be a category of "domains" and "maps" all that is required is an associative composition $g \circ f$ of maps with identity maps $I : \mathcal{D} \to \mathcal{D}$ for each domain of the category. And this we certainly have for the systems indicated. And there are many other categories waiting around: for instance, restrict systems to those where $\emptyset \notin \mathcal{D}$. This is not much of a restriction, as every system is isomorphic to one like this. Or restrict the maps to being the strict maps $f : \mathcal{D} \to \mathcal{D}'$ where $f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}'}$. This is an essentially different, though related category. We shall find many others.

What examples 6.1 and 6.2 suggest is the notion of a construct which makes new domains out of old. For example, with $\mathcal{D}$ fixed, 6.1 suggests for any domain X over $\Gamma \subseteq \Sigma^*$ a domain

$$T(X) = \mathcal{D} + (X \times X).$$

More specifically (converting from $\Sigma = \{0,1,2\}$ to $\Sigma = \{0,1\}$) we could write

$$T(X) = \{\Gamma'\} \cup \{0X \mid X \in \mathcal{D}\} \cup \{10X \cup 11Y \mid X,Y \in X\},$$

where we have $\Gamma' = 0\Delta \cup 10\Gamma \cup 11\Gamma$. (By the way, here we definitely want to assume $\emptyset \notin \mathcal{D}$ and $\emptyset \notin X$ and to get $\emptyset \notin T(X)$.) This construct is an example of a *functor*, a notion that can be defined abstractly on any category.

DEFINITION 6.3. A *functor* on a category (into itself) associates with every domain X in the category another domain $T(X)$ *and* to every map

$$f : X \to Y$$

another map

$$T(f) : T(X) \to T(Y)$$

in such a way that identity maps and compositions are preserved:

$$T(I_X) = I_{T(X)}, \text{ and}$$
$$T(g \circ f) = T(g) \circ T(f),$$

whenever $f : X \to Y$ and $g : Y \to Z$. □

In the example from 6.1 we have not checked how the special T is a functor. The hint is that whenever $f : X \to Y$, then there is a map

$$f \times f : X \times X \to Y \times Y$$

But there is also a map

$$I_{\mathcal{D}} + f \times f : \mathcal{D} + (X \times X) \to \mathcal{D} + (Y \times Y)$$

and this suggests the definition of T(f). The details are left to the exercises. Note that the map T(f) just suggested is always strict, so T is a functor also for the category of strict maps.

One good reason for a little of the category-theoretic language is that the next definition becomes very neat indeed.

DEFINITION 6.4. A T-*algebra* is a domain E in the category together with a map

$$k : T(E) \to E.$$

If $m : T(F) \to F$ is another T-algebra, then a *homomorphism* is a map $h : E \to F$ in the category such that the diagram

$$\begin{array}{ccc} T(E) & \xrightarrow{k} & E \\ {\scriptstyle T(h)}\downarrow & & \downarrow{\scriptstyle h} \\ T(F) & \xrightarrow{m} & F \end{array}$$

commutes; that is, the equation

$$h \circ k = m \circ T(h)$$

holds. □

In our example from 6.1 a T-algebra is a *strict* map

$$k : \mathcal{D} + (E \times E) \to E \ .$$

But such strict maps are in a one-one correspondence with pairs of (not necessarily strict) maps

$$n : \mathcal{D} \to E \quad \text{and} \quad p : E \times E \to E \ .$$

And the structure $\langle E,n,p\rangle$ is what we called a tree algebra. Definition 6.4 just makes this abstract. The reader should also work out the details showing that 6.4's definition of homomorphism is just what we ought to expect.

Note that the T-algebras and homomorphisms form a category. (Why?) The following definition is so abstract that it could be given for *any* category.

DEFINITION 6.5. A T-algebra is *initial* if and only if there is a unique homomorphism from it into any other T-algebra. □

The word "other" here is not meant to imply "distinct". For an initial algebra there is one and only one homomorphism into itself: the identity map. As we already indicated in 6.1 it is a general fact that the next proposition holds.

PROPOSITION 6.6. Any two initial T-algebras are uniquely isomorphic. □

Slightly more interesting is the behaviour of T on initial algebras.

PROPOSITION 6.7. If $i : T(\mathcal{D}) \to \mathcal{D}$ is an initial T-algebra, then so is $T(i) : T^2(\mathcal{D}) \to T(\mathcal{D})$ and i is the isomorphism from $T(\mathcal{D})$ to $\mathcal{D}$.

Proof: Clearly since T is a functor, the map T(i) has the right mapping character to make $T(\mathcal{D})$ a T-algebra. Since $\mathcal{D}$ is initial, we have a commuting diagram :

$$\begin{array}{ccc} T(\mathcal{D}) & \xrightarrow{\ i\ } & \mathcal{D} \\ {\scriptstyle T(j)}\downarrow & & \downarrow{\scriptstyle j} \\ T^2(\mathcal{D}) & \xrightarrow{\ T(i)\ } & T(\mathcal{D}) \end{array}$$

But we also have the trivial diagram:

$$\begin{array}{ccc} T^2(\mathcal{D}) & \xrightarrow{\ T(i)\ } & T(\mathcal{D}) \\ {\scriptstyle T(i)}\downarrow & & \downarrow{\scriptstyle i} \\ T(\mathcal{D}) & \xrightarrow{\ i\ } & \mathcal{D} \end{array}$$

It follows that $i \circ j$ is a homomorphism, so

$$i \circ j = I_{\mathcal{D}}.$$

But then because T is a functor we find:

$$T(i) \circ T(j) = I_{T(\mathcal{D})},$$

and, since j is a homomorphism, we have

$$j \circ i = I_{T(\mathcal{D})}.$$

This shows that i is an isomorphism. □

From 6.7 we see that if we are going to have initial algebras at all we have to satisfy the domain equation

$$\mathcal{D} \cong T(\mathcal{D}).$$

But generally that is not enough to assure that $\mathcal{D}$ is initial. There is a condition that our functors satisfy, however, which guarantees the existence of homomorphisms.

DEFINITION 6.8. On the category of domains and strict approximable maps a functor T is *continuous on maps* if for any systems $\mathcal{D}$ and $\mathcal{E}$ the induced mapping

$$\lambda f.\ T(f) : (\mathcal{D} \to_{\perp} \mathcal{E}) \to (T(\mathcal{D}) \to_{\perp} T(\mathcal{E}))$$

is approximable.

THEOREM 6.9. If the functor T is continuous on maps and if $\mathcal{D} \cong T(\mathcal{D})$, so in particular $\mathcal{D}$ is a T-algebra, then for any T-algebra $k : T(E) \to E$ there is a homomorphism $h : \mathcal{D} \to E$.

Proof: Let $i : T(\mathcal{D}) \to \mathcal{D}$ make $\mathcal{D}$ a T-algebra, where $j : \mathcal{D} \to T(\mathcal{D})$ is the inverse so that i is an isomorphism of domains. Suppose that $k : T(E) \to E$ is any T-algebra. A homomorphism $h : \mathcal{D} \to E$ would satisfy

$$h \circ i = k \circ T(h).$$

Rewrite this equation as

$$h = k \circ T(h) \circ j.$$

In the domain of strict maps $(\mathcal{D} \to E)$ this is a fixed-point equation for an approximable map

$$\lambda h.\ k \circ T(h) \circ j$$

by our assumption on T. Thus, the desired homomorphism exists. □

The final question we have to answer is why in our category the minimal $\mathcal{D}$ exist. The reason is that the functors T that we have in mind possess further continuity properties on domains. This is conveniently expressed in terms of a notion of "subdomain".

DEFINITION 6.10. For two neighbourhood systems $\mathcal{D}$ and E we write

$$\mathcal{D} \lhd E$$

to mean that these are neighbourhood systems over the same set of tokens Δ and not only is $\mathcal{D} \subseteq E$ but whenever $X, Y \in \mathcal{D}$ and $X \cap Y \in E$, then $X \cap Y \in \mathcal{D}$. □

For the subdomain relation $\mathcal{D} \lhd E$ to hold, $\mathcal{D}$ has to be a smaller family of neighbourhoods, but the notion of consistency in $\mathcal{D}$ also has to be the same as in E. Note that if $\mathcal{D}_0 \lhd E$ and $\mathcal{D}_1 \lhd E$ then

$$\mathcal{D}_0 \lhd \mathcal{D}_1 \text{ iff } \mathcal{D}_0 \subseteq \mathcal{D}_1 .$$

It is also easy to prove that the union of a directed family of subdomains of E is again a subdomain. As a consequence of this remark we have:

PROPOSITION 6.11. For a given neighbourhood system E, the set of subsystems

$$\{\mathcal{D} \mid \mathcal{D} \lhd E\}$$

forms a domain in its own right. □

The subdomain relationship implies a mapping relationship between the domains.

PROPOSITION 6.12. If $\mathcal{D} \lhd E$, then there exists a projection pair of approximable mappings:

$$i : \mathcal{D} \to E \quad \text{and} \quad j : E \to \mathcal{D}$$

where $j \circ i = I_{\mathcal{D}}$ and $i \circ j \subseteq I_E$, which are determined as element-wise functions by these equations:

$$i(x) = \{Y \in E \mid \exists X \in x \,.\, X \subseteq Y\}, \quad \text{and}$$

$$j(y) = y \cap \mathcal{D},$$

for all $x \in |\mathcal{D}|$ and $y \in |E|$. □

The proof is left for the exercises.

DEFINITION 6.13. A functor T is *monotone on domains* iff whenever $\mathcal{D} \lhd E$, then not only do we have $T(\mathcal{D}) \lhd T(E)$ but the projection pair i, j of 6.12 is mapped to the same kind of projection pair T(i), T(j). A monotone functor is *continuous on domains* iff whenever E is a domain, then the mapping

$$\lambda\mathcal{D}.\, T(\mathcal{D}) : \{\mathcal{D} \mid \mathcal{D} \lhd E\} \to \{\mathcal{D}' \mid \mathcal{D}' \lhd T(E)\}$$

is approximable. □

We can now state an existence theorem that covers in fairly wide generality the examples of this lecture.

THEOREM 6.14. If the functor T is continuous on maps and monotone and continuous on domains, and if there is a set Γ such that

$$\{\Gamma\} \lhd T(\{\Gamma\}),$$

then there exists an initial T-algebra.

Proof: We proceed as in the proof of the fixed-point theorem by iterating the functor. The assumption about Γ means that, as a neighbourhood system, $T(\{\Gamma\})$ is a system over the *same* set Γ. Thus, if we iterate T to form $T^n(\{\Gamma\})$, all these systems are over Γ and indeed

$$T^n(\{\Gamma\}) \lhd T^{n+1}(\{\Gamma\})$$

for all n. We can thus introduce

$$\mathcal{D} = \bigcup_{n=0}^{\infty} T^n(\{\Gamma\}),$$

and it is easy to check that $\mathcal{D}$ is a system over Γ and

$$T^n(\{\Gamma\}) \lhd \mathcal{D}$$

holds for all n. But then we have for all n:

$$T^n(\{\Gamma\}) \lhd T^{n+1}(\{\Gamma\}) \lhd T(\mathcal{D}),$$

which implies $\mathcal{D} \lhd T(\mathcal{D})$. But T is continuous on domains, so

$$T(\mathcal{D}) = T\Big(\bigcup_{n=0}^{\infty} T^n(\{\Gamma\})\Big)$$

$$= \bigcup_{n=0}^{\infty} T^{n+1}(\{\Gamma\})$$

$$= \mathcal{D}\ .$$

Thus, not only is $\mathcal{D}$ a T-algebra, but the isomorphism we get for $\mathcal{D}$ and $T(\mathcal{D})$ is just the identity mapping. We know by 6.9 that homomorphisms exist; what remains to show is that homomorphism from $\mathcal{D}$ are unique. As in the examples, we will show in effect they are determined uniquely on the finite elements of $\mathcal{D}$.

Since each $T^n(\{\Gamma\}) \lhd \mathcal{D}$, there are projection mappings

$$i_n : T^n(\{\Gamma\}) \to \mathcal{D} \text{ and } j_n : \mathcal{D} \to T^n(\{\Gamma\}).$$

Define $\rho_n : \mathcal{D} \to \mathcal{D}$ by $\rho_n = i_n \circ j_n$. Projection pairs are always pairs of strict mappings (Why?), and so are in the category. By assumption and 6.13, the functor **T** preserves these maps, so we have

$$T(\rho_n) = T(i_n) \circ T(j_n) = i_{n+1} \circ j_{n+1} = \rho_{n+1} .$$

As a neighbourhood relation ρ_n can be characterized by :

$$X \rho_n Y \text{ iff } \exists Z \in T^n(\{\Gamma\}).\ X \subseteq Z \subseteq Y.$$

We thus see that $\rho_n \subseteq \rho_{n+1}$ and

$$\bigcup_{n=0}^{\infty} \rho_n = I_{\mathcal{D}}.$$

Now suppose $k : T(E) \to E$ is any T-algebra and $h : \mathcal{D} \to E$ is a homomorphism. The mapping will satisfy the fixed-point equation

$$h = k \circ T(h),$$

where no other mappingsneed be written in because $\mathcal{D} = T(\mathcal{D})$ and so

$$T(h) : \mathcal{D} \to T(E) .$$

We wish to show that h really is the least fixed point of this equation.

Define $h_n = h \circ \rho_n : \mathcal{D} \to E$. For $n = 0$, the map ρ_0 is the trivial map where $\rho_0(x) = \perp_{\mathcal{D}}$ for all $x \in |\mathcal{D}|$. But h must be strict, so $h_0(x) = \perp_E$ for all $x \in |\mathcal{D}|$; that is, h_0 is the least element of $|\mathcal{D} \to_{\perp} E|$. Now calculate :

$$k \circ T(h_n) = k \circ T(h) \circ T(\rho_n)$$

$$= h \circ \rho_{n+1}$$

$$= h_{n+1} .$$

This shows that the union of the h_n is the least fixed point of $\lambda h.k \circ T(h)$. But

$$\bigcup_{n=0}^{\infty} h_n = \bigcup_{n=0}^{\infty} h \circ \rho_n$$

$$= h \circ \bigcup_{n=0}^{\infty} \rho_n$$

$$= h \circ I_{\mathcal{D}} = h,$$

so the given h is in fact the least fixed point. The homomorphism is uniquely determined, and $\mathcal{D}$ is the initial T-algebra. □

Having the existence of initial T-algebras, we can prove one more result that shows just how minimal they are. We need a lemma about projection pairs, first, that shows where subdomains fit it. We write $\mathcal{D} \lesssim E$ as short for $\mathcal{D} \cong \mathcal{D}'$ for some $\mathcal{D}' \lhd E$ in the following. The lemma gives a converse to 6.12.

LEMMA 6.15. For two neighbourhood systems $\mathcal{D}$ and E, if there exist a projection pair

$$i : \mathcal{D} \to E \text{ and } j : E \to \mathcal{D}$$

with $j \circ i = I_{\mathcal{D}}$ and $i \circ j \subseteq I_E$, then $\mathcal{D} \lesssim E$.

Proof. What we want to show is that i maps finite elements to finite elements, and that the desired $\mathcal{D}'$ is the image of $\mathcal{D}$ in E.

Suppose $X \in \mathcal{D}$. We can write:

$$i(\uparrow X) = \bigcup \{\uparrow Y \mid Y \in i\ (\uparrow X)\}.$$

Applying j to both sides we have:

$$\uparrow X = j \circ i\,(\uparrow X) = \bigcup \{ j(\uparrow Y) \mid Y \in i(\uparrow X) \}.$$

But then, since $X \in \uparrow X$, we find $X \in j\,(\uparrow Y)$ for some $Y \in i\,(\uparrow X)$. This implies

$$\uparrow X \subseteq j\,(\uparrow Y)\ ;\ \text{and so}\ i(\uparrow X) \subseteq i \circ j\,(\uparrow Y) \subseteq \uparrow Y.$$

Since $\uparrow Y \subseteq i\,(\uparrow X)$ in any case, we conclude $i\,(\uparrow X) = \uparrow Y$. This proves finite elements *are* mapped to finite elements.

What of Δ; that is, what is $i\,(\uparrow \Delta)$? We find, supposing E to be a neighbourhood system over a set Δ', that since $\uparrow \Delta \subseteq j\,(\uparrow \Delta')$, then $i\,(\uparrow \Delta) \subseteq \uparrow \Delta'$ and so $i\,(\uparrow \Delta) = \uparrow \Delta'$. This means that Δ corresponds to Δ'. So we have established that $\mathcal{D}$ is in an inclusion preserving one-one correspondence with a subset $\mathcal{D}'$ of E where $\Delta' \in \mathcal{D}'$. But it remains to show that $\mathcal{D}'$ is a neighbourhood system and that $\mathcal{D}' \lhd E$ holds. All we really have to show is that $\mathcal{D}'$ is closed under intersection whenever the intersection belongs to E.

Suppose $Y', Z' \in \mathcal{D}'$ and $Y' \cap Z' \in E$. Let $X' = Y' \cap Z'$. We have, for suitable $Y, Z \in \mathcal{D}$,

$$i(\uparrow Y) = \uparrow Y',\ \text{and so}\ \uparrow Y = j(\uparrow Y');\ \text{and}$$

$$i(\uparrow Z) = \uparrow Z',\ \text{and so}\ \uparrow Z = j(\uparrow Z').$$

But $\uparrow Y' \subseteq \uparrow X'$ and $j(\uparrow Y') \subseteq j(\uparrow X')$; thus $Y \in j(\uparrow X')$. For similar reasons $Z \in j\,(\uparrow X')$. But then $X = Y \cap Z \in j(\uparrow X')$, and therefore $Y \cap Z \in \mathcal{D}$. (The element $j(\uparrow X')$ must be a filter.) Notice, however, that

$$\uparrow Y \subseteq \uparrow X,\ \text{and so}\ \uparrow Y' \subseteq i\,(\uparrow X)\ ;\ \text{and}$$
$$\uparrow Z \subseteq \uparrow X,\ \text{and so}\ \uparrow Z' \subseteq i\,(\uparrow X).$$

It follows that $Y' \cap Z' = X' \in i(\uparrow X)$. On the other hand we already knew $X \in j\,(\uparrow X')$, which implies $i(\uparrow X) \subseteq \uparrow X'$. We may thus conclude that $i(\uparrow X) = \uparrow X'$. In other words $X' \in \mathcal{D}'$. □

THEOREM 6.16. If on the category of domains and strict approximable maps the functor T is continuous on maps, and if $\mathcal{D}$ is an initial T-algebra, then for any system $E \cong T(E)$ we have $\mathcal{D} \trianglelefteq E$.

Proof: There is a homomorphism $h : \mathcal{D} \to E$. By 6.9 there is a homomorphism $g : E \to \mathcal{D}$. Now $g \circ h : \mathcal{D} \to \mathcal{D}$ is also a homomorphism, so $g \circ h = I_{\mathcal{D}}$ because $\mathcal{D}$ is initial. In view of 6.15, all we have to prove now is that $h \circ g \sqsubseteq I_E$.

Let the maps $i : T(\mathcal{D}) \to \mathcal{D}$ and $j : \mathcal{D} \to T(\mathcal{D})$ give the isomorphism for $\mathcal{D}$, and let $u : T(E) \to E$ and $v : E \to T(E)$ do the same for E. By the proof of 6.9 we know

$$g = i \circ T(g) \circ v \quad \text{and} \quad h = u \circ T(h) \circ j$$

and each of these maps is the least fixed point of its respective equation. Let

$$g_0 = \bot_{E \to \mathcal{D}} \quad \text{and} \quad h_0 = \bot_{\mathcal{D} \to E}$$

and define by recursion

$$g_{n+1} = i \circ T(g_n) \circ v \quad \text{and} \quad h_{n+1} = u \circ T(h_n) \circ j.$$

By the fixed-point calculation

$$g = \bigsqcup_{n=0}^{\infty} g_n \quad \text{and} \quad h = \bigsqcup_{n=0}^{\infty} h_n.$$

Now we see that

$$h_0 \circ g_0 = \bot_{E \to E},$$

and for each n that

$$\begin{aligned} h_{n+1} \circ g_{n+1} &= u \circ T(h_n) \circ j \circ i \circ T(g_n) \circ v \\ &= u \circ T(h_n) \circ T(g_n) \circ v \\ &= u \circ T(h_n \circ g_n) \circ v. \end{aligned}$$

But this means that

$$h \circ g = \bigsqcup_{n=0}^{\infty} (h_n \circ g_n)$$

is the least fixed point for the equation

$$k = u \circ T(k) \circ v.$$

But I_E is *one* of the fixed points; whence $h \circ g \sqsubseteq I_E$ must follow. □

EXERCISES

EXERCISE 6.17. What are the algebras for which C is initial? If A of 6.2 is a generalization of B, what is the corresponding generalization of C? Prove that it exists and explain what are the algebras involved.

EXERCISE 6.18. With reference back to Exercise 3.16 discuss the construction of $\mathcal{D}^\infty$ as an initial algebra and as a solution to the domain equation

$$\mathcal{D}^\infty \cong \mathcal{D} \times \mathcal{D}^\infty \ .$$

(I do not know whether all solutions must be of the form $\mathcal{D}^\infty \times E$.)

EXERCISE 6.19. For the sake of uniformity restrict attention to systems $\mathcal{D}$ on sets $\Delta \subseteq \{0,1\}^*$, where $\Lambda \in \Delta$ and $\emptyset \notin \mathcal{D}$, and to the category of strict maps. Define sum and product by:

$$\mathcal{D}_0 + \mathcal{D}_1 = \{\{\Lambda\} \cup 0\,\Delta_0 \cup 0\,\Delta_1\} \cup \{0\,X \mid X \in \mathcal{D}_0\} \cup \{1\,Y \mid Y \in \mathcal{D}_1\},$$

$$\mathcal{D}_0 \times \mathcal{D}_1 = \{\{\Lambda\} \cup 0\,X \cup 1\,Y \mid X \in \mathcal{D}_0 \text{ and } Y \in \mathcal{D}_1\}.$$

Are these correct up to isomorphism? Now generate all constructs $T(X)$ formed by the constants (that is, $T(X) = \mathcal{D}$ for a fixed $\mathcal{D}$), by the identity ($T(X) = X$), and by sums and products ($T_0(X) + T_1(X)$, etc.) Show that these are all functors, continuous on maps, and monotone and continuous on domains.

EXERCISE 6.20. For any system $\mathcal{D}$ let $\mathrm{tok}(\mathcal{D})$ be the underlying set of tokens, so that $\mathcal{D}$ is a system over $\mathrm{tok}(\mathcal{D})$. For the category of Exercise 6.19 show that the function

$$\lambda\Gamma.\ \mathrm{tok}(T(\{\Gamma\}))$$

is continuous on the domain $\{\Gamma \subseteq \{0,1\}^* \mid \Lambda \in \Gamma\}$, where T is any of the functors generated in 6.19. Conclude that there must exist a set

$$\Gamma = \mathrm{tok}(T(\{\Gamma\})),$$

so that $\{\Gamma\} \lhd T(\{\Gamma\})$, and so 6.14 applies.

EXERCISE 6.21. Do the same as 6.19 and 6.20 when the functors are also allowed to be generated by the operations:

$$\mathcal{D}_0 \oplus \mathcal{D}_1 = \{\{\Lambda\} \cup 0\Delta_0 \cup 1\Delta_1\} \cup \{0X \mid X \in \mathcal{D}_0 \setminus \{\Delta_0\}\} \cup \{1Y \mid Y \in \mathcal{D}_1 \setminus \{\Delta_1\}\} ,$$

$$\mathcal{D}_0 \otimes \mathcal{D}_1 = \{\{\Lambda\} \cup 0\Delta_0 \cup 1\Delta_1\} \cup \{\{\Lambda\} \cup 0X \cup 1Y \mid X \in \mathcal{D}_0 \setminus \{\Delta_0\} \text{ and } Y \in \mathcal{D}_1 \setminus \{\Delta_1\}\}.$$

Generalize all of $+$, $\times$, $\oplus$, $\otimes$ to combinations of several terms, not just the binary sums and products.

EXERCISE 6.22. Comment on these domain equations:

$$N \cong \{\{0\}, \{0, \Lambda\}\} \oplus N,$$

$$M \cong \{\{\Lambda\}\} + M,$$

$$N^* \cong N \oplus (N \otimes N^*).$$

EXERCISE 6.23. Construe the initial solution to

$$Exp \cong N \oplus ((Exp \times Exp) + (Exp \times Exp))$$

as a "syntactical domain" of *expressions* generated from infinitely many "variables" by means of two binary "operation symbols". Given an algebra $\mathcal{D}$ with two operations

$$u : \mathcal{D} \times \mathcal{D} \to \mathcal{D} \text{ and } v : \mathcal{D} \times \mathcal{D} \to \mathcal{D} ,$$

show how any strict map $s : N \to \mathcal{D}$ determines a unique map

$$val(s) : Exp \to \mathcal{D}$$

that can be regarded as the "evaluation of an expression".

EXERCISE 6.24. Show that there must exist domains satisfying:

$$\mathcal{D} \cong \mathcal{D} + (\mathcal{D} \times E), \text{ and}$$

$$E \cong \mathcal{D} + E,$$

by using a double fixed-point method. First decide what the underlying set of tokens should be, and then define $\mathcal{D}$ and E by simultaneous fixed points. (Syntactical domains as in 6.23 may very well require several simultaneous equations.)

EXERCISE 6.25. For a projection pair $g : \mathcal{D} \to E$ and $h : E \to \mathcal{D}$ show that for $x \in |\mathcal{D}|$ and $y \in |E|$ we have:

$$g(x) \sqsubseteq y \text{ iff } x \sqsubseteq h(y).$$

Thus, conclude that:

$$h(y) = \bigsqcup \{x \in |\mathcal{D}| \mid g(x) \sqsubseteq y\}, \text{ and}$$

$$g(x) = \bigsqcap \{y \in |E| \mid x \sqsubseteq h(y)\},$$

for all $x \in |\mathcal{D}|$ and $y \in |E|$. So each of the functions determines the other. In the first equation check that the set on the right is directed, and in the second equation that the set on the right is non empty. Prove also that g maps consistent sets to consistent sets and preserves $\sqcup$ (not just directed unions).

EXERCISE 6.26. For systems $\mathcal{D}$ as in 6.19 define

$$\mathcal{D}_\perp = \{\{\Lambda\} \cup 0\,\Delta\} \cup \{0X \mid X \in \mathcal{D}\}$$

Describe the construct in terms of elements. Is this a suitable functor? Prove that

$$\mathcal{D}_\perp \oplus E_\perp \cong \mathcal{D} + E \ .$$

What is

$$\mathcal{D}_\perp \otimes E_\perp \cong \ ??$$

EXERCISE 6.27. Which of the following relationships are true:

$$(\mathcal{D} \otimes E) \lhd (\mathcal{D} \times E) \ ; \ \mathcal{D} \lhd \mathcal{D} \times E \ ;$$

$$(\mathcal{D} \oplus E) \lhd (\mathcal{D} + E) \ ; \ \mathcal{D} \lhd \mathcal{D} \oplus E \ ;$$

$$(\mathcal{D} \to_\perp E) \lhd (\mathcal{D} \to E); \ \mathcal{D} \lhd \mathcal{D} \otimes E \ ?$$

EXERCISE 6.28. (Suggested by G. Plotkin). Show that if and are *finite* systems and

$$\mathcal{D} \lhd E \lhd \mathcal{D} \ ,$$

then $\mathcal{D} \cong E$. Need the same be true of infinite systems?

EXERCISE 6.29. Generalize + and × to infinitary operations on domains:

$$\sum_{n=0}^{\infty} \mathcal{D}_n \quad \text{and} \quad \prod_{n=0}^{\infty} \mathcal{D}_n \ .$$

Would a similar generalization be possible for $\oplus$ and $\otimes$?

LECTURE VII

COMPUTABILITY IN EFFECTIVELY GIVEN DOMAINS

For the domain N the strict functions from N into N, the strict maps $f : N \to N$, correspond exactly to the partial functions $g : \mathbb{N} \to \mathbb{N}$ (as we wrote in 5.6 we had $f = \overline{g}$). For such functions there is a standard theory of computability: g is called computable if it can be defined as a partial recursive function with its "program" written down in a certain standard form. The non-strict maps $h : N \to N$ are all constant, and so are intuitively computable; so we know all about computable maps in $|N \to N|$ in general. The question is: what are the computable maps on (elements of) other domains?

The answer will of course depend on how the domain is presented to us. Even with N, there are continuum many isomorphisms $\pi : N \to N$ of N onto itself, not all of which can be computable. That is, if we permute N and, so to speak, present the integers in a different order, then a well-behaved computable function $f : N \to N$ may well be transformed into a non-computable function,

$$\pi \circ f \circ \pi^{-1} : N \to N.$$

(Hint: Consider the characteristic function e of the even numbers. Take $f = \overline{e}$ and let π be very horrid.) The reason we imagined we knew which were the computable $f : N \to N$ is that N is always thought of in a standard presentation. We must thus define "in general" a concept of an *effectively given domain*, that is to say, one with a sufficiently computable presentation to represent the additional knowledge about the domain.

The main idea will be that the *finite* elements of $|\mathcal{D}|$ should be regarded as the ones initially known. Abstractly, to know a finite element is to know how it is *related to* other finite elements.

Of course, this will mean that we will allow at most a countable infinity of finite elements - but this restriction well accords with intuition. To make precise the terminology "related to" it proves most convenient to go back to the neighbourhoods (in any case they are in a one-one correspondence with the finite elements).

DEFINITION 7.1. A neighbourhood system $\mathcal{D}$ has a *computable presentation* provided we can write

$$\mathcal{D} = \{ X_n \mid n \in \mathbb{N} \},$$

where the following two relations

(i) $X_n \cap X_m = X_k$; and

(ii) $\exists k \in \mathbb{N} . X_k \subseteq X_n$ and $X_k \subseteq X_m$

are recursively decidable (in integer indices n, m, k and in n, m, respectively). □

More strictly the sequence,

$$\langle X_n \rangle_{n=0}^{\infty},$$

is the presentation. Even more strictly, when it is required to cope with infinitely many domains at a time, it would be necessary to give the actual Gödel numbers of the recursive relations (i) and (ii) (rather than just saying there exists some way of showing them to be recursively decidable).

The intuitive idea of 7.1 is that the system is effectively given if you know how to do elementary "calculations" with neighbourhoods. The basic calculations are the forming of intersections. The neighbourhoods have to be laid out in a systematic way; and, if we are asked for an intersection of two given neighbourhoods, we have to be able to locate it in the standard sequence. Relation (ii) is the *consistency condition*, which is the necessary and sufficient condition for the intersection to exist in $\mathcal{D}$. When (ii) is true, therefore, we have only to try k = 0,1,2, ... until we discover that we have found the intersection. We are

assuming that these basic decisions can be carried out in "finite time". Note that the obvious biconditional,

$$X_n \subseteq X_m \text{ iff } X_n \cap X_m = X_n,$$

assures us that the inclusion relation between neighbourhoods is itself decidable in terms of the indices. So in (ii) *if* k exists, *then* it (or the first one) can indeed be found in finite time. The rub is that if it *does not exist*, no finite number of inclusion checks will determine that fact. That is why we have to *assume* that (ii) is always decidable. The information contained in (ii) is a fundamental part of the neighbourhood structure. (An axiomatic characterization of neighbourhood structures is given in Exercise 7.12, which may make clearer what we are assuming and what a presentation is.)

DEFINITION 7.2. Given two recursively presented domains,

$$\mathcal{D} = \{X_n \mid n \in \mathbb{N}\} \text{ and } E = \{Y_m \mid m \in \mathbb{N}\},$$

an approximable mapping $f : \mathcal{D} \to E$ is said to be *computable* iff the relation

$$X_n \, f \, Y_m$$

is recursively enumerable in n and m. □

The question to ask first is why "recursively enumerable" rather than "recursive" (= "recursively decidable")? The answer will become clear when we let $\mathcal{D}$ degenerate to the one-element domain, $\mathcal{D} = \{\Delta\}$. Then what we are considering is merely a single element

$$y = f(\{\Delta\}) \in |E|.$$

Therefore, 7.2 incorporates the notion of a *computable element* of a domain. And the condition reduces to the statement that the filter $y \in |E|$ is such that the set

$$\{m \in \mathbb{N} \mid Y_m \in y\}$$

is a recursively enumerable set of integers. The point is that the elements of $|E|$ are finite or infinite. If y were finite, the set of indices above would indeed be recursive in view of

our assumptions on E. But an infinite element can in general only be approximated "a little at a time". We cannot expect to know the whole story of its approximations in a flash. What it means to be recursively enumerable is that there is a primitive recursive function (hence, a *total* function), $r : \mathbb{N} \to \mathbb{N}$, such that

$$y = \{Y_{r(i)} \mid i \in \mathbb{N}\}.$$

That is to say, *all* the approximations to y can *eventually* be listed. In the case of the mapping f we could write

$$f = \{(X_{s(i)}, Y_{r(i)}) \mid i \in \mathbb{N}\},$$

for a suitable pair of primitive recursive functions s and r.

Definitions 7.1 and 7.2 may very well irritate the person hearing them for the first time: instead of explaining computability in direct terms, the whole question is thrown into the lap of recursion theory! There are several answers. "You have to start somewhere" is one thing I always say. Recursion on the integers is a well-understood theory, and we shall not need the refined parts of the development, fortunately. In any case, our definitions apply to *many* domains of quite different structure, not just to the domain N. And the next step we shall take is to show how to build up computable functions (and also effectively given domains) from simpler ones. Thus, often it will not be necessary to go back to the seemingly over-precise definitions involving the indices but to appeal to some broad general principles.

PROPOSITION 7.3. The identity map on an effectively given domain is computable; the composition of computable mappings on effectively given domains is again computable. □

The proofs for 7.3 are so trivial they are hardly worth an exercise. Note the immediate and useful consequence: if $f : \mathcal{D} \to E$ is computable and $x \in |\mathcal{D}|$ is computable, then $f(x) \in |E|$ is also computable. The next result is, however, worth working out even though it is quite easy.

THEOREM 7.4. If $\mathcal{D}_0$ and $\mathcal{D}_1$ are effectively given, then so are

$$(\mathcal{D}_0 + \mathcal{D}_1) \text{ and } (\mathcal{D}_0 \times \mathcal{D}_1).$$

Moreover the combinators in_i and out_i and $proj_i^2$ are all computable; further if f and g are computable maps, then so are $f + g$ and $f \times g$.

Proof: Let the computable presentations be given as:

$$\mathcal{D}_i = \{X_n^i \mid n \in \mathbb{N}\}.$$

We can assume that the sets of tokens Δ_0 and Δ_1 are disjoint and $\emptyset \notin \mathcal{D}_i$. Then the construction of the sum is just

$$\mathcal{D}_0 + \mathcal{D}_1 = \{\Delta_0 \cup \Delta_1\} \cup \mathcal{D}_0 \cup \mathcal{D}_1.$$

As an enumeration we define for $n \in \mathbb{N}$:

$$Z_0 = \Delta_0 \cup \Delta_1 \; ; \; Z_{2n+1} = X_n^0 \; ; \; Z_{2n+2} = X_n^1 \; .$$

We leave as an exercise the check of 7.1(i)-(ii).

For the product we want:

$$\mathcal{D}_0 \times \mathcal{D}_1 = \{X_n^0 \cup X_m^1 \mid n, m \in \mathbb{N}\}$$

What we then need are recursive functions $p : \mathbb{N} \to \mathbb{N}$, $q : \mathbb{N} \to \mathbb{N}$, and $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ where for $m, n, k \in \mathbb{N}$ we have:

$$p(r(n, m)) = n \text{ and } q(r(n, m)) = m, \text{ and } r(p(k), q(k)) = k.$$

Thus r is a "one-one pairing function"; there are many ways to find such functions (see Exercise 5.13). We can then define for $k \in \mathbb{N}$:

$$W_k = X^0_{p(k)} \cup X^1_{q(k)} \; .$$

Again we leave as an exercise the check that this provides a computable presentation of $\mathcal{D}_0 \times \mathcal{D}_1$.

As for the combinators, the neighbourhood relations have to be worked out in terms of the indices. For example

$$X_n^0 \; in_0 \; Z_m \text{ iff either } m = 0 \text{ or for some } k$$
$$m = 2k + 1 \text{ and } X_n^0 \subseteq X_k^0 \; .$$

$$W_k \; proj_1^2 \; X_m^1 \text{ iff } X^1_{q(k)} \subseteq X_m^1 \; .$$

The reader needs to check that these are recursively enumerable

relations in the indices. For this purpose it may be convenient to recall some closure properties of these relations: taking conjunctions, disjunctions, substituting recursive functions, applying an existential quantifier to the front. □

Products give us a way of providing an immediate meaning to the notion of a computable function of several variables. Note that the proof of 3.7 is "effective" and shows that substitution of computable functions of several variables into each other always gives computable functions. We turn next to the function spaces.

THEOREM 7.5. If $\mathcal{D}_0$ and $\mathcal{D}_1$ are effectively given, then so is $(\mathcal{D}_0 \to \mathcal{D}_1)$. The combinators eval and curry are computable, provided all the domains involved are effectively given. The computable elements $f \in |\mathcal{D}_0 \to \mathcal{D}_1|$ are exactly the computable maps $f : \mathcal{D}_0 \to \mathcal{D}_1$.

Proof : The proofs of 3.9, 3.11, and 3.12 were set up with this theorem in mind. If

$$\mathcal{D}_0 = \{X_n | n \in \mathbb{N}\} \text{ and } \mathcal{D}_1 = \{Y_m | m \in \mathbb{N}\}$$

are two effectively given neighbourhood systems, then the neighbourhoods of $(\mathcal{D}_0 \to \mathcal{D}_1)$, by Definition 3.8, are non-empty intersections like

$$\bigcap_{i<q} [X_{n_i}, Y_{m_i}],$$

where $\langle n_0, n_1, \ldots, n_{q-1} \rangle$ and $\langle m_0, m_1, \ldots, m_{q-1} \rangle$ are two finite sequences of integers determining the choice of the function-space neighbourhood. In 3.9(i) the test for nonemptiness is given. Assuming the decidability of relations in $\mathcal{D}_0$ and $\mathcal{D}_1$, one remarks that the consistency of *finite sequences* of neighbourhoods is also decidable. (Hint: Test the first two, then form their intersection. Next test the third given neighbourhood against this one set; if consistent, form the intersection, and carry on.) By 3.9(i) at most 2.2^q such sequential checks must be carried out to determine whether the function-space neighbourhood is non empty.

It may not be fun, but the checks can be carried out in finite time. Owing to this decidability, we can therefore enumerate in a systematic way *all* the pairs of finite sequences $\langle n_0, \ldots \rangle$ and $\langle m_0, \ldots \rangle$ that determine neighbourhoods: that is the way that $(\mathcal{D}_0 \to \mathcal{D}_1)$ obtains its enumeration.

Concerning the decidability of the required relations on $(\mathcal{D}_0 \to \mathcal{D}_1)$, we remark first off that consistency is more of the same: to test two finite intersections against each other, just form one big intersection and test it for non-emptiness as before. Secondly, the testing for intersection comes down in the end to testing one typical intersection of $[X, Y]$ - neighbourhoods for equality with another. But equality amounts to two inclusions; inclusion in an intersection amounts to inclusion in each term. Therefore, what we need to do is to check a finite number of statements of the form:

$$\bigcap_{i<q} [X_{n_i}, Y_{m_i}] \subseteq [X_k, Y_\ell].$$

As we pointed out after the proof of 3.9, this inclusion is equivalent to

$$\bigcap \{Y_{m_i} \mid X_k \subseteq X_{n_i}\} \subseteq Y_\ell.$$

By decidability in $\mathcal{D}_0$, we can effectively find the n_i that are needed. Then in $\mathcal{D}_1$ we form the intersection of the corresponding Y_{m_i}. Finally, we check the inclusion. Again, one check in $(\mathcal{D}_0 \to \mathcal{D}_1)$ requires a whole sequence of checks in $\mathcal{D}_0$ and in $\mathcal{D}_1$, but the process is finite. So we have argued that $(\mathcal{D}_0 \to \mathcal{D}_1)$ is effectively given.

In showing that the combinators are computable, we refer first to the proof of 3.11. The typical pair of neighbourhoods possibly belonging to eval is

$$\bigcap_{i<q} [X_{n_i}, Y_{m_i}] \cup X_k \text{ eval } Y_\ell.$$

As we needed not to be so specific, we expressed the holding of this relationship in terms of *all* the functions in the function-

space neighbourhood. But we know that the neighbourhood, by 3.9(ii), has a minimal element; it is then sufficient to test for the holding of $X_k \, f_0 \, Y_\ell$ at this minimal function f_0. But this test, we have already seen, is decidable. So the pairs in eval actually form a recursive set, not just a recursively enumerable set; thus, eval is a computable function.

The case of curry involves three domains and is a bit more messy. But again, if the required neighbourhoods are written out in full, it will be seen that curry, too, is computable. We leave this minor struggle to the exercises.

The final statement is an easy consequence of the fundamental connection between approximable $f : \mathcal{D}_0 \to \mathcal{D}_1$ as relations and as elements . Recall, as in the proof of 3.10, that we have

$$f \in [X, Y] \quad \text{iff} \quad X \, f \, Y,$$

for all $X \in \mathcal{D}_0$ and $Y \in \mathcal{D}_1$. Therefore,

$$f \in \bigcap_{i<q} [X_{n_i}, Y_{m_i}] \quad \text{iff} \quad \forall i < q . \, X_{n_i} \, f \, Y_{m_i} .$$

It follows that if f is recursively enumerable as a set of pairs, then, by forming all the non-empty intersections (as shown), we get an enumeration of all the neighbourhoods to which f belongs; and this is the same as the filter corresponding to f as an element of the function space. The converse direction is clear. □

We have nearly all our favourite combinators computable, but perhaps the most important one - since it is the key to recursive definitions - is the fixed-point combinator. It is not left out.

THEOREM 7.6. For any effectively given domain D, the combinator $\mathrm{fix} : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ is computable.

Proof : Referring back to the proof of Theorem 4.2 and thinking of

$$\mathcal{D} = \{ X_n \mid n \in \mathbb{N} \}$$

as effectively given, fix as a relation comes down to

$\bigcap_{i<q}[X_{n_i},X_{m_i}]$ fix X_ℓ iff for some finite sequence $\Delta = X_{k_0}, \ldots, X_{k_p} = X_\ell$ we have, for each $j<p$,

$$\bigcap \{X_{m_i} \mid X_{k_j} \subseteq X_{n_i}\} \subseteq X_{k_{j+1}}.$$

Inside the "for some finite sequence" all the checks are decidable by assumption on $\mathcal{D}$. But the existential quantification of a decidable predicate always gives a recursively enumerable predicate. (And, as there is no implied bound on the size of the finite sequence we are looking for, this really *is* an enumerable set and not generally a recursive set.) □

The major consequence of what we have done up to this point concerns typed λ-calculus. *Any* expression involving only *effectively given types* and, perhaps, some basic *computable constants* using only the λ, ! -notation defines a computable function of its free variables. And such functions applied to computable arguments give computable values. And such functions have computable least fixed points. Etc., etc. In a definite sense then we have in the "metalanguage", as people say, a quite precise and fully *mathematical programming language* for defining computable operators. It is not a machine implemented language, but it is a mathematically well-defined and easy-to-use language. And when we combine the usual type-definition facility together with *domain equations*, we have an especially powerful language.

PROPOSITION 7.7. For any effectively given domain $\mathcal{D}$, the domain $\mathcal{D}^{\S}$ is also effectively given, and all the combinators of Example 6.1 prove to be computable.

Proof: This proof is essentially an exercise, but it is useful to have an easy-to-grasp example. Indeed, to make things easy to reason about, we can assume that $\mathcal{D}$ is a system over $\Delta = \mathbb{N}$, and that in the presentation where

$$\mathcal{D} = \{X_n \mid n \in \mathbb{N}\},$$

the relation $k \in X_n$ is *recursive* in k and n. (It is worth thinking why this is so.) Of course, a lot of other things are recursive also.

Now what kind of a system is $\mathcal{D}^{\S}$? The construction of 6.1 made it a system over a certain set of strings Γ. For the sake of checking various assertions about computability, we are transposing everything back to $\mathbb{N}$. (These are all denumerable sets in any case.) The set Γ is divided into three equally big parts, and we can do the same for $\mathbb{N}$. Let us write for any $m, k \in \mathbb{N}$ and subset $X \subseteq \mathbb{N}$: $mX+k = \{m.n + k \mid n \in X\}$.
Then by splitting the integers modulo 3 we have:

$$\mathbb{N} = 3\mathbb{N} \cup (3\mathbb{N}+1) \cup (3\mathbb{N}+2),$$

and this equation is quite analogous to that for Γ. We then propose this definition for $\mathcal{D}^{\S}$

$$\mathcal{D}^{\S} = \{\mathbb{N}\} \cup \{3X \mid X \in \mathcal{D}\} \cup \{(3X+1) \cup (3Y+2) \mid X, Y \in \mathcal{D}^{\S}\},$$

but this does not make the enumeration of $\mathcal{D}^{\S}$ all that obvious. This is one way to do it:

$$V_0 = \mathbb{N} \; ; \; V_{2n+1} = 3X_n \; ; \; V_{2n+2} = (3V_{p(n)}+1) \cup (3V_{q(n)}+2).$$

Here p and q are the inverse of the pairing functions mentioned in 7.4 They must be chosen so that $'(n) \leqslant n$ and $q(n) \leqslant n$ for all $n \in \mathbb{N}$. Thus, in calculating V_k where $k = 2n+2$ we will be using $V_{p(n)}$ and $V_{q(n)}$ where *both* subscripts are strictly less than k. This observation is required so that $m \in V_k$ is going to be a recursive relation. What we claim is that

$$\mathcal{D}^{\S} = \{V_k \mid k \in \mathbb{N}\}.$$

It should be clear that everything on the right belongs to $\mathcal{D}^{\S}$. What needs an inductive argument is that everything in $\mathcal{D}^{\S}$ is eventually of the form V_k. But this should be fairly obvious owing to the properties of $r : \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$.

The reader also has to check that 7.=(i)-(ii) hold for the V_k. The idea is that any such check is either (1) trivial, or (2) something already assumed about $\mathcal{D}$ and the X_n, or (3) can be thrown back to some sets V_m with strictly smaller subscripts. Therefore, the checks will give an answer in finite time according to an effective reduction.

Next for the combinators, we have to translate neighbourhood relations into relations among integer indices. A selection of examples must suffice.

$$X_n(\lambda x.x^{\S}) \; V_k \text{ iff } V_{2n+1} \subseteq V_k$$

$V_m \text{ proj}_0 V_k$ iff $k = 0$ or $\exists n \in \mathbb{N}.\ m = 2n+2$ and $V_{p(n)} \subseteq V_k$.

The reader should write out other cases. □

EXAMPLE 7.8. We have often made reference to the powerset $P\mathbb{N}$ as a domain and we should check here that it is effectively given. One easy way to see this is to note

$$P\,\mathbb{N} \cong |T^{\infty}|.$$

The (slight) trouble with $P\,\mathbb{N}$ is that we usually think of it in terms of *elements* rather than *neighbourhoods*. Going back to Exercise 1.16, we can argue that the neighbourhoods of $P\mathbb{N}$ are ordered not like the finite sets of integers but in the partial ordering *converse* to that. But this is of no trouble, since all will be decidable. What we need first is an enumeration of all finite sets of integers. We can do this by:

$$E_n = \{k \mid \exists i, j\ .\ i < 2^k \text{ and } n = i + 2^k + j.2^{k+1}\ \}.$$

The idea is that $k \in E_n$ means that the exponent k does occur in the binary expansion of n as a sum of powers of 2. All finite subsets of $\mathbb{N}$ are of the form E_n. We then find that as a neighbourhood system

$$(P\mathbb{N}) = \{\mathbb{N} \setminus E_n \mid n \in \mathbb{N}\}.$$

As the relationship $E_n \cup E_m = E_k$ is recursive, there is no trouble in proving that this is a computable presentation. In this system, of course, any two neighbourhoods are consistent. Various combinators on $P\mathbb{N}$ are suggested in Exercise 7.23. □

This construct is known as the *Smyth Power Domain*. It is defined for any neighbourhood system $\mathcal{D}$ and results in a new system we shall call here $\mathbb{P}\,\mathcal{D}$. The elements of $\mathbb{P}\,\mathcal{D}$ behave rather like *sets of elements* of $\mathcal{D}$, but since our elements can be either partial or total, there are certain dangers to pushing the analogy too far. For some purposes a rival construct called the *Plotkin Power Domain* is better, but it leads outside the category of neighbourhood systems as defined in these lectures. Do not confuse $P\mathbb{N}$ with $\mathbb{P}\,\mathcal{D}$.

DEFINITION 7.9. Let $\mathcal{D}$ be any neighbourhood system and define

$$\mathbb{P}\mathcal{D} = \{\bigcup_{i<n} (\downarrow X_i) \mid \forall\, i<n.\ X_i \in \mathcal{D}\}.$$

We recall that for any $X \in \mathcal{D}$

$$\downarrow X = \{Y \in \mathcal{D} \mid Y \subseteq X\}.$$

The finite unions in $\mathbb{P}\mathcal{D}$ can be empty (i.e. if $n = 0$). □

Formally, the system $\mathbb{P}\mathcal{D}$ is just more or less the closure of $\mathcal{D}$ under finite unions; however, this would not be an isomorphism-invariant construct unless $\mathcal{D}$ is "prepared". The preparation consists of replacing $\mathcal{D}$ by the isomorphic domain

$$\mathcal{D}^{\downarrow} = \{\downarrow X \mid X \in \mathcal{D}\}.$$

(In this connection refer back to Exercise 1.20.) We remark that

$$\downarrow X \cap \downarrow Y \neq \emptyset \text{ iff } \{X,Y\} \text{ is consistent in } \mathcal{D},$$

and in that case

$$\downarrow X \cap \downarrow Y = \downarrow (X \cap Y).$$

PROPOSITION 7.10. The power domain $\mathbb{P}\mathcal{D}$ is a neighbourhood system if $\mathcal{D}$ is, and it is effectively given if $\mathcal{D}$ is.

Proof : The system $\mathcal{D}^{\downarrow}$ is a neighbourhood system as we just remarked; indeed it is a positive neighbourhood system. It is easy to prove that the closure of any positive system under finite unions is a neighbourhood system, because the resulting family of sets is closed under *all* finite intersections. (If we left out the empty union, the result would be a positive system.) The proof is obvious since intersection of sets distributes over finite union. So $\mathbb{P}\mathcal{D}$ is a neighbourhood system.

For the second half of the proposition, we just have to constructivize the previous argument. Thus, if

$$\mathcal{D} = \{X_n \mid n \in N\},$$

then the elements of $\mathbb{P}\mathcal{D}$ can be written as:

$$\bigcup_{i<q} (\downarrow X_{n_i}),$$

and hence are indexed by the finite sequences $\langle n_0, \ldots, n_{q-1} \rangle$ of integers. Now one of the standard devices of recursion theory is to put the finite sequences of integers into a recursive one-one correspondence with the integers themselves. This is the start of the recursive presentation of $\mathbb{P}\mathcal{D}$, since it means we can list effectively all the required neighbourhoods.

Next consider an intersection

$$\bigcup_{i<q} (\dagger X_{n_i}) \cap \bigcup_{j<r} (\dagger X_{m_j}) = \bigcup_{\substack{i<q \\ j<r}} \dagger (X_{n_i} \cap X_{m_j}) \ .$$

Some of the terms which are $\emptyset$ have to be thrown out - but this requires only a finite number of decisions all computable by assumption. Now we have to rewrite

$$X_{n_i} \cap X_{m_j} = X_{k_{ij}} \ ,$$

but the finding of k_{ij} is also computable. *Finally*, we have to re-order the doubly indexed sequence into a singly indexed sequence of length q.r, but this is easily seen to be computable also. Therefore, intersections can be "calculated".

It remains to be shown that equality between neighbourhoods in $\mathbb{P}\mathcal{D}$ is decidable. The question really comes down to deciding something like:

$$\dagger X_k \subseteq \bigcup_{i<q} \dagger X_{n_i} \ .$$

Now since $X_k \in \dagger X_k$, we find that the above is just equivalent to:

$$\exists\, i < q.\ X_k \subseteq X_{n_i} \ .$$

By our assumptions on $\mathcal{D}$, this is decidable. (It is this part of the argument that required the passage to $\mathcal{D}^\dagger$. It does not seem to be generally true that the closure under finite unions of an effectively given system is again effectively given.) □

One of the main reasons that $\mathbb{P}\mathcal{D}$ is like a power domain is the possibility of forming "finite sets".

DEFINITION 7.11. For elements $x_0, \ldots, x_{n-1} \in |\mathcal{D}|$ we define

$$\{x_0, \ldots, x_{n-1}\} = \{Z \in \mathbb{P}\mathcal{D} \mid \exists X_0 \in x_0 \ldots \exists X_{n-1} \in x_{n-1} \bigcup_{i<n} (\downarrow X_i) \subseteq Z\}.$$

(Note, we could also write $\forall i < n . X_i \in Z$). □

PROPOSITION 7.12. The mapping

$$\lambda x_0, \ldots, x_{n-1} . \{x_0, \ldots, x_{n-1}\} : \mathcal{D}^n \to \mathbb{P}\mathcal{D}$$

is approximable and is computable if $\mathcal{D}$ is effectively given. Moreover, the map $\lambda x . \{x\}$ shows that $\mathcal{D} \trianglelefteq \mathbb{P}\mathcal{D}$, and we also have the law:

$$\{x_0, \ldots, x_{n-1}\} = \{x_0\} \cap \cdots \cap \{x_{n-1}\}$$

as an intersection of filters.

Proof : The second part shows that everything reduces to $\lambda x . \{x\}$. We see that

$$X_k \, (\lambda x . \{x\}) \bigcup_{i<q} (\downarrow X_{n_i}) \text{ iff } \exists i<q . \; X_k \subseteq X_{n_i} .$$

Thus, $\lambda x . \{x\}$ is an approximable mapping and is computable in the effectively given case.

The proof of the law can be reduced to the special case

$$\{x\} \cap \{y\} = \{x, y\}$$

for the sake of illustration. In terms of finite elements of the two domains $\mathcal{D}$ and $\mathbb{P}\mathcal{D}$ we find

$$\{\uparrow X\} = \uparrow\downarrow X ,$$

and so,

$$\begin{aligned} \{\uparrow X\} \cap \{\uparrow Y\} &= \uparrow\downarrow X \cap \uparrow\downarrow Y \\ &= \uparrow(\downarrow X \cup \downarrow Y) \\ &= \{\uparrow X, \uparrow Y\} . \end{aligned}$$

An equation between approximable functions that checks for finite elements also holds for all elements.

Finally, we note that

$$\mathcal{D} \cong \mathcal{D}^{\downarrow} \lhd \mathbb{P}\mathcal{D}$$

and that the isomorphism involved is just $\lambda x.\{x\}$ by what we saw on the finite elements. □

Further combinators on the power domain are given in the exercises.

EXERCISES

EXERCISE 7.13. Show that an effectively given domain can always be identified with a relation

$$INCL(n,m)$$

on integers, where the two derived relations

$CONS(n,m)$ iff $\exists k.\ INCL(k,n)$ and $INCL(k,m)$;

$MEET(n,m,k)$ iff $\forall j\,[INCL(j,k)$ iff $INCL(j,n)$ and $INCL(j,m)]$

are both recursively decidable, and where the following axioms hold:

(i) $\forall n.\ INCL(n,n)$;

(ii) $\forall n,m,k.\ INCL(n,m)$ and $INCL(m,k)$ imply $INCL(n,k)$;

(iii) $\exists m\ \forall n.\ INCL(n,m)$

(iv) $\forall n,m.\ CONS(n,m)$ implies $\exists k.\ MEET(n,m,k)$.

(Hint: Consider the neighbourhood system

$$\mathcal{D} = \{\{m\in\mathbb{N} \mid INCL(m,n)\} \mid n\in\mathbb{N}\}.$$

Is this essentially any effectively given system?)

EXERCISE 7.14. (For recursive-function theorists.) Prove the statements after definition 7.2 about the existence of primitive recursive functions for showing things recursively enumerable. (Recall that a non-empty set is r.e. iff it is the range of a primitive recursive function.) Show also that every computable element $y\in|E|$ can be written

$$y = \bigcup\{\uparrow Y_{t(i)} \mid i\in\mathbb{N}\},$$

where $t:\mathbb{N}\to\mathbb{N}$ is primitive recursive and where we may assume

$$Y_{t(i+1)} \subseteq Y_{t(i)}$$

for all $i \in \mathbb{N}$.

EXERCISE 7.15. Finish the proof of 7.4 and establish similar results for the constructs $(\mathcal{D}_0 \otimes \mathcal{D}_1)$, $(\mathcal{D}_0 \oplus \mathcal{D}_1)$ and $\mathcal{D}^\infty$. Take into account the various appropriate combinators.

EXERCISE 7.16. Let $\mathcal{D}_0 = \{ X_n | n \in \mathbb{N} \}$, $\mathcal{D}_1 = \{ Y_m | m \in \mathbb{N} \}$ and $\mathcal{D}_2 = \{ Z_k | k \in \mathbb{N} \}$ be three effectively given domains. Complete the proof of 7.5 by writing out curry as a relation between neighbourhoods. Is it a recursive set or only a recursively enumerable set?

EXERCISE 7.17. Complete the proof of 7.7 for showing that $\mathcal{D}^\S$ is effectively given if $\mathcal{D}$ is. Include all the combinators of 6.2. Prove also that if E is effectively given and

$$u : \mathcal{D} \to E \text{ and } v : E \times E \to E$$

are computable, then the unique strict mapping

$$g : \mathcal{D}^\S \to E ,$$

where, for $x \in |\mathcal{D}|$ and $y, z \in |E|$,

$$g\ (\text{in}\ (x)) = u\ (g\ (x)) , \text{ and}$$

$$g\ (\text{pair}\ (y,z)) = v\ (g(y), g\ (z)),$$

is a computable mapping.

EXERCISE 7.18. Two effectively given systems $\mathcal{D}$ and E are *effectively isomorphic* iff ... (complete the sentence!). Show that if $\mathcal{D}$ is effectively given then the isomorphism

$$\mathcal{D}^\infty \cong (\mathcal{D}^\infty)^\infty$$

is effective.

EXERCISE 7.19. Prove that $\mathcal{D} \longmapsto \mathbb{P}\,\mathcal{D}$ is a functor by defining for each $f : \mathcal{D} \to E$ a mapping

$$\mathbb{P}\,f : \mathbb{P}\,\mathcal{D} \to \mathbb{P}\,E$$

by the formula

$$\bigcup_{i<n} \uparrow X_i \quad \mathbb{P}\,f \quad \bigcup_{j<m} \uparrow Y_j \quad \text{iff } \forall i<n \exists j<m.\ X_i f Y_j \ .$$

Be sure to check that $\mathbb{P}\,f$ is approximable and that $\mathbb{P}$ preserves identity maps and composition. If f is computable is $\mathbb{P}\,f$? Is there a combinator $\lambda f.\mathbb{P}\,f$? What is

$$\mathbb{P}\,f(\{x,y\}) = \ ??$$

EXERCISE 7.20. Show that there is a combinator

$$\text{union} : \mathbb{P}\,(\mathbb{P}\,\mathcal{D}) \to \mathbb{P}\,\mathcal{D}$$

where for suitable neighbourhoods

$$\bigcup_{i<n} \uparrow\Big(\bigcup_{j<m_i} \uparrow X_{ij}\Big) \ \text{union} \ \bigcup_{k<q} \uparrow Y_k \quad \text{iff } \forall i<n\ \forall j<m_i\ \exists\, k<q.\ X_{ij} \subseteq Y_k \ .$$

Is union computable if $\mathcal{D}$ is effectively given? What is

$$\text{union}\,(\{\{x\}, \{y,z\}\}) = \ ??$$

Are $\mathbb{P}\,(\mathbb{P}\,\mathcal{D})$ and $\mathbb{P}\,\mathcal{D}$ generally isomorphic??

EXERCISE 7.21. Is there a non-trivial combinator of type

$$\mathbb{P}\,(\mathcal{D} \to E) \to (\mathbb{P}\,\mathcal{D} \to \mathbb{P}\ E) \ ?$$

Are there in general any isomorphisms between the systems

$$(\mathcal{D} \to \mathbb{P}\,E), \ \mathbb{P}\,(\mathcal{D} \times E), \ \ \mathbb{P}\,\mathcal{D} \times \mathbb{P}\ E \ ??$$

Is there a non-trivial combinator of type

$$\mathbb{P}\,(\mathcal{D} \times E) \times \mathbb{P}\,(E \times F) \to \mathbb{P}\,(\mathcal{D} \times F) \ ???$$

Is there any connection between

$$\mathbb{P}\ \mathbb{N} \ \text{ and } \ P\ \mathbb{N} \ ????$$

EXERCISE 7.22. (For algebraists.) Let $\Sigma = \{0,1\}^*$ be the free semigroup. A new domain is constructed by defining a family of sets by the least fixed point theorem as follows

$$S = \{\Sigma\} \cup \{\{\sigma\} \mid \sigma \in \Sigma\} \cup \{XY \mid X, Y \in S\} \cup$$
$$\{X \cap Y \mid X, Y \in S \text{ and } X \cap Y \neq \emptyset\}.$$

Here we write:

$$XY = \{\sigma\tau \mid \sigma \in X \text{ and } \tau \in Y\}.$$

Prove that S is an effectively given, positive neighbourhood system. (Hint: The sets in S are each "regular events" in the terminology of automata theory, and we have a decision method for the set algebra of regular events.) Define multiplication on $|S|$ by

$$x\,y = \{Z \in S \mid \exists X \in x\ \exists Y \in y.\ XY \subseteq Z\},$$

and show $|S|$ becomes a semigroup with Σ embedded into $|S|$ by the homomorphism $\sigma \mapsto \{X \in S \mid \sigma \in X\}$. Investigate some *infinite words* in S, say those defined by least fixed points such as:

$$\vec{\sigma} = \sigma\vec{\sigma} \text{ and } \overleftarrow{\sigma} = \overleftarrow{\sigma}\sigma.$$

Are these equations true:

$$\vec{0}\vec{0} = \vec{0}, \quad \vec{0}\vec{0}\vec{0} = \vec{0}, \quad \vec{0}\vec{1}\vec{0}\vec{1} = \vec{0}\vec{1},$$
$$\text{and } \overleftarrow{01}\ \overrightarrow{01}\ \overleftarrow{01}\ \overrightarrow{01} = \overleftarrow{01}\ \overrightarrow{01} \text{ ?}$$

EXERCISE 7.23. Complete the discussion of $P\mathbb{N}$ of Example 7.8. Show that the combinators fun and graph of Exercise 5.14 are computable. Also do the same for

$$\lambda x, y.\ x \cap y, \quad \lambda x, y.\ x \cup y, \quad \text{and } \lambda x, y.\ x + y,$$

where for $x, y \in P\mathbb{N}$ we define

$$x + y = \{n+m \mid n \in x \text{ and } m \in y\}.$$

What are the computable elements of $P\mathbb{N}$?

EXERCISE 7.24. (Suggested by the LUCID language of Ashcroft and Wadge: SIAM Jour. Comp. vol. 5 (1976).) Define a set Γ by

$$\Gamma = \bigcup_{i=0}^{\infty} (\{i\} \times \Gamma) \cup \{\star\}.$$

Define a system

$$L = \{\Gamma\} \cup \{\{i\} \times X \mid i \in \mathbb{N} \text{ and } X \in L\}.$$

Show that L is effectively given. Show that the elements of $|L|$ can be identified with the finite and infinite sequences of natural numbers. What is the connection between B and L? Show that the combinators of LUCID can be construed as computable mappings of type

$$(L \to T) \to (L \to T)$$

or of type

$$(L \to T) \times (L \to T) \to (L \to T)$$

Conclude that programs in LUCID define computable maps.

LECTURE VIII

RETRACTS OF THE UNIVERSAL DOMAIN

In order to be able to have a fully flexible method of solving domain equations *and* to be able to see why the domains obtained are effectively given, we shall embed all the desired domains in one "largest" domain. This universal domain will be easily shown to be effectively given, and the mappings needed to extract the other domains will be found to be computable. In order to be able to carry out this programme, we investigate first how certain subdomains correspond to mappings - the so-called *retracts* . An advantage of this analysis is that all the necessary definitions can be written out in λ-calculus notation, thus demonstrating the power of our mathematical programming language.

DEFINITION 8.1. A *retraction* of a given domain E is an approximable mapping $a : E \to E$ such that $a \circ a = a$. □

PROPOSITION 8.2. If $\mathcal{D} \lhd E$ and if $a : E \to E$ is defined by

$$X \, a \, Z \quad \text{iff} \quad \exists \, Y \in \mathcal{D}. \; X \subseteq Y \subseteq Z$$

for all $X, Z \in E$, then a is a retraction and $|\mathcal{D}|$ is isomorphic to the fixed-point set of a, the set $\{y \in |E| \mid a(y) = y\}$, under inclusion.

Proof: That a is an approximable mapping is a direct consequence of Definition 6.10. Indeed, in the notation of Proposition 6.12, we have

$$a = i \circ j,$$

and this is another proof that a is approximable. This remark is also convenient, since we know from 6.10

$$j \circ i = I_{\mathcal{D}} .$$

Therefore, we find:

$$a \circ a = i \circ j \circ i \circ j = i \circ j = a ;$$

and so a is a retraction.

We can also employ i and j to give the isomorphism on $|\mathcal{D}|$. If $x \in |\mathcal{D}|$, then $i(x) \in |E|$ and we calculate:

$$a(i(x)) = i \circ j \circ i(x) = i(x).$$

Thus, $i(x)$ belongs to the fixed-point set of a. In the other direction, if $a(y) = y$, then $i(j(y)) = y$. But $j(y) \in |\mathcal{D}|$, so i maps $|\mathcal{D}|$ one-one and onto the fixed-point set of a. As i and j are monotone, the map is an isomorphism with respect to $\subseteq$. □

Not every retraction comes from a relationship like $\mathcal{D} \lhd E$; in fact, we can see from the definition of a above that $a \subseteq I_E$. But, as is indicated in Exercise 8.11 , even this condition is not sufficient to characterize the kind of retractions provided by 8.2. The characterization is as follows.

DEFINITION 8.3. A retraction $a : E \to E$ is called a *projection* provided

$$a \subseteq I_E ;$$

it is *finitary* iff its fixed-point set is isomorphic to a domain. □

EXAMPLES 8.4. If a system $\mathcal{D}$ over Δ is not trivial, then the two element system $\mathcal{O} = \{\{0\}, \{0,1\}\}$ comes from a retraction on $\mathcal{D}$. Specifically, define a combinator

$$\text{check} : \mathcal{D} \to \mathcal{O}$$

by the relation

$$X \text{ check } Y \quad \text{iff either } Y = \{0,1\} \text{ or } X \neq \Delta.$$

We see $\text{check}(x) = \bot_{\mathcal{O}}$ iff $x = \bot_{\mathcal{D}}$. We leave to the reader the definition of a combinator:

$$\text{fade} : \mathcal{O} \times \mathcal{D} \to \mathcal{D} ,$$

where we have for $t \in |\mathcal{O}|$ and $x \in |\mathcal{D}|$:

$$\text{fade}(t,x) = \bot_{\mathcal{D}}, \text{ if } t = \bot_{\mathcal{O}} ;$$

$$= x, \text{ if not.}$$

Now, take any $u \in |\mathcal{D}|$ with $u \neq \bot$, and define

$$a(x) = \text{fade}(\text{check}(x), u).$$

Then a is a retraction (not a projection in general) and the range of a is isomorphic to $\mathcal{O}$.

Another way of using these combinators is to find $(\mathcal{D} \to_{\perp} E)$ as a retraction of $(\mathcal{D} \to E)$. Specifically, define a combinator

$$\text{strict} : (\mathcal{D} \to E) \to (\mathcal{D} \to E)$$

by the equation

$$\text{strict}(f) = \lambda x.\ \text{fade}\ (\text{check}(x),\ f(x)),$$

where this time

$$\text{fade} : \mathcal{O} \times E \to E\ .$$

The range of strict consists exactly of the strict functions and this time strict is a projection whose range is indeed a domain.

Similarly, we can find a projection on $\mathcal{D} \times E$ with a range isomorphic to $\mathcal{D} \otimes E$ by the combinator such that:

$$\text{smash}(x,y) = \text{fade}\ (\text{check}(x), \text{fade}\ (\text{check}(y), \langle x,y \rangle)),$$

for $x \in |\mathcal{D}|$ and $y \in |E|$. □

THEOREM 8.5. For an approximable mapping $a : E \to E$ the following are equivalent:

(i) a is a finitary projection;

(ii) $a(x) = \{Y \in E \mid \exists\, X \in x.\ \ X \mathrel{a} X \subseteq Y\}$, for all $x \in |E|$.

Proof: Suppose a satisfies (ii) first. Inasmuch as

$$X \in x \text{ and } X \subseteq Y \text{ always imply } Y \in x,$$

for all $x \in |E|$, we see $a(x) \subseteq x$ must always hold. Moreover, it is obvious that

$$X \in x \text{ and } X \mathrel{a} X \text{ always imply } X \in a(x);$$

therefore, $a(x) \subseteq a\,(a\,(x))$ for all $x \in |E|$. This shows that a

is indeed a projection.

Let $\mathcal{D} = \{X \in E \mid X\, a\, X\}$, then it is easy to check that $\mathcal{D} \lhd E$ and that a is determined from $\mathcal{D}$ exactly as in 8.2; thus, the fixed-point set of a is isomorphic to a domain, by what we have already proved. So we have shown (ii) implies (i).

In the converse direction, assume that a is a finitary projection. And let the system $\mathcal{D}$ be isomorphic to the fixed point set of a. We have the situation of Theorem 6.15 There is a projection pair,

$$i : \mathcal{D} \to E \text{ and } j : E \to \mathcal{D},$$

where the connection with a gives:

$$j \circ i = I_{\mathcal{D}} \text{ and } i \circ j = a \subseteq I_E.$$

By 6.15 $\mathcal{D} \cong \mathcal{D}' \lhd E$ and we want to identify $\mathcal{D}'$ in terms of a as follows:

$$\mathcal{D}' = \{X \in E \mid X\, a\, X\}.$$

Now from a reading of the proof of 6.15 the neighbourhoods of $\mathcal{D}'$ are just those corresponding to the finite elements of $\mathcal{D}$. But any such element is a fixed point of a. We have

$$X \in \mathcal{D}' \text{ implies } a(\uparrow X) = \uparrow X \text{ implies } X\, a\, X.$$

Conversely, if $X\, a\, X$ holds, then $\uparrow X \subseteq a(\uparrow X)$. But a is a projection, so $\uparrow X$ is a fixed point. But $i(j(\uparrow X)) = \uparrow X$ means $j(\uparrow X)$ is a finite element of $|\mathcal{D}|$. So $X \in \mathcal{D}'$, and we have $\mathcal{D}'$ identified as desired.

Finally, if we calculate $a = i \circ j$ by the formulae of 6.12 (with $\mathcal{D}'$ for $\mathcal{D}$, of course), we obtain our formula (ii). □

The criterion for being a finitary projection just obtained provides us with a very interesting new combinator.

THEOREM 8.6. For any domain E define

$$\text{sub} : (E \to E) \to (E \to E)$$

by the formula

$$X \text{ sub } (f)\ Z \text{ iff } \exists Y \in E.\ \ X \subseteq Y\, f\, Y \subseteq Z,$$

for all $X, Z \in E$ and all $f : E \to E$. Then the range of sub consists exactly of the finitary projections on E, and moreover sub itself is a finitary projection on $(E \to E)$. If E is effectively given, then sub is computable.

Proof: It is trivial to check that sub(f) is always approximable. Also, it is obvious from the definition that the correspondence

$$f \mapsto \text{sub}(f)$$

preserves directed unions of f's. Thus, sub is itself approximable. We note that

$$X \subseteq Y \; f \; Y \subseteq Z \text{ always implies } X \; f \; Z;$$

hence, $\text{sub}(f) \subseteq f$ holds. Also

$$Y \; f \; Y \text{ always implies } Y \; \text{sub}(f) \; Y,$$

hence, $\text{sub}(f) \subseteq \text{sub}(\text{sub}(f))$ holds. This shows sub to be a projection on $(E \to E)$. The effectiveness of the definition makes it also clear that sub is computable when E has a computable presentation.

Since, sub is a projection, its range is the same as its fixed-point set. If

$$\text{sub}(a) = a,$$

then there is no problem in checking that a satisfies 8.5(ii) *and conversely*. So the range of sub picks out exactly the finitary projections in view of 8.5.

Finally, to prove that sub is a finitary projection of $(E \to E)$, we invoke 6.11 and remark that, in view of 8.2, the fixed point set (range) of sub is in a one-one inclusion-preserving correspondence with the domain $\{\mathcal{D} \mid \mathcal{D} \lhd E\}$. □

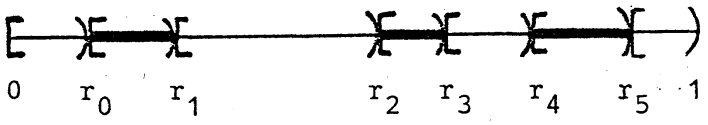
These results have almost completely translated the theory of $\lhd$-subdomains into λ-calculus *via* the sub-combinator. One last step will complete the passage, and then we shall be able to return to solving domain equations.

DEFINITION 8.7. Let $\mathbb{Q}$ be the set of rational numbers, and let

$$[0, 1) = \{q \in \mathbb{Q} \mid 0 \leqslant q < 1\},$$

and similarly for $[r, s)$ for any $r < s$ in $\mathbb{Q}$. The neighbourhood system $\mathcal{U}$ over $[0,1)$ is the set of all non-empty *unions* of intervals of rational intervals $[r, s)$ with $0 \leqslant r < s \leqslant 1$. □

A picture of a typical element of $\mathcal{U}$ could be drawn like this:



Note that any union can be taken as a *disjoint* union of the form

$$\bigcup_{i \leqslant n} [r_{2i}, r_{2i+1})$$

where $0 \leqslant r_0 < r_1 < r_2 < \cdots < r_{2n} < r_{2n+1} \leqslant 1$. (Hint: Any overlapping intervals or abutting intervals can always be combined into one long interval.) It is a most elementary exercise to show that, by virtue of this representation, the system $\mathcal{U}$ has a computable presentation. (Some isomorphic versions of $\mathcal{U}$ - equally effective - are recorded in the exercises.) Note that $\mathcal{U}$ has no minimal neighbourhoods: every set in $\mathcal{U}$ can be written as the union of two disjoint sets in $\mathcal{U}$. (Hint: Use the density of the ordering of $\mathbb{Q}$.) The significance of $\mathcal{U}$ can now be explained.

THEOREM 8.8. The system $\mathcal{U}$ is universal in the sense that, for every *countable* neighbourhood system $\mathcal{V}$, we have

$$\mathcal{V} \trianglelefteq \mathcal{U}.$$

Moreover, if $\mathcal{V}$ is effectively given, then the projection pair making the embedding can be taken as computable. Indeed there is a correspondence between effectively presented domains and the computable, finitary projections of $\mathcal{U}$.

Proof: As $\mathcal{V}$ is countable, we can assume that

$$\mathcal{V} = \{X_n \mid n \in \mathbb{N}\},$$

where $\mathcal{D}$ is a system over a set Δ (say, $X_0 = \Delta$). We shall do the effective and general cases together, where for the latter all remarks on recursiveness are just left out. So, if we want $\mathcal{D}$ effectively given, the above enumeration should be taken as the computable presentation.

Without loss of generality we can assume $\mathcal{D} \cong \mathcal{D}^{\downarrow}$, since otherwise we would just replace $\mathcal{D}$ by $\mathcal{D}^{\downarrow}$. The advantage of this preparation is that unions in $\mathcal{D}^{\downarrow}$ keep things rather *separate* (as we noticed in constructing $\mathbb{P}\mathcal{D}$). In particular, we can be sure of this equivalence:

$$(\blacklozenge) \qquad X_m \subseteq \bigcup_{i<k} X_{n_i} \quad \text{iff} \quad \exists i<k.\ X_m \subseteq X_{n_i}.$$

This property, for example, fails for the system $\mathcal{U}$ as presented in Definition 8.7. However, that observation is of no moment, because we are employing the assumption with respect to $\mathcal{D}$ not $\mathcal{U}$.

The reason for the assumption is this: for $\delta \in \{+,-\}$ define for $X \in \mathcal{D}$:

$$\begin{aligned} \delta X &= X \quad \text{if } \delta = +\ ; \\ &= \Delta \setminus X \quad \text{if } \delta = -\ . \end{aligned}$$

(A similar notation will be used for $Y \in \mathcal{U}$.) Then for $\delta \in \{+,-\}^n$ the sets of the form

$$\bigcap_{i<n} \delta_i X_i \quad (= X_\delta, \text{ for short})$$

form a partition of Δ into (at most) 2^n parts. The reason for assumption $(\blacklozenge)$ is that we can effectively decide for each $\delta \in \{+,-\}^n$ whether one of these intersections is empty or not. (Why? - assuming that $\mathcal{D}$ is effectively given, of course). If for some reason we had not wanted to pass to $\mathcal{D}^{\downarrow}$, we could have made this stronger assumption of decidability on the (positive) system $\mathcal{D}$. ($\mathcal{U}$, for example, satisfies it.)

Suppose, corresponding to $X_0, X_1, \ldots, X_{n-1}$, we have selected $Y_0, Y_1, \ldots, Y_{n-1}, \in \mathcal{U}$ so that, for all $\delta \in \{+,-\}^n$,

$$(\blacksquare) \qquad \bigcap_{i<n} \delta_i X_i = \emptyset \quad \text{iff} \quad \bigcap_{i<n} \delta_i Y_i = \emptyset .$$

We wish to show - effectively - how to choose Y_n corresponding to X_n, so that (■) holds with n+1 replacing n. Proceeding inductively, we obtain a recursive enumeration of sets $Y_n \in \mathcal{U}$ so that

$$\mathcal{D} \cong \{Y_n \mid n \in \mathbb{N}\} \lhd \mathcal{U}.$$

Clearly the isomorphism (matching X_i to Y_i) will be computable and the projection is computable. (It will then remain only to consider the arbitrary finitary computable projection to complete the proof of the theorem.)

So, consider X_n; for each $\delta \in \{+,-\}^n$ there are four cases:

$$X_\delta \cap X_n = \emptyset\ , \quad X_\delta \cap -X_n = \emptyset\ ,$$
$$X_\delta \cap X_n \neq \emptyset\ , \quad X_\delta \cap -X_n \neq \emptyset\ .$$

Corresponding to X_δ is a similar intersection Y_δ. If X_δ were $\emptyset$, then Y_δ would be also. If not, $Y_\delta \subseteq [0,1)$ is a union of rational intervals that can be written down explicitly. (Why?) In our four cases on X_n, the first implies the fourth. (Why?) Thus, we need only make some choices in these circumstances:

$X_\delta \cap X_n = \emptyset$: choose $I_{\delta,n} = \emptyset$;

$X_\delta \cap -X_n = \emptyset$: choose $I_{\delta,n} = Y_\delta$;

otherwise : choose $I_{\delta,n} \subseteq Y_\delta$, with $\emptyset \neq I_{\delta,n} \neq Y_\delta$.

All these cases are decidable by assumption on $\mathcal{D}$, and the effective choice of (unions of) intervals is effective by construction of $\mathcal{U}$. Now set

$$Y_n = \bigcup_{\delta \in \{+,-\}^n} I_{\delta,n} \neq \emptyset$$

The set $Y_n \in \mathcal{U}$, it can be found effectively, and (■) is obviously satisfied for n+1 .

Finally, suppose that a is a computable, finitary projection of $\mathcal{U}$. As we have seen in the proof of 8.5, the domain corresponding to the range of a is isomorphic to the neighbourhood system

$$\{Y \in \mathcal{U} \mid Y\, a\, Y\} \lhd \mathcal{U}.$$

Clearly, if a as a set of ordered pairs of neighbourhoods is recursively enumerable, then the above set is also recursively enumerable (because equality between neighbourhoods is decidable). It follows easily that the subsystem is effectively given as a neighbourhood system in its own right. □

We have now proved that U is a nice and big domain that is nicely behaved with respect to computable mappings. It has some very interesting subdomains; to name a few:

$$U+U,\quad U\oplus U,\quad U\times U,\quad U\otimes U$$
$$U_{\perp},\quad U^{\infty},\quad U^{\S},\quad \mathbb{P}U,\quad U\to U.$$

That all of these are $\lhd U$ follows from knowing that they are all effectively presented. What we wish to check next is that they all combine well with respect to projections. To this end the explicit definitions are given for the constructs $+$, $\times$, and $\to$, and the details of the others are left for the exercises.

DEFINITION 8.9. Let the computable projection pairs

$$i_+ : U+U\to U \text{ and } j_+ : U\to U+U$$

be fixed. Similarly choose $i_\times, j_\times$ and $i_\to, j_\to$ for $U\times U$ and $U\to U$. Define:

$$a+b = \mathsf{cond}\circ\langle \mathsf{which},\ i_+\circ \mathsf{in}_0\circ a\circ \mathsf{out}_0,\ i_+\circ \mathsf{in}_1\circ b\circ \mathsf{out}_1\rangle\circ j_+ ;$$
$$a\times b = i_\times\circ\langle a\circ \mathsf{proj}_0, b\circ \mathsf{prog}_1\rangle\circ j_\times ;$$
$$a\to b = i_\to\circ(\lambda f.\ b\circ f\circ a)\circ j_\to ,$$

for all $a, b : U\to U$. □

These interesting (computable!) combinators on elements of $U\to U$ have many, many properties. We shall, however, only see what they do to projections.

PROPOSITION 8.10. If $a, b : U\to U$ are projections, then so are $a+b$, $a\times b$, and $a\to b$. If a and b are finitary, then so are the others; for the fixed-point set of each of them is isomorphic to the corresponding construct applied to the domains determined by a and b.

Proof: Suppose that $a, b \subseteq I_U$ (= I for short). Then

$$a+b \subseteq I + I = i_+ \circ j_+ \subseteq I.$$

The other cases are similar.

Suppose $a = a \circ a$ and $b = b \circ b$, then, for example,

$$\begin{aligned}(a\times b) \circ (a\times b) &= i_\times \circ \langle a \circ proj_0,\ b \circ proj_1\rangle \circ \langle a \circ proj_0,\ b \circ proj_1\rangle \circ j_\times \\ &= i_\times \circ \langle a \circ a \circ proj_0,\ b \circ b \circ proj_1\rangle \circ j_\times \\ &= a \times b.\end{aligned}$$

The other cases are similar.

Now in case the fixed-point sets of a and b are domains, they are respectively isomorphic to

$$\mathcal{D}_a = \{X \in U \mid X\ a\ X\} \text{ and}$$
$$\mathcal{D}_b = \{Y \in U \mid Y\ b\ Y\}.$$

We have to show, for example, that

$$\mathcal{D}_a \to \mathcal{D}_b \cong \mathcal{D}_{a\to b}$$

Now to simplify matters, remark that the fixed-point set of $a \to b$ on U is isomorphic to the fixed-point set of $\lambda f.\ b \circ f \circ a$ on $(U \to U)$. (Hint: use $i_\to$ and $j_\to$ to set up the isomorphism.) So we have to think what it is for an $f : U \to U$ to satisfy

$$f = b \circ f \circ a.$$

Notice that we might as well say that $a : U \to \mathcal{D}_a$ and that this map is the other half of an obvious projection pair where

$$i_a : \mathcal{D}_a \to U,$$

and $i_a \circ a = a$ and $a \circ i_a = i_a$. So if $g : \mathcal{D}_a \to \mathcal{D}_b$, let

$$f = i_b \circ g \circ a,$$

then $b \circ f \circ a = f$. Conversely, if f is like this, then let

$$g = b \circ f \circ i_a.$$

Thus, $i_b \circ g \circ a = b \circ f \circ a = f$; so there is an order-preserving isomorphism between the $g : \mathcal{D}_a \to \mathcal{D}_b$ and the $f = b \circ f \circ a$.

The isomorphism proofs for + and × are similar. □

Well, this was a lot of work, but the pay-off is rather handsome. What we have done is transpose all the

$$\mathcal{D}_a \lhd U$$

over to finitary projections $a : U \to U$. This transposition is an isomorphism, because

$$\mathcal{D}_a \lhd \mathcal{D}_b \quad \text{iff} \quad a \subseteq b.$$

Moreover, by the method of 8.9 and 8.10, *all* our favourite constructs have been made into *combinators*, that is, approximable - even computable - maps on the domain of finitary projections. *ALL APPROXIMABLE (COMPUTABLE) MAPS HAVE (COMPUTABLE) FIXED POINTS.* And there you are! The standard fixed-point method is available to obtain computable (i.e. effectively given) solutions to *all* domain equations (even sets of equations) where the constructs can be reworked in this way to be defined on projections. Examples are suggested in the exercises.

Another pay-off concerns the λ-calculus itself. Inasmuch as

$$U + U, \quad U \times U, \quad U \to U \trianglelefteq U,$$

we might just as well forget the outside world and regard all these useful domains as being part of U. For example, on the left we have the new notation and on the right the old notation:

$$\begin{aligned}
\mathsf{which}\ (z) &= \mathsf{which}(j_+(z)) \ ; \\
\mathsf{in}_i\ (x) &= i_+(\mathsf{in}_i(x)), \quad i = 0, 1 \ ; \\
\mathsf{out}_i\ (x) &= \mathsf{out}_i(j_+(x)), \quad i = 0, 1 \ ; \\
\langle x, y\rangle &= i_\times(\langle x, y\rangle) \ ; \\
\mathsf{proj}_i\ (z) &= \mathsf{proj}_i(j_\times(z)), i = 0, 1 \ ; \\
u\,(x) &= j_\leftarrow(u)\,(x) \ ; \\
\lambda\, x.\, \tau &= i_\to(\lambda\, x.\, \tau).
\end{aligned}$$

And, there is no reason to stop here. The system

$$T \cong \{[0,1/2), [1/2,1), [0,1)\} \lhd U \ ,$$

so we might as well think of

$$\text{true}, \text{false} \in |U|$$

and think of cond: $U \times U \times U \to U$. No! that is wrong: under the new regime *EVERYTHING IS AN ELEMENT OF* U. With the new meaning of λ, all functions, all pairs, all combinators, all constructs became *elements* of U.

It takes a little time to get used to "universal conscription" with all elements doing (at least) double duty in the same domain, but there are many advantages, both notational and conceptual.

EXERCISES

EXERCISE 8.11. Let $\mathbb{Q}$ be the set of *rational* numbers and define a neighbourhood system by the equation

$$R = \{[0, r) \mid r \in \mathbb{Q} \text{ and } 0 < r \leqslant 1\}.$$

Show that the following defines an approximable map $a : R \to R$:

$$[0,r)\ a\ [0,s) \text{ iff } r < s \text{ or } r = s = 1.$$

Show in addition that a is a projection where the fixed-point set of a is in a one-one correspondence with the *real* numbers between 0 and 1 inclusive. (Hint: Recall Dedekind cuts and show $\subseteq$ matches $\leqslant$.) Conclude that a is *NOT* finitary. (Hint: Aside from $\perp$ there areno finite elements for $\{x \mid x = a(x)\}$.)

EXERCISE 8.12. Generalize the notation $2X+1$ for subsets $X \subseteq \mathbb{N}$ to sets of the form

$$2^k X + \ell, \text{ where } \ell < 2^k.$$

Let V be the non-empty finite unions of sets $2^k \mathbb{N} + \ell$. Show that $U \cong V$ and that the isomorphism is effective, thus obtaining another presentation of U.

EXERCISE 8.13. (For logicians.) Prove that the universal domain U is isomorphic to the domain of all proper filters of the free Boolean algebra on $\aleph_0$-generators (= the Lindenbaum algebra of propositional calculus). (For topologists.) Connect this

representation of U with the collection of non-empty open subsets of the product space $2^{\mathbb{N}}$ (= Cantor space).

EXERCISE 8.14. A retraction $a : \mathcal{D} \to \mathcal{D}$ is called a *closure operator* iff $I_{\mathcal{D}} \sqsubseteq a$. On a domain like $P\mathbb{N}$, give some examples of closure operators. (Hint: Close up a set of integers under addition. Is this continuous on $P\mathbb{N}$?) Prove in general for any closure $a : \mathcal{D} \to \mathcal{D}$ that the fixed-point set of a is always a finitary domain. (Hint: Show that the fixed-point set is closed under intersections and directed unions.) What are the finite elements of the fixed-point set?

EXERCISE 8.15. Give a direct proof that the domain $\{X \mid X \lhd \mathcal{D}\}$ is effectively presented if $\mathcal{D}$ is. (Hint: The finite elements of the domain correspond exactly to the finite systems $X \lhd \mathcal{D}$.) In the case of $\mathcal{D} = U$, show that the computable elements of the domain correspond exactly to the effectively presented domains (up to effective isomorphism).

EXERCISE 8.16. For finitary projections $a : E \to E$, write

$$\mathcal{D}_a = \{X \in E \mid X \, a \, X\}$$

(cf. 8.5.). Show that for any two such projections $a, b : E \to E$ we have

$$a \sqsubseteq b \quad \text{iff} \quad \mathcal{D}_a \lhd \mathcal{D}_b.$$

(This fills in the gap at the end of the proof of 8.6.) Also finish off the proof of 8.8 by showing that if E is effectively given and $a : E \to E$ is computable, then $\mathcal{D}_a$ is effectively given.

EXERCISE 8.17. Find explicitly (if possible) the projection pairs for $U + U$, $U \times U$, and $U \to U$ needed for 8.9. Are any of these domains isomorphic with U? (The author does not know a really good construction for $U \to U$.) Find a universal domain $V \not\cong U$.

EXERCISE 8.18. Many of the cases of 8.10 were left unproved. Please establish these assertions explicitly.

EXERCISE 8.19. Suppose we know both

$$T \text{ and } E \to E \trianglelefteq E \ .$$

Does it follow that $E + E$ and $E \times E \trianglelefteq E$?

EXERCISE 8.20. For any system we know $\mathcal{D} \trianglelefteq \mathcal{D} + \mathcal{D}$, but what about

$$\mathcal{D} \trianglelefteq \mathcal{D} \times \mathcal{D} \text{ and } \mathcal{D} \trianglelefteq \mathcal{D} \to \mathcal{D} \ ?$$

Would these projections be computable if $\mathcal{D}$ is effectively given? Are there more than one projection pair in each case?

EXERCISE 8.21. Using the fixed-point construction, show that there is a continuous and computable operator $\lambda a.\ a^{\S}$, such that if a is a finitary projection of U, then

$$\mathcal{D}_{a^{\S}} \cong (\mathcal{D}_a)^{\S} \ .$$

EXERCISE 8.22. Which of the two relations hold:

$$B \trianglelefteq C \text{ or } C \trianglelefteq B \ ?$$

Or do they both hold? In general if we use domain equations

$$\mathcal{D} = T(\mathcal{D}) + S(\mathcal{D}), \text{ and}$$
$$E = T(E) \ ,$$

will $E \trianglelefteq \mathcal{D}$ hold? What projections do you see in the examples in 6.2?

EXERCISE 8.23. Suppose a construct T on domains can be made into a computable operator $t : (U \to U) \to (U \to U)$ so that whenever $a : U \to U$ is a finitary projection, then so is t(a) and

$$\mathcal{D}_{t(a)} \cong T(\mathcal{D}_a).$$

Does it follow that $||t|| = fix(t)$ is such that

$$\mathcal{D}_{||t||} \cong T(\mathcal{D}_{||t||})$$

really is the initial solution of the domain equation with respect to projections? Since t is computable, will this solution be effectively given?

EXERCISE 8.24. Suppose S and T are two (binary-argument) constructs on domains that can be made into computable operators on projections of the universal domain. Show that we can therefore find a pair of effectively presented domains such that

$$\mathcal{D} \cong S(\mathcal{D},E) \text{ and } E \cong T(\mathcal{D},E)^{\cdot}.$$

EXERCISE 8.25. The problem is to find non-trivial solutions to the domain equation

$$(\clubsuit) \qquad \mathcal{D} \cong \mathcal{D} \to \mathcal{D}$$

Show that the "obvious" solution by retracts is of no use because

$$\perp \to \perp = \perp$$

for projections. Change the method as follows. Show first

$$U^{\infty} \times U^{\infty} \cong U^{\infty} .$$

Next solve

$$\mathcal{D} \cong \mathcal{D} \to U^{\infty}$$

and remark that $U \lhd \mathcal{D}$; so $\mathcal{D}$ is universal and non-trivial. Finally prove ($\clubsuit$) for this $\mathcal{D}$. (Hint: First show

$$\mathcal{D} \times \mathcal{D} \cong \mathcal{D},$$

and then show $\mathcal{D}$ satisfies ($\clubsuit$).) Is this $\mathcal{D}$ effectively given?

EXERCISE 8.26. Discuss in more detail the "pay-off" for U, namely the translation of "untyped" λ-calculus into U as shown by the equations at the end of the lecture after the proof of 8.9. In particular show how the whole of the *typed* λ-calculus can be retranslated back into U with the aid of projections. (Hint: Whenever you want to write

$$f : \mathcal{D}_a \to \mathcal{D}_b,$$

write instead

$$f = b \circ f \circ a,$$

where a, b are finitary projections. Whenever you want to form a λ-abstraction

$$\lambda x^{\mathcal{D}_a}.\, \sigma,$$

where σ is of type $\mathcal{D}_b$, instead form

$$\lambda x.\, b(\sigma'[a(x)/x]),$$

where σ' is the further translation of σ into untyped λ-calculus. Be sure to show that this result "has the right type" in the sense defined above.)

EXERCISE 8.27. (Suggested by James Donahue.) Finite cartesian products of domains are formed by the $\mathcal{D}_0 \times \mathcal{D}_1$- construct we have used so often. The problem is to define - computably - some *infinite* cartesian products. In particular, as applied to the universal domain U, the combinator sub is to be regarded as a finitary projection of U whose fixed points are exactly *all* the finitary projections. A map

$$d = \mathsf{sub} \circ d \circ \mathsf{sub}$$

can be regarded as a *polymorphic type* (because, whenever t is a finitary projection (= type), then so is d(t)). The *continuous product* of all these types would be the domain of all approximable functions x such that

$$x(t) = d(t)(x(t))$$

for all types t. (Why does this equation mean that x is in the product?) Define Π as a combinator by

$$\Pi = \lambda\, d\, \lambda\, x\, \lambda\, t.\ \mathsf{sub}\ (d\ (\mathsf{sub}\ (t)))\ (x\ (\mathsf{sub}(t))).$$

Show that for d a polymorphic type, $\Pi(d)$ is a type. (Hint: It is easy to check that $\Pi(d)$ is a projection; the problem is to show it is *finitary*.)