Domain Theory: An Introduction

Robert Cartwright

Rebecca Parsons

Rice University

This monograph is an unauthorized revision of "Lectures On A Mathematical Theory of Computation" by Dana Scott [3]. Scott's monograph uses a formulation of domains called *neighborhood systems* in which finite elements are selected subsets of a master set of objects called "tokens". Since tokens have little intuitive significance, Scott has discarded neighborhood systems in favor of an equivalent formulation of domains called *information systems* [4]. Unfortunately, he has not rewritten his monograph to reflect this change.

We have rewritten Scott's monograph in terms of *finitary bases* (see Cartwright [2]) instead of *information systems*. A finitary basis is an information system that is closed under least upper bounds on finite consistent subsets. This convention ensures that every finite answer is represented by a single basis object instead of a set of objects.

1 The Rudiments of Domain Theory

Motivation

Programs perform computations by repeatedly applying primitive operations to data values. The set of primitive operations and data values depends on the particular programming language. Nearly all languages support a rich collection of data values including *atomic* objects, such as booleans, integers, characters, and floating point numbers, and *composite* objects, such as arrays, records, sequences, tuples, and infinite streams. More advanced languages also support functions and procedures as data values. To define the meaning of programs in a given language, we must first define the building blocks—the primitive data values and operations—from which computations in the language are constructed.

Domain theory is a comprehensive mathematical framework for defining the data values and primitive operations of a programming language. A critical feature of domain theory (and expressive programming languages like Scheme and **ML**) is the fact that program operations are also data values; both operations and values are elements of computational *domains*.

In a language implementation, every data value and operation is represented by a finite configuration of symbols (*e.g.*, a bitstring). But the choice of data representations should not affect the observable behavior of programs. Otherwise, a programmer cannot reason about the behavior of programs independent of their implementations.

To achieve this goal, we must abandon finite representations for some data values. The abstract meaning of a procedure, for example, is typically defined as a function from an infinite domain to an infinite codomain. Although the graph of this function is recursively enumerable, it does not have an effective *finite* canonical representation; otherwise we could decide the equality of recursively enumerable sets by generating their canonical descriptions and comparing them.

Data values that do not have finite canonical representations are called *infinite* data values. Some common examples are functions over an infinite domain, infinite streams, and infinite trees. To describe an infinite data value, we must use an infinite sequence of progressively better finite approximations. Each finite approximation obviously has a canonical representation.

We can interpret each finite approximation as a proposition asserting that a certain property is true of the approximated value. By stating enough different properties (a countably infinite number in general), every higher order data value can be uniquely identified.

Higher order data domains also contain ordinary finite values. There are two kinds of finite values.

- First, the finite elements used to approximate infinite values are legitimate data values themselves. Even though these approximating elements are only "partially defined", they can be produced as the *final* results fo computations. For example, a tree of the form $cons(\alpha, \beta)$, where α and β are arbitrary values, is a data value in its own right, because a computation yielding $cons(\alpha, \beta)$ may subsequently diverge without producing any information about the values α and β.

- Second, higher order domains may contain "maximal" finite elements that do not approximate any other values. These "maximal" values correspond to conventional finite data objects. For example, in the domain of potentially infinite binary trees of integers, the leaf consisting of the integer 42 does not approximate any element other than itself.

In summary, a framework for defining data values and operations must accommodate infinite elements, partially defined elements, and finite maximal elements. In addition, the framework should support the construction of more complex values from simpler values, and it should support a notion of computation on these objects. This paper describes a framework—called *domain theory*—satisfying all of these properties.

Notation

The following notation is used throughout the paper:

$\Rightarrow$	means	logical implication
$\Longleftrightarrow$	means	if and only if, used in mathematical formulas
iff	means	if and only if, used in text
$\sqsubseteq$	means	approximation ordering
$\mathbb{N}$	means	the natural numbers

Basic Definitions

To support the idea of describing data values by generating "better and better" approximations, we need to specify an ordering relation among the finite approximations to data values. The following definitions describe the structure of the sets of finite approximations corresponding to domains; these sets of finite approximations are called *finitary bases*.

Definition 1.1: [Partial Order] A *partial order* $\mathbf{B}$ is a pair $\langle B, \sqsubseteq \rangle$ consisting of (i) a set B called the *universe* and (ii) a binary relation $\sqsubseteq$ on the set B called the *approximation ordering* that is

- *reflexive:* $\forall x \in B[x \sqsubseteq x]$,
- *antisymmetric:* $\forall x, y \in B[x \sqsubseteq y]$ and $[y \sqsubseteq x]$ implies $x = y$, and
- *transitive:* $\forall x, y, z \in B[x \sqsubseteq y]$ and $[y \sqsubseteq z]$ implies $x \sqsubseteq z$.

Definition 1.2: [Upper bounds, Lower bounds, Consistency] Let S be a subset of a partial order $\mathbf{B}$. An element $b \in B$ is an *upper bound* of S iff $\forall s \in S\ s \sqsubseteq b$. An element $b \in B$ is a *lower bound* of S iff $\forall s \in S\ b \sqsubseteq s$. S is *consistent* (sometimes called *bounded*) iff S has an upper bound. An upper bound b of S is the *least upper bound* of S (denoted $\bigsqcup S$) iff b approximates all upper bounds of S. A lower bound b of S is the *greatest lower bound* of S (denoted $\bigsqcap S$) iff every lower bound of S approximates b.

Remark 1.3: Upper bounds are much more important in domain theory than lower bounds.

Definition 1.4: [Directed Subset] A subset S of a partial order $\mathbf{B}$ is *directed* iff every finite subset of S has an upper bound in S. A directed subset of $\mathbf{B}$ is *progressive* iff it does not contain a maximum element. A directed subset of $\mathbf{B}$ is a *chain* iff it is *totally ordered*: $\forall a, b \in \mathbf{B} a \sqsubseteq b$ or $b \sqsubseteq a$.

Claim 1.5: The empty set $\emptyset$ is directed.

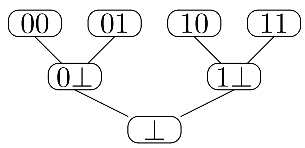
Definition 1.6: [Complete Partial Order] A *complete partial order*, abbreviated *cpo*, is a partial order $\langle B, \sqsubseteq \rangle$ such that every directed subset has a least upper bound in B.

Claim 1.7: A *cpo* has a least element.

Definition 1.8: [Finitary Basis] A *finitary basis* $\mathbf{B}$ is a partial order $\langle B, \sqsubseteq \rangle$ such that B is countable and every finite consistent subset $S \subseteq B$ has a least upper bound in B.

We call the elements of a finitary basis $\mathbf{B}$ *propositions* because they can be interpreted as logical assertions about domain elements. In propositional logic, the least upper bound of a set of propositions is the conjunction of all the propositions in the set. Since the empty set, $\emptyset$, is a finite consistent subset of $\mathbf{B}$, it has a least upper bound, which is denoted $\bot_B$. The $\bot$ proposition holds for all domain elements; hence it does not give any "information" about an element.

Example 1.9: Let $B = \{\bot, 0\bot, 1\bot, 00, 01, 10, 11\}$ where $0\bot$ describes strings that start with 0 and are indeterminate past that point; 00 describes the string that consisting of two consecutive 0's. The other propositions are defined similarly. Let $\sqsubseteq$ denote the obvious approximation (implication) relation between propositions. Thus, $0\bot \sqsubseteq 00$ and $0\bot \sqsubseteq 01$. In pictorial form, the partial order $\langle B, \sqsubseteq \rangle$ looks like:



$\langle B, \sqsubseteq \rangle$ is clearly a partial order. To show that $\langle B, \sqsubseteq \rangle$ is a finitary basis, we must show that B is countable and that all finite consistent subsets of B have least upper bounds.

Since B is finite, it is obviously countable. It is easy to confirm that every finite consistent subset has a least upper bound by inspection. In fact, the least upper bound of any consistent subset S of B is simply the greatest element of S.[1] Thus, $\langle B, \sqsubseteq \rangle$ is a finitary basis. $\Box$

Example 1.10: Let $B = \{(n, m) \mid n, m \in \mathbb{N} \cup \{\infty\}, n \leq m\}$ where the proposition (n, m) represents an integer x such that $n \leq x \leq m$. $\bot$ in this example is the proposition $(0, \infty)$. Let $\sqsubseteq$ be defined as

$$(n, m) \sqsubseteq (j, k) \iff n \leq j \text{ and } m \geq k$$

For example, $(1, 10) \sqsubseteq (2, 6)$ but $(2, 6)$ and $(7, 12)$ are incomparable, as are $(2, 6)$ and $(4, 8)$. It is easy to confirm that B is countable and that $\langle B, \sqsubseteq \rangle$ is a partial order. A subset S of B is consistent if there is an integer for which the proposition in S is true. Thus, $(2, 6)$ and $(4, 8)$ are consistent since either 4, 5, or 6 could be represented by these propositions. The least upper bound of these elements is $(4, 6)$. In general, for a consistent subset $S = \{(n_i, m_i) \mid i \in I\}$ of B, the least upper bound of S is defined as

$$\bigsqcup S = (\mathsf{max}\,\{n_i \mid i \in I\}, \mathsf{min}\,\{m_i \mid i \in I\})\,.$$

Therefore, $\langle B, \sqsubseteq \rangle$ is a finitary basis. $\Box$

Given a finitary basis $\mathbf{B}$, the corresponding *domain* $\mathcal{D}_{\mathbf{B}}$ is constructed by forming all consistent subsets of $\mathbf{B}$ that are "closed" under implication (where $a \sqsubseteq b$ corresponds to $b \Rightarrow a$ and $a \sqcup b$ corresponds to $a \wedge b$). More precisely, a consistent subset $S \subseteq \mathbf{B}$ is an element of the corresponding domain $\mathcal{D}_{\mathbf{B}}$ iff

[1]This is a *special* property of B. It is not true of finitary bases in general.

- $\forall s \in S \;\; \forall b \in \mathbf{B} \;\; b \sqsubseteq s \;\Rightarrow\; b \in S$

- $\forall r, s \in S \;\; r \sqcup s \in S$

In $\mathcal{D}_{\mathbf{B}}$, there is a unique element $\mathcal{I}_p$ corresponding to each proposition $p \in B$: $\mathcal{I}_p = \{b \in \mathbf{B} \mid b \sqsubseteq p\}$. In addition, $\mathcal{D}_{\mathbf{B}}$ contains elements ("closed" subsets of B) corresponding to the "limits" of all progressive directed subsets of $\mathbf{B}$. This construction "completes" the finitary basis $\mathbf{B}$ by adding limit elements for all progressive directed subsets of B.

In $\mathcal{D}_{\mathbf{B}}$, every element d is represented by the set of all the propositions in the finitary basis $\mathbf{B}$ that describe d. These sets are called *ideals*.

Definition 1.11: [Ideal] For finitary basis $\mathbf{B}$, a subset $\mathcal{I}$ of B is an *ideal over* $\mathbf{B}$ iff

- $\mathcal{I}$ is downward closed: $e \in \mathcal{I} \;\Rightarrow\; (\forall b \in B \; b \sqsubseteq e \;\Rightarrow\; b \in \mathcal{I})$

- $\mathcal{I}$ is closed under least upper bounds on finite subsets (conjunction).

We construct *domains* as partially ordered sets of ideals.

Definition 1.12: [Constructed Domain] Let $\mathbf{B}$ be a finitary basis. The *domain* $\mathcal{D}_{\mathbf{B}}$ *determined* by $\mathbf{B}$ is the partial order $\langle D, \sqsubseteq_D \rangle$ where D is the set of all ideals $\mathcal{I}$ over $\mathbf{B}$ and $\sqsubseteq_D$ is the subset relation. We will frequently write $\mathcal{D}$ instead of $\mathcal{D}_{\mathbf{B}}$.

The proof of the following two claims are easy; they are left to the reader.

Claim 1.13: The least upper bound of two ideals $\mathcal{I}_1$ and $\mathcal{I}_2$, if it exists, is found by closing $\mathcal{I}_1 \cup \mathcal{I}_2$ to form an ideal over $\mathbf{B}$.

Claim 1.14: The domain $\mathcal{D}$ determined by a finitary basis $\mathbf{B}$ is a *complete partial order*.

Each proposition b in a finitary basis B determines an ideal consisting of the set of propositions implied by b. An ideal of this form called a *principal ideal* of B.

Definition 1.15: [Principal Ideals] For finitary basis $\mathbf{B} = \langle B, \sqsubseteq \rangle$, the *principal ideal* determined by $b \in B$, is the ideal $\mathcal{I}_b$ such that

$$\mathcal{I}_b = \{b' \in B \mid b' \sqsubseteq b\} \; .$$

We will use the notation $\mathcal{I}_b$ to denote the principal ideal determined by b throughout the monograph.

Since there is a natural one-to-one correspondence between the propositions of a finitary basis $\mathbf{B}$ and the principal ideals over $\mathbf{B}$, the following theorem obviously holds.

Theorem 1.16: The principal ideals over a finitary basis $\mathbf{B}$ form a finitary basis under the subset ordering.

Within the domain $\mathcal{D}$ determined by a finitary basis $\mathbf{B}$, the principal ideals are characterized by an important topological property called *finiteness*.

Definition 1.17: [Finite Elements] An element e of a *cpo* $\mathcal{D} = \langle D, \sqsubseteq \rangle$ is *finite*[2] iff for every directed subset S of D, $e = \bigsqcup S$ implies $e \in S$. The set of finite elements in a *cpo* $\mathcal{D}$ is denoted $\mathcal{D}^0$.

[2]This property is weaker in general than the corresponding property (called *isolated* or *compact*) that is widely used in topology. In the context of *cpos*, the two properties are equivalent.

The proof of the following theorem is left to the reader.

Theorem 1.18: An element of the domain $\mathcal{D}$ of ideals determined by a finitary basis $\mathbf{B}$ is finite iff it is principal.

In $\mathcal{D}$, the principal ideal determined by the least proposition $\perp$ is the set $\{\perp\}$. This ideal is the least element in the domain (viewed as a *cpo*). In contexts where there is no confusion, we will abuse notation and denote this ideal by the symbol $\perp$ instead of $\mathcal{I}_\perp$.

The next theorem identifies the relationship between an ideal and all the principal ideals that approximate it.

Theorem 1.19: Let $\mathcal{D}$ be the domain determined by a finitary basis $\mathbf{B}$. For any $\mathcal{I} \in \mathcal{D}$, $\mathcal{I} = \bigsqcup \{\mathcal{I}' \in \mathcal{D}^0 \mid \mathcal{I}' \sqsubseteq \mathcal{I}\}$.

Proof See Exercise 9. □

The approximation ordering in a partial order allows us to differentiate partial elements from total elements.

Definition 1.20: [Partial and Total Elements] Let $\mathbf{B}$ be a partial order. An element $b \in B$ is *partial* iff there exists an element $b' \in B$ such that $b \neq b'$ and $b \sqsubseteq b'$. An element $b \in B$ is *total* iff for all $b' \in B$, $b \sqsubseteq b'$ implies $b = b'$.

Example 1.21: The domain determined by the finitary basis defined in Example 1.9 consists only of elements for each proposition in the basis. The four total elements are the principal ideals for the propositions $00, 01, 10$, and 11. In general, a finite basis determines a domain with this property. □

Example 1.22: The domain determined by the basis defined in Example 1.10 contains total elements for each of the natural numbers. These elements are the principal ideals for propositions of the form (n, n). In this case as well, there are no ideals formed that are not principal. □

Example 1.23: Let $\Sigma = \{0, 1\}$ and $\Sigma^\star$ be the set of all finite strings over Σ with ϵ denoting the empty string. $\Sigma^\star$ forms a finitary basis under the prefix ordering on strings. ϵ is the least element in the $\Sigma^\star$. The domain $\mathcal{S}$ determined by $\Sigma^\star$ contains principal ideals for all the finite bitstrings. In addition, $\mathcal{S}$ contains nonprincipal ideals corresponding to all infinite bitstrings. Given any infinite bitstring s, the corresponding ideal $\mathcal{I}_s$ is the set of all finite prefixes of s. In fact, these prefixes form a chain.[3] □

If we view *cpo*s abstractly, the names we associate with particular elements in the universe are unimportant. Consequently, we introduce the notion of *isomorphism*: two domains are *isomorphic* iff they have exactly the same structure.

Definition 1.24: [Isomorphic Partial Orders] Two partial orders $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic*, denoted $\mathbf{A} \approx \mathbf{B}$, iff there exists a one-to-one onto function $m : A \to B$ that preserves the approximation ordering:

$$\forall a, b \in A \; a \sqsubseteq_{\mathbf{A}} b \iff m(a) \sqsubseteq_{\mathbf{B}} m(b)\,.$$

[3] Nonprincipal ideals in other domains are not necessarily chains. Strings are a special case because finite elements can "grow" in only one direction. In contrast, the ideals corresponding to infinite trees—other than vines—are not chains.

Theorem 1.25: Let $\mathcal{D}$ be the domain determined by a finitary basis $\mathbf{B}$. $\mathcal{D}^0$ forms a finitary basis $\mathbf{B}'$ under the approximation ordering $\sqsubseteq$ (restricted to $\mathcal{D}^0$). Moreover, the domain $\mathcal{E}$ determined by the finitary basis $\mathbf{B}'$ is isomorphic to $\mathcal{D}$.

Proof Since the finite elements of $\mathcal{D}$ are precisely the principal ideals, it is easy to see that $\mathbf{B}'$ is isomorphic to $\mathbf{B}$. Hence, $\mathbf{B}'$ is a finitary basis and $\mathcal{E}$ is isomorphic to $\mathcal{D}$. The isomorphism between $\mathcal{D}$ and $\mathcal{E}$ is given by the mapping $\delta : \mathcal{D} \to \mathcal{E}$ is defined by the equation

$$\delta(d) = \{e \in \mathcal{D}^0 \mid e \sqsubseteq d\} \, .$$

□

The preceding theorem justifies the following comprehensive definition for domains.

Definition 1.26: [Domain] A *cpo* $\mathcal{D} = \langle D, \sqsubseteq \rangle$ is a *domain* iff

- $\mathcal{D}^0$ forms a finitary basis under the approximation ordering $\sqsubseteq$ restricted to $\mathcal{D}^0$, and
- $\mathcal{D}$ is isomorphic to the domain $\mathcal{E}$ determined by $\mathcal{D}^0$.

In other words, a domain is a partial order that is isomorphic to a constructed domain.

To conclude this section, we state some closure properties on $\mathcal{D}$ to provide more intuition about the approximation ordering.

Theorem 1.27: Let $\mathcal{D}$ the the domain determined by a finitary basis $\mathbf{B}$. For any subset S of $\mathcal{D}$, the following properties hold:

1. $\bigcap S \in \mathcal{D}$ and $\bigcap S = \bigsqcap S$.
2. if S is directed, then $\bigcup S \in \mathcal{D}$ and $\bigcup S = \bigsqcup S$.

Proof The conditions for ideals specified in Definition 1.11 must be satisfied for these properties to hold. The intersection case is trivial. The union case requires the stated restriction since ideals require closure under lubs. □

For the remainder of this monograph, we will ignore the distinction between principal ideals and the corresponding elements of the finitary basis whenever it is convenient.

Exercises

1. Let
$$B = \{s_n \mid s_n = \{m \in \mathbb{N} \mid m \geq n\}, \; n \in \mathbb{N}\}$$
What is the approximation ordering for B? Verify that B is a finitary basis. What are the total elements of the domain determined by B. Draw a partial picture demonstrating the approximation ordering in the basis.

2. Example 1.9 can be generalized to allow strings of any finite length. Give the finitary basis for this general case. What is the approximation ordering? What does the domain look like? What are the total elements in the domain? Draw a partial picture of the approximation ordering.

3. Let B be all finite subsets of $\mathbb{N}$ with the subset relation as the approximation relation. Verify that this is a finitary basis. What is the domain determined by B. What are the total elements? Draw a partial picture of the domain.

4. Construct two non-isomorphic infinite domains in which all elements are finite but there are no infinite chains of elements ($(\langle x_n \rangle_{n=0}^{\infty}$ with $x_n \sqsubseteq x_{n+1}$ but $x_n \neq x_{n+1}$ for all n).

5. Let B be the set of all non-empty open intervals on the real line with rational endpoints plus a "bottom" element. What would a reasonable approximation ordering be? Verify that B is a finitary basis. For any real number t, show that

$$\{x \in B \mid t \in x\} \cup \{\bot\}$$

is an ideal element. Is it a total element? What are the total elements? (Hint: When t is rational consider all intervals with t as a right-hand end point.)

6. Let $\mathbf{D}$ be a finitary basis for domain $\mathcal{D}$. Define a new basis, $\mathbf{D}' = \{\downarrow X \mid X \in \mathbf{D}\}$ where $\downarrow X = \{Y \in \mathbf{D} \mid X \sqsubseteq Y\}$. Show that $\mathbf{D}'$ is a finitary basis and that $\mathbf{D}$ and $\mathbf{D}'$ are isomorphic.

7. Let $\langle B, \sqsubseteq \rangle$ be a finitary basis where

$$B = \{X_0, X_1, \ldots, X_n, \ldots\}.$$

Suppose that consistency of finite sequences of elements is decidable. Let

$$\begin{aligned} Y_0 &= X_0 \\ Y_{n+1} &= \begin{cases} X_{n+1} & \text{if } X_{n+1} \text{ is consistent with } Y_0, Y_1, \ldots, Y_n \\ Y_n & \text{otherwise}. \end{cases} \end{aligned}$$

Show that $\{Y_0, \ldots, Y_n, \ldots\}$ is a total element in the domain determined by B. (Hint: Show that $Y_0, \ldots, Y_{n-1}$ is consistent for all n.) Show that all ideals can be determined by such sequences.

8. Devise a finitary basis $\mathbf{B}$ with more than two elements such that every pair of elements in B is consistent, but B is not consistent.

9. Prove Theorem 1.19.

2 Operations on Data

Since program operations perform computations *incrementally* on data values that correspond to sets of approximations (ideals), they obey some critical *topological* constraints. For any approximation x' to the input value x, a program operation f must produce the output $f(x')$. Since program output cannot be withdrawn, every program operation f is a *monotonic* function: $x_1 \sqsubseteq x_2$ implies $f(x_1) \sqsubseteq (x_2)$.

We can decribe this process in more detail by examining the structure of computations. Recall that every value in a domain D can be interpreted as a set of finite elements in D that is closed under implication. When an operation f is applied to the input value x, f gathers information about x by asking the program computing x to generate a countable chain of finite elements C where $\bigsqcup \{\mathcal{I}_c \mid c \in C\} = x$. For the sake of simplicity, we can force the chain C describing the input value x to be infinite: if C is finite, we can convert it to an equivalent infinite chain by repeating the last element. Then we can view f as a function on infinite streams that repeatedly "reads" the next element in an infinite chain C and "writes" the next element element in an infinite chain C' where $\bigsqcup \{\mathcal{I}_{c'} \mid c' \in C'\} = f(x)$. Any such function f on D is clearly monotonic. In addition, f

obeys the stronger property that for any directed set S, $f(\bigsqcup S) = \bigsqcup\{f(s) \mid s \in S\}$. This property is called *continuity*.

This formulation of computable operations as functions on streams of finite elements is concrete and intuitive, but it is not canonical. There are many different functions on streams of finite elements corresponding to the same continuous function f over a domain D. For this reason, we will use a slightly different model of incremental computation as the basis for defining the form of operations on domains.

To produce a canonical representation for computable operations, we must represent values as ideals rather than chains of finite elements. In addition, we must perform computations in parallel, producing finite answers incrementally in non-deterministic order. It is important to emphasize that the result of every computation, which is an ideal $\mathcal{I}$, is still deterministic; only the order in which the elements of $\mathcal{I}$ are enumerated is non-deterministic. When an operation f is applied to the input value x, f gathers information about x by asking x to enumerate the ideal of finite elements $I_x = \{d \in \mathcal{D}^0 \mid d \sqsubseteq x\}$. In response to each input approximation $d \sqsubseteq x$, f enumerates the ideal $I_{f(d)} = \{e \in \mathcal{D}^0 \mid e \sqsubseteq f(d)\}$. Since $I_{f(d)}$ may be infinite, each enumeration is an independent computation. The operation f merges all of these enumerations yielding an enumeration of the ideal $I_{f(x)} = \{e \in \mathcal{D}^0 \mid e \sqsubseteq f(x)\}$.

A computable operation f mapping $\mathcal{A}$ into $\mathcal{B}$ can be formalized as a consistent relation $F \subseteq \mathbf{A} \times \mathbf{B}$ such that

- the image $F(a) = \{b \in \mathbf{B} | a\, F\, b\}$ of any input element $a \in \mathbf{A}$ is an ideal
- F is monotonic: $a \sqsubseteq a' \;\Rightarrow\; F(a) \subseteq F(a')$.

These closure properties ensure that the relation F uniquely identifies a continuous function f on D. Relations satisfying these closure properties are called *approximable mappings.*

The following set of definitions restates the preceding descriptions in more rigorous form.

Definition 2.1: [Approximable Mapping] Let $\mathcal{A}$ and $\mathcal{B}$ be the domains determined by finitary bases $\mathbf{A}$ and $\mathbf{B}$, respectively. An *approximable mapping* $F \subseteq \mathbf{A} \times \mathbf{B}$ is a binary relation over $\mathbf{A} \times \mathbf{B}$ such that

1. $\bot_A\, F\, \bot_B$
2. If $a\, F\, b$ and $a\, F\, b'$ then $a\, F\, (b \sqcup b')$
3. If $a\, F\, b$ and $b' \sqsubseteq_B b$, then $a\, F\, b'$
4. If $a\, F\, b$ and $a \sqsubseteq_A a'$, then $a'\, F\, b$

The partial order of approximable mappings $F \subseteq \mathbf{A} \times \mathbf{B}$ under the subset relation is denoted by the expression $\mathsf{Map}(\mathbf{A}, \mathbf{B})$.

Conditions 1, 2, and 3 force the image of an input ideal to be an ideal. Condition 4 states that the function on ideals associated with F is monotonic.

Definition 2.2: [Continuous Function] Let $\mathcal{A}$ and $\mathcal{B}$ be the domains determined by finitary bases $\mathbf{A}$ and $\mathbf{B}$, respectively. A function $f : \mathcal{A} \to \mathcal{B}$ is *continuous* iff for any ideal $\mathcal{I}$ in $\mathcal{A}$, $f(\mathcal{I}) = \bigsqcup\{f(\mathcal{I}_a) \mid a \in \mathcal{I}\}$. The partial ordering $\sqsubseteq_B$ from $\mathcal{B}$ determines a partial ordering $\sqsubseteq$ on continuous functions:

$$f \sqsubseteq g \iff \forall x \in \mathcal{A}\; f(x) \sqsubseteq_{\mathcal{A}} g(x)\,.$$

The partial order consisting of the continuous functions from $\mathcal{A}$ to $\mathcal{B}$ under the pointwise ordering is denoted by the expression $\mathsf{Fun}(A, B)$.

It is easy to show that continuous functions satisfy a stronger condition than the definition given above.

Theorem 2.3: If a function $f : \mathcal{A} \to \mathcal{B}$ is continuous, then for every directed subset S of $\mathcal{A}$, $f(\bigsqcup S) = \bigsqcup \{f(\mathcal{I}) \mid \mathcal{I} \in S\}$.

Proof By Theorem 1.27, $\bigsqcup S$ is simply $\bigcup S$. Since f is continuous, $f(\bigsqcup S) = \bigsqcup \{f(\mathcal{I}_a) \mid \exists \mathcal{I} \in S \; a \in \mathcal{I}\}$. Similarly, for every $\mathcal{I} \in \mathcal{A}$, $f(\mathcal{I}) = \bigsqcup \{f(\mathcal{I}_a) \mid a \in \mathcal{I}\}$. Hence, $\bigsqcup \{f(\mathcal{I}) \mid \mathcal{I} \in S\} = \bigsqcup \{\bigsqcup \{f(\mathcal{I}_a) \mid a \in \mathcal{I}\} \mid \mathcal{I} \in S\} = \bigsqcup \{f(\mathcal{I}_a) \mid \exists \mathcal{I} \in S \; a \in \mathcal{I}\}$. □

Every approximable mapping F over the finitary bases $\mathbf{A} \times \mathbf{B}$ determines a continuous function $f : \mathcal{A} \to \mathcal{B}$. Similarly, every continuous function $f : \mathcal{A} \to \mathcal{B}$ determines a an approximable mapping F over the finitary bases $\mathbf{A} \times \mathbf{B}$.

Definition 2.4: [Image of Approximable Mapping] For approximable mapping

$$F \subseteq \mathbf{A} \times \mathbf{B}$$

the *image* of $d \in \mathcal{A}$ under F (denoted $\mathsf{apply}(F, d)$) is the ideal $\{b \in \mathbf{B} \mid \exists a \in \mathbf{A},\; a \in d,\; \wedge a\, F\, b\}$. The *function* $f : \mathcal{A} \to \mathcal{B}$ *determined by* F is defined by the equation:

$$f(d) = \mathsf{apply}(F, d)\,.$$

Remark 2.5: It is easy to confirm that $\mathsf{apply}(F, d)$ is an element of $\mathcal{B}$ and that the function f is continuous. Given any ideal $d \in \mathcal{A}$, $\mathsf{apply}(F, d)$ is the subset of $\mathbf{B}$ consisting of all the elements related by F to finite elements in d. The set $\mathsf{apply}(F, d)$ is an ideal in $\mathcal{B}$ since (i) the set $\{b \in \mathbf{B} | a\, F\, b\}$ is downward closed for all $a \in \mathbf{A}$, and (ii) $a\, F\, b \wedge a\, F\, b'$ implies $a\, F\, (b \sqcup b')$. The continuity of f is an immediate consequence of the definition of f and the definition of continuity.

The following theorem establishes that the partial order of approximable mappings over $\mathbf{A} \times \mathbf{B}$ is isomorphic to the partial order of continuous functions in $\mathcal{A} \to \mathcal{B}$.

Theorem 2.6: Let $\mathbf{A}$ and $\mathbf{B}$ be finitary bases. The partial order $\mathsf{Map}(\mathbf{A}, \mathbf{B})$ consisting of the set of approximable mappings over $\mathbf{A}$ and $\mathbf{B}$ is isomorphic to the partial order $\mathcal{A} \to_c \mathcal{B}$ of continuous functions mapping $\mathcal{A}$ into $\mathcal{B}$. The isomorphism is produced by the function $\mathbb{F} : \mathsf{Map}(\mathbf{A}, \mathbf{B}) \to (\mathcal{A} \to_c \mathcal{B})$ defined by

$$\mathbb{F}(F) = f$$

where f is the function defined by the equation

$$f(d) = \mathsf{apply}(F, d)$$

for all $d \in \mathcal{A}$.

Proof The theorem is an immediate consequence of the following lemma. □

Lemma 2.7:

1. For any approximable mappings $F, G \subseteq \mathbf{A} \times \mathbf{B}$
   (a) $\forall a \in \mathbf{A}, b \in \mathbf{B} \;\; a\, F\, b \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(F)(\mathcal{I}_a)$.
   (b) $F \subseteq G \iff \forall a \in \mathbf{A} \;\; \mathbb{F}(F)(a) \sqsubseteq \mathbb{F}(G)(a)$

2. The function $\mathbb{F} : \mathsf{Map}(\mathbf{A}, \mathbf{B}) \rightarrow (\mathcal{A} \rightarrow_c \mathcal{B})$ is one-to-one and onto.

Proof (lemma)

1. Part (a) is the immediate consequence of the definition of f ($b \in f(\mathcal{I}_a) \iff a\,F\,b$) and the fact that $f(\mathcal{I}_a)$ is downward closed. Part (b) follows directly from Part (a): $F \subseteq G \iff \forall a \in \mathcal{A}\ \{b \mid a\,F\,b\} \subseteq \{b \mid a\,G\,b\}$. But the latter holds iff $\forall a \in \mathbf{A}\ \ (f(a) \subseteq g(a) \iff f(a) \sqsubseteq g(a))$.

2. Assume $\mathbb{F}$ is not one-to-one. Then there are distinct approximable mappings F and G such that $\mathbb{F}(F) = \mathbb{F}(G)$. Since $\mathbb{F}(F) = \mathbb{F}(G)$,

$$\forall a \in \mathbf{A}, b \in \mathbf{B}\ \ (\mathcal{I}_b \sqsubseteq \mathbb{F}(F)(\mathcal{I}_a) \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(G)(\mathcal{I}_a))\ .$$

By Part 1 of the lemma,

$$\forall a \in \mathbf{A}, b \in \mathbf{B}\ \ (a\,F\,b \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(F)(\mathcal{I}_a) \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(G)(\mathcal{I}_a) \iff a\,G\,b)\ .$$

We can prove that $\mathbb{F}$ is *onto* as follows. Let f be an arbitary continuous function in $\mathcal{A} \rightarrow \mathcal{B}$. Define the relation $F \subseteq \mathbf{A} \times \mathbf{B}$ by the rule

$$a\,F\,b \iff \mathcal{I}_b \sqsubseteq f(\mathcal{I}_a)\ .$$

It is easy to verify that F is an approximable mapping. By Part 1 of the lemma,

$$a\,F\,b \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(F)(\mathcal{I}_a)\ .$$

Hence

$$\mathcal{I}_b \sqsubseteq f(\mathcal{I}_a) \iff \mathcal{I}_b \sqsubseteq \mathbb{F}(F)(\mathcal{I}_a)\ ,$$

implying that f and $\mathbb{F}(F)$ agree on finite inputs. Since f and $\mathbb{F}(F)$ are continuous, they are equal. $\Box$

The following examples of show how approximable mappings and continuous functions are related.

Example 2.8: Let $\mathcal{B}$ be the domain of infinite strings from the previous section and let $\mathcal{T}$ be the truth value domain with two total elements, `true` and `false` where $\bot_{\mathcal{T}}$ denotes that there is insufficient information to determine the outcome. Let $p : \mathcal{B} \rightarrow \mathcal{T}$ be the function defined by the equation:

$$p(x) = \begin{cases} \texttt{true} & \text{if } x = 0^n 1y \text{ where } n \text{ is even} \\ \texttt{false} & \text{if } x = 0^n 1y \text{ where } n \text{ is odd} \\ \bot_{\mathcal{T}} & \text{otherwise} \end{cases}$$

The function p determines whether or not there are an even number of 0's before the first 1 in the string. If there is no one in the string, the result is $\bot_{\mathcal{T}}$. It is easy to show that p is continuous. The corresponding binary relation P is defined by the rule:

$$a\,P\,b \quad \iff \quad b \sqsubseteq_{\mathcal{T}} \bot_{\mathcal{T}}\ \vee\ 0^{2n}1 \sqsubseteq_B a \wedge b \sqsubseteq_{\mathcal{T}} \texttt{true}\ \vee\ 0^{2n+1}1 \sqsubseteq_B a \wedge b \sqsubseteq_{\mathcal{T}} \texttt{false}$$

The reader should verify that P is an approximable mapping and that p is the continuous function determined by P. $\Box$

Example 2.9: Given the domain $\mathcal{B}$ from the previous example, let $g : \mathcal{B} \to \mathcal{B}$ be the function defined by the equation:

$$g(x) = \begin{cases} 0^{n+1}y & \text{if } x = 0^n 1^k 0 y \\ \bot_D & \text{otherwise} \end{cases}$$

The function g eliminates the first substring of the form 1^k $(k > 0)$ from the input string x. If $x = 1^\infty$, the infinite string of ones, then $g(x) = \bot_D$. Similarly, if $x = 0^n 1^\infty$, then $g(x) = \bot_D$. The reader should confirm that g is continuous and determine the approximable mapping G corresponding to g. $\square$

Approximable mappings and continuous functions can be composed and manipulated just like any other relations and functions. In particular, the composition operators for approximable mappings and continuous functions behave as expected. In fact, both the approximable mappings and the continuous functions form a *category*.

Theorem 2.10: The approximable mappings form a *category* over finitary bases where the *identity mapping* for finitary basis B, $\mathsf{I}_B \subseteq \mathbf{B} \times \mathbf{B}$, is defined for $a, b \in \mathbf{B}$ as

$$a \, \mathsf{I}_B \, b \iff b \sqsubseteq a$$

and the composition $G \circ F \subseteq \mathbf{B_1} \times \mathbf{B_3}$ of approximable mappings $F \subseteq \mathbf{B_1} \times \mathbf{B_2}$ and $G \subseteq \mathbf{B_2} \times \mathbf{B_3}$ is defined for $a \in \mathbf{B_1}$ and $c \in \mathbf{B_3}$ by the rule:

$$a \, (G \circ F) \, c \iff \exists b \in \mathbf{B_2} \; a \, F \, b \wedge b \, G \, c \, .$$

To show that this structure is a category, we must establish the following properties:

1. the identity mappings are approximable mappings,
2. the identity mappings composed with an approximable mapping defines the original mapping,
3. the mappings formed by composition are approximable mappings, and
4. function composition is *associative*.

Proof Let $F \subseteq \mathcal{D}_1 \times \mathcal{D}_2$ and $G \subseteq \mathcal{D}_2 \times \mathcal{D}_3$ be approximable mappings. Let $\mathsf{I}_1, \mathsf{I}_2$ be identity mappings for $\mathbf{B_1}$ and $\mathbf{B_2}$ respectively.

1. The verification that the identity mappings satisfy the requirements for approximable mappings is straightforward and left to the reader.
2. To show $F \circ \mathsf{I}_1$ and $\mathsf{I}_2 \circ F$ are approximable mappings, we prove the following equivalence:

   $$F \circ \mathsf{I}_1 = \mathsf{I}_2 \circ F = F$$

   For $a \in \mathbf{B_1}$ and $b \in \mathbf{B_2}$,

   $$a \, (F \circ \mathsf{I}_1) \, b \iff \exists c \in \mathcal{D}_1 \;\; (c \sqsubseteq a \wedge c \, F \, b) \, .$$

   By the definition of approximable mappings, this property holds iff $a \, F \, b$, implying that F and $F \circ \mathsf{I}_1$ are the same relation. The proof of the other half of the equivalence is similar.

3. We must show that the relation $G \circ F$ is approximable given that the relations F and G are approximable. To prove the first condition, we observe that $\bot_1 \, F \, \bot_2$ and $\bot_2 \, G \, \bot_3$ by assumption, implying that $\bot_1 \, (G \circ F) \, \bot_3$. Proving the second condition requires a bit more work. If $a \, (G \circ F) \, c$ and $a \, (G \circ F) \, c'$, then by the definition of composition, $a \, F \, b$ and $b \, G \, c$ for some b and $a \, F \, b'$ and $b' \, G \, c'$ for some b'. Since F and G are approximable mappings, $a \, F \, (b \sqcup b')$ and since $b' \sqsubseteq (b \sqcup b')$, it must be true that $(b \sqcup b') \, G \, c$. By an analogous argument, $(b \sqcup b') \, G \, c'$. Therefore, $(b \sqcup b') \, G \, (c \sqcup c')$ since G is an approximable mapping, implying that $a \, (G \circ F) \, (c \sqcup c')$. The final condition asserts that $G \circ F$ is monotonic. We can prove this as follows. If $a \sqsubseteq a'$, $c' \sqsubseteq c$ and $a \, (G \circ F) \, c$, then $a \, F \, b$ and $b \, G \, c$ for some b. So $a' \, F \, b$ and $b \, G \, c'$ and thus $a' \, (G \circ F) \, c'$. Thus, $G \circ F$ satisfies the conditions of an approximable mapping.

4. Associativity of composition implies that for approximable mapping H with F, G as above and $H : \mathcal{D}_3 \to \mathcal{D}_4$, $H \circ (G \circ F) = (H \circ G) \circ F$. Assume $a \, (H \circ (G \circ F)) \, z$. Then,

$$\begin{array}{rcl} a \, (H \circ (G \circ F)) \, z & \iff & \exists c \in \mathcal{D}_3 \; a \, (G \circ F) \, c \wedge c \, H \, z \\ & \iff & \exists c \in \mathcal{D}_3 \, \exists b \in \mathcal{D}_2 \; a \, F \, b \wedge b \, G \, c \wedge c \, H \, z \\ & \iff & \exists b \in \mathcal{D}_2 \, \exists c \in \mathcal{D}_3 \; a \, F \, b \wedge b \, G \, c \wedge c \, H \, z \\ & \iff & \exists b \in \mathcal{D}_2 \; a \, F \, b \wedge b \, (H \circ G) \, z \\ & \iff & a \, ((H \circ G) \circ F) \, z \end{array}$$

□

Since finitary bases correspond to domains and approximable mappings correspond to continuous functions, we can restate the same theorem in terms of domains and continuous functions.

Corollary 2.11: The continuous functions form a *category* over domains determined by finitary bases; the *identity function* for domain $\mathcal{B}$, $\mathsf{I}_B : \mathcal{B} \to \mathcal{B}$, is defined for by the equation

$$\mathsf{I}_B(d) = d$$

and the composition $g \circ f \in \mathcal{B}_1 \to \mathcal{B}_3$ of continuous functions $f : \mathcal{B}_1 \to \mathcal{B}_2$ and $g : \mathcal{B}_2 \to \mathcal{B}_3$ is defined for $a \in \mathcal{B}_1$ by the equation

$$(g \circ f)(a) = g(f(a)) \, .$$

Proof The corollary is an immediate consequence of the preceding theorem and two facts:

- The partial order of finitary bases is isomorphic to the partial order of domains determined by finitary bases; the ideal completion mapping established the isomorphism.
- The partial order of approximable mappings over $\mathbf{A} \times \mathbf{B}$ is isomorphic to the partial order of continuous functions in $\mathcal{A} \to \mathcal{B}$.

□

Isomorphisms between domains are important. We briefly state and prove one of their most important properties.

Theorem 2.12: Every isomorphism between domains is characterized by an approximable mapping between the finitary bases. Additionally, finite elements are always mapped to finite elements.

Proof Let $f : \mathcal{D} \rightarrow \mathcal{E}$ be a one-to-one and onto function that preserves the approximation ordering. Using the earlier theorem characterizing approximable mappings and their associated functions, we can define the mapping as $a\,F\,b \iff \mathcal{I}_b \sqsubseteq f(\mathcal{I}_a)$ where $\mathcal{I}_a, \mathcal{I}_b$ are the principal ideals for a, b. As shown in Exercise 2.7, monotone functions on finite elements always determine approximable mappings. Thus, we need to show that the function described by this mapping, using the function image construction defined earlier, is indeed the original function f. To show this, the following equivalence must be established for $a \in \mathcal{D}$:

$$f(a) = \{b \in \mathbf{E} \mid \exists a' \in a\, \mathcal{I}_b \sqsubseteq f(\mathcal{I}_{a'})\}$$

The right-hand side of this equation, call it e, is an ideal—for a proof of this, see Exercise 2.10. Since f is an onto function, there must be some $d \in \mathcal{D}$ such that $f(d) = e$. Since $a' \in a$, $\mathcal{I}_{a'} \sqsubseteq a$ holds. Thus, $f(\mathcal{I}_{a'}) \sqsubseteq f(a)$. Since this holds for all $a' \in a$, $f(d) \sqsubseteq f(a)$. Now, since f is an order-preserving function, $d \sqsubseteq a$. In addition, since $a' \in a$, $f(\mathcal{I}_{a'}) \sqsubseteq f(d)$ by the definition of $f(d)$ so $\mathcal{I}_{a'} \sqsubseteq d$. Thus, $a' \in d$ and thus $a \sqsubseteq d$ since a' is an arbitrary element of a. Thus, $a = d$ and $f(a) = f(d)$ as desired.

To show that finite elements are mapped to finite elements, let $\mathcal{I}_a \in \mathcal{D}$ for $a \in \mathbf{D}$. Since f is one-to-one and onto, every $b \in f(\mathcal{I}_a)$ has a unique $\mathcal{I}_{b'} \sqsubseteq \mathcal{I}_a$ such that $f(\mathcal{I}_{b'}) = \mathcal{I}_b$. This element is found using the inverse of f which must exist. Now, let

$$z = \bigsqcup \{\mathcal{I}_{b'} \mid b \in f(\mathcal{I}_a)\}$$

Since $p \sqsubseteq q$ implies $\mathcal{I}_{p'} \sqsubseteq \mathcal{I}_{q'}$, z is also an ideal (see Exercise 2.10 again). Since $\mathcal{I}_{b'} \sqsubseteq \mathcal{I}_a$ holds for each $\mathcal{I}_{b'}$, $z \sqsubseteq \mathcal{I}_a$ must also hold. Also, since each $\mathcal{I}_{b'} \sqsubseteq z$, $\mathcal{I}_b = f(\mathcal{I}_{b'}) \sqsubseteq f(z)$. Therefore, $b \in f(z)$. Since b is an arbitrary element in $f(\mathcal{I}_a)$, $f(\mathcal{I}_a) \sqsubseteq f(z)$ must hold and thus $\mathcal{I}_a \sqsubseteq z$. Therefore, $\mathcal{I}_a = z$ and $a \in z$. But then $a \in \mathcal{I}_{c'}$ for some $c \in f(\mathcal{I}_a)$ by the definition of z. Thus, $\mathcal{I}_a \sqsubseteq \mathcal{I}_{c'}$ and $f(\mathcal{I}_a) \sqsubseteq \mathcal{I}_c$. Since c was chosen such that $\mathcal{I}_c \sqsubseteq f(\mathcal{I}_a)$, $\mathcal{I}_{c'} \sqsubseteq \mathcal{I}_a$ and therefore $\mathcal{I}_c = f(\mathcal{I}_a)$ and $f(\mathcal{I}_a)$ is finite. The same argument holds for the inverse of f; therefore, the isomorphism preserves the finiteness of elements. □

Exercises

Exercise 2.13: Show also that partial order of monotonic functions mapping $\mathcal{D}^0$ to $\mathcal{E}^0$ (using the pointwise ordering) is isomorphic to the partial order of approximable mappings over $f : \mathbf{D} \times \mathbf{E}$.

Exercise 2.14: Prove that, if $F \subseteq \mathbf{D} \times \mathbf{E}$ is an approximable mapping, then the corresponding function $f : \mathcal{D} \rightarrow \mathcal{E}$ satisfies the following equation:

$$f(x) = \bigsqcup \{e \mid \exists d \in x\; dFe\}$$

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

Exercise 2.15: Prove the following claim: if $F, G \subseteq \mathbf{D} \times \mathbf{E}$ are approximable mappings, then there exists $H \subseteq \mathbf{D} \times \mathbf{E}$ such that $H = F \cap G = F \sqcap G$.

Exercise 2.16: Let $\langle I, \leq \rangle$ be a non-empty partial order that is directed and let $\langle D, \sqsubseteq \rangle$ be a finitary basis. Suppose that $a : I \rightarrow \mathcal{D}$ is defined such that $i \leq j \Rightarrow a(i) \sqsubseteq a(j)$ for all $i, j \in I$. Show that

$$\bigcup \{a(i) \mid i \in I\}$$

is an ideal in $\mathcal{D}$. This says that the domain is closed under directed unions. Prove also that for $f : \mathbf{D} \to \mathbf{E}$ an approximable mapping, then for any directed union,

$$f(\bigcup\{a(i) \mid i \in I\}) = \bigcup\{f(a(i)) \mid i \in I\}$$

This says that approximable mappings preserve directed unions. If an elementwise function preserves directed unions, must it come from an approximable mapping? (Hint: see Exercise 2.8).

Exercise 2.17: Let $\langle I, \leq \rangle$ be a directed partial order with $f_i : \mathbf{D} \to \mathbf{E}$ as a family of approximable mappings indexed by $i \in I$. We assume $i \leq j \Rightarrow f_i(x) \sqsubseteq f_j(x)$ for all $i, j \in I$ and all $x \in \mathcal{D}$. Show that there is an approximable mapping $g : \mathbf{D} \to \mathbf{E}$ where

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

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

Exercise 2.18: Let $f : \mathcal{D} \to \mathcal{E}$ be an isomorphism between domains. Let $\phi : \mathbf{D} \to \mathbf{E}$ be the one-to-one correspondence from Theorem 2.6 where

$$f(\mathcal{I}_a) = \mathcal{I}_{\phi(a)}$$

for $a \in \mathbf{D}$. Show that the approximable mapping determined by f is the relationship $\phi(x) \sqsubseteq b$. Show also that if a and a' are consistent in $\mathcal{D}$ then $\phi(a \sqcup a') = \phi(a) \sqcup \phi(a')$. Show how this means that isomorphisms between domains correspond to isomorphisms between the bases for the domains.

Exercise 2.19: Show that the mapping defined in Example 2.9 is approximable. Is it uniquely determined by the following equations or are some missing?

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

Exercise 2.20: Define in words the affect of the approximable mapping $h : \mathbf{B} \to \mathbf{B}$ using the bases defined in Example 2.9 where

$$\begin{aligned} h(0x) &= 00h(x) \\ h(1x) &= 10h(x) \end{aligned}$$

for all $x \in \mathcal{B}$. Is h an isomorphism? Does there exist a map $k : \mathbf{B} \to \mathbf{B}$ such that $k \circ h = \mathsf{I}_B$ and is k a one-to-one function?

Exercise 2.21: Generalize the definition of approximable mappings to define a mapping

$$f : \mathbf{D_1} \times \mathbf{D_2} \to \mathbf{D_3}$$

of two variables. (Hint: f can be a ternary relation $f \subseteq \mathbf{D_1} \times \mathbf{D_2} \times \mathbf{D_3}$ where the relation among the basis elements is denoted $(a, b)\, F\, c$. State a modified version of the theorem characterizing these mappings and their corresponding functions.

Exercise 2.22: Modify the construction of the domain $\mathcal{B}$ from Example 2.8 to construct a domain $\mathcal{C}$ with both finite and infinite total elements ($\mathcal{B} \subseteq \mathcal{C}$). Define an approximable map, C, on this domain corresponding to the concatenation of two strings. (Hint: Use 011 as an finite total element, $011\bot$ as the corresponding finite partial element). Recall that ϵ, the empty sequence, is different from $\bot$, the undefined sequence. Concatenation should be defined such that if x is an infinite element from $\mathcal{C}$, then $\forall y \in \mathbf{C}\ (x, y)\, C\, x$. How does the concatenation behave on partial elements on the left?

Exercise 2.23: Let $\mathbf{A}$ and $\mathbf{B}$ be arbitrary finitary bases. Prove that the partial order of approximable mappings over $\mathbf{A} \times \mathbf{B}$ is a domain. (Hint: the finite elements are the closures of finite consistent relations.) Prove that the partial order of continuous functions in $\mathcal{A} \rightarrow \mathcal{B}$ is a domain.

3 Domain Constructors

Now that the notion of a domain has been defined, we need to develop convenient methods for constructing specific domains. The strategy that we will follow is to define flat domains directly (as term algebras) and construct more complex domains by applying domain constructors to simpler domains. Since domains are completely determined by finitary bases, we will focus on how to construct composite finitary bases from simpler ones. These constructions obviously determine corresponding constructions on domains.

The two most important constructions on finitary bases are Cartesian products of finitary bases and approximable mappings on finitary bases.

Definition 3.1: [Product Basis] Let $\mathbf{D}$ and $\mathbf{E}$ be finitary bases generating domains $\mathcal{D}$ and $\mathcal{E}$. The *product basis*, $\mathbf{D} \times \mathbf{E}$ is the partial order consisting of the universe

$$D \times E = \{[d, e] \mid d \in \mathbf{D}, e \in \mathbf{E}\}$$

and the approximation ordering

$$[d, e] \sqsubseteq [i, j] \iff d \sqsubseteq_D i \text{ and } e \sqsubseteq_E j$$

Theorem 3.2: The product basis of two finitary bases as defined above is a finitary basis.

Proof Let $\mathbf{D}$ and $\mathbf{E}$ be finitary bases and let $\mathbf{D} \times \mathbf{E}$ be defined as above. Since $\mathbf{D}$ and $\mathbf{E}$ are countable, the universe of $\mathbf{D} \times \mathbf{E}$ must be countable. It is easy to show that $\mathbf{D} \times \mathbf{E}$ is a partial order. By the construction, the bottom element of the product basis is $[\bot_D, \bot_E]$. For any finite bounded subset R of $\mathbf{D} \times \mathbf{E}$ where $R = \{[d_i, e_i]\}$, the lub of R is $[\sqcup\{d_i\}, \sqcup\{e_i\}]$ which must be defined since $\mathbf{D}$ and $\mathbf{E}$ are finitary bases and for R to be bounded, each of the component sets must be bounded. $\Box$

It is straightforward to define projection mappings on product bases, corresponding to the standard projection functions defined on Cartesian products of sets.

Definition 3.3: [Projection Mappings] For a finitary basis $\mathbf{D} \times \mathbf{E}$, *projection mappings* $\mathsf{P}_0 \subseteq (\mathbf{D} \times \mathbf{E}) \times \mathbf{D}$ and $P_1 \subseteq (\mathbf{D} \times \mathbf{E}) \times \mathbf{E}$ are the relations defined by the rules

$$\begin{aligned} [d, e]\, P_0\, d' &\iff d' \sqsubseteq_D d \\ [d, e]\, P_1\, e' &\iff e' \sqsubseteq_E e \end{aligned}$$

where d and d' are arbitrary elements of $\mathbf{D}$ and and e and e' are arbitrary elements of $\mathbf{E}$.

Let $\mathbf{A}$, $\mathbf{D}$, and $\mathbf{E}$ be finitary bases and let $F \subseteq \mathbf{A} \times \mathbf{D}$ and $G \subseteq \mathbf{A} \times \mathbf{E}$ be approximable mappings. The *paired mapping* $\langle F, G \rangle \subseteq \mathbf{A} \times (\mathbf{D} \times \mathbf{E})$ is the relation defined by the rule

$$a \, \langle F, G \rangle \, [d, e] \iff a \, F \, d \wedge a \, G \, e$$

for all $a \in \mathbf{A}, d \in \mathbf{D}$, and all $e \in \mathbf{E}$.

It is an easy exercise to show that projection mappings and paired mappings are approximable mappings (as defined in the previous section).

Theorem 3.4: The mappings P_0, P_1, and $\langle F, G \rangle$ are approximable mappings if F, G are. In addition,

1. $\mathsf{P}_0 \circ \langle F, G \rangle = F$ and $\mathsf{P}_1 \circ \langle F, G \rangle = G$.
2. For $[d, e] \in \mathbf{D} \times \mathbf{E}$ and $d' \in \mathbf{D}$, $[d, e] \, \mathsf{P}_0 \, d' \iff d' \sqsubseteq d$.
3. For $[d, e] \in \mathbf{D} \times \mathbf{E}$ and $e' \in \mathbf{E}$, $[d, e] \, \mathsf{P}_1 \, e' \iff e' \sqsubseteq d$.
4. For approximable mapping $H \subseteq \mathbf{A} \times (\mathbf{D} \times \mathbf{E})$, $H = \langle (\mathsf{P}_0 \circ H), (\mathsf{P}_1 \circ H) \rangle$.
5. For $a \in \mathcal{A}$ and $[d, e] \in \mathbf{D} \times \mathbf{E}$, $[a, [d, e]] \in \langle F, G \rangle \iff [a, d] \in F \wedge [a, e] \in G$.

Proof The proof is left as an exercise to the reader. $\square$

The projection mappings and paired mappings on finitary mappings obviously correspond to continuous functions on the corresponding domains. We will denote the continous functions corresponding to P_0 and P_1 by the symbols p_1 and p_2. Similarly, we will denote the function corresponding to the paired mapping $\langle F, G \rangle$ by the expression $\langle f, g \rangle$.

It should be clear that the definition of projection mappings and paired mappings can be generalized to products of more than two domains. This generalization enables us to treat a multi-ary continous function (or approximable mapping) as a special form of a unary continuous function (or approximable mapping) since multi-ary inputs can be treated as single objects in a product domain. Moreover, it is easy to show that a relation $R \subseteq (\mathbf{A_1} \times \ldots \times \mathbf{A_n}) \times \mathbf{B}$ of arity $n+1$ (as in Exercise 2.18) is an approximable mapping iff every restriction of R to a single input argument (by fixing the other input arguments) is an approximable mapping.

Theorem 3.5: A relation $F \subseteq (\mathbf{A} \times \mathbf{B}) \times \mathbf{C}$ is an approximable mapping iff for every $a \in \mathbf{A}$ and every $b \in \mathbf{B}$, the derived relations

$$\begin{aligned} F_{a,*} &= \{[y, z] \mid [[a, y], z] \in F\} \\ F_{*,b} &= \{[x, z] \mid [[x, b], z] \in F\} \end{aligned}$$

are approximable mappings.

Proof Before we prove the theorem, we need to introduce the class of constant relations $\mathsf{K}_e \subseteq \mathbf{D} \times \mathbf{E}$ for arbitrary finitary bases $\mathbf{D}$ and $\mathbf{E}$ and show that they are approximable mappings.

Lemma 3.6: For each $e \in \mathbf{E}$, let the "constant" relation $\mathsf{K}_e \subseteq \mathbf{D} \times \mathbf{E}$ be defined by the equation

$$\mathsf{K}_e = \{[d, e'] \mid d \in \mathbf{D}, e' \sqsubseteq e\}.$$

In other words,

$$d \, \mathsf{K}_e \, e' \iff e' \sqsubseteq e.$$

For $e \in \mathbf{E}$, the constant relation $\mathsf{K}_e \subseteq \mathbf{D} \times \mathbf{E}$ is approximable.

Proof (lemma) The proof of this lemma is left to the reader. □(lemma)

To prove the "if" direction of the theorem , we observe that we can construct the relations $F_{a,*}$ and $F_{*,b}$ for all $a \in \mathbf{A}$ and $b \in \mathcal{B}$ by by composing and pairing primitive approximable mappings. In particular, $F_{a,*}$ is the relation

$$F \circ \langle K_a, \mathsf{I}_B \rangle$$

where I_B denotes the identity relation on $\mathbf{B}$. Similary, $F_{*,b}$ is the relation

$$F \circ \langle \mathsf{I}_A, K_b \rangle$$

where I_A denotes the identity relation on $\mathbf{A}$.

To prove the "only-if" direction, we assume that for all $a \in \mathbf{A}$ and $b \in \mathbf{B}$, the relations $F_{a,*}$ and $F_{*,b}$ are approximable. We must show that the four closure properties for approximable mappings hold for F.

1. Since $F_{\perp_A,*}$ is approximable, $[\perp_B, \perp_C] \in F_{\perp_A,*}$, which implies $[[\perp_A, \perp_B], \perp_C] \in F$.
2. If $[[x,y],z] \in F$ and $[[x,y],z'] \in F$, then $[y,z] \in F_{x,*}$ and $[y,z'] \in F_{x,*}$. Since $F_{x,*}$ is approximable, $[y, z \sqcup z'] \in F_{x,*}$, implying $[[x,y], z \sqcup z'] \in F$.
3. If $[[x,y],z] \in F$ and $z' \sqsubseteq z$, then $[y,z] \in F_{x,*}$. Since $F_{x,*}$ is approximable, $[y,z'] \in F_{x,*}$, implying $[[x,y],z'] \in F$.
4. If $[[x,y],z] \in F$ and $[x,y] \sqsubseteq [x',y']$, then $[y,z] \in F_{x,*}$, $x \sqsubseteq x'$, and $y \sqsubseteq y'$. Since $F_{x,*}$ is approximable, $[y',z] \in F_{x,*}$, implying $[x,y'],z] \in F$, which is equivalent to $[x,z] \in F_{*,y'}$. Since $F_{*,y'}$ is approximable, $[x',z] \in F_{*,y'}$, implying $[[x',y'],z] \in F$.

□

The same result can be restated in terms of continous functions.

Theorem 3.7: A function of two arguments, $f : \mathcal{A} \times \mathcal{B} \to \mathcal{C}$ is continuous iff for every $a \in \mathcal{A}$ and every $b \in \mathcal{B}$, the unary functions

$$x \mapsto f[a,x] \text{ and } y \mapsto f[y,b]$$

are continuous.

Proof Immediate from the previous theorem and the fact that the domain of approximable mappings over $(\mathbf{A} \times \mathbf{B}) \times \mathbf{C}$ is isomorphic to the domain of continuous functions over $(\mathcal{A} \times \mathcal{B}) \times \mathcal{C}$. □

The composition of functions as defined in Theorem 2.10 can be generalized to functions of several arguments. But we need some new syntactic machinery to describe more general forms of function composition.

Definition 3.8: [Cartesian types] Let S be a set of symbols used to denote primitive types. The set S^* of *Cartesian* types over S consists of the set of expressions denoting all finite non-empty Cartesian products over primitive types in S:

$$S^* ::= S \mid S \times S \mid \ldots .$$

A *signature* Σ is a pair $\langle S, O \rangle$ consisting of a set S of type names $\{s_1, \ldots, s_m\}$ used to denote domains and a set O of function symbols $\{o_i^{\rho_i \to \sigma_i} \mid 1 \le i \le m,\ \rho_i \in S^*,\ \sigma_i \in S\}$ used to denote first order functions over the domains S. Let V be a countably infinite set of symbols (variables) $\{v_i^\tau \mid \tau \in S,\ i \in \mathbb{N}\}$ distinct from the symbols in Σ. The *typed expressions* over Σ (denoted $\mathcal{E}\S(\Sigma)$) is the set "typed" terms determined by the following inductive definition:

1. $v_i^\tau \in V$ is a term of type τ,
2. for $M_1^{\tau_1}, \ldots, M_n^{\tau_n} \in \mathcal{E}\S(\Sigma)$ and $o^{(\tau_1 \times \ldots \times \tau_n) \to \tau_0} \in O$ then

$$o^{(\tau_1 \times \ldots \times \tau_n) \to \tau_0}(M_1^{\tau_1}, \ldots, M_n^{\tau_n})^{\tau_0}$$

   is a term of type τ_0.

We will restrict our attention to terms where every instance of a variable v_i has the same type τ. To simplify notation, we will drop the type superscripts from terms whenever they can easily be inferred from context.

Definition 3.9: [Finitary algebra] A *finitary* algebra with signature Σ is a function $\mathbf{A}$ mapping

- each primitive type $\tau \in S$ to a finitary basis $\mathbf{A}[\![\tau]\!]$,
- each operation type $\tau^1 \times \ldots \times \tau^n \in S^*$ to the finitary basis $\mathbf{A}[\![\tau^1]\!] \times \ldots \times \mathbf{A}[\![\tau^n]\!]$,
- each function symbol $o_i^{\rho_i \to \sigma_i} \in O$ to an approximable mapping $\mathbf{A}[\![o_i]\!] \subseteq (\mathbf{A}[\![\rho_i]\!] \times \mathbf{A}[\![\sigma_i]\!])$. Recall that $\mathbf{A}[\![\rho_i]\!]$ is a product basis.

Definition 3.10: [Closed term] A term $M \in \mathcal{E}\S\Sigma$ is *closed* iff it contains no variables in V.

The finitary algebra $\mathbf{A}$ implicitly assigns a meaning to every closed term M in $\mathcal{E}\S(\Sigma)$. This extension is inductively defined by the equation:

$$\mathbf{A}[\![o[M_1, \ldots, M_n]]\!] = \mathbf{A}[\![o]\!][\mathbf{A}[\![M_1]\!], \ldots, \mathbf{A}[\![M_n]\!]] = \{b_0 \mid \exists [b_1, \ldots, b_n] \in \mathbf{A}[\![\rho_i]\!]\ [b_1, \ldots, b_n]\mathbf{A}[\![o]\!]b_0\}\,.$$

We can extend $\mathbf{A}$ to terms M with free variables by implicitly abstracting over all of the free variables in M.

Definition 3.11: [Meaning of terms] Let M be a term in $\mathcal{E}\S\Sigma$ and let $l = x_1^{\tau_1}, \ldots x_n^{\tau_n}$ be a list of distinct variables in V containing all the free variables of M. Let $\mathbf{A}$ be a finitary algebra with signature Σ and for each tuple $[d_1, \ldots, d_n] \in \mathbf{A}[\![\tau_1]\!] \times \ldots \times \mathbf{A}[\![\tau_n]\!]$, let $\mathbf{A}_{\{x_1 := d_1, \ldots, x_n := d_n\}}$ denote the algebra $\mathbf{A}$ extended by defining

$$\mathbf{A}[\![x_i]\!] = d_i$$

for $1 \leq i \leq n$. The meaning of M with respect to l, denoted $\mathbf{A}[\![x_1^{\tau_1}, \ldots, x_n^{\tau_n} \mapsto M]\!]$, is the relation $F_M \subseteq (\mathbf{A}[\![\tau_1]\!] \times \mathbf{A}[\![\tau_n]\!]) \times \mathbf{A}[\![\tau_0]\!]$ defined by the equation:

$$F_M[d_1, \ldots, d_n] = \mathbf{A}_{\{x_1 := d_1, \ldots, x_1 := d_n\}}[\![M]\!]$$

The relation denoted by $\mathbf{A}[\![x_1^{\tau_1}, \ldots, x_n^{\tau_n} \mapsto M]\!]$ is often called a *substitution*. The following theorem shows that the relation $\mathbf{A}[\![x_1^{\tau_1}, \ldots, x_n^{\tau_n} \mapsto M]\!]$ is approximable.

Theorem 3.12: (Closure of continuous functions under substitution) Let M be a term in $\mathcal{E}\S\Sigma$ and let $l = x_1^{\tau_1}, \ldots x_n^{\tau_n}$ be a list of distinct variables in V containing all the free variables of M. Let $\mathbf{A}$ be a finitary algebra with signature Σ. Then the relation F_M denoted by the expression

$$x_1^{\tau_1}, \ldots, x_n^{\tau_n} \mapsto M$$

is approximable.

Proof The proof proceeds by induction on the structure of M. The base cases are easy. If M is a variable x_i, the relation F_M is simply the projection mapping P_i. If M is a constant c of type τ, then F_M is the contant relation K_c of arity n. The induction step is also straightforward. Let M have the form $g[M_1^{\sigma_1}, \ldots, M_m^{\sigma_m}]$. By the induction hypothesis,

$$x_1^{\tau_1}, \ldots, x_n^{\tau_n} \mapsto M_i^{\sigma_i}$$

denotes an approximable mapping $F_{M_i} \subseteq (\mathbf{A}[\![\tau_1]\!] \times \mathbf{A}[\![\tau_n]\!]) \times \mathbf{A}[\![\sigma_i]\!]$. But F_M is simply the composition of the approximable mapping $\mathbf{A}[\![g]\!]$ with the mapping $\langle F_{M_1}, \ldots F_{M_m} \rangle$. Theorem 2.10 tells us that the composition must be approximable. □

The preceding generalization of composition obviously carries over to continuous functions. The details are left to the reader.

The next domain constructor, the function-space constructor, allows approximable mappings (or equivalently continuous functions) to be regarded as objects. In this framework, standard operations on approximable mappings such as *application* and *composition* are approximable mappings. Indeed, the definitions of ideals and of approximable mappings are quite similar. The space of approximable mappings is built by looking at the actions of mappings on finite sets, and then using progressively larger finite sets to construct the mappings in the limit. To this end, the notion of a *finite step mapping* is required.

Definition 3.13: [Finite Step Mapping] Let $\mathbf{A}$ and $\mathbf{B}$ be finitary bases. An approximable mapping $F \subseteq \mathbf{A} \times \mathbf{B}$ is a *finite step mapping* iff there exists finite set $S \subseteq \mathbf{A} \times \mathbf{B}$ and F is the least approximable mapping such that $S \subseteq F$.

It is easy to show that for every *consistent* finite set $S \subseteq \mathbf{A} \times \mathbf{B}$, a least mapping F always exists. F is simply the closure of S under the four conditions that an approximable mapping must satisfy. The least approximable mapping respecting the empty set is the relation $\{\langle a, \bot_B \rangle \mid a \in \mathbf{A}\}$. The space of approximable mappings is built from these finite step mappings.

Definition 3.14: [Partial Order of Finite Step Mappings] For finitary bases $\mathbf{A}$ and $\mathbf{B}$ the *mapping basis* is the partial order $\mathbf{A} \Rightarrow \mathbf{B}$ consisting of

- the universe of all finite step mappings, and
- the approximation ordering

$$F \sqsubseteq G \iff \forall a \in \mathbf{A}\ F(a) \sqsubseteq_B G(a)\,.$$

The following theorem establishes that the constructor $\Rightarrow$ maps finitary bases into finitary bases.

Theorem 3.15: Let $\mathbf{A}$ and $\mathbf{B}$ be finitary bases. Then, the mapping basis $\mathbf{A} \Rightarrow \mathbf{B}$ is a finitary basis.

Proof Since the elements are finite subsets of a countable set, the basis must be countable. It is easy to confirm that $\mathbf{A} \Rightarrow \mathbf{B}$ is a partial order; this task is left to the reader. We must show that every finite consistent subset of $\mathbf{A} \Rightarrow \mathbf{B}$ has a least upper bound in $\mathbf{A} \Rightarrow \mathbf{B}$. Let § be a finite consistent subset of the universe of $\mathbf{A} \Rightarrow \mathbf{B}$. Each element of § is a set of ordered pairs $\langle a, b \rangle$ that meets the approximable mapping closure conditions. Since § is consistent, it has an upper bound $\S' \in \mathbf{A} \Rightarrow \mathbf{B}$. Let $U = \bigcup \S$. Clearly, $U \subseteq \S'$. But U may not be approximable. Let S be the intersection of all relations in $\mathbf{A} \Rightarrow \mathbf{B}$ above §. Clearly $U \subseteq S$, implying S is a superset of every element of §. It is easy to verify that S is approximable, because all the approximable mapping closure conditions are preserved by infinite intersections. □

Definition 3.16: [Domain] We will denote the *domain* of ideals determined by the finitary basis $\mathbf{A} \Rightarrow \mathbf{B}$ by the expression $\mathcal{A} \Rightarrow \mathcal{B}$. The justification for this notation will be explained shortly.

Since the partial order of approximable mappings is isomorphic to the partial order of continuous functions, the preceding definitions and theorems about approximable mappings can be restated in terms of continuous functions.

Definition 3.17: [Finite Step Function] Let $\mathcal{A}$ and $\mathcal{B}$ be the domains determined by the finitary bases $\mathbf{A}$ and $\mathbf{B}$, respectively. A continuous function f in $\mathsf{Fun}(\mathcal{A}, \mathcal{B})$ is *finite* iff there exists a finite step mapping $F \subseteq \mathbf{A} \times \mathbf{B}$ such that f is the function determined by F.

Definition 3.18: [Function Basis] For domains $\mathcal{A}$ and $\mathcal{B}$, the *function basis* is the partial order $(\mathcal{A} \rightarrow_c \mathcal{B})^0$ consisting of

- a universe of all finite step functions, and
- the approximation order

$$f \sqsubseteq g \iff \forall a \in \mathcal{A}\; f\, a \sqsubseteq_B g\, a\,.$$

Corollary 3.19: (to Theorem 3.15) For domains $\mathcal{A}$ and $\mathcal{B}$, the function basis $(\mathcal{A} \rightarrow_c \mathcal{B})^0$ is a finitary basis.

We can prove that the domain constructed by generating the ideals over $\mathbf{A} \Rightarrow \mathbf{B}$ is isomorphic to the partial order $\mathsf{Map}(\mathbf{A}, \mathbf{B})$ of approximable mappings defined in Section 2. This result is not surprising; it merely demonstrates that $\mathsf{Map}(\mathbf{A}, \mathbf{B})$ is a domain and that we have identified the finite elements correctly in defining $\mathbf{A} \Rightarrow \mathbf{B}$.

Theorem 3.20: The domain of ideals determined by $\mathbf{A} \Rightarrow \mathbf{B}$ is isomorphic to the partial order of the approximable mappings $\mathsf{Map}(\mathbf{A}, \mathbf{B})$. Hence, $\mathsf{Map}(\mathbf{A}, \mathbf{B})$ is a domain.

Proof We must establish an isomorphism between the domain determined by $\mathbf{A} \Rightarrow \mathbf{B}$ and the partial order of mappings from $\mathbf{A}$ to $\mathbf{B}$. Let $h : \mathcal{A} \Rightarrow \mathcal{B} \rightarrow \mathsf{Map}(\mathbf{A}, \mathbf{B})$ be the function defined by rule

$$h\, \mathcal{F} = \bigcup \{F \in \mathcal{F}\}\,.$$

It is easy to confirm that the relation on the right hand side of the preceding equation is approximable mapping: if it violated any of the closure properties so would a finite approximation in $\mathcal{F}$. We must prove that the function h is one-to-one and onto. To prove the former, we note that each pair of distinct ideals has a witness $\langle a, b\rangle$ that belongs to a set in one ideal but not in any set in the other. Hence, the images of the two ideals are distinct. The function h is onto because every approximable mapping is the image of the set of finite step maps that approximate it. □

The preceding theorem can be restated in terms of continuous functions.

Corollary 3.21: (to Theorem 3.20) The domain of ideals determined by the finitary basis $(\mathcal{A} \rightarrow_c \mathcal{B})^0$ is isomorphic to the partial order of continuous functions $\mathcal{A} \rightarrow_c \mathcal{B}$. Hence, $\mathcal{A} \rightarrow_c \mathcal{B}$ is a domain.

Now that we have defined the approximable map and continous function domain constructions, we can show that operators on maps and functions introduced in Section 2 are continuous functions.

Theorem 3.22: Given finitary bases, $\mathbf{A}$ and $\mathbf{B}$, there is an approximable mapping

$$\mathit{Apply} : ((\mathbf{A} \Rightarrow \mathbf{B}) \times \mathbf{A}) \times \mathcal{B}$$

such that for all $F : \mathbf{A} \Rightarrow \mathbf{B}$ and $a \in \mathbf{A}$,

$$\mathit{Apply}[F, a] = F\ a\,.$$

Recall that for any approximable mapping $G \subseteq \mathbf{C} \times \mathbf{D}$ and any element $c \in \mathbf{C}$

$$G\,c = \{d \mid c\,G\,d\}\,.$$

Proof For $F \in (\mathbf{A} \Rightarrow \mathbf{B})$, $a \in \mathbf{A}$ and $b \in \mathbf{B}$, define the *Apply* relation as follows:

$$[F, a]\ \mathit{Apply}\ b \iff a\ F\ b\,.$$

It is easy to verify that *Apply* is an approximable mapping; this step is left to the reader. From the definition of *Apply*, we deduce

$$\mathit{Apply}[F, a] = \{b \mid [F, a]\ \mathit{Apply}\ b\} = \{b \mid a\ F\ b\} = F\ a\,.$$

□

This theorem can be restated in terms of continuous functions.

Corollary 3.23: Given domains, $\mathcal{A}$ and $\mathcal{B}$, there is an continuous function

$$\mathit{apply} : ((\mathcal{A} \to_c \mathcal{B}) \times \mathcal{A}) \to_c \mathcal{B}$$

such that for all $f : \mathcal{A} \to_c \mathcal{B}$ and $a \in \mathcal{A}$,

$$\mathit{apply}[f, a] = f\ a\,.$$

Proof of corollary. Let $\mathit{apply} : ((\mathcal{A} \to_c \mathcal{B}) \times \mathcal{A}) \to_c \mathcal{B}$ be the continuous function (on functions rather than relations!) corresponding to *Apply*. From the definition of *apply* and Theorem 2.6 which relates approximable mappings on finitary bases to continous functions over the corresponding domains, we know that

$$\mathit{apply}[f, \mathcal{I}_A] = \{b \in \mathbf{B} \mid \exists F' \in (\mathbf{A} \Rightarrow \mathbf{B})\ ; \exists a \in \mathcal{I}_A \wedge F' \subseteq F \wedge [F', a]\ \mathit{Apply}\ b\}$$

where F denotes the approximable mapping corresponding to f. Since f is the continuous function corresponding to F,

$$f\,\mathcal{I}_A = \{b \in \mathbf{B} \mid \exists a \in \mathcal{I}_A\ a\ F\ b\}$$

So, by the definition of the *Apply* relation, $\mathit{apply}[f, \mathcal{I}_A] \subseteq f\ \mathcal{I}_A$. For every $b \in f\ \mathcal{I}_A$, there exists $a \in \mathcal{I}_A$ such that $a\ F\ b$. Let F' be the least approximable mapping such that $a\ F' b$. By definition, F' is a finite step mapping. Hence $b \in \mathit{apply}[f, \mathcal{I}_A]$, implying $f\ \mathcal{I}_A \subseteq \mathit{apply}[f, \mathcal{I}_A]$. Therefore, $f\ \mathcal{I}_A = \mathit{apply}[f, \mathcal{I}_A]$ for arbitrary $\mathcal{I}_A$. □

The preceding theorem and corollary demonstrate that approximable mappings and continous functions can operate on other approximable mappings or continuous functions just like other data objects. The next theorem shows that the *currying* operation is a continuous function.

Definition 3.24: [The Curry Operator] Let **A**,**B**, and **C** be finitary bases. Given an approximable mapping G in the domain $(\mathbf{A} \times \mathbf{B}) \Rightarrow \mathcal{C}$,

$$Curry_G : \mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{C})$$

is the relation defined by the equation

$$Curry_G\, a = \{F \in \mathbf{B} \Rightarrow \mathbf{C} \mid \forall [b, c] \in F\ [a, b]\, G\, c\}$$

for all $a \in \mathbf{A}$. Similarly, given any continuous function $g : (\mathcal{A} \times \mathcal{B}) \rightarrow_c \mathcal{C}$,

$$curry_g : \mathcal{A} \rightarrow (\mathcal{B} \rightarrow_c \mathcal{C})$$

is the function defined by the equation

$$curry_g[\mathcal{I}_A] = (y \mapsto g[\mathcal{I}_A, y])\,.$$

By theorem 2.7, $(y \mapsto g[\mathcal{I}_A, y])$ is a continous function.

Lemma 3.25: $Curry_G$ is an approximable mapping and $curry_g$ is the continuous function determined by $Curry_G$.

Proof A straightforward exercise.

It is more convenient to discuss the *currying* operation in the context of continuous functions than approximable mappings.

Theorem 3.26: Let $g \in (\mathcal{A} \times \mathcal{B}) \rightarrow_c \mathcal{C}$ and $h \in (\mathcal{A} \rightarrow_c (\mathcal{B} \rightarrow_c \mathcal{C})$. The *curry* operation satisfies the following two equations:

$$\begin{aligned} apply \circ \langle curry_g \circ p_0, p_1 \rangle &= g \\ curry_{apply \circ \langle h \circ p_0, p_1 \rangle} &= h\,. \end{aligned}$$

In addition, the function

$$curry : (\mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}) \rightarrow (\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{C}))$$

defined by the equation

$$curry\, g\, \mathcal{I}_A\, \mathcal{I}_B = curry_g\, \mathcal{I}_A\, \mathcal{I}_B$$

is continous.

Proof Let g be any continuous function in the domain $(\mathcal{A} \times \mathcal{B}) \rightarrow \mathcal{C}$. Recall that

$$curry_g\, a = (y \mapsto g[a, y])\,.$$

Using this definition and the definition of operations in the first equation, we can deduce

$$\begin{aligned} apply \circ \langle curry_g \circ p_0, p_1 \rangle [a, b] &= apply[\langle curry_g \circ p_0, p_1 \rangle [a, b]] \\ &= apply[(curry_g \circ p_0)[a, b], p_1[a, b]] \\ &= apply[curry_g p_0[a, b], b] \\ &= apply[curry_g\, a, b] \\ &= curry_g\, a\, b \\ &= g[a, b]\,. \end{aligned}$$

Hence, the first equation holds.

The second equation follows almost immediately from the first. Define $g' : (\mathcal{A} \times \mathcal{B}) \to_c \mathcal{C}$ by the equation

$$g'[a, b] = h\ a\ b\,.$$

The function g' is defined so that $curry_{g'} = h$. This fact is easy to prove. For $a \in \mathcal{A}$:

$$\begin{aligned} curry_{g'}\ a &= (y \mapsto g'[a, y]) \\ &= (y \mapsto h\ a\ y) \\ &= h\ a\,. \end{aligned}$$

Since $h = curry_{g'}$, the first equation implies that

$$\begin{aligned} apply \circ \langle h \circ p_0, p_1 \rangle &= apply \circ \langle curry_{g'} \circ p_0, p_1 \rangle \\ &= g'\,. \end{aligned}$$

Hence,

$$curry_{apply} \circ \langle h \circ p_0, p_1 \rangle = curry_{g'} = h\,.$$

These two equations show that $(\mathcal{A} \times \mathcal{B}) \to_c \mathcal{C}$ is isomorphic to $(\mathcal{A} \to_c (\mathcal{B} \to \mathcal{C})$ under the *curry* operation. In addition, the definition of *curry* shows that

$$curry\ g \sqsubseteq curry\ g' \iff g \sqsubseteq g'\,.$$

Hence, *curry* is an isomorphism. Moreover, *curry* must be continuous. □

The same theorem can be restated in terms of approximable mappings.

Corollary 3.27: The relation $Curry_G$ satisfies the following two equations:

$$\begin{aligned} Apply \circ \langle Curry_G \circ \P_0, \P_1 \rangle &= G \\ Curry_{Apply \circ \langle G \circ \P_0, \P_1 \rangle} &= G\,. \end{aligned}$$

In addition, the relation

$$Curry : (\mathbf{A} \times \mathbf{B}) \Rightarrow \mathbf{C}) \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{C}))$$

defined by the equation

$$Curry(G) = \{[a, F] \mid a \in \mathbf{A},\ F \in (\mathbf{B} \Rightarrow \mathbf{C}),\ \forall [b, c] \in F\ [a, b]\ G\ c\}$$

is approximable.

Exercises

Exercise 3.28: We assume that there is a countable basis. Thus, the basis elements could without loss of generality be defined in terms of $\{0, 1\}^*$. Show that the product space $\mathbf{A} \times \mathbf{B}$ could be defined as a finitary basis over $\{0, 1\}^*$ such that

$$\mathbf{A} \times \mathbf{B} = \{[0a, 1b] \mid a \in \mathbf{A}, b \in \mathbf{B}\}$$

Give the appropriate definition for the elements in the domain. Also show that there exists an approximable mapping $diag : \mathbf{D} \to \mathbf{D} \times \mathbf{D}$ where $diag\ x = [x, x]$ for all $x \in \mathcal{D}$.

Exercise 3.29: Establish some standard isomorphisms:

1. $\mathbf{A} \times \mathbf{B} \approx \mathbf{B} \times \mathbf{A}$
2. $\mathbf{A} \times (\mathbf{B} \times \mathbf{C}) \approx (\mathbf{A} \times \mathbf{B}) \times \mathbf{C}$
3. $\mathbf{A} \approx \mathbf{A}', \mathbf{B} \approx \mathbf{B}' \Rightarrow \mathbf{A} \times \mathbf{B} \approx \mathbf{A}' \times \mathbf{B}'$

for all finitary bases.

Exercise 3.30: Let $B \subseteq \{0,1\}^*$ be a finitary basis. Define

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

Thus, B^∞ contains infinitely many disjoint copies of B. Now let D^∞ be the least family of subsets over $\{0,1\}*$ such that

1. $B^\infty \in D^\infty$
2. if $b \in \mathbf{B}$ and $d \in D^\infty$, then $0X \cup 1Y \in D^\infty$.

Show that, with the superset relation as the approximation ordering, D^∞ is a finitary basis. State any assumptions that must be made. Show then that $D^\infty \approx D \times D^\infty$.

Exercise 3.31: Using the product construction as a guide, generate a definition for the separated sum system $\mathbf{A} + \mathbf{B}$. Show that there are mappings $in_A : \mathbf{A} \to \mathbf{A} + \mathbf{B}$, $in_B : \mathbf{B} \to \mathbf{A} + \mathbf{B}$, $out_A : \mathbf{A} + \mathbf{B} \to \mathbf{A}$, and $out_B : \mathbf{A} + \mathbf{B} \to \mathbf{B}$ such that $out_A \circ in_A = \mathsf{I}_A$ where I_A is the identity function on $\mathbf{A}$. State any necessary assumptions to ensure this function equation is true.

Exercise 3.32: For approximable mappings $f : \mathbf{A} \to \mathbf{A}'$ and $g : \mathbf{B} \to \mathbf{B}'$, show that there exist approximable mappings, $f \times g : \mathbf{A} \times \mathbf{B} \to \mathbf{A}' \times \mathbf{B}'$ and $f + g : \mathbf{A} + \mathbf{B} \to \mathbf{A}' + \mathbf{B}'$ such that

$$(f \times g)[a, b] = [f\ a, g\ b]$$

and thus

$$f \times g = \langle f \circ p_0, g \circ p_1 \rangle$$

Show also that

$$out_A \circ (f + g) \circ in_A = f$$

and

$$out_B \circ (f + g) \circ in_B = g$$

Is $f + g$ uniquely determined by the last two equations?

Exercise 3.33: Prove that the composition operator is an approximable mapping. That is, show that $comp : (\mathbf{B} \to \mathbf{C}) \times (\mathbf{A} \to \mathbf{B}) \to (\mathbf{A} \to \mathbf{C})$ is an approximable mapping where for $f : \mathbf{A} \to \mathbf{B}$ and $g : \mathbf{B} \to \mathbf{C}$, $comp[g, f] = g \circ f$. Show this using the approach used in showing the result for *apply* and *curry*. That is, define the relation and then build the function from *apply*, *curry*, using $\circ$ and paired functions. (Hint: Fill in mappings according to the following sequence of domains).

$$\begin{gathered}
(\mathbf{A} \to \mathbf{B}) \times \mathbf{A} \to \mathbf{B} \\
(\mathbf{B} \to \mathbf{C}) \times ((\mathbf{A} \to \mathbf{B}) \times \mathbf{A}) \to (\mathbf{B} \to \mathbf{C}) \times \mathbf{B} \\
((\mathbf{B} \to \mathbf{C}) \times (\mathbf{A} \to \mathbf{B})) \times \mathbf{A} \to (\mathbf{B} \to \mathbf{C}) \times \mathbf{B} \\
((\mathbf{B} \to \mathbf{C}) \times (\mathbf{A} \to \mathbf{B})) \times \mathbf{A} \to \mathbf{C} \\
(\mathbf{B} \to \mathbf{C}) \times (\mathbf{A} \to \mathbf{B}) \to (\mathbf{A} \to \mathbf{C}).
\end{gathered}$$

This map shows only one possible solution.

Exercise 3.34: Show that for every domain $\mathcal{D}$ there is an approximable mapping

$$cond : \mathbf{T} \times \mathbf{D} \times \mathbf{D} \rightarrow \mathbf{D}$$

called the conditional operator such that

1. $cond[true, a, b] = a$
2. $cond[false, a, b] = b$
3. $cond[\bot_T, a, b] = \bot_D$

and $\mathbf{T} = \{\bot_T, true, false\}$ such that $\bot_T \sqsubseteq true$, $\bot_T \sqsubseteq false$, but $true$ and $false$ are incomparable. (Hint: Define a *cond* relation).

4 Fixed Points and Recursion

Functions can now be constructed by composing basic functions. However, we wish to be able to define functions recursively as well. The technique of recursive definition will also be useful for defining domains. Recursion can be thought of as (possibly infinite) interated composition. The primary result is the following *Fixed Point Theorem*.

Theorem 4.1: For any approximable mapping $f : \mathbf{D} \rightarrow \mathbf{D}$ on any domain, there exists a least element $x \in \mathcal{D}$ such that

$$f(x) = x.$$

Proof Let f^n stand for the function f composed with itself n times. Thus,

$$\begin{array}{lcl} f^0 & = & \mathsf{I}_D \text{ and} \\ f^{n+1} & = & f \circ f^n \end{array}$$

Define

$$x = \{d \in \mathbf{D} \mid \exists n \in \mathbb{N}.\Delta\ f^n\ d\}.$$

To show that $x \in \mathcal{D}$, we must show it to be an ideal. f is an approximable mapping, so $\Delta \in x$ since $\Delta\ f\ \Delta$. For $d \in x$ and $d' \sqsubseteq d$, $d' \in x$ must hold since, for $d \in x$, there must exist an $a \in \mathbf{D}$ such that $a\ f\ d$. But by the definition of an approximable mapping, $a\ f\ d'$ must hold as well so $d' \in x$. Closure under lubs is direct since f must include lubs to be approximable.

To see that $f(x) = x$, note that for any $d \in x$, if $d\ f\ d'$, then $d' \in x$. Thus, $f(x) \sqsubseteq x$. Now, x is constructed to be the least element in $\mathcal{D}$ with this property. To see this is true, let $a \in \mathcal{D}$ such that $f(a) \sqsubseteq a$. We want to show that $x \sqsubseteq a$. Let $d \in x$ be an arbitrary element. Therefore, there exists an n such that $\Delta\ f^n\ d$ and therefore

$$\Delta\ f\ d_1\ f\ d_n\ \ldots\ f\ d_{n-1}\ f\ d.$$

Since $\Delta \in a$, $d_1 \in f(a)$. Thus, since $f(a) \sqsubseteq a$, $d_1 \in a$. Thus, $d_2 \in f(a)$ and therefore $d_2 \in a$. Using induction on n, we can show that $d \in f(a)$. Therefore, $d \in a$ and thus $x \sqsubseteq a$.

Since f is monotonic and $f(x) \sqsubseteq x$, $f(f(x)) \sqsubseteq f(x)$. Since x is the least element with this property, $x \sqsubseteq f(x)$ and thus $x = f(x)$. □

Since the element x above is the least element, it must be unique. Thus we have defined a function mapping the domain $\mathcal{D} \rightarrow \mathcal{D}$ into the domain $\mathcal{D}$. The next step is to show that this mapping is approximable.

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

$$fix : (\mathbf{D} \to \mathbf{D}) \to \mathbf{D}$$

such that if $f : \mathbf{D} \to \mathbf{D}$ is an approximable mapping,

$$fix(f) = f(fix(f))$$

and for $x \in \mathcal{D}$,

$$f(x) \sqsubseteq x \;\Rightarrow\; fix(f) \sqsubseteq x$$

This property implies that fix is unique. The function fix is characterized by the equation

$$fix(f) = \bigcup_{n=0}^{\infty} f^n(\bot)$$

for all $f : \mathbf{D} \to \mathbf{D}$.

Proof The final equation can be simplified to

$$fix(f) = \{d \in \mathbf{D} \mid \exists n \in \mathbb{N}.\Delta\ f^n\ d\}$$

which is the equation used in the previous theorem to define the fixed point. Using the formula from Exercise 2.8 on the above definition for fix yields the following equation to be shown:

$$fix(f) = \bigcup \{fix(\mathcal{I}_F) \mid \exists F \in (\mathbf{D} \to \mathbf{D}).F \in f\}$$

where $\mathcal{I}_F$ denotes the ideal for F in $\mathbf{D} \to \mathbf{D}$.

From its definition, fix is monotonic since, if $f \sqsubseteq g$, then $fix(f) \sqsubseteq fix(g)$ since $f^n \sqsubseteq g^n$. Since $F \in f$, $\mathcal{I}_F \sqsubseteq f$ and since fix is monotonic, $fix(\mathcal{I}_F) \sqsubseteq fix(f)$.

Let $x \in fix(f)$. Thus, there is a finite sequence of elements such that $\Delta\ f\ x_1\ f\ \ldots\ f\ x'\ f\ x$. Define F as the basis element encompassing the step functions required for this sequence. Clearly, $F \in f$. In addition, this same sequence exists in $fix(\mathcal{I}_F)$ since we constructed F to contain it, and thus, $x \in fix(\mathcal{I}_F)$ and $fix(f) \sqsubseteq fix(\mathcal{I}_F)$. The equality is therefore established.

The first equality is direct from the Fixed Point Theorem since the same definition is used. Assume $f(x) \sqsubseteq x$ for some $x \in \mathcal{D}$. Since $\Delta \in x$, $x \neq \emptyset$. Since f is an approximable mapping, for $x' \in x$ and $x'\,f\,y$, $y \in x$ must hold. By induction, for any $\Delta\,f\,y$, $y \in x$ must hold. Thus, $fix(f) \sqsubseteq x$.

To see that the operator is unique, define another operator fax that satisfies the first two equations. It can easily be shown that

$$\begin{array}{rcl} fix(f) & \sqsubseteq & fax(f) \text{ and} \\ fax(f) & \sqsubseteq & fix(f) \end{array}$$

Thus the two operators are the same. □

Recursion has played a part already in these definitions. Recall that f^n was defined for all $n \in \mathbb{N}$. More complex examples of recursion are given below.

Example 4.3: Define a basis $\mathbf{N} = \langle N, \sqsubseteq_N \rangle$ where

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

and the approximation ordering is the superset relation. This generates a flat domain with $\bot = \{\{\mathbb{N}\}\}$ and the total elements being in a one-to-one correspondence with the natural numbers.

Using the construction outlined in Exercise 3.30, construct the basis $F = N^{\infty}$. Its domain is the domain of partial functions over the natural numbers. To see this, let Φ be the set of all finite partial functions $\varphi \subseteq \mathbb{N} \times \mathbb{N}$. Define

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

Consider the finitary basis $\langle F', \sqsubseteq'_F \rangle$ where

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

and the approximation order is the superset relation. The reader should satisfy himself that F' and F are isomorphic and that the elements are the partial functions. The total elements are the total functions over the natural numbers.

The domains $\mathcal{F}$ and $(\mathcal{N} \to \mathcal{N})$ are not isomorphic. However, the following mapping val : $F \times \mathbf{N} \to \mathbf{N}$ can be defined as follows:

$$(\uparrow \varphi, \{n\})\ val\ \{m\} \iff (n, m) \in \varphi$$

and

$$(\uparrow \varphi, \mathbb{N})\ val\ \mathbb{N}$$

Define also as the ideal for $m \in \mathcal{N}$,

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

It is easy to show then that for $\pi \in \mathcal{F}$ and $n \in \mathcal{N}$ we have

$$\begin{aligned} val(\pi, \hat{n}) &= \hat{\pi(n)} \quad \text{if } \pi(n) \neq \bot \\ &= \bot \quad \text{otherwise} \end{aligned}$$

Thus,

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

is a one-to-one function on elements. (The problem is that $(\mathbf{N} \to \mathbf{N})$ has more elements than $\mathbf{F}$ does as the reader should verify for himself).

Now, what about mappings $f : \mathbf{F} \to \mathbf{F}$? Consider the function

$$\begin{aligned} f(\pi)(n) &= 0 & \text{if } n = 0 \\ &= \pi(n-1) + n - 1 & \text{for } n > 0 \end{aligned}$$

If π is a total function, $f(\pi)$ is a total function. If $\pi(k)$ is undefined, then $f(\pi)(k+1)$ is undefined. The function f is approximable since it is completely determined by its actions on partial functions. That is

$$f(\pi) = \bigcup \{f(\varphi) \mid \exists \varphi \in \Phi. \varphi \subseteq \pi\}$$

The Fixed Point Theorem defines a least fixed point for any approximable mapping. Let $\sigma = f(\sigma)$. Now, $\sigma(0) = 0$ and

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

By induction, $\sigma(n) = \sum_{i=0}^{n} i$ and therefore, σ is a total function. Thus, f has a unique fixed point.

Now, in looking at $(\mathbf{N} \to \mathbf{N})$, we have $\hat{0} \in \mathcal{N}$ (The symbols n and $\hat{n}$ will no longer be distinguished, but the usage should be clear from context.). Now define the two mappings, $succ, pred : \mathbf{N} \to \mathbf{N}$ as approximable mappings such that

$$\begin{array}{lcl} n\; succ\; m & \iff & \exists p \in \mathbb{N}. n \sqsubseteq p, m \sqsubseteq p+1 \\ n\; pred\; m & \iff & \exists p+1 \in \mathbb{N}. n \sqsubseteq p+1, m \sqsubseteq p \end{array}$$

In more familiar terms, the same functions are defined as

$$\begin{array}{rcll} succ(n) & = & n+1 & \\ pred(n) & = & n-1 & \text{if } n > 0 \\ & = & \perp & \text{if } n = 0 \end{array}$$

The mapping $zero : \mathbf{N} \to \mathbf{T}$ is also defined such that

$$\begin{array}{rcll} zero(n) & = & true & \text{if } n = 0 \\ & = & false & \text{if } n > 0 \end{array}$$

where $\mathcal{T}$ is the domain of truth value defined in an earlier section. The *structured domain* $\langle N, 0, succ, pred, zero \rangle$ is called "The Domain of the Integers" in the present context. The function element σ defined as the fixed point of the mapping f can now be defined directly as a mapping $\sigma : \mathbf{N} \to \mathbf{N}$ as follows:

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

where the function $+$ must be suitably defined. Recall that *cond* was defined earlier as part of the structure of the domain $\mathcal{T}$. This equation is called a *functional equation*; the next section will give another notation, the $\lambda - calculus$ for writing such equations. □

Example 4.4: The domain $\mathcal{B}$ defined in Example **??** contained only infinite elements as total elements. A related domain, $\mathcal{C}$ defined in Exercise 2.20, can be regarded as a generalization on $\mathcal{N}$. To demonstrate this, the structured domain corresponding to the domain of integers must be presented. The total elements in $\mathcal{C}$ are denoted σ while the partial elements are denoted $\sigma\perp$ for any $\sigma \in \{0, 1\}*$.

The empty sequence ϵ assumes the role of the number 0 in $\mathcal{N}$. Two approximable mappings can serve as the successor function: $x \mapsto 0x$ denoted $succ_0$ and $x \mapsto 1x$ denoted $succ_1$. The predecessor function is filled by the *tail* mapping defined as follows:

$$\begin{array}{rcll} tail(0x) & = & x, & \\ tail(1x) & = & x & \text{and} \\ tail(\epsilon) & = & \perp. & \end{array}$$

The *zero* predicate is defined using the *empty* mapping defined as follows:

$$\begin{array}{rcll} empty(0x) & = & false, & \\ empty(1x) & = & false & \text{and} \\ empty(\epsilon) & = & true. & \end{array}$$

To distinguish the other types of elements in $\mathcal{C}$, the following mappings are also defined:

$$\begin{array}{rcll} zero(0x) & = & true, & \\ zero(1x) & = & false & \text{and} \\ zero(\epsilon) & = & false. & \\ one(0x) & = & false, & \\ one(1x) & = & true & \text{and} \\ one(\epsilon) & = & false. & \end{array}$$

The reader should verify the conditions for an approximable mapping are met by these functions.

An element of $\mathcal{C}$ can be defined using a fixed point equation. For example, the total element representing an infinite sequence of alternating zeroes and ones is defined by the fixed point of the equation

$$a = 01a.$$

This same element is defined with the equation

$$a = 0101a.$$

Is the element defined as

$$b = 010b$$

the same as the previous two? Approximable mappings in $\mathcal{C} \to \mathcal{C}$ can also be defined using equations. For example, the mapping

$$\begin{array}{lcl} d(\epsilon) & = & \epsilon, \\ d(0x) & = & 00d(x) \quad \text{and} \\ d(1x) & = & 11d(x) \end{array}$$

can be characterized with the functional equation

$$d(x) = cond(empty(x), \epsilon, cond(zero(x), succ_0(succ_0(d(tail(x)))), succ_1(succ_1(d(tail(x))))))$$

The concatenation function of Exercise 2.20 over $\mathcal{C} \times \mathcal{C} \to \mathcal{C}$ can be defined with the functional equation

$$C(x, y) = cond(empty(x), y, cond(zero(x), succ_0(C(tail(x), y)), succ_1(C(tail(x), y))))$$

The reader should verify that this definition is consistent with the properties required in the exercise.

These definitions all use *recursion*. They rely on the object being defined for a base case (ϵ for example) or on earlier values ($tail(x)$ for example). These equations characterize the object being defined, but unless a theorem is proved to show that a solution to the equation exists, the definition is meaningless. However, the Fixed Point Theorem for domains was established earlier in this section. Thus, solutions exist to these equations provided that the variables in the equation range over domains and any other functions appearing in the equation are known to be continuous (that is approximable).

To illustrate one use of the Fixed Point Theorem as well as show the use of recursion in a more familiar setting, we will show that all second order models of Peano's axioms are isomorphic. Recall that

Definition 4.5: [Model for Peano's Axiom] A structured set $\langle \mathbb{N}, 0, succ \rangle$ for $0 \in \mathbb{N}$ and $succ : \mathbb{N} \times \mathbb{N}$ is a *model for Peano's axioms* if all the following conditions are satisfied:

1. $\forall n \in \mathbb{N}. 0 \neq succ(n)$
2. $\forall n, m \in \mathbb{N}. succ(n) = succ(m) \ \Rightarrow \ n = m$
3. $\forall x \subseteq \mathbb{N}. 0 \in x \wedge succ(x) \subseteq x \ \Rightarrow \ x = \mathbb{N}$

where $succ(x) = \{succ(n) \mid n \in x\}$. The final clause is usually referred to as the principle of mathematical induction.

Theorem 4.6: All second order models of Peano's axioms are isomorphic.

Proof Let $\langle N, 0, +\rangle$ and $\langle M, \bullet, \#\rangle$ be models for Peano's axioms. Let $N \times M$ be the cartesian product of the two sets and let $\mathcal{P}(N \times M)$ be the powerset of $N \times M$. Recall from Exercise **??** that the powerset can be viewed as a domain with the subset relation as the approximation order. Define the following mapping:

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

The reader should verify that this mapping is approximable. Since it is indeed approximable, a fixed point exists for the function. Let r be the least fixed point:

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

But r defines a binary relation which establishes the isomorphism. To see that r is an isomorphism, the one-to-one and onto features must be established. By construction,

1. $0\, r\, \bullet$ and
2. $n\, r\, m \;\Rightarrow\; +(n)\, r\, \#(m)$.

Now, the sets $\{(0, \bullet)\}$ and $\{(+(n), \#(m)) \,|\, (n, m) \in r\}$ are disjoint by the first axiom. Therefore, 0 corresponds to only one element in m. Let $x \subseteq N$ be the set of all elements of N that correspond to only one element in m. Clearly, $0 \in x$. Now, for some $y \in x$ let $z \in M$ be the element in M that y uniquely corresponds to (that is $y\, r\, z$). But this means that $+(y)\, r \#(z)$ by the construction of the relation. If there exists $w \in M$ such that $+(y)\, r\, w$ and since $(+(y), w) \neq (0, \bullet)$, the fixed point equation implies that $(+(y) = +(n_0))$ and $(w = \#(m_0))$ for some $(n_0, m_0) \in r$. But then by the second axiom, $y = n_0$ and since $y \in x$, $z = m_0$. Thus, $\#(z)$ is the unique element corresponding to $+(y)$. The third axiom can now be applied, and thus every element in N corresponds to a unique element in M. The roles of N and M can be reversed in this proof. Therefore, it can also be shown that every element of M corresponds to a unique element in N. Thus, r is a one-to-one and onto correspondence. $\square$

Exercises

Exercise 4.7: In Theorem 4.2, an equation was given to find the least fixed point of a function $f : \mathcal{D} \to \mathcal{D}$. Suppose that for $a \in \mathcal{D}$, $a \sqsubseteq f(a)$. Will the fixed point $x = f(x)$ be such that $a \sqsubseteq x$? (Hint: How do we know that $\bigsqcup_{n=0}^{\infty} f^n(a) \in \mathcal{D}$?)

Exercise 4.8: Let $f : \mathcal{D} \to \mathcal{D}$ and $S \subseteq \mathcal{D}$ satisfy

1. $\bot \in S$
2. $x \in S \;\Rightarrow\; f(x) \in S$
3. $[\forall n. \{x_n\} \subseteq S \wedge x_n \sqsubseteq x_{n+1}] \;\Rightarrow\; \bigsqcup_{n=0}^{\infty} x_n \in S$

Conclude that $fix(f) \in S$. This is sometimes called the principle of fixed point induction. Apply this method to the set

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

where $a, b : \mathcal{D} \to \mathcal{D}$ are approximable, $a(\bot) = b(\bot)$, 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 $f : \mathcal{D} \to \mathcal{D}$,

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

Prove also 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 the domain $\mathcal{D}_a$ where

$$\mathcal{D}_a = \{x \in \mathcal{D} \mid x \sqsubseteq a\}$$

Show that if $f : \mathcal{D} \to \mathcal{D}$ is approximable, then f can be restricted to another approximable map $f' : \mathcal{D}_{fix(f)} \to \mathcal{D}_{fix(f)}$ where $\forall x \in \mathcal{D}_{fix(f)}.f'(x) = f(x)$ How many fixed points does f' have in $\mathcal{D}_{fix(f)}$?

Exercise 4.11: The mapping **fix** can be viewed as assigning a fixed point operator to any domain $\mathcal{D}$. Show that **fix** can be uniquely characterized by the following conditions on an assignment $\mathcal{D} \rightsquigarrow F_D$:

1. $F_D : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$
2. $F_D(f) = f(F_D(f))$ for all $f : \mathcal{D} \to \mathcal{D}$
3. when $f_0 : \mathcal{D}_0 \to \mathcal{D}_0$ and $f_1 : \mathcal{D}_1 \to \mathcal{D}_1$ are given and $h : \mathcal{D}_0 \to \mathcal{D}_1$ is such that $h(\bot) = \bot$ and $h \circ f_0 = f_1 \circ h$, then
$$h(F_{D_0}(f_0)) = F_{D_1}(f_1).$$

Hint: Apply Exercise 4.7 to show **fix** satisfies the conditions. For the other direction, apply Exercise 4.10.

Exercise 4.12: Must an approximable function have a maximum fixed point? Give an example of an approximable function that has many fixed points.

Exercise 4.13: Must a monotone function $f : \mathcal{P}(A) \to \mathcal{P}(A)$ have a maximum fixed point? (Recall $\mathcal{P}(A)$ is the powerset of the set A).

Exercise 4.14: Verify the assertions made in the first example of this section.

Exercise 4.15: Verify the assertions made in the second example, in particular those in the discussion of "Peano's Axioms". Show that the predicate function $one : \mathcal{C} \to \mathcal{T}$ could be defined using a fixed point equation from the other functions in the structure.

Exercise 4.16: Prove that

$$fix(f \circ g) = f(fix(g \circ f))$$

for approximable functions $f, g : \mathcal{D} \to \mathcal{D}$.

Exercise 4.17: Show that the less-than-or-equal-to relation $l \subseteq \mathbb{N} \times \mathbb{N}$ is uniquely determined by

$$l = \{(n, n) \mid n \in \mathbb{N}\} \cup \{(n, succ(m) \mid (n, m) \in l\}$$

for the structure called the "Domain of Integers".

Exercise 4.18: Let N^* be a structured set satisfying only the first two of the axioms referred to as "Peano's". Must there be a subset $S \subseteq N^*$ such that all three axioms are satisfied? (Hint: Use a least fixed point from $\mathcal{P}(N^*)$).

Exercise 4.19: Let $f : \mathcal{D} \to \mathcal{D}$ be an approximable map. Let $a_n : \mathcal{D} \to \mathcal{D}$ be a sequence of approximable maps such that

1. $a_0(x) = \bot$ for all $x \in \mathcal{D}$
2. $a_n \sqsubseteq a_{n+1}$ for all $n \in \mathbb{N}$
3. $\bigcup_{n=0}^{\infty} a_n = \mathsf{I}_D$ in $\mathcal{D} \to \mathcal{D}$
4. $a_{n+1} \circ f = a_{n+1} \circ f \circ a_n$ for all $n \in \mathbb{N}$

Show 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 $n \in \mathbb{N}$. Show this by induction on n.)

5 Typed λ-Calculus

As shown in the previous section, functions can be characterized by recursion equations which combine previously defined functions with the function being defined. The expression of these functions is simplified in this section by introducing a notation for specifying a function without having to give the function a name. The notation used is that of the *typed* λ-Calculus; a function is defined using a λ-*abstraction*.

An informal characterization of the λ-calculus suffices for this section; more formal descriptions are available elsewhere in the literature [1]. Thus, examples are used to introduce the notation.

An infinite number of variables, x,y,z,... of various types are required. While a variable has a certain type, type subscripts will not be used due to the notational complexity. A distinction must also be made between type symbols and domains. The domain $\mathcal{A} \times \mathcal{B}$ does not uniquely determine the component domains $\mathcal{A}$ and $\mathcal{B}$ even though these domains are uniquely determined by the symbol for the domain. The domain is the *meaning* that we attribute to the symbol.

In addition to variables, constants are also present. For example, the symbol 0 is used to represent the zero element from the domain $\mathcal{N}$. Another constant, present in each domain by virtue of Theorem 4.2, is $fix^{\mathcal{D}}$, the least fixed point operator for domain $\mathcal{D}$ of type $(\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$. The constants and variables are the atomic (non-compound) terms. Types can be associated with all atomic terms.

There are several constructions for compound terms. First, given $\tau, \ldots, \sigma$, a list of terms, the *ordered tuple*

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

is a compound term. If the types of $\tau, \ldots, \sigma$ are $\mathcal{A}, \ldots, \mathcal{B}$, the type of the tuple is $\mathcal{A} \times \ldots \times \mathcal{B}$ since the tuple is to be an element of this domain. The tuple notation for combining functions given earlier should be disregarded here.

The next construction is function application. If the term τ has type $\mathcal{A} \to \mathcal{B}$ and the term σ has the type $\mathcal{A}$, then the compound term

$$\tau(\sigma)$$

has type $\mathcal{B}$. Function application denotes the value of a function at a given input. The notation $\tau(\sigma_0, \ldots, \sigma_n)$ abbreviates $\tau(\langle\sigma_0, \ldots, \sigma_n\rangle)$. Functions applied to tuples allows us to represent applications of multi-variate functions.

The λ-abstraction is used to define functions. Let $x_0, \ldots, x_n$ be a list of distinct variables of type $\mathcal{D}_0, \ldots, \mathcal{D}_n$. Let τ be a term of some type $\mathcal{D}_{n+1}$. τ can be thought of as a function of $n+1$ variables with type $(\mathcal{D}_0 \times \ldots \times \mathcal{D}_n) \to \mathcal{D}_{n+1}$. The name for this function is written

$$\lambda x_0, \ldots, x_n.\tau$$

This expression denotes the entire function. To look at some familiar functions in the new notation, consider

$$\lambda x, y.x$$

This notation is read "lambda ex wye (pause) ex". If the types of x and y are $\mathcal{A}$ and $\mathcal{B}$ respectively, the function has type $(\mathcal{A} \times \mathcal{B}) \to \mathcal{A}$. This function is the first projection function p_0. This function and the second projection function can be defined by the following equations:

$$\begin{aligned} p_0 &= \lambda x, y.x \\ p_1 &= \lambda x, y.y \end{aligned}$$

Recalling the function tuple notation introduced in an earlier section, the following equation holds:

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

which defines a function of type $\mathcal{D}_1 \to (\mathcal{D}_2 \times \mathcal{D}_3)$.

Other familiar functions are defined by the following equations:

$$\begin{aligned} eval &= \lambda f, x.f(x) \\ curry &= = \lambda g \lambda x \lambda y.g(x, y) \end{aligned}$$

The *curry* example shows that this notation can be iterated. A distinction is thus made between the terms $\lambda x, y.x$ and $\lambda x \lambda y.x$ which have the types $\mathcal{D}_0 \times \mathcal{D}_1 \to \mathcal{D}_0$ and $\mathcal{D}_0 \to \mathcal{D}_1 \to \mathcal{D}_0$ respectively. Thus, the following equation also holds:

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

which relates the multi-variate form to the iterated or *curried* form. Another true equation is

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

where fix has type $(\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ and $\mathbf{fix}$ has 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 notation can now be used to define functions using recursion equations. For example, the function σ in Example 4.3 can be characterized by the following equation:

$$\sigma = fix(\lambda f \lambda n.cond(zero(n), 0, f(pred(n)) + pred(n))$$

which states that σ is the least recursively defined function f whose value at n is $cond(\ldots)$. The variable f occurs in the body of the *cond* expression, but this is just the point of a recursive definition. f is defined in terms of its value on "smaller" input values. The use of the fixed point operator makes the definition explicit by forcing there to be a unique solution to the equation.

In an abstraction $\lambda x, y, z.\tau$, the variables x,y, and z are said to be *bound* in the term τ. Any other variables in τ are said to be *free* variables in τ unless they are bound elsewhere in τ. Bound variables are simply placeholders for values; the particular variable name chosen is irrelevant. Thus, the equation

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

is true provided y is not free in τ. The notation $\tau[y/x]$ specifies the substitution of y for x everywhere x occurs in τ. The notation $\tau[\sigma/x]$ for the substitution of the term σ for the variable x is also legitimate.

To show that these equations with λ–terms are indeed meaningful, the following theorem relating λ–terms and approximable mappings must be proved.

Theorem 5.1: Every typed λ–term defines an approximable function of its free variables.

Proof Induction on the length of the term and its structure will be used in this proof.

Variables Direct since $x \mapsto x$ is an approximable function.

Constants Direct since $x \mapsto k$ is an approximable function for constant k.

Tuples Let $\tau = \langle \sigma_0, \ldots, \sigma_n \rangle$. Since the σ_i terms are less complex, they are approximable functions of their free variables by the induction hypothesis. Using Theorem 3.4 (generalized to the multi-variate case) then, τ which takes tuples as values also defines an approximable function.

Application Let $\tau = \sigma_0(\sigma_1)$. We assume that the types of the terms are appropriately matched. The σ_i terms define approximable functions again by the induction hypothesis. Recalling the earlier equations, the value of τ is the same as the value of $eval(\sigma_0, \sigma_1)$. Since $eval$ is approximable, Theorem 3.7 shows that the term defines an approximable function.

Abstraction Let $\tau = \lambda x.\sigma$. By the induction hypothesis, σ defines a function of its free variables. Let those free variables be of types $\mathcal{D}_0, \ldots, \mathcal{D}_n$ where $\mathcal{D}_n$ is the type of x. Then σ defines an approximable function

$$g : \mathcal{D}_0 \times \ldots \times \mathcal{D}_n \to \mathcal{D}'$$

where $\mathcal{D}'$ is the type of σ. Using Theorem **??**, the function

$$curry(g) : \mathcal{D}_0 \times \ldots \times \mathcal{D}_{n-1} \to (\mathcal{D}_n \to \mathcal{D}')$$

yields an approximable function, but this is just the function defined by τ. The reader can generalize this proof for multiple bound variables in τ.

□

Given this, the equation $\tau = \sigma$ states that the two terms define the same approximable function of their free variables. As an example,

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

provided y is not free in τ since the generation of the approximable function did not depend on the name x but only on its location in τ. Other equations such as these are given in the exercises. The most basic rule is shown below.

Theorem 5.2: For appropriately typed terms, the following equation is true:

$$(\lambda x_0, \ldots, x_1.\tau)(\sigma_0, \ldots, \sigma_{n-1}) = \tau[\sigma_0/x_0, \ldots, \sigma_{n-1}/x_{n-1}]$$

Proof The proof is given for $n = 1$ and proceeds again by induction on the length of the term and the structure of the term.

Variables This means $(\lambda x.x)(\sigma) = \sigma$ must be true which it is.

Constants This requires $(\lambda x.k)(\sigma) = k$ must be true which it is for any constant k.

Tuples Let $\tau = \langle \tau_0, \tau_1 \rangle$. This requires that

$$(\lambda x.\langle \tau_0, \tau_1 \rangle)(\sigma) = \langle \tau_0[\sigma/x], \tau_1[\sigma/x] \rangle$$

must be true. This equation holds since the left-hand side can be transformed using the following true equation:

$$(\lambda x.\langle \tau_0, \tau_1 \rangle)(\sigma) = \langle (\lambda x.\tau_0)(\sigma), (\lambda x.\tau_1)(\sigma) \rangle$$

Then the inductive hypothesis is applied to the τ_i terms.

Applications Let $\tau = \tau_0(\tau_1)$. Then, the result requires that the equation

$$(\lambda x.\tau_0(\tau_1))(\sigma) = \tau_0[\sigma/x](\tau_1[\sigma/x])$$

hold true. To see that this is true, examine the approximable functions for the left-hand side of the equation.

$$\begin{array}{lll}
\tau_0 & \mapsto & \bar{V}, x \to t_0 \\
\tau_1 & \mapsto & \bar{V}, x \to t_1 \\
\sigma & \mapsto & \bar{V} \to s \\
\text{so} & & \\
(\lambda x.\tau_0(\tau_1))(\sigma) & \mapsto & \bar{V} \to [(x \to t_0(t_1))(s)] \\
 & = & \bar{V}, x \to [(x \to t_0)(s)]([(x \to t_1)(s)])
\end{array}$$

From this last term, we use the induction hypothesis. To see why the last step holds, start with the set representing the left-hand side and using the aprroximable mappings for the terms:

$$\begin{array}{ll}
 & (\lambda x.\tau_0(\tau_1))(\sigma) \\
\mapsto & \bar{V} \to [(x \to t_0(t_1))(s)] \\
= & \{b \mid \exists a.a \in s \wedge a\,[x \to t_0(t_1)]\,b\} \\
= & \{b \mid \exists a.a \in s \wedge a\,\{(x, u) \mid v \in x \to t_1) \wedge v\,(x \to t_0)\,u\}\,b\} \\
= & \{b \mid \exists a.a \in s \wedge v \in (x \to t_1)(a) \wedge v\,(x \to t_0)(a)\,b\} \\
= & \{b \mid \exists a, c.a \in s \wedge a\,(x \to t_1)\,v \wedge a\,(x \to t_0)\,c \wedge v\,c\,b\} \\
= & \{b \mid v \in [(x \to t_1)(s)] \wedge c \in (x \to t_0)(s) \wedge v\,c\,b\} \\
= & \{b \mid v \in [(x \to t_1)(s)] \wedge v\,[(x \to t_0)(s)]\,b\} \\
= & [(x \to t_0)(s)]([(x \to t_1)(s)])
\end{array}$$

Abstractions Let $\tau = \lambda y.\tau_0$. The required equation is

$$(\lambda x.\lambda y.\tau_0)(\sigma) = \lambda y.\tau_0[\sigma/x]$$

provided that y is not free in σ. The following true equation applies here:

$$(\lambda x.\lambda y.\tau)(\sigma) = \lambda y.((\lambda x.\tau)(\sigma))$$

To see that this equation holds, let g be a function of $n+2$ free variables defined by τ. By Theorem 5.1, the term $\lambda x.\lambda y.\tau$ defines the function $curry(curry(g))$ of n variables. Call this function h. Thus,

$$h(v)(\sigma)(y) = g(v, \sigma, y)$$

where v is the list of the other free variables. Using a combinator inv which inverts the order of the last two arguments,

$$h(v)(\sigma)(y) = curry(inv(g))(v, y)(\sigma)$$

But, $curry(inv(g))$ is the function defined by $\lambda x.\tau$. Thus, we have shown that

$$(\lambda x.\lambda y.\tau)(\sigma)(y) = (\lambda x.\tau)(\sigma)$$

is a true equation. If y is not free in α and $\alpha(y) = \beta$ is true, then $\alpha = \lambda y.\beta$ must also be true.

□

If τ' is the term $\lambda x, y.\tau$, then $\tau'(x, y)$ is the same as τ. This specifies that x and y are not free in τ. This notation is used in the proof of the following theorem.

Theorem 5.3: The least fixed point of

$$\lambda x, y.\langle \tau(x, y), \sigma(x, y)\rangle$$

is the pair with coordinates $fix(\lambda x.\tau(x, fix(\lambda y.\sigma(x, y))))$ and $fix(\lambda y.\sigma(fix(\lambda x.\tau(x, y)), y))$.

Proof We are thus assuming that x and y are **not** free in τ and σ. The purpose here is to find the least solution to the pair of equations:

$$x = \tau(x, y) \text{ and } y = \sigma(x, y)$$

This generalizes the fixed point equation to two variables. More variables could be included using the same method. Let

$$y_* = fix(\lambda y.\sigma(fix(\lambda x.\tau(x, y)), y))$$

and

$$x_* = fix(\lambda x.\tau(x, y))$$

Then,

$$x_* = \tau(x_*, y_*)$$

and

$$\begin{aligned} y_* &= \sigma(fix(\lambda x.\tau(x, y_*), y_*)) \\ &= \sigma(x_*, y_*). \end{aligned}$$

This shows that the pair $\langle x_*, y_*\rangle$ is one fixed point. Now, let $\langle x_0, y_0\rangle$ be the least solution. (Why must a least solution exist? Hint: Consider a suitable mapping of type $(\mathcal{D}_x \times \mathcal{D}_y) \to (\mathcal{D}_x \times \mathcal{D}_y)$.) Thus, we know that $x_0 = \tau(x_0, y_0)$, $y_0 = \sigma(x_0, y_0)$, and that $x_0 \sqsubseteq x_*$ and $y_0 \sqsubseteq y_*$. But this means that $\tau(x_0, y_0) \sqsubseteq x_0$ and thus $fix(\lambda x.\tau(x, y_0)) \sqsubseteq x_0$ and consequently

$$\sigma(fix(\lambda x.\tau(x, y_0), y_0)) \sqsubseteq \sigma(x_0, y_0) \sqsubseteq y_0$$

By the fixed point definition of y_*, $y_* \sqsubseteq y_0$ must hold as well so $y_0 = y_*$. Thus,

$$x_* = fix(\lambda x.\tau(x, y*)) = fix(\lambda x.\tau(x, y_0)) \sqsubseteq x_0.$$

Thus, $x* = x_0$ must also hold. A similar argument holds for x_0. □

The purpose of the above proof is to demonstrate the use of least fixed points in proofs. The following are also true equations:

$$fix(\lambda x.\tau(x)) = \tau(fix(\lambda x.\tau(x)))$$

and

$$\tau(y) \sqsubseteq y \;\Rightarrow\; fix(\lambda x.\tau(x)) \sqsubseteq y$$

if x is not free in τ. These equations combined with the monotonicity of functions were the methods used in the proof above. Another example is the proof of the following theorem.

Theorem 5.4: Let x,y, and $\tau(x, y)$ be of type $\mathcal{D}$ and let $g : \mathcal{D} \to \mathcal{D}$ be a function. Then the equation

$$\lambda x.fix(\lambda y.\tau(x, y)) = fix(\lambda g.\lambda x.\tau(x, g(x)))$$

holds.

Proof Let f be the function on the left-hand side. Then,

$$f(x) = fix(\lambda y.\tau(x, y)) = \tau(x, f(x))$$

holds using the equations stated above. Therefore,

$$f = \lambda x.\tau(x, f(x))$$

and thus

$$g_0 = fix(\lambda g.\lambda x.\tau(x, g(x))) \sqsubseteq f.$$

By the definition of g_0 we have

$$g_0(x) = \tau(x, g_0(x))$$

for any given x. By the definition of f we find that

$$f(x) = fix(\lambda y.\tau(x, y)) \sqsubseteq g_0(x)$$

must hold for all x. Thus $f \sqsubseteq g_0$ and the equation is true. □

This proof illustrates the use of inclusion and equations between functions. The following principle was used:

$$(\forall x.\tau \sqsubseteq \sigma) \;\Rightarrow\; \lambda x.\tau \sqsubseteq \lambda x.\sigma$$

This is a restatement of the first part of Theorem **??**.

Below is a list of various combinators with their definitions in λ-notation. The meanings of those combinators not previously mentioned should be clear.

$$
\begin{array}{lcl}
p_0 & = & \lambda x, y.x \\
p_1 & = & \lambda x, y.y \\
pair & = & \lambda x.\lambda y.\langle x, y\rangle \\
n-tuple & = & \lambda x_0\lambda \ldots \lambda x_{n-1}.\langle x_0, \ldots, x_{n-1}\rangle \\
diag & = & \lambda x.\langle x, x\rangle \\
funpair & = & \lambda f.\lambda g.\lambda x.\langle f(x), g(x)\rangle \\
proj_i^n & = & \lambda x_0, \ldots, x_{n-1}.x_i \\
inv_{i,j}^n & = & \lambda x_0, \ldots, x_i, \ldots, x_j, \ldots, x_{n-1}.\langle x0, \ldots, x_j, \ldots, x_i, \ldots, x_{n-1}\rangle \\
eval & = & \lambda f, x.f(x) \\
curry & = & \lambda g.\lambda x.\lambda y.g(x, y) \\
comp & = & \lambda f, g.\lambda x.g(f(x)) \\
const & = & \lambda k.\lambda x.k \\
\mathbf{fix} & = & \lambda f.fix(\lambda x.f(x))
\end{array}
$$

These combinators are actually schemes for combinators since no types have been specified and thus the equations are ambiguous. Each scheme generates an infinite number of combinators for all the various types.

One interest in combinators is that they allow expressions without variables—if enough combinators are used. This is useful at times but can be clumsy. However, defining a combinator when the same combination of symbols repeatedly appears is also useful.

There are some familiar combinators that do not appear in the table. Combinators such as *cond*, *pred*, and *succ* cannot be defined in the pure λ-calculus but are instead specific to certain domains. They are thus regarded as primitives. A large number of other functions can be defined using these primitives and the λ-notation, as the following theorem shows.

Theorem 5.5: For every partial recursive function $h : \mathbb{N} \to \mathbb{N}$, there is a λ-term τ of type $\mathcal{N} \to \mathcal{N}$ such that the only constants occurring in τ are *cond*, *succ*, *pred*, *zero*, and 0 and if $h(n) = m$ then $\tau(n) = m$. If $h(n)$ is undefined, then $\tau(n) = \bot$ holds. $\tau(\bot) = \bot$ is also true.

Proof It is convenient in the proof to work with strict functions $f : \mathcal{N}^k \to \mathcal{N}$ such that if any input is $\bot$, the result of the function is $\bot$. The composition of strict functions is easily shown to be strict. It is also easy to see that any partial function $g : \mathbb{N}^k \to \mathbb{N}$ can be extended to a strict approximable function $\bar{g} : \mathcal{N}^k \to \mathcal{N}$ which yields the same values on inputs for which g is defined. Other input values yield $\bot$. We want to show that $\bar{g}$ is definable with a λ-expression.

First we must show that *primitive* recursive functions have λ-definitions. Primitive recursive functions are formed from starting functions using composition and the scheme of primitive recursion. The starting functions are the constant function for zero and the identity and projection functions. These functions, however, must be strict so the term $\lambda x, y.x$ is not sufficient for a projection function. The following device reduces a function to its strict form. Let $\lambda x.cond(zero(x), x, x)$ be a function with x of type $\mathcal{N}$. This is the strict identity function. The strict projection function attempted above can be defined as

$$\lambda x, y.cond(zero(y), x, x)$$

The three variable projection function can be defined as

$$\lambda x, y, z.cond(zero(x), cond(zero(z), y, y), cond(zero(z), y, y))$$

While not very elegant, this device does produce strict functions. Strict functions are closed under substitution and composition. Any substitution of a group of functions into another function can be defined with a λ-term if the functions themselves can be so defined. Thus, we need to show that functions obtained by primitive recursion are definable. Let $f : \mathcal{N} \to \mathcal{N}$, and $g : \mathcal{N}^3 \to \mathcal{N}$ be total functions with $\bar{f}$ and $\bar{g}$ being λ-definable. We obtain the function $h : \mathcal{N}^2 \to \mathcal{N}$ by primitive recursion where

$$\begin{aligned} h(0, m) &= f(m) \\ h(n+1, m) &= g(n, m, h(n, m)) \end{aligned}$$

for all $n, m \in \mathcal{N}$. The λ-term for $\bar{h}$ is

$$fix(\lambda k.\lambda x, y.cond(zero(x), \bar{f}(y), \bar{g}(pred(x), y, k(pred(x), y))))$$

Note that the fixed point operator for the domain $\mathcal{N}^2 \to \mathcal{N}$ was used. The variables x and y are of type $\mathcal{N}$. The *cond* function is used to encode the function requirements. The fixed point function is easily seen to be strict and this function is $\bar{h}$.

Primitive recursive functions are now λ-definable. To obtain *partial* recursive functions, the μ-scheme (the least number operator) is used. Let $f(n, m)$ be a primitive recursive function. Then, define h, a partial function, as $h(m) =$ the least n such that $f(n, m) = 0$. This is written as $h(m) = \mu n.f(n, m) = 0$. Since $\bar{f}$ is λ-definable as has just been shown, let

$$\bar{g} = fix(\lambda g.\lambda x, y.cond(zero(\bar{f}(x, y)), x, g(succ(x), y)))$$

Then, the desired function $\bar{h}$ is defined as $\bar{h} = \lambda y.\bar{g}(0, y)$. It is easy to see that this is a strict function. Note that, if $h(m)$ is defined, clearly $h(m) = \bar{g}(0, m)$ is also defined. If $h(m)$ is undefined, it is also true that $\bar{g}(0, m) = \bot$ due to the fixed point construction but it is less obvious. This argument is left to the reader. □

Theorem 5.5 does not claim that all λ-terms define partial recursive functions although this is also true. Further examples of recursion are found in the exercises.

Exercises

Exercise 5.6: Find the definitions of

$$\lambda x, y.\tau \text{ and } \sigma(x, y)$$

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, and *pair*. Generalize the result to functions of many variables.

Exercise 5.7: The table of combinators 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 application as the only mode of combination?

Exercise 5.8: 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: See Exercise 4.16.) If, in addition, $f(\bot) = g(\bot)$, show that $fix(f) = fix(g)$. Will $fix(f) = fix(f^2)$? What if the assumption is weakened to $f \circ g = g^2 \circ f$?

Exercise 5.9: For any domain $\mathcal{D}$, $\mathcal{D}^\infty$ can be regarded as consisting of bottomless stacks of elements of $\mathcal{D}$. Using this view, define the following combinators with their obvious meaning: $head : \mathcal{D}^\infty \to \mathcal{D}$, $tail : \mathcal{D}^\infty \to \mathcal{D}^\infty$ and $push : \mathcal{D} \times \mathcal{D}^\infty \to \mathcal{D}^\infty$. Using the fixed point theorem, argue that there is a combinator $diag : \mathcal{D} \to \mathcal{D}^\infty$ where for all $x \in \mathcal{D}$, $diag(x) = \langle x \rangle_{n=0}^\infty$. (Hint: Try a recursive definition, such as

$$diag(x) = push(x, diag(x))$$

but be sure to prove that all terms of $diag(x)$ are x.) Also introduce by an appropriate recursive definition a combinator $map : (\mathcal{D} \to \mathcal{D})^\infty \times \mathcal{D} \to \mathcal{D}^\infty$ where for elements of the proper type

$$map(\langle f_n \rangle_{n=0}^\infty, x) = \langle f_n(x) \rangle_{n=0}^\infty$$

Exercise 5.10: For any domain $\mathcal{D}$ introduce, as a least fixed point, a combinator

$$while : (\mathcal{D} \to \mathcal{T}) \times (\mathcal{D} \to \mathcal{D}) \to (\mathcal{D} \to \mathcal{D})$$

by the recursion equation

$$while(p, f)(x) = cond(p(x), while(p, f)(f(x)), x)$$

Prove that

$$while(p, while(p, f)) = while(p, f)$$

Show how $while$ could be used to obtain the least number operator,μ, mentioned in the proof of Theorem 5.5. Generalize this idea to define a combinator

$$find : \mathcal{D}^\infty \times (\mathcal{D} \to \mathcal{T}) \to \mathcal{D}$$

which means "find the first term in the sequence (if any) which satisfies the given predicate".

Exercise 5.11: Prove the existence of a one-one function $num : \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$ such that

$$\begin{aligned} num(0,0) &= 0 \\ num(n, m+1) &= num(n+1, m) \\ num(n+1, 0) &= num(0, n) + 1 \end{aligned}$$

Draw a descriptive picture (an infinite matrix) for the function. Find a closed form for the values if possible. Use the function to prove the isomorphism between $\mathcal{P}(\mathbb{N})$,$\mathcal{P}(\mathbb{N} \times \mathbb{N})$, and $\mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N})$.

Exercise 5.12: Show that there are approximable mappings

$$graph : (\mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})) \to \mathcal{P}(\mathbb{N})$$

and

$$fun : \mathcal{P}(\mathbb{N}) \to (\mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N}))$$

where $fun \circ graph = \lambda f.f$ and $graph \circ fun \sqsubseteq \lambda x.x$. (Hint: Using the notation $[n_0, \ldots, n_k] = num(n_0, [n_1, \ldots, n_k])$, two such combinators can be given by the formulas

$$\begin{aligned} fun(u)(x) &= \{m \mid \exists n_0, \ldots, n_{k-1} \in x.[n_0+1, \ldots, n_{k-1}+1, 0, m] \in u\} \\ graph(f) &= \{[n_0+1, \ldots, n_{k-1}+1, 0, m] \mid m \in f(\{n_0, \ldots, n_{k-1}\})\} \end{aligned}$$

where k is a variable - meaning all finite sequences are to be considered.)

6 Introduction to Domain Equations

As stressed in the introduction, the notion of computation with potentially infinite elements is an integral part of domain theory. The previous sections have defined the notion of functions over domains, as well as a notation for expressing these functions. In addition, the notion of computation through series of approximations has been addressed. This computation is possible since the functions defined have been approximable and thus continuous. This section addresses the construction of more complex domains with infinite elements. The next section looks specifically at the notion of computability with respect to these infinite elements. The last section looks at another approach to domain construction.

New domains have been constructed from existing ones using domain constructors such as the product construction ($\times$), the function space construction ($\rightarrow$) and the sum construction ($+$) of Exercise 3.31. These constructors can be *iterated* similar to the way that function application was iterated to form recursive function definitions. In this way, domains can be characterized using recursion equations, called *domain equations*.

A domain equation represents an isomorphism between the domain as a whole and the combination of domains that comprise it. These recursive domains are frequently termed *reflexive* domains since, as in the following example, the domain contains a copy of itself in its structure.

Example 6.1: Consider the following domain equation:

$$\mathcal{T} = \mathcal{A} + (\mathcal{T} \times \mathcal{T})$$

where $\mathcal{A}$ is a previously defined domain. This domain can be thought of as containing atomic elements from $\mathcal{A}$ or pairs of elements of $\mathcal{T}$. What do the elements of this domain look like? In particular, what are the finite elements of this domain? How is the domain constructed? What is an appropriate approximation ordering for the domain? What do lubs in this domain look like? What is the appropriate notion of consistency? Does this domain even exist? Is there a unique solution to this equation? Each of these questions is examined below.

The domain equation tells us that an element of the domain is either an element from $\mathcal{A}$ or is a pair of "smaller" elements from $\mathcal{T}$. One method of constructing a sum domain is using pairs where some distinguished element denotes what type an element is. Thus, for some $a \in \mathcal{A}$, the pair $\langle \pi, a \rangle$ might represent the element in $\mathcal{T}$ for the given element a. For some $s, t \in \mathcal{T}$, the pair $\langle \langle s, t \rangle, \pi \rangle$ might then represent the element in $\mathcal{T}$ for the pair s, t. Thus, π is the distinguished element, and the location of π in the pair specifies the type of the element. The finite elements are either elements in $\mathcal{T}$ representing the (finite) elements of $\mathcal{A}$ or the pair elements from $\mathcal{T}$ whose components are also finite elements in $\mathcal{T}$.

The question then arises about infinite elements. Are there infinite elements in this domain? Consider the following fixed point equation for some element for $a \in \mathcal{A}$:

$$x = \langle \langle a, x \rangle, \pi \rangle.$$

The fixed point of this equation is the infinite product of the element a. Does this element fit the definition for $\mathcal{T}$? From the informal description of the elements of $\mathcal{T}$ given so far, x does qualify as a member of $\mathcal{T}$.

Now that some intuition has been developed about this domain, a formal construction is required. Let $\langle \mathbf{A}, \sqsubseteq_A \rangle$ be the finitary basis used to generate the domain $\mathcal{A}$. Let π be an object such that $\pi \notin \mathbf{A}$. Define the bottom element of the finitary basis $\mathbf{T}$ as $\Delta_T = \langle \pi, \pi \rangle$. Next, all the elements of $\mathcal{A}$ must be included so define an element in $\mathbf{T}$ for each $a \in \mathbf{A}$ as $\langle \pi, a \rangle$. Finally, pair elements for all elements in $\mathbf{T}$ must exist in $\mathbf{T}$ to complete the construction. The set $\mathbf{T}$ can be defined inductively as the *least* set such that:

1. $\Delta_T \in \mathbf{T}$
2. $\langle \pi, a \rangle \in \mathbf{T}$ whenever $a \in \mathbf{A}$
3. $\langle \langle \Delta_T, s \rangle, \pi \rangle \in \mathbf{T}$ whenever $s \in \mathbf{T}$
4. $\langle \langle t, \Delta_T \rangle, \pi \rangle \in \mathbf{T}$ whenever $t \in \mathbf{T}$

The set can also be characterized by the following fixed point equation:

$$\mathbf{T} = \{\Delta_T\} \cup \{\langle \pi, a \rangle \mid a \in \mathbf{A}\} \cup \{\langle \langle \Delta_T, s \rangle, \pi \rangle \mid s \in \mathbf{T}\} \cup \{\langle \langle t, \Delta_T \rangle, \pi \rangle \mid t \in \mathbf{T}\}.$$

A solution must exist for this equation by the fixed point theorem.

Now that the basis elements have been defined, we must show how to find lubs. We will again use an inductive definition.

1. $\langle \pi, \pi \rangle \sqcup t = t$ for all $t \in \mathbf{T}$
2. For $a, b \in \mathbf{A}$, $\langle \pi, a \rangle \sqcup \langle \pi, b \rangle = \langle \pi, a \sqcup b \rangle$ if $a \sqcup b$ exists in $\mathbf{A}$
3. $\langle \langle s, t \rangle, \pi \rangle \sqcup \langle \langle s', t' \rangle, \pi \rangle = \langle \langle s \sqcup s', t \sqcup t' \rangle, \pi \rangle$ if $s \sqcup s'$ and $t \sqcup t'$ exist in $\mathbf{T}$.
4. The lub $\langle \pi, a \rangle \sqcup \langle \langle s, t \rangle, \pi \rangle$ does not exist.

Next, the notion of consistency needs to be explored. From the definition of lubs given above, the following sets are consistent:

1. The empty set is consistent.
2. Everything is consistent with the bottom element.
3. A set of elements all from the basis $\mathbf{A}$ is consistent in $\mathbf{T}$ if the set of elements is consistent in $\mathbf{A}$.
4. A set of product elements in $\mathbf{T}$ is consistent if the left component elements are consistent and the right component elements are consistent.

These conditions derive from the sum and product nature of the domain.

The approximation ordering in the basis has the following inductive definition:

1. $\Delta_T \sqsubseteq_T s$ for all $s \in \mathbf{T}$
2. $y \sqsubseteq_T u \sqcup \Delta_T$ whenever $y \sqsubseteq_T u$
3. $\langle \pi, a \rangle \sqsubseteq_T \langle \pi, b \rangle$ whenever $a \sqsubseteq_A b$
4. $\langle \langle s, t \rangle, \pi \rangle \sqsubseteq_T \langle \langle u, v \rangle, \pi \rangle$ whenever $s \sqsubseteq_T u$ and $t \sqsubseteq_T v$

The next step is to verify that $\mathbf{T}$ is indeed a finitary basis. The basis is still countable. The approximation is clearly a partial order. The existence of lubs of finite bounded subsets must be verified. The definition of consistency gives us the requirements for a bounded subset. Each of the conditions for consistency are examined inductively since the definitions are all inductive:

1. The lub of the empty set is the bottom element Δ_T.

2. The lub of a set containing the bottom element is the lub of the set without the bottom element which must exist by the induction hypothesis.

3. The lub of a set of elements all from the **A** is the element in **T** for the lub in **A**. This element must exist since **A** is a finitary basis and all elements from **A** have corresponding elements in **T**.

4. The lub of a set of product elements is the pair of the lub of the left components and the lub of the right components. These exist by the induction hypothesis.

Thus, a finitary basis has been created; the domain is formed as always from the basis. The solution to the domain equation has been found since any element in the domain $\mathcal{T}$ is either an element representing an element in $\mathcal{A}$ or is the product of two other elements in $\mathcal{T}$. Similarly, any element found on the left-hand side must also be in the domain $\mathcal{T}$ by the construction. Thus, the domain $\mathcal{T}$ is *identical* to the domain $\mathcal{A} + (\mathcal{T} \times \mathcal{T})$.

To look at the question concerning the existence and uniqueness of the solution to this domain equation, recall the fixed point theorem. This theorem states that a fixed point set exists for any approximable mapping over a domain. In the final section, the concept of a *universal domain* is introduced. A universal domain is a domain which contains all other domains as sub-domains. These sub-domains are, roughly speaking, the image of approximable functions over the universal domain. The domain equation for $\mathcal{T}$ can be viewed as an approximable mapping over the universal domain. As such, the fixed point theorem states that a least fixed point set for the function does exist and is unique. Sub-domains are defined formally below.

Looking again at the informal discussion concerning the elements of the domain $\mathcal{T}$, the infinite element proposed does fit into the formal definition for elements of $\mathcal{T}$. This element is an infinite tree with all left sub-trees containing only the element a. For this infinite element to be computable, it must be the lub of some ascending chain of finite approximations to it. The element x can, in fact, be defined by the following ascending sequence of finite trees:

$$\begin{array}{lcl} x_0 & = & \bot \\ x_{n+1} & = & \langle\langle a, x_n\rangle, \pi\rangle \\ x & = & \bigsqcup_{n=0}^{\infty} x_n \end{array}$$

Thus, using domain equations, a domain has been defined recursively. This domain includes infinite as well as finite elements and allows computation on the infinite elements to proceed using the finite approximations, as with the more conventionally defined domains presented earlier.

The final topic of this section is the notion of a sub-domain. Informally, a sub-domain is a structured part of a larger domain. Earlier, a domain was described as a sub-domain of the universal domain. Thus, the sub-domain starts with a subset of the elements of the larger domain while retaining the approximation ordering, consistency relation and lub relation, suitably restricted to the subset elements.

Definition 6.2: [Sub-Domain] A domain $\langle \mathcal{R}, \sqsubseteq_R \rangle$ is a *sub-domain* of a domain $\langle \mathcal{D}, \sqsubseteq_D \rangle$, denoted $\mathcal{R} \lhd \mathcal{D}$ iff

1. $\mathcal{R} \subseteq \mathcal{D}$ - The elements of $\mathcal{R}$ are a subset of the elements of $\mathcal{D}$.

2. $\bot_R = \bot_D$ - The bottom elements are the same.

3. For $x, y \in \mathcal{R}$, $x \sqsubseteq_R y \iff x \sqsubseteq_D y$ - The approximation ordering for $\mathcal{R}$ is the approximation ordering for $\mathcal{D}$ restricted to elements in $\mathcal{R}$.

4. For $x, y, z \in \mathcal{R}$, $x \sqcup_R y = z$ iff $x \sqcup_D y = z$ - The lub relation for $\mathcal{R}$ is the lub relation for $\mathcal{D}$ restricted to elements in $\mathcal{R}$.

5. $\mathcal{R}$ is a domain.

Equivalently, a sub-domain can be thought of as the image of an approximable function which approximates the identity function (also termed a *projection*). The notion of a sub-domain is used in the final section in the discussions about the universal domain. This mapping between the domains can be formalized as follows:

Theorem 6.3: If $\mathcal{D} \lhd \mathcal{E}$, then there exists a projection pair of approximable mappings $i : \mathcal{D} \to \mathcal{E}$ and $j : \mathcal{E} \to \mathcal{D}$ where $j \circ i = \mathsf{I}_D$ and $i \circ j \sqsubseteq \mathsf{I}_E$ where i and j are determined by the following equations:

$$\begin{array}{rcl} i(x) & = & \{y \in \mathbf{E} \mid \exists z \in x . z \sqsubseteq y\} \\ j(y) & = & \{x \in \mathbf{D} \mid x \in y\} \end{array}$$

for all $x \in \mathcal{D}$ and $y \in \mathcal{E}$.

The proof is left as an exercise.

By the definition of a sub-domain, it should be clear that

$$\mathcal{D}_0 \lhd \mathcal{E} \wedge \mathcal{D}_1 \lhd \mathcal{E} \Rightarrow (\mathcal{D}_0 \lhd \mathcal{D}_1 \iff \mathcal{D}_0 \subseteq \mathcal{D}_1)$$

Using this observation, the sub-domains of a domain can be ordered. Indeed, the following theorem is a consequence of this ordering.

Theorem 6.4: For a given domain $\mathcal{D}$, the set of sub-domains $\{\mathcal{D}_0 \mid \mathcal{D}_0 \lhd \mathcal{D}\}$ form a domain.

The proof proceeds using the ordering relation defined as an approximation ordering and is left as an exercise.

Finally, a converse of Theorem 6.3 can also be established:

Theorem 6.5: For two domains $\mathcal{D}$ and $\mathcal{E}$, if there exists a projection pair $i : \mathcal{D} \to \mathcal{E}$ and $j : \mathcal{E} \to \mathcal{D}$ with $j \circ i = \mathsf{I}_D$ and $i \circ j \sqsubseteq \mathsf{I}_E$, then $\mathcal{D}' \lhd \mathcal{E}$ where $\mathcal{D} \approx \mathcal{D}'$.

Proof We want to show that i maps finite elements to finite elements and that $\mathcal{D}'$ is the image of $\mathcal{D}$ in $\mathcal{E}$.

For some $x \in \mathbf{D}$ with $\mathcal{I}_x$ as the principal ideal of x, we can write

$$i(\mathcal{I}_x) = \sqcup \{\mathcal{I}_y \mid y \in i(\mathcal{I}_x)\}$$

Applying j to both sides we get

$$\mathcal{I}_x = j \circ i(\mathcal{I}_x) = \sqcup \{j(\mathcal{I}_y) \mid y \in i(\mathcal{I}_x)\}$$

since $j \circ i = \mathsf{I}_D$ and j is continuous by assumption. But, since $x \in \mathcal{I}_x$, $x \in j(\mathcal{I}_y)$ for some $y \in i(\mathcal{I}_x)$. This means that

$$\mathcal{I}_x \subseteq j(\mathcal{I}_y)$$

and thus

$$i(\mathcal{I}_x) \subseteq i \circ j(\mathcal{I}_y) \subseteq \mathcal{I}_y$$

Since $\mathcal{I}_y \subseteq i(\mathcal{I}_x)$ must hold by the construction, $i(\mathcal{I}_x) = \mathcal{I}_y$. This proves that finite elements are mapped to finite elements.

Next, consider the value for $i(\perp_D)$. Since $\perp_D \sqsubseteq_D j(\perp_E)$, $i(\perp_D) \sqsubseteq \perp_E$. Thus $i(\perp_D) = \perp_E$. Thus, $\mathcal{D}$ is isomorphic to the image of i in $\mathcal{E}$. We still must show that $\mathcal{D}'$ is a domain. Thus, we need to show that if a lub exists in $\mathcal{E}$ for a finite subset in $\mathcal{D}'$, then the lub is also in $\mathcal{D}'$. Let $y', z' \in \mathbf{D}'$ and $y' \sqcup z' = x' \in \mathbf{E}$. Then, there exists $y, z \in \mathbf{D}$ such that $i(\mathcal{I}_y) = \mathcal{I}_{y'}$ and $i(\mathcal{I}_z) = \mathcal{I}_{z'}$ which implies that $\mathcal{I}_y = j(\mathcal{I}_{y'})$ and $\mathcal{I}_z = j(\mathcal{I}_{z'})$. Since $\mathcal{I}_{y'} \sqsubseteq \mathcal{I}_{x'}$ and $j(\mathcal{I}_{y'}) \sqsubseteq j(\mathcal{I}_{x'})$ by monotonicity, $y \in j(\mathcal{I}_{x'})$ must hold. By the same reasoning, $z \in j(\mathcal{I}_{x'})$. But then $x = y \sqcup z \in j(\mathcal{I}_{x'})$ must also hold and thus $y \sqcup z \in \mathcal{D}$ since the element $j(\mathcal{I}_{x'})$ must be an ideal. But,

$$\begin{array}{lcl} \mathcal{I}_y \sqsubseteq \mathcal{I}_x & \Rightarrow & \mathcal{I}_{y'} \sqsubseteq i(\mathcal{I}_x) \\ \mathcal{I}_z \sqsubseteq \mathcal{I}_x & \Rightarrow & \mathcal{I}_{z'} \sqsubseteq i(\mathcal{I}_x) \end{array}$$

This implies that $y' \sqcup z' = x' \in i(\mathcal{I}_x)$. We already know that $x \in j(\mathcal{I}_{x'})$ so $i(\mathcal{I}_x) \sqsubseteq \mathcal{I}_{x'}$. Thus, $i(\mathcal{I}_x) = \mathcal{I}_{x'}$ and thus, $x' \in \mathcal{D}'$. □

Exercises

Exercise 6.6: Show that there must exist domains satisfying

$$\begin{array}{lcl} \mathcal{A} & = & \mathcal{A} + (\mathcal{A} \times \mathcal{B}) \quad \text{and} \\ \mathcal{B} & = & \mathcal{A} + \mathcal{B} \end{array}$$

Decide what the elements will look like and define $\mathcal{A}$ and $\mathcal{B}$ using simultaneous fixed points.

Exercise 6.7: Prove Theorem 6.4

Exercise 6.8: Prove Theorem 6.3

Exercise 6.9: Show that if $\mathcal{A}$ and $\mathcal{B}$ are finite systems, that

$$\mathcal{D} \trianglelefteq \mathcal{E} \trianglelefteq \mathcal{D} \Rightarrow \mathcal{D} \approx \mathcal{E}$$

where $\mathcal{D} \approx \mathcal{D}'$ and $\mathcal{D}' \triangleleft \mathcal{E}$ is denoted $\mathcal{D}' \trianglelefteq \mathcal{E}$.

7 Computability in Effectively Given Domains

In the previous sections, we gave considerable emphasis to the notion of computation using increasingly accurate approximations of the input and output. This section defines this notion of computability more formally. In Section 5, we found that partial functions over the natural numbers were expressible in the λ-notation. This relationship characterizes computation for a particular domain. To describe computation over domains in general, a broader definition is required.

The way a domain is presented impacts the way computations are performed over it. Indeed, the theorems of recursive function theory rely in part on the normal presentation of the natural numbers. A presentation for a domain is an enumeration of the elements of the domain. The standard presentation of the natural numbers is simply the numbers in ascending order beginning with 0. There are many permutations of the natural numbers, each of which can be considered a presentation. Computation with these non-standard presentations may be impossible; that is a computable function on the standard presentation may be non-computable over a non-standard presentation. Therefore, an *effective presentation for a domain* is defined as a presentation which makes the required information computable.

Information about elements in a domain can be characterized completely by looking at the finite elements and their relationships. Thus a presentation must enumerate the finite elements and allow the consistency and lub relationships on these elements to be computed to allow this style of computation.

The consistency relation and the lub relation depend on each other. For example, if a set of elements is consistent, a lub must exist for the set. Given that a set is consistent, the lub can be found in finite time by just enumerating the elements and checking to see if this element is the lub. However, if the set is inconsistent, the enumeration will not reveal this fact. Thus, the consistency relation must be assumed to be recursive in an effective presentation. Exercise 7.10 provides a description of presentations that should clarify the assumptions made. Formally, a presentation is defined as follows:

Definition 7.1: [Effective Presentation] The *presentation* of a finitary basis $\mathbf{D}$ is a function $\pi : \mathbb{N} \rightarrow \mathbf{D}$ such that $\pi(0) = \Delta_D$ and the range of π is the set of finite elements of $\mathbf{D}$. The definition holds for a domain $\mathcal{D}$ as well.

A presentation π is *effective* iff

1. The consistency relation $(\exists k.\pi_i \sqsubseteq \pi_k \wedge \pi_j \sqsubseteq \pi_k)$ for elements π_i and π_j is recursive[4] over i and j.

2. The lub relation $(\pi_k = \pi_i \sqcup \pi_j)$ is recursive over i, j, and k.

This definition supports our intuition about domains; we have stated that the important information about a domain is the set of finite elements, the ordering and consistency relationships between the elements and the lub relation. Thus, an effective presentation provides, in a suitable (that is computable) form, the basic information about the structure and elements of a domain. A presentation can also be viewed as an enumeration of the elements of the domain with the position of an element in the enumeration given by the index corresponding to the integer input for that element in the presentation function with the 0 element representing $\perp$. This perspective is used in the majority of the proofs.

Now that the presentation of a domain has been formalized, the notion of computability can be formally defined. Thus,

Definition 7.2: [Computable Mappings] Given two domains, $\mathcal{D}$ and $\mathcal{E}$ with effective presentations π_1 and π_2 respectively, an approximable mapping $f : \mathbf{D} \rightarrow \mathbf{E}$ is *computable* iff the relation

$$x_n \, f \, y_m$$

is recursively enumerable in n and m.

By considering the domain $\mathcal{D}$ to be a single element domain, the above definition applies not only to computable functions but also to computable elements. For $d \in \mathcal{D}$ where d is the only element in the domain, the element

$$e = f(d) \in \mathcal{E}$$

defines an element in $\mathcal{E}$. The definition states that e is a computable iff the set

$$\{m \in \mathbb{N} \mid y_m \sqsubseteq e\}$$

[4] Recursive in this context means that the relation is decidable.

is a recursively enumerable set of integers. Clearly if the set of elements approximating another is finite, the set is recursive. The notion of a recursively enumerable set simply requires that all elements approximating the element in question be listed eventually. The computation then proceeds by accepting an enumeration representing the input element and enumerating the elements that approximate the desired output element.

Now that the notions of computability and effective presentations have been formalized, the methods of constructing domains and functions will be addressed.

The proof of the next theorem is trivial and is left to the reader.

Theorem 7.3: The identity map on an effectively given domain is computable. The composition of computable mappings on effectively given domains are also computable.

The following corollary is a consequence of this theorem:

Corollary 7.4: For computable function $f : \mathcal{D} \to \mathcal{E}$ and a computable element $x \in \mathcal{D}$, the element $f(x) \in \mathcal{E}$ is computable.

In addition, the standard domain constructors maintain effective presentations.

Theorem 7.5: For domains $\mathcal{D}_0$ and $\mathcal{D}_1$ with effective presentations, the domains

$$\mathcal{D}_0 + \mathcal{D}_1 \text{ and } \mathcal{D}_0 \times \mathcal{D}_1$$

are also effectively given. In addition, the projection functions are all computable. Finally, if f and g are computable maps, then so are $f + g$ and $f \times g$.

Proof Let $\{X_i \mid i \in \mathbb{N}\}$ be the enumeration of $\mathcal{D}_0$ and $\{Y_i \mid i \in \mathbb{N}\}$ be the enumeration of $\mathcal{D}_1$. Another method of sum construction is to use two distinguishing elements in the first position to specify the element type. Thus, a sum domain can be defined as follows:

$$\mathcal{D}_0 + \mathcal{D}_1 = \{(\Delta_0, \Delta_1)\} \cup \{(0, x) \mid x \in \mathcal{D}_0\} \cup \{(1, y) \mid y \in \mathcal{D}_1\}$$

The enumeration can then be defined as follows for $n \in \mathbb{N}$:

$$\begin{array}{lcl} Z_0 & = & (\Delta_0, \Delta_1) \\ Z_{2n+1} & = & (0, X_n) \\ Z_{2n+2} & = & (1, Y_n) \end{array}$$

The proof that Z_i is an effective presentation is left as an exercise.

For the product construction, the domain appears as follows:

$$\mathcal{D}_0 \times \mathcal{D}_1 = \{(x, y) \mid x \in \mathcal{D}_0, y \in \mathcal{D}_1\}$$

The enumeration can be defined in terms of the 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, $k \in \mathbb{N}$:

$$\begin{array}{lcl} p(r(n, m)) & = & n \\ q(r(n, m)) & = & m \\ r(p(k), q(k)) & = & k \end{array}$$

Thus, r is a one-to-one pairing function (see Exercise 5.11) of which there are several. The functions p and q extract the indices from the result of the pairing function. The enumeration for the product domain is then defined as follows:

$$W_i = (X_{p(i)}, Y_{q(i)})$$

The proof that this is an effective presentation is also left as an exercise.

For the combinators, the relations will be defined in terms of the enumeration indices. For example,

$$\begin{array}{lcl} X_n \, in_0 \, Z_m & \iff & m = 0 \text{ or} \\ & & \exists k. m = 2k+1 \wedge X_k \sqsubseteq X_n \\ W_k \, proj_1 \, Y_m & \iff & Y_m \sqsubseteq Y_{q(k)} \end{array}$$

The reader should verify that these sets are recursively enumerable. For this proof, recall that recursively enumerable sets are closed under conjunction, disjunction, substituting recursive functions, and applying an existential quantifier to the front of a recursive predicate. The proof for the other combinators is left as an exercise. □

Product spaces formalize the notion of computable functions of several variables. Note that the proof of Theorem 3.7 shows that substitution of computable functions of severable variables into other computable functions are still computable. The next step is to show that the function space constructor preserves effectiveness.

Theorem 7.6: For domains $\mathcal{D}_0$ and $\mathcal{D}_1$ with effective presentations, the domain $\mathcal{D}_0 \to \mathcal{D}_1$ also has an effective presentation. The combinators *apply* and *curry* are computable if all input domains are effectively given. The computable elements of the domain $\mathcal{D}_0 \to \mathcal{D}_1$ are the computable maps for $\mathbf{D_0} \to \mathbf{D_1}$.

Proof Let $\mathcal{D}_0 = \{X_i \mid i \in \mathbb{N}\}$ and $\mathcal{D}_1 = \{Y_i \mid i \in \mathbb{N}\}$ be the presentations for the domains. The elements of $\mathbf{D_0} \to \mathbf{D_1}$ are finite step functions which respect the mapping of some subset of $\mathbf{D_0} \times \mathbf{D_1}$. Given the enumeration, each element can be associated with a set

$$\{(X_{n_i}, Y_{m_i}) \mid \exists q. 1 \leq i \leq q\}$$

Thus, there is a finite set of integers pairs that determine the element. Given the definition of consistency from Theorem 3.15 for elements in the function space domain and the decidability of consistency in $\mathcal{D}_0$ and $\mathcal{D}_1$, consistency of any finite set of this form is decidable (tedious but decidable since all elements must be checked with all others, etc). Since consistency is decidable, a systematic enumeration of pair sets which are consistent can be made; this enumeration is simply the enumeration of $\mathcal{D}_0 \to \mathcal{D}_1$. Finding the lub consists of making a finite series of tests to find the element that is the lub, which must exist since the set is consistent and we have closure on lubs of finite consistent subsets. Finding the lub requires a finite series of checks in both $\mathcal{D}_0$ and $\mathcal{D}_1$ but these checks are decidable. Thus, the lub relation is also decidable in $\mathcal{D}_0 \to \mathcal{D}_1$. This shows that $\mathcal{D}_0 \to \mathcal{D}_1$ is effectively given.

To show that *apply* and *curry* are computable, the mappings need to be examined. The mapping defined for apply is

$$(F, a) \, apply \, b \iff a \, F \, b$$

The function F is the lub of all the finite step functions that are consistent with it. As such, F can be viewed as the canonical representative of this set. Since F is a finite step function, this relation is decidable. As such, the *apply* relation is recursive and not just recursively enumerable and *apply* is a computable function.

The reasoning for *curry* is similar in that the relations are studied. Given the increase in the number of domains, the construction is more tedious and is left for the exercises.

To see that the computable elements correspond to the computable maps, recall the relationship shown in Theorem 3.20 between the maps and the elements in the function space. Thus, we have

$$a \, f \, b \iff b \in f(\mathcal{I}_a) \text{ or } \mathcal{I}_b \sqsubseteq f(\mathcal{I}_a)$$

Since f is a computable map, we know that the pairs in the map are recursively enumerable. Using the previous techniques for deciding consistency of finite sets, the set of elements consistent with f can be enumerated. But this set is simply the ideal for f in the function space. The converse direction is trivial. □

The final combinator to be discussed, and perhaps the most important, is the fixed point combinator.

Theorem 7.7: For any effectively given domain, $\mathcal{D}$, the combinator $fix : (\mathcal{D} \to \mathcal{D}) \to \mathcal{D}$ is computable.

Proof Let $\{X_n \mid n \in \mathbb{N}\}$ be the presentation of the domain $\mathcal{D}$. Recall that for $f \in \mathcal{D} \to \mathcal{D}$,

$$f\ fix\ X \iff \exists k \in \mathbb{N}.\Delta\ f\ X_1\ f \ldots f\ X_k \wedge X_k = X$$

All of the checks in this finite sequence are decidable since $\mathcal{D}$ is effectively given. In addition, existential quantification of a decidable predicate gives a recursively enumerable predicate. Thus, fix is computable. □

Now that this has been formalized, what has been accomplished? The major consequence of the theorems to this point is that any expression over effectively given domains (that is effectively given types) combined with computable constants using the λ-notation and the fixed point combinator is a computable function of its free variables. Such functions, applied to computable arguments, yield computable values. These functions also have computable least fixed points. All this gives us a mathematical programming language for defining computable operations. Combining this language with the specification of types with domain equations gives a powerful language.

As an example, the effectiveness of the domain $\mathcal{T}$ from Example 6.1 is studied. The complete proof is left as an exercise.

Example 7.8: Recall the domain $\mathcal{T}$ from the previous section. This domain is characterized by the domain equation

$$\mathcal{T} = \mathcal{A} + (\mathcal{T} \times \mathcal{T})$$

for some domain $\mathcal{A}$. If $\mathcal{A}$ is effectively given, we wish to show that $\mathcal{T}$ is effectively given as well. The elements are either atomic elements from $\mathcal{A}$ or are pairs from $\mathcal{T}$. Let $A = \{A_i \mid i \in \mathbb{N}\}$ be the enumeration for $\mathcal{A}$. An enumeration for $\mathcal{T}$ can be defined as follows:

$$\begin{aligned} T_0 &= \bot_{\mathcal{T}} \\ T_{2n+1} &= 3 * A_n \\ T_{2n+2} &= 3 * T_{p(n)} + 1 \cup 3 * T_{q(n)} + 2 \end{aligned}$$

where for A, a set of indices, $m * A + k = \{m * n + k \mid n \in A\}$. The functions p and q here are the inverses of the pairing function r defined in Theorem 7.5. These functions must be defined such that $p(n) \le n$ and $q(n) \le n$ so that the recursion is well defined by taking smaller indices. The rest of the proof is left to the exercises. Specifically, the claim that $\mathcal{T} = \{T_i\}$ should be verified as well as the effectiveness of the enumeration. These proofs rely either on the effectiveness of $\mathcal{A}$, on the effectiveness of elements in $\mathcal{T}$ with smaller indices, or are trivial.

The final example uses the powerset construction. We have repeatedly used the fact that a powerset is a domain. Its effectiveness is now verified.

Example 7.9: Specifically, the powerset of the natural numbers, $\mathcal{P}(\mathbb{N})$ is considered. In this domain, all elements are consistent, and there is a top element, denoted ω, which is the set of all natural numbers. The ordering is the subset relation. The lub of two subsets is the union of the two subsets, which is decidable. To enumerate the finite subsets, the following enumeration is used:

$$E_n = \{k \mid \exists i, j. i < 2^k \wedge n = i + 2^k + j * 2^{k+1}\}$$

This says that $k \in E_n$ if the k bit in the binary expansion of n is a 1. All finite subsets of $\mathbb{N}$ are of the form E_n for some n. Various combinators for $\mathcal{P}(\mathbb{N})$ are presented in Exercise 7.14.

Exercises

Exercise 7.10: Show that an effectively given domain can always be identified with a relation

$$INCL(n, m)$$

on integers where the derived relations

$$\begin{array}{lcl} CONS(n, m) & \iff & \exists k. INCL(k, n) \wedge INCL(k, m) \\ MEET(n, m, k) & \iff & \forall j.[INCL(j, k) \iff INCL(j, n) \wedge INCL(j, m)] \end{array}$$

are recursively decidable and where the following axioms hold:

1. $\forall n. INCL(n, n)$
2. $\forall n, m, k. INCL(n, m) \wedge INCL(m, k) \Rightarrow INCL(n, k)$
3. $\exists m. \forall n. INCL(n, m)$
4. $\forall n, m. CONS(n, m) \Rightarrow \exists k. MEET(n, m, k)$

Exercise 7.11: Finish the proof of Theorem 7.5.

Exercise 7.12: Complete the proof of Theorem 7.6 by defining *curry* as a relation and showing it computable. Is the set recursively enumerable or is it recursive?

Exercise 7.13: Two effectively given domains are *effectively isomorphic* iff ... Complete the statement of the theorem and prove it.

Exercise 7.14: Complete the proof about the powerset in Example 7.9. Show that the combinators fun and *graph* from Exercise 5.12 are computable. Show the same for

1. $\lambda x, y. x \cap y$
2. $\lambda x, y. x \cup y$
3. $\lambda x, y. x + y$

where for $x, y \in \mathcal{P}(\mathbb{N})$,

$$x + y = \{n + m \mid n \in x, m \in y\}$$

What are the computable elements of $\mathcal{P}(\mathbb{N})$?

8 Sub-Spaces of the Universal Domain

To have a flexible method of solving domain equations and yielding effectively given domains as the solutions, the domains will be embedded in a *universal* domain which is "big" enough to hold all other domains as sub-domains. This universal domain is shown to be effectively presented, and the mappings which define the sub-spaces are shown to be computable. First, the correspondence between sub-spaces and mappings called *retractions* is investigated. It is then shown that these definitions can be written out using the λ-calculus notation, demonstrating the power of our mathematical programming language.

We start with the definition of *retractions*.

Definition 8.1: **[Retractions]** A *retraction* of a given domain $\mathcal{E}$ is an approximable mapping $a : \mathbf{E} \to \mathbf{E}$ such that $a \circ a = a$.

Thus, a retraction is the identity function on objects in the range of the retraction and maps other elements into range. The next theorem relates these sets to sub-spaces.

Theorem 8.2: If $\mathcal{D} \lhd \mathcal{E}$ and if $a : \mathbf{E} \to \mathbf{E}$ is defined such that

$$X \; a \; Z \iff \exists Y \in \mathcal{D}. Z \sqsubseteq Y \sqsubseteq X$$

for all $X, Z \in \mathbf{E}$, then a is a retraction and $\mathcal{D}$ is isomorphic to the fixed point set of a, the set $\{y \in \mathcal{E} \mid a(y) = y\}$, ordered under inclusion.

Proof That a is an approximable map is a direct consequence of the definition of sub-space (Definition 6.2). By Theorem 6.3, a projection pair, i and j, exist for $\mathcal{D}$ and this tells us that $a = i \circ j$ (also showing a approximable since approximable mappings are closed under composition). Theorem 6.3 also tells us that $j \circ i = \mathsf{I}_D$. To show that a is a retraction, $a \circ a = a$ must be established. Thus,

$$a \circ a = i \circ j \circ i \circ j = i \circ \mathsf{I}_D \circ j = i \circ j = a$$

holds, showing that a is a retraction.

We now need to show the isomorphism to $\mathcal{D}$. For $x \in \mathcal{D}$, $i(x) \in \mathcal{E}$ and we can calculate:

$$a(i(x)) = i \circ j \circ i(x) = i \circ \mathsf{I}_D(x) = i(x)$$

Thus, $i(x)$ is in the fixed point set of a. For the other direction, let $a(y) = y$. Then $i(j(y)) = y$ holds. But, $j(y) \in \mathcal{D}$, so i must map $\mathcal{D}$ one-to-one and onto the fixed point set of a. Since i and j are approximable, they are certainly monotonic, and thus the map is an isomorphism with respect to set inclusion. $\Box$

Not all retractions are associated with a sub-domain relationship. The retractions defined in the above theorem are all subsets as relations of the identity relation. The retractions for sub-domains are characterized by the following definition:

Definition 8.3: **[Projections]** A retraction $a : \mathcal{E} \to \mathcal{E}$ is a *projection* if $a \subseteq \mathsf{I}_E$ as relations. The retraction is *finitary* iff its fixed point set is isomorphic to some domain.

An example is in order.

Example 8.4: Consider a two element system, $\mathbf{O}$ with objects Δ and 0. For any basis $\mathbf{D}$ that is not trivial (has more than one element), $\mathbf{O}$ comes from a retraction on $\mathbf{D}$. Define a combinator $check : \mathbf{D} \to \mathbf{O}$ by the relation

$$x \; check \; y \iff y = \Delta \text{ or } x \neq \Delta_D$$

Thus, $check(x) = \bot_O \iff x = \bot_D$. Another combinator can be defined,

$$fade : \mathbf{O} \times \mathbf{D} \to \mathbf{D}$$

such that for $t \in \mathcal{O}$ and $x \in \mathcal{D}$

$$\begin{array}{rcll} fade(t, x) & = & \bot_D & \text{if } t = \bot_O \\ & = & x & \textit{otherwise} \end{array}$$

For $u \in \mathcal{D}$ and $u \neq \bot_D$, the mapping a is defined as

$$a(x) = fade(check(x), u)$$

It can be seen that a is a retraction, but not a projection in general, and the range of a is isomorphic to $\mathbf{O}$.

These combinators can also be used to define the subset of functions in $\mathbf{D} \to \mathbf{E}$ that are strict. Define a combinator $strict : (\mathbf{D} \to \mathbf{E}) \to (\mathbf{D} \to \mathbf{E})$ by the equation

$$strict(f) = \lambda x.fade(check(x), f(x))$$

with $fade$ defined as $fade : \mathbf{O} \times \mathbf{E} \to \mathbf{E}$. The range of $strict$ is all the strict functions; $strict$ is a projection whose range is a domain.

The next theorem characterizes projections.

Theorem 8.5: For approximable mapping $a : \mathbf{E} \to \mathbf{E}$, the following are equivalent:

1. a is a finitary projection
2. $a(x) = \{y \in \mathbf{E} \mid \exists x' \in x.x' \, a \, x' \wedge y \sqsubseteq x'\}$ for all $x \in \mathcal{E}$.

Proof Assume that (2) holds. We want to show that a is a finitary projection. By the closure properties on ideals, we know that for all $x \in \mathcal{E}$,

$$x' \in x \wedge y \sqsubseteq x' \;\Rightarrow\; y \in x$$

Thus, $a(x) \subseteq x$ must hold. In addition, the following trivially holds:

$$x' \in x \wedge x' \, a \, x' \;\Rightarrow\; x' \in a(x)$$

thus $a(x) \subseteq a(a(x))$ holds for all $x \in \mathcal{E}$. This shows that a is indeed a projection. Let $D = \{x \in \mathbf{E} | x a x\}$. It is easy to show that $\mathbf{D} \lhd \mathbf{E}$ and that a is determined from $\mathbf{D}$ as required in Theorem 8.2. Thus, the fixed point set of a is isomorphic to a domain from the previous proofs. Thus, (2) $\Rightarrow$ (1).

For the converse, assume that a is a finitary projection. Let $\mathcal{D}$ be isomorphic to the fixed point set of a. This means there is a projection pair i and j such that $j \circ i = \mathsf{I}_D$ and $i \circ j = a$ and $a \subseteq \mathsf{I}_E$. From Theorem 6.5 then we have that $\mathcal{D} \approx \mathcal{D}'$ and $\mathcal{D}' \lhd \mathcal{E}$. We want to identify $\mathcal{D}'$ as follows:

$$\mathcal{D}' = \{x \in \mathcal{E} \mid x \, a \, x\}$$

From the proof of Theorem 6.5, the basis elements of $\mathbf{D}'$ are the finite elements of $\mathbf{D}$. Each of these elements is in the fixed point set of a. Thus,

$$x \in \mathbf{D}' \;\Rightarrow\; a(\mathcal{I}_x) = \mathcal{I}_x \;\Rightarrow\; x \, a \, x$$

Since a is a projection, $\mathcal{I}_x$ must also be a fixed point. Since $i(j(\mathcal{I}_x)) = \mathcal{I}_x$ implies that $j(\mathcal{I}_x)$ is a finite element of $\mathcal{D}$, $x \in \mathcal{D}'$ must hold. Thus, the identification of $\mathcal{D}'$ holds.

Finally, using $a = i \circ j$ in the formula in Theorem 6.3, the formula in (2) is obtained, proving the converse. □

This characterization of projections provides a new and interesting combinator.

Theorem 8.6: For any domain $\mathcal{E}$, define $sub : (\mathcal{E} \to \mathcal{E}) \to (\mathcal{E} \to \mathcal{E})$ using the relation

$$x\, sub(f)\, z \iff \exists y \in \mathbf{E}. y\ f\ y \wedge y \sqsubseteq x \wedge z \sqsubseteq y$$

for all $x, z \in \mathbf{E}$ and all $f : \mathbf{E} \to \mathbf{E}$. Then the range of sub is exactly the set of finitary projections on $\mathcal{E}$. In addition, sub is a finitary projection on $\mathcal{E} \to \mathcal{E}$. If $\mathcal{E}$ is effectively given, then sub is computable.

Proof Clearly, $sub(f)$ is approximable. It is obvious from the definition that $f \mapsto sub(f)$ preserves lubs and thus is approximable as well. Thus,

$$y\ f\ y \wedge y \sqsubseteq x \wedge z \sqsubseteq y \;\Rightarrow\; x\ f\ z$$

obviously holds. Thus, $sub(f) \subseteq f$ holds. Also

$$y\ f\ y \;\Rightarrow\; y\, sub(f)\, y$$

thus, $sub(f) \subseteq sub(sub(f))$ holds as well. Thus, sub is a projection on $\mathcal{E} \to \mathcal{E}$. The definition of the relation shows that it is computable when $\mathcal{E}$ is effectively given.

Since sub is a projection, its range is the same as its fixed point set. If $sub(a) = a$, it is easy to see that clause (2) of Theorem 8.5 holds and *conversely*. Thus, the range of sub is the finitary projections.

To see that sub is a finitary projection, we use Theorem 6.4 and Theorem 8.2 to say that the fixed point set of sub is in a one-to-one inclusion preserving correspondence with the domain $\{D \mid D \lhd \mathcal{E}\}$. □

With these results and the universal domain to be defined next, the theory of sub-domains is translated into the λ-calculus notation using the sub combinator. The universal domain is defined by first defining a domain which has the desired structure but has a *top* element. The top element is then removed to give the universal domain.

Definition 8.7: [Universal Domain] As in the section on domain equations, an inductive definition for a domain $\mathcal{V}$ is given as follows:

1. $\Delta, \top \in \mathbf{V}$
2. $\langle u, v \rangle \in \mathbf{V}$ whenever $u, v \in \mathbf{V}$

Thus, we are starting with two objects, a bottom element and a top element, and making two flavors of copies of these objects. Intuitively, we end up with finite binary trees with either the top or the bottom element as the leaves. To simplify the definitions below, the pairs should be reduced such that:

1. All occurrences of $\langle \Delta, \Delta \rangle$ are replaced by Δ and
2. All occurrences of $\langle \top, \top \rangle$ are replaced by $\top$.

These rewrite rules are easily shown to be finite Church-Rosser.[5] As an example of the reduction the pair

$$\langle\langle\langle \top, \langle \top, \top \rangle\rangle, \langle \top, \Delta \rangle\rangle, \langle\langle \Delta, \Delta \rangle, \langle \top, \top \rangle\rangle\rangle$$

reduces to

$$\langle\langle \top, \langle \top, \Delta \rangle\rangle, \langle \Delta, \top \rangle\rangle$$

. The approximation ordering is defined as follows:

[5]The finitary basis should be defined as the equivalence classes induced by the reduction. The presentation is simplified by considering only reduced trees.

1. $\Delta \sqsubseteq v$ for all $v \in \mathbf{V}$

2. $v \sqsubseteq \top$ for all $v \in \mathbf{V}$.

3. $\langle u, v \rangle \sqsubseteq \langle u', v' \rangle$ iff $u \sqsubseteq u'$ and $v \sqsubseteq v'$

Since the top element is approximated by everything, all finite sets of trees are consistent. The lub for a pair of trees is defined as follows:

1. $u \sqcup \top = \top$ for $u \in \mathbf{V}$

2. $\top \sqcup u = \top$ for $u \in \mathbf{V}$

3. $u \sqcup \Delta = u$ for $u \in \mathbf{V}$

4. $\Delta \sqcup u = u$ for $u \in \mathbf{V}$

5. $\langle u, v \rangle \sqcup \langle u', v' \rangle = \langle u \sqcup u', v \sqcup v' \rangle$ for $u, v \in \mathbf{V}$

The proof that this forms a finitary basis follows the same guidelines as the proofs in Section 6. In addition, it should be clear that the presentation is effective.

To form the universal domain, the top element is simply removed. Thus, the system $\mathbf{U} = \mathbf{V} - \{\top\}$ is the basis used to form the universal domain. The proof that this is still a finitary basis with an effective presentation is also straightforward and left to the exercises. Note that inconsistent sets can now exist since there is no top element. A set is inconsistent iff its lub is $\top$.

We shall now prove the claims made for the universal domain.

Theorem 8.8: The domain $\mathcal{U}$ is universal in the sense that for every domain $\mathcal{D}$, $\mathcal{D} \lhd \mathcal{U}$. If $\mathcal{D}$ is effectively given, then the projection pair for the embedding is computable. In fact, there is a correspondence between the effectively presented domains and the computable finitary projections of $\mathcal{U}$.

Proof Recall that $\mathbf{D}$ must be countable to be a finitary basis. Thus, we can assume that the basis has an enumeration

$$D = \{X_n \mid n \in \mathbb{N}\}$$

where $X_0 = \Delta$. The effective and general cases are considered together in the proof; comments about computability are included for the effective case as required. Thus, if $\mathcal{D}$ is effectively given, the enumeration above is assumed to be computable.

To prove that the domain can be embedded in $\mathcal{U}$, the embedding will be shown. To start, for each finite element d_i in the basis, define two sets, d_i^+ and d_i^- as follows:

$$\begin{array}{rcl} d_i^+ & = & \{d \in \mathbf{D} \mid d_i \sqsubseteq d\} \\ d_i^- & = & D - d_i^+ \end{array}$$

The d_i^+ set contains all the elements that d_i approximates, while the d_i^- set contains all the other elements, partitioning $\mathbf{D}$ into two disjoint sets. Sets for different elements can be intersected to form finer partitions of $\mathbf{D}$. For $k > 0$, let $R \in \{+, -\}^k$, let R_i be the *ith* symbol in the string R, and define a region D_R as

$$D_R = \bigcap_{i=1}^{k} d_i^{R_i}$$

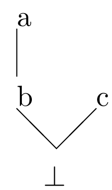


Figure 1: Example Finite Domain

where k is the length of R. The set $\{D_R \mid R \in \{+,-\}^k\}$ of regions partitions $\mathbf{D}$ into 2^k disjoint sets. Thus, for each element e_i in the enumeration there is a corresponding partition of the basis given by the family of sets $\{D_R \mid R \in \{+,-\}^i\}$. For strings $R, S \in \{+,-\}^*$ such that R is a prefix of S, denoted $R \leq S$, $D_S \subseteq D_R$. It is important to realize that the composition of these sets is dependent on the order in which the elements are enumerated. Some of these regions are empty, but it is decidable if a given intersection is empty if $\mathcal{D}$ is effectively presented. It is also decidable if a given element is in a particular region.

To see the function these regions are serving, consider the finite domain in Figure 1:[6] Consider the enumeration with

$$d_0 = \bot, d_1 = b, d_2 = c, d_3 = a$$

The d_i^+ and d_i^- sets are as follows:

$$\begin{array}{rcl} d_1^+ & = & \{a, b\} \\ d_1^- & = & \{c, \bot\} \\ d_2^+ & = & \{c\} \\ d_2^- & = & \{a, b, \bot\} \\ d_3^+ & = & \{a\} \\ d_3^- & = & \{b, c, \bot\} \end{array}$$

The regions are as follows:

$$\begin{array}{rcl} D_+ & = & \{a, b\} \\ D_- & = & \{\bot, c\} \\ D_{++} & = & \{\} \\ D_{+-} & = & \{a, b\} \\ D_{-+} & = & \{c\} \\ D_{--} & = & \{\bot\} \end{array} \qquad \begin{array}{rcl} D_{+++} & = & \{\} \\ D_{++-} & = & \{\} \\ D_{+-+} & = & \{a\} \\ D_{+--} & = & \{b\} \\ D_{-++} & = & \{\} \\ D_{-+-} & = & \{c\} \\ D_{--+} & = & \{\} \\ D_{---} & = & \{\bot\} \end{array}$$

The regions generated by each successive element encode the relationships induced by the approximation ordering between the new element and all elements previously added. The reader is encouraged to try this example with other enumerations of this basis and compare the results.

The embedding of the elements proceeds by building a tree based on the regions corresponding to the element. The regions are used to find locations in the tree and to determine whether a $\top$ or a

[6]This example is taken from Cartwright and Demers [2].

Δ element is placed in the location. These trees preserve the relationships specified by the regions and thus, the tree embedding is isomorphic to the domain in question. Once the tree is built, the reduction rules are applied until a non-reducible tree is reached. This tree is the representative element in the universal domain, and the set of these trees form the sub-space.

The function to determine the location in the tree for a given domain,

$$Loc_D : \{+,-\}^* \rightarrow \{l,r\}^*$$

takes strings used to generate regions and outputs a path in a tree where l stands for left sub-tree and r stands for right sub-tree. This path is computed using the following inductive definition:

$$\begin{array}{lcll} Loc_D(\epsilon) & = & \epsilon. & \\ Loc_D(R+) & = & Loc_D(R)l & \text{if } D_{R+} \neq \emptyset \text{ and } D_{R-} \neq \emptyset. \\ & = & Loc_D(R) & \text{otherwise.} \\ Loc_D(R-) & = & Loc_D(R)r & \text{if } D_{R+} \neq \emptyset \text{ and } D_{R-} \neq \emptyset. \\ & = & Loc_D(R) & \text{otherwise.} \end{array}$$

The set of locations for each non-empty region is the set of paths to all leaves of some finite binary tree. An induction argument is used to show the following properties of Loc_D that ensure this:

1. If $R \leq S$ for $R, S \subseteq \{+,-\}^*$, then $Loc_D(R) \leq Loc_D(S)$.

2. Let $S = \{Loc_D(R) \mid R \in \{+,-\}^k \wedge D_R \neq \emptyset\}$ for $k > 0$ be a set of location paths for a given k. For any $p \in \{l,r\}^*$ there exists $q \in S$ such that either $p \leq q$ or $q \leq p$. That is, every potential path is represented by some finite path.

3. Finally, for all $p, q \in S$ if $p \leq q$ then $p = q$. This means that a unique leaf is associated with each location.

To find the tree for a given element d_k in the enumeration, apply the following rules to each $R \in \{+,-\}^{k-1}$.

1. If $D_{R-} \neq \emptyset$ then the leaf for path $Loc_D(R-)$ is labeled $\top$.

2. If $D_{R+} \neq \emptyset$ then the leaf for path $Loc_D(R+)$ is labeled Δ.

These rules are used to assign a tree in $\mathbf{U}$, which is then reduced using the reduction rules, for each element in the enumeration of $\mathbf{D}$. To see that the top element is never assigned by these rules, note that some region of the form $R+$ for every length k must be non-empty since it must contain the element e_k being embedded.

Returning to the example, the location function defines paths for these elements as follows:

$$\begin{array}{lcl@{\qquad}lcl} Loc_D(+) & = & l & Loc_D(+-+) & = & ll \\ Loc_D(-) & = & r & Loc_D(+--) & = & lr \\ Loc_D(+-) & = & l & Loc_D(-+-) & = & rl \\ Loc_D(-+) & = & rl & Loc_D(---) & = & rr \\ Loc_D(--) & = & rr & & & \end{array}$$

The trees generated for each of the elements are:

$$\begin{array}{lcl} d_0 & \mapsto & \Delta \\ d_1 & \mapsto & \langle \Delta, \top \rangle \\ d_2 & \mapsto & \langle \top, \langle \Delta, \top \rangle \rangle \\ d_3 & \mapsto & \langle \langle \Delta, \top \rangle, \langle \top, \top \rangle \rangle \\ & \mapsto & \langle \langle \Delta, \top \rangle, \top \rangle \end{array}$$

To verify that the space generated is a valid sub-space, we must verify that the bottom element is mapped to $\bot_U$ and that the consistency and lub relations are maintained. The tree Δ is clearly assigned to X_0, the bottom element for the basis being embedded, since there are no strings of length -1. The embedding preserves inconsistency of elements by forcing the lub of the embedded elements to be $\top$. The D_{R-} regions represent the elements that the element being embedded does not approximate. Note that the D_{R-} sets cause the $\top$ element to be added as the leaf. Since the D_R sets are built using the approximation ordering, it is straightforward to see that the approximation ordering is preserved by the embedding. Lubs are also maintained by the embedding, although the reduction is required to see that this is the case. It should be clear that, if the domain $\mathcal{D}$ is effectively given, the sub-space can be computed since the embedding procedure uses the relationships given in the presentation.

Finally, suppose that a is a computable, finitary projection on $\mathcal{U}$. From the proof of Theorem 8.5, the domain of this projection is characterized by the set

$$\{y \in \mathbf{U} \mid y\, a\, y\}$$

If a is computable, the set of pairs for a is recursively enumerable. Thus, the set above is also recursively enumerable since equality among basis elements is decidable. Thus, the domain given by the projection must also be effectively given. $\square$

Thus, the domain $\mathcal{U}$ is an effectively presented *universal* domain in which all other domains can be embedded. The sub-domains of $\mathcal{U}$ include $\mathcal{U} \to \mathcal{U}$, $\mathcal{U} \times \mathcal{U}$, etc. These domains must be sub-domains of $\mathcal{U}$ since they are effectively presented based on our earlier theorems. The next step is to see how to define the constructors commonly used.

Definition 8.9: [Domain Constructors] Let the computable projection pair,

$$i_+ : \mathcal{U} + \mathcal{U} \to \mathcal{U} \text{ and } j_+ : \mathcal{U} \to \mathcal{U} + \mathcal{U}$$

be fixed. Fix suitable projection pairs $i_\times, j_\times, i_\to$, and $j_\to$ as well. Define

$$\begin{aligned}
a + b &= cond \circ \langle which, i_+ \circ in_0 \circ a \circ out_0, i_+ \circ in_1 \circ b \circ out_1 \rangle \circ j_+ \\
a \times b &= i_x \circ \langle a \circ proj_0, b \circ proj_1 \rangle \circ j_x \\
a \to b &= i_\to \circ (\lambda f. b \circ f \circ a) \circ j_\to
\end{aligned}$$

for all $a, b : \mathcal{U} \to \mathcal{U}$.

From earlier theorems, we know that these combinators are all computable over an effectively presented domain. The next theorem characterizes the effect these combinators have on projection functions.

Theorem 8.10: If $a, b : \mathcal{U} \to \mathcal{U}$ are projections, then so are $a + b$, $a \times b$, and $a \to b$. If a and b are finitary, then so are the compound projections.

Proof Since a and b are retractions, $a = a \circ a$ and $b = b \circ b$. Then for $a \times b$ using the definition of $\times$,

$$\begin{aligned}
(a \times b) \circ (a \times b) &= i_x \circ \langle a \circ proj_0, b \circ proj_1 \rangle \circ \langle a \circ proj_0, b \circ proj_1 \rangle \circ j_x \\
&= i_x \circ \langle a \circ a \circ proj_0, b \circ b \circ proj_1 \rangle \circ j_x \\
&= a \times b
\end{aligned}$$

Thus, $a \times b$ is a retraction. The other cases follow similarly.

Since a and b are projections, $a, b \subseteq \mathsf{I}_U$ (denoted simply I for the remainder of the proof). Using the definition for $+$ along with the above relation and the definition of projection pairs, we can see that

$$a + b \subseteq \mathsf{I} + \mathsf{I} = i_+ \circ j_+ \subseteq \mathsf{I}$$

Thus, $a + b$ is a projection. The other cases follow similarly.

To show that the projections are finitary, we must show that the fixed point sets are isomorphic to a domain. Since a and b are assumed finitary, their fixed point sets are isomorphic to

$$\begin{aligned} D_a &= \{x \in \mathbf{U} \mid x\, a\, x\} \\ D_b &= \{y \in \mathbf{U} \mid y\, b\, y\} \end{aligned}$$

We wish to show that $\mathcal{D}_a \to \mathcal{D}_b \approx \mathcal{D}_{a \to b}$. By the definition of the $\to$ constructor, the fixed point set of $a \to b$ over $\mathcal{U}$ is the same as the fixed point set of $\lambda f.b \circ f \circ a$ on $\mathcal{U} \to \mathcal{U}$. (Hint: $i_\to$ and $j_\to$ set up the isomorphism.) So, the fixed points for $f : \mathcal{U} \to \mathcal{U}$ are of the form:

$$f = b \circ f \circ a$$

We can think of a as a function in $\mathcal{U} \to \mathcal{D}_a$ and define the other half of the projection pair as $i_a : \mathcal{D}_a \to \mathcal{U}$ where $i_a \circ a = a$ and $a \circ i_a = i_a$. Define a function i_b for the projection pair for b similarly. For some $g : \mathcal{D}_a \to \mathcal{D}_b$ let

$$f = i_b \circ g \circ a$$

Substituting this definition for f yields

$$b \circ f \circ a = b \circ i_b \circ g \circ a \circ a = i_b \circ g \circ a = f$$

by the definition of i_b and since a is a retraction by assumption. Conversely, for a function f such that $i_b \circ g \circ a = f$, let

$$g = b \circ f \circ i_a$$

Substituting again,

$$i_b \circ g \circ a = i_b \circ g \circ f \circ i_a \circ a = b \circ f \circ a = f$$

Thus, there is an order preserving isomorphism between $g : \mathcal{D}_a \to \mathcal{D}_b$ and the functions $f = b \circ f \circ a$. The proofs of the isomorphisms for the other constructs are similar. □

Thus, the sub-domain relationship with the universal domain has been stated in terms of finitary projections over the universal domain. In addition, all the domain constructors have been shown to be computable combinators on the domain of these finitary projections. Recalling that all computable maps have computable fixed points, the standard fixed point method can be used to solve domain equations of all kinds if they can be defined on projections.

Returning to the λ-calculus for a moment, all objects in the λ-calculus are considered functions. Since $\mathcal{U} \to \mathcal{U}$ is a part of $\mathcal{U}$, every object in the λ-calculus is also an object of $\mathcal{U}$. Transposing some of the familiar notation, where the old notation appears on the left, the new combinators are defined as follows:

$which(z) = which(j_+(z))$
$in_i(x) = i_+(in_i(x))$ where $i = 0, 1$
$out_i(x) = out_i(j_+(x))$ where $i = 0, 1$
$\langle x, y\rangle = i_x(\langle x, y\rangle)$
$proj_i = proj_i(j_x(z))$ where $i = 0, 1$
$u(x) = j_\to(u)(x)$
$\lambda x.\tau = i_\to(\lambda x.\tau)$

Thus, all functions, all constants, all combinators, and all constructs are elements of $\mathcal{U}$. Indeed, **everything** is an element of $\mathcal{U}$. Elements in $\mathcal{U}$ play multiple roles by representing different objects under different projections. While this notion may be difficult to get used to, there are many advantages, both notational and conceptual.

Exercises

Exercise 8.11: A retraction $a : \mathcal{D} \to \mathcal{D}$ is a *closure operator* iff $\mathsf{I}_D \subseteq a$ as relations. On a domain like $\mathcal{P}(\mathbb{N})$, give some examples of closure operators. (Hint: Close up the integers under addition. Is this continuous on $\mathcal{P}(\mathbb{N})$?) Prove in general that for any closure $a : \mathcal{D} \to \mathcal{D}$, the fixed point set of a is always a finitary domain. (Hint: Show that the fixed point set is closed as required for a domain.) What are the finite elements of the fixed point set?

Exercise 8.12: 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 domains $X \lhd \mathcal{D}$.) In the case of $\mathcal{D} = \mathcal{U}$, show that the computable elements of the domain correspond exactly to the effectively presented domains (up to effective isomorphism).

Exercise 8.13: For finitary projections $a : \mathcal{E} \to \mathcal{E}$, write

$$\mathcal{D}_a = \{x \in \mathcal{E} \mid x\, a\, x\}$$

Show that for any two such projections a and b, that

$$a \subseteq b \iff \mathcal{D}_a \lhd \mathcal{D}_b$$

Exercise 8.14: Find another universal domain that is not isomorphic to $\mathcal{U}$.

Exercise 8.15: Prove the remains cases in Theorem 8.10.

Exercise 8.16: Suppose S and T are two binary constructors on domains that can be made into computable operators on projections over the universal domain. Show that we can find a pair of effectively presented domains such that

$$D \approx S(D, E) \text{ and } E \approx T(D, E).$$

Exercise 8.17: Using the translations shown after the proof of Theorem 8.10, show how the whole *typed*-λ-calculus can be translated into $\mathcal{U}$. (Hint: for $f : \mathcal{D}_a \to \mathcal{B}$, write $f = b \circ f \circ a$ for finitary projections a and b. For $\lambda x^{\mathcal{D}_a}.\sigma$, write $\lambda x.b(\sigma'[a(x)/x])$ where σ' is the translation of σ into the untyped λ-calculus. Be sure that the resulting term has the right type.)

Exercise 8.18: Show that the basis presented for the universal domain **U** is indeed a finitary basis and that it has an effective presentation.

Exercise 8.19: Work out the embedding for the other enumerations for the example given in the proof of Theorem8.8.

References

[1] Barendregt, H.P. *The Lambda Calculus: Its Syntax and Semantics*, Revised Edition. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, 1984.

[2] Cartwright, Robert, and Demers, Alan. The Topology of Program Termination. Manuscript.

[3] Scott, Dana. Lectures on a Mathematical Theory of Computation. Technical Monograph PRG-19, Oxford University Computing Laboratory, Oxford.

[4] Scott, Dana. Domains for Denotational Semantics. Technical Report, Computer Science Department, Carnegie Mellon University, 1983.