Uniqueness of Scott's Reflexive Domain in $P\omega$

Thomas Drakengren

April 20, 1994

Abstract

Domains for the pure λ-calculus, sometimes called *reflexive domains*, can be constructed in several ways, but the D_∞ domains found by Scott in 1969 still seem to be the ones with the best properties. The main disadvantage of D_∞ has been that the construction is considered somewhat artificial.
On the other hand, a very concrete domain was found by Scott in 1976, in fact being a subset of $P\omega$, the set of subsets of the natural numbers, by means of *retracts*. The advantage of such a concrete domain is that one may more easily extract its properties, since one can analyse the domain in terms of natural numbers.
The main results of this thesis are the following.

- The reflexive domain in $P\omega$ is unique, in the sense that it is the only nontrival solution to the retract equation $d = d{\circ\!\!\rightarrow} d$.
- The $P\omega$ domain is isomorphic to one of the D_∞ models. This implies that we may adopt results about the D_∞ domains for the $P\omega$ domain.
- The domain possesses no *second least elements*.
- An algorithm for deciding relations between basis elements of the domain is provided.

Acknowledgements

First of all, I want to thank Karl-Johan Bäckström, my supervisor, for support, valuable ideas and enthusiasm during my work, giving me new insights into the topic.
Thanks also to Pontus Gagge, for submitting very constructive criticism, enabling me to make the proofs more understandable, and to improve my language.
Several persons have helped me with comments and criticism, proofreading, finding LaTeX for PC, using LaTeX, typographical matters, enthusiasm and sending me an article, namely Peter Jonsson, David Byers, Helena Drakengren and Rebecca Clark.

Contents

Chapter 1

Preliminaries

1.1 Prerequisites

The reader should have some knowledge of λ-calculus, lattice theory and Scott's $P\omega$ model. It should be enough to read [9], which is a good introduction to the topic. For more details, consult e.g. [7].

1.2 Motivation

In 1969, Scott provided the first "mathematical" model for the λ-calculus, using an *inverse limit construction* (see [6]). However, it was considered somewhat contrived, although the domain in question, called D_∞, was shown to have many "nice" properties, e.g. in [10]. Later, he constructed a model in $P\omega$, the set of all subsets of the natural numbers (see [7]), by means of *retracts* and *fixed points.* Since natural numbers are a lot more concrete than inverse limits of lattices, one may think that it should be easier to extract properties of the latter model, and indeed, that is the key to most of the results in this thesis. If in addition we knew that the two models were isomorphic, we would have a model that is both concrete and has excellent properties, and what more could we want? This desired isomorphism will be proved in chapter 2.
These models are *extensional,* i.e. every function is uniquely determined by its input-output behaviour[1]. There are many subtleties in the notion of a model for the λ-calculus, but in [2], Hindley and Longo, after having analysed several extensionality properties of λ-calculus models, make the following remark.

> In contrast to the above, with the splendid D_∞ everything goes perfectly smoothly;...

Other properties in favour of the reflexive domain in $P\omega$ is that it is a *continuous lattice* (even an *algebraic lattice*), and has a *countable basis.* For an explanation of these concepts, see [9] or [7].

1.3 Coding Continuous Functions

Pairs of integers will be coded by the pairing function

$$(\cdot,\cdot) : \omega \times \omega \to \omega$$

[1] The definition of extensionality can be argued; e.g. Scott in [8], who advocates a weaker axiom, that allows a function to have several representatives.

defined by

$$(n, m) = \frac{1}{2}(n+m)(n+m+1) + m.$$

The finite sets of integers will be written as e_n, with n as a code for the members of the set, satisfying

$$e_n = \{a_1, a_2, \cdots, a_k\} \Leftrightarrow n = \sum_{1 \leq i \leq k} 2^{a_i}$$

These codings have some properties that will be used extensively in the following.

Theorem 1.1

$$\begin{array}{l} x < (x, y) \vee (x \leq 1 \wedge y = 0) \\ y < (x, y) \vee x = y = 0 \\ x \in e_s \Rightarrow x < s \\ (\cdot, \cdot) \text{ is bijective} \\ \lambda n.e_n \text{ is bijective} \end{array}$$

Proof. Left to the reader. ■

Remark 1.2 The second property in theorem 1.1 tells us that if we recurse through the right components of a pair[2] (x, y), we will eventually reach zero, and thus, the *sequences* produced by iterated pairing are always finite[3]. ■

Further on, we will use this to define recursive functions on integers, seen as pairs. Using these codings, we code continuous functions as sets of integers. Given a continuous function $f : P\omega \rightarrow P\omega$, it is represented as

$$graph(f) = \{(n, m) | m \in f(e_n)\},$$

and given any $u, x \in P\omega$, application is defined as

$$u(x) = \{m | \exists e_n \subseteq x : (n, m) \in u\}.$$

Note that *any* element $u \in P\omega$ represents a continuous function on $P\omega$, using the definition of application above, even if it is not the *graph* of any continuous function on $P\omega$. Some properties of *graph* that will be used are the following.

Theorem 1.3 *Given continuous functions f and g on $P\omega$ we have*

$$f = g \Leftrightarrow graph(f) = graph(g);$$

thus we may speak of elements of $P\omega$ such that

$$u = graph(u)$$

as being continuous functions on $P\omega$.
Furthermore, we always have

$$u \subseteq graph(u).$$

Proof. See e.g. [9], chapter 7. ■

[2] Note that the y in the right component also can be seen as a pair.

[3] For a survey of properties of general codings, not necessarily well-behaved like this one, see [5].

Corollary 1.4 *Provided $f, g \in P\omega$ are continuous functions on $P\omega$, i.e. that*

$$f = graph(f) \wedge g = graph(g),$$

we have

$$(\forall u \in P\omega : f(u) = g(u)) \Leftrightarrow f = g.$$

Proof. Follows from theorem 1.3 and the definition of equality between functions. ■

Corollary 1.5 *For $u \in P\omega$,*

$$graph(u) = \lambda x.u(x).$$

Proof. Follows directly from the definitions of *graph* and of application in $P\omega$. ■

Corollary 1.6

$$\begin{array}{rcl} \top & = & graph(\top) \quad and \\ \perp & = & graph(\perp) \end{array}$$

Proof. By theorem 1.3,

$$\top \subseteq graph(\top),$$

implying

$$\top = graph(\top).$$

Also

$$\begin{array}{rcll} graph(\perp) & = & \lambda x.\perp(x) & \text{cor. 1.5} \\ & = & \lambda x.\perp & \\ & = & \perp. & \end{array}$$

■

1.4 Notation

We are going to study domains defined by *retracts*, i.e. fixed points of continuous functions $f \in P\omega$ such that $f = f \circ f$. To obtain a domain for the λ-calculus, we want a domain that is a solution to the *isomorphic domain equation*

$$D \cong D \rightarrow D,$$

i.e. a domain isomorphic to its own space of continuous functions. With retracts, reflexive domains can be obtained as solutions to the equation

$$d = d \circ\!\!\rightarrow d,$$

where $a \circ\!\!\rightarrow b$, for retracts a and b, is the retract defining the domain of continuous functions from the domain defined by a to the domain defined by b. $a \circ\!\!\rightarrow b$ is defined by

$$a \circ\!\!\rightarrow b = \lambda v.b \circ v \circ a = \lambda v \lambda x.b(v(a(x))).$$

In [9], such domains are called *reflexive domains*. The retracts that define them (e.g. d above) are called *reflexive retracts* in [3], and we will adopt that terminology.
The notation and terminology used is close to that of Stoy in [9], but some conventions need to be stated.

- The set of natural numbers is written as ω.
- The set of all subsets of ω is written as $P\omega$.
- A notation for the set of *finite* sets of integers, i.e. the e_n sets above, will be needed, written as $P_F\omega$, with the F for "finite".
- $\mathcal{T}$ and $\mathcal{F}$ denote the truth values.
- $\subseteq$ will be used for the subset relation and $\subset$ for the strict subset relation.
- For the image set of a function f on a *set* X, we will write $f(X)$, when no ambiguity arises.
- We will make use of the language *LAMBDA* for defining elements in $P\omega$. However, instead of writing its conditional expression as

  $$z \supset x, y$$

  we will write it as

  $$0 \in z \rightarrow x, y$$

  to make the meaning clear, which is

  $$\{n \in x | 0 \in z\} \cup \{m \in y | \exists k : k + 1 \in z\}.$$

  The language *LAMBDA* is described in [9] and [7].

Chapter 2

The Domain

The main results of this chapter are that Scott's nontrivial solution to the retract equation

$$d = d \circ\!\!\rightarrow d$$

is unique, and that the domain it defines is isomorphic to one of the D_∞ models.

2.1 Domain Uniqueness

Definition 2.1 *Let d be any solution to the retract equation*

$$d = d \circ\!\!\rightarrow d$$

such that $d \neq \top$ and $d \neq \bot$. Then d is a **normal reflexive retract**[1].
We refer to the domain defined by a given d, i.e. $d(P\omega)$, as D. ■

Normally, d is of course defined on $P\omega$, but since applying d to singleton sets of $P\omega$ will turn out to be a key to proving the uniqueness, and therefore will be used frequently, it will be convenient to use the following notation.

Notation 2.2 *The following abbreviations will be used, provided that no ambiguity arises: $d(\{x\})$ will be written as $d(x)$, $d(\{(s,t)\})$ as $d(s,t)$, $\{(s,t)\}(u)$ as $(s,t)u$, and similarly for other cases.* ■

Lemma 2.3

$$d(s,t) = \{(n,m) | m \in d(t) \wedge e_s \subseteq d(e_n)\}$$

Proof.

$$\begin{array}{rcl} d(s,t) & = & (d \circ\!\!\rightarrow d)(s,t) \\ & = & \lambda x.d((s,t)d(x)) \\ & = & \{(n,m) | m \in d((s,t)d(e_n))\} \\ & = & \{(n,m) | m \in d(\{b | \exists a : (e_a \subseteq d(e_n) \wedge (a,b) \in \{(s,t)\})\})\} \\ & = & \{(n,m) | m \in d(\{t | e_s \subseteq d(e_n)\})\} \\ & = & \{(n,m) | m \in d(t) \wedge e_s \subseteq d(e_n)\} \end{array}$$

■

Lemma 2.4

$$a = \lambda x.a \Leftrightarrow a = \bot \vee a = \top$$

[1] The term used in [3].

Proof. The proof is taken almost directly from [7].
$\Leftarrow$) Trivially,

$$\begin{aligned}\lambda x.\bot &= \{(n,m)|m \in \bot\} = \bot \quad \text{and} \\ \lambda x.\top &= \{(n,m)|m \in \top\} = \top.\end{aligned}$$

$\Rightarrow$) Suppose that $a = \lambda x.a$ and $a \neq \top$. Then there exists a least integer k such that $k \notin a$. We have $k = (n, m)$ for some n, m. If $m \in a$, then
$k = (n, m) \in \lambda x.a = a$, hence $m \notin a$. We have $m \leq k$ and k minimal, therefore $m = k$, which is only possible when $k = m = n = 0$. Conclusion: $a \neq \top \Rightarrow 0 \notin a$. Suppose further that $a \neq \bot$. Then there exists a least integer l such that $l \in a$. We have $l = (i, j)$ for some i, j, and we have $j \in a$ and $j \leq l$. Thus $j = l$ since l is minimal, which is only possible when $l = i = j = 0$. Conclusion: $a \neq \bot \Rightarrow 0 \in a$. We combine the two results and get

$$a \neq \top \wedge a \neq \bot \Rightarrow 0 \in a \wedge 0 \notin a,$$

and the results follows from the contradiction. ∎

Lemma 2.5 *The following equalities hold:*

$$\begin{aligned}d(\bot) &= \lambda x.d(\bot) \\ d(\top) &= \lambda x.d(\top) \\ d(0) &= \lambda x.d(0)\end{aligned}$$

Proof.

$$\begin{aligned}d(\bot) &= (d \circ\!\!\rightarrow d)\bot \\ &= \lambda x.d(\bot d(x)) \\ &= \lambda x.d(\bot)\end{aligned}$$

$$\begin{aligned}d(\top) &= (d \circ\!\!\rightarrow d)\top \\ &= \lambda x.d(\top d(x)) \\ &= \lambda x.d(\top)\end{aligned}$$

$$\begin{aligned}d(0) &= (d \circ\!\!\rightarrow d)0 \\ &= \lambda x.d(0(d(x))) \\ &= \lambda x.d((0,0)d(x)) \\ &= \lambda x.d(0)\end{aligned}$$

∎

Lemma 2.6 *For any normal reflexive retract d,*

$$d = graph(d).$$

Proof.

$$\begin{aligned}graph(d) &= \lambda x.d(x) && \text{cor. 1.5} \\ &= \lambda x.(d \circ\!\!\rightarrow d)(x) \\ &= \lambda x \lambda u.d(x(d(u))) \\ &= d \circ\!\!\rightarrow d \\ &= d\end{aligned}$$

∎

Lemma 2.7 $d(\bot) = \bot$, *i.e. d is strict.*

Proof. By lemma 2.4 and lemma 2.5, we have to show that $d(\bot) \neq \top$. If we had $d(\bot) = \top$, then since d is monotonic, by lemma 2.6 and corollary 1.6 we would have that $d = \top$, a case that was excluded. ■

Lemma 2.8

$$d(\top) = \top$$

Proof. Follows from lemma 2.4 and lemma 2.5, for otherwise we would have $d(\top) = \bot$, implying $d = \bot$, by monotonicity, lemma 2.6 and corollary 1.6. ■

Lemma 2.9

$$u \in P\omega \Rightarrow (e_t = e_s(u) \Rightarrow t < s \vee s \leq 1)$$

Proof. We have

$$e_s(u) \subseteq e_s(\top) = \{m | \exists n : (n, m) \in e_s\},$$

and by the properties of the basis element coding, we must have $t \leq a$ if we set

$$e_a = \{m | \exists n : (n, m) \in e_s\},$$

and therefore it is sufficient to show that $a < s \vee s \leq 1$.
We know by the basis element coding that

$$s = \sum_{(n,m) \in e_s} 2^{(n,m)}$$

and

$$a = \sum_{(n,m) \in e_s} 2^m.$$

For every element $(n, m) \in e_s$, we know that $m < (n, m) \vee n = m = 0$, thus if e_s contains at least one element such that $m < (n, m)$, we are done, since then $a < s$ by the sums above. Otherwise we must have $s \leq 2^0 = 1$. ■

Theorem 2.10 *d is additive, i.e.*

$$u \in P\omega \Rightarrow d(u) = \bigcup_{x \in u} d(x).$$

Proof. Since d is continuous, it suffices to show that for all $s \in \omega$,

$$d(e_s) = \bigcup_{x \in e_s} d(x).$$

Induction on s. We have that

$$d(e_s) = (d \multimap d) e_s = \{(n, m) | m \in d(e_s d(e_n))\},$$

and by lemma 2.9, if we set $e_t = e_s d(e_n)$, we will have $t < s \vee s \leq 1$. If $s = 1$, the result follows trivially, and also when $s = 0$, by lemma 2.7. Otherwise, by induction we may assume that

$$d(e_t) = \bigcup_{a \in e_t} d(a),$$

and therefore

$$\begin{aligned} d(e_s) &= \{(n,m) | m \in d(e_s d(e_n))\} \\ &= \{(n,m) | m \in d(\bigcup_{x \in e_s} x d(e_n))\} \\ &= \{(n,m) | m \in \bigcup_{x \in e_s} d(x d(e_n))\} \quad \text{since } e_t = e_s d(e_n) \\ &= \bigcup_{x \in e_s} \{(n,m) | m \in d(x d(e_n))\} \\ &= \bigcup_{x \in e_s} (d \circ\!\!\rightarrow d) x \\ &= \bigcup_{x \in e_s} d(x). \end{aligned}$$

■

Lemma 2.11 *$d(0) = \top$, and if $u \in D$, then $0 \in u \Leftrightarrow u = \top$.*

Proof. By lemma 2.5 and lemma 2.4, we must have that

$$d(0) = \bot \vee d(0) = \top,$$

and thus it suffices to show the contradiction

$$d(0) = \bot \Rightarrow d = \bot,$$

and since d is additive by theorem 2.10, showing that by corollary 1.6 and lemma 2.6 amounts to showing

$$d(0) = \bot \Rightarrow \forall k \in \omega : d(k) = \bot.$$

Suppose $d(0) = \bot$.
Induction on $k = (s,t)$.
For $k = 0$, we already know $d(k) = \bot$ by the assumption. Now suppose $k > 0$.
By lemma 2.3, we have

$$d(k) = d(s,t) = \{(n,m) | m \in d(t) \wedge e_s \subseteq d(e_n)\},$$

with $t < k$, by theorem 1.1, since $k > 0$. Thus we know by induction that $d(t) = \bot$, implying $d(k) = \bot$.
Now suppose $u \in D$, i.e. $u = d(u)$. If $u = \top$, trivially, $0 \in u$. If $0 \in u$, then $\top = d(0) \subseteq d(u) = u$, by monotonicity; hence $u = \top$. ■

Lemma 2.12

$$\forall k \in \omega : k \in d(k)$$

Proof. Induction on $k = (s,t)$.
If $k = 0$, then $0 \in d(0)$ by lemma 2.11. Now suppose $k > 0$. By lemma 2.3, we have

$$d(k) = d(s,t) = \{(n,m) | m \in d(t) \wedge e_s \subseteq d(e_n)\},$$

thus we have to show that

$$t \in d(t) \text{ and } e_s \subseteq d(e_s).$$

By the properties of the pairing function and since $k > 0$,

$$\begin{aligned} &t < (s,t) = k \quad \text{and} \\ &s \le (s,t) = k. \end{aligned}$$

By induction, we know that $t \in d(t)$. If $s = 0$, then $e_s \subseteq d(e_s)$ trivially holds. We have for every $a \in e_s$ that $a < s \leq k$. By induction we know that $\forall a \in e_s : a \in d(a)$, and by monotonicity, $e_s \subseteq d(e_s)$. ■

Corollary 2.13 *d is a finitary closure, i.e. $I \subseteq d$.*

Proof. Since d is additive by theorem 2.10 and $\forall k \in \omega : k \in d(k)$ by lemma 2.12, the result follows. ■

Now, we have extracted the basic properties of d needed for showing the uniqueness of d. The proof will be a simple induction, but first we need some auxiliary notation and some lemmas enabling us to perform the details of the proof.

Notation 2.14 *Henceforth, $d'(x)$ will be a shorthand notation for $\{k | x \in d(k)\}$.* ■

Lemma 2.15

$$d'(0) = \{0\}$$

Proof. By lemma 2.11,

$$d'(0) = \{k | 0 \in d(k)\} = \{k | d(k) = \top\}.$$

We know that $d'(0)$ is nonempty, since $0 \in d'(0)$ by lemma 2.11. Let $k = (s, t)$ be the least integer in $d'(0)$ such that $k > 0$. Then we have $d(k) = \top$. Furthermore, by lemma 2.3,

$$d(k) = d(s, t) = \{(n, m) | m \in d(t) \wedge e_s \subseteq d(e_n)\},$$

and since $d(k) = \top$, in particular, $0 = (0, 0) \in d(k)$, i.e.

$$0 \in d(t) \wedge e_s \subseteq d(\emptyset),$$

and then by lemma 2.7, $s = 0$.
Since $0 \in d(t)$ we have $t \in d'(0)$, and $t < (s, t) \vee t = 0$. Since $k = (s, t) > 0$, we must have $t > 0$ since $s = 0$ and consequently $t < (s, t) = k$, contradicting the choice of k. ■

Lemma 2.16

$$e_s \subseteq d(e_n) \Leftrightarrow \forall a \in e_s \exists b \in e_n : a \in d(b)$$

Proof.
$\Rightarrow$) Suppose $e_s \subseteq d(e_n)$. By theorem 2.10, d is additive, and thus

$$e_s \subseteq \bigcup_{x \in e_n} d(x).$$

Each $a \in e_s$ now must be contained in $d(b)$ for at least one $b \in e_n$.
$\Leftarrow$) Suppose $\forall a \in e_s \exists b \in e_n : a \in d(b)$. By monotonicity, we must have $e_s \subseteq d(e_n)$. ■

Lemma 2.17

$$d(s, t) = \{(n, m) | m \in d(t) \wedge \forall a \in e_s : d'(a) \cap e_n \neq \emptyset\}$$

Proof.

$$\begin{array}{rcll} d(s,t) & = & \{(n,m)|m \in d(t) \wedge e_s \subseteq d(e_n)\} & \text{lem. 2.3} \\ & = & \{(n,m)|m \in d(t) \wedge \forall a \in e_s \exists b \in e_n : a \in d(b)\} & \text{lem. 2.16} \\ & = & \{(n,m)|m \in d(t) \wedge \forall a \in e_s : \{k|a \in d(k)\} \cap e_n \neq \emptyset\} & \\ & = & \{(n,m)|m \in d(t) \wedge \forall a \in e_s : d'(a) \cap e_n \neq \emptyset\} & \end{array}$$

■

Lemma 2.18

$$d'(s,t) = \{(n,m)|m \in d'(t) \wedge e_n \subseteq \bigcup_{a \in e_s} d(a)\}$$

Proof.

$$\begin{array}{rcll} d'(s,t) & = & \{(n,m)|(s,t) \in d(n,m)\} & \\ & = & \{(n,m)|(s,t) \in \{(a,b)|b \in d(m) \wedge e_n \subseteq d(e_a)\}\} & \text{lem. 2.3} \\ & = & \{(n,m)|t \in d(m) \wedge e_n \subseteq d(e_s)\} & \\ & = & \{(n,m)|m \in d'(t) \wedge e_n \subseteq \bigcup_{a \in e_s} d(a)\} & \text{th. 2.10} \end{array}$$

■

Theorem 2.19 *Let d and e be normal reflexive retracts. Then $d = e$.*

Proof. We will prove

$$\forall k \in \omega : d(k) = e(k) \wedge d'(k) = e'(k).$$

The result then follows by lemma 2.6 and corollary 1.4, since d is additive by theorem 2.10.
Induction on $k = (s,t)$.
By lemma 2.11 we know that

$$d(0) = e(0) = \top,$$

and by lemma 2.15 that

$$d'(0) = e'(0) = \{0\}.$$

For the induction, suppose $k = (s,t) > 0$. We first have

$$d(s,t) = \{(n,m)|m \in d(t) \wedge \forall a \in e_s : d'(a) \cap e_n \neq \emptyset\} \tag{2.1}$$

by lemma 2.17, and since $a < s \leq (s,t) = k$, by induction we may assume that $d'(a) = e'(a)$ for every $a \in e_s$. Similarly, $t < (s,t) = k$, since $k > 0$. By induction, we have $d(t) = e(t)$, and therefore by (2.1)

$$d(s,t) = \{(n,m)|m \in e(t) \wedge \forall a \in e_s : e'(a) \cap e_n \neq \emptyset)\} = e(s,t).$$

By a similar argument, using lemma 2.18, we get

$$d'(s,t) = e'(s,t),$$

and the result follows. ■

Discussion 2.20 What theorem 2.19 tells us is that there are exactly three solutions to the retract equation $d = d \circ\!\!\rightarrow d$, since it is easily verified that $\perp$ and $\top$ are solutions. This seems to contradict a statement by Scott in [7], page 553.

There are many other fixed points of $[d = d \circ\!\!\rightarrow d]$ besides this least closure operation, but their connection with inverse limits is not fully investigated.

However, this result cannot allow us to conclude that our D domain is the only nontrivial reflexive domain as a continuous lattice with a countable basis that can be embedded in $P\omega$, since by our construction, we have forced the isomorphism between D and $D \to D$ to be I, the identity function.
To eliminate this restriction, the retract equation $d = d \circ\!\!\rightarrow d$ can be generalised to

$$d = f^{-1} \circ (d \circ\!\!\rightarrow d) \circ f,$$

where $f : D \to (D \to D)$ is meant to be the (continuous) isomorphism. However, this equation is not as simple to analyse as the former, for instance, the uniqueness proof above relies heavily on recursion through pair sequences, and that seems to be blocked by introducing the f.
One may also think that if the coding of pairs and basis elements used in the $P\omega$ model were changed, one would get a different structure. Schellinx in [5] indicates that λ-calculus models may change due to changes in coding, and even if one uses I as the isomorphism, with a change in coding, one may not even have pair sequences to perform induction on. ∎

2.2 Relating the Domain D to Scott's D_∞ Domains

We are now about to prove that the domain D_∞ obtained by the inverse limit construction, mentioned in the introduction, is essentially equivalent to our D domain. We will not bother to explain everything about inverse limits here; the reader may consult e.g. [1] for a general introduction, and [6] for a detailed description of the construction of D_∞.
The idea is to start with any initial complete lattice, and to inductively define a sequence of lattices, each being the lattice of continuous functions on the preceding lattice, with the pointwise partial ordering of functions. The lattices in this sequence will be written as $D_0, D_1, \ldots$, with D_0 as the initial lattice. Then the D_i:s are provided with *connecting maps* between each pair of lattices, defined recursively as follows.

$$\begin{array}{rcll}
\phi_0(x) & = & \lambda y \in D_0.x & x \in D_0 \\
\psi_0(x') & = & x'(\perp_{D_0}) & x' \in D_1 \\
\\
\phi_{n+1}(x) & = & \phi_n \circ x \circ \psi_n, & x \in D_{n+1} \\
\psi_{n+1}(x') & = & \psi_n \circ x' \circ \phi_n & x' \in D_{n+2}
\end{array}$$

The *inverse limit* of the D_i:s and the maps is

$$D_\infty = \{\langle x_n \rangle_0^\infty | x_n = \psi(x_{n+1}), x_n \in D_n\}.$$

The favourable properties of D_∞ are independent of D_0, as stated in [10], so we may select an arbitrary initial lattice, and then prove that our domain, D, is isomorphic to the resulting inverse limit[2].
The following construction is due to Jonsson in [3].

[2] It it not necessary to use the ϕ maps, but with them, we can express the domain as a *direct limit*. Therefore, we will also include those maps in the proofs.

Definition 2.21 *Let $\hat{0} \in P\omega$ be defined*[3] *by*

$$\hat{0} = \lambda x.0 \in x \to 0, \perp = \{(2n+1, 0) | n \in \omega\}.$$

Then define, inductively,

$$\begin{array}{rcl} \hat{0}_0 & = & \hat{0} \\ \hat{0}_{i+1} & = & \hat{0}_i \circ\!\!\to \hat{0}_i. \end{array}$$

■

Lemma 2.22 *For every $i \in \omega$, the following holds.*

$$\begin{array}{l} \hat{0}_i \text{ is a retract} \\ \hat{0}_i \subseteq \hat{0}_{i+1} \end{array}$$

Proof. We refer to [3] for proofs. ■

The properties in lemma 2.22 are in fact what is required for Jonsson's construction to work. He shows that the we obtain Scott's reflexive domain in $P\omega$ (from [7]) by the normal reflexive retract

$$\hat{0}_\infty = \bigcup_{i=0}^{\infty} \hat{0}_i,$$

and by our preceding uniqueness result, $d = \hat{0}_\infty$, and our D is the same domain. Now, we may relate this construction to the inverse limit construction.

Theorem 2.23 *D and D_∞, obtained with $\{\perp, 0\}$ as the initial lattice D_0, are lattice isomorphic.*

Proof. The fixed points of $\hat{0}_0$ are $\{\perp, 0\}$, so the sequence of domains in definition 2.21 is the same as the domain sequence in the inverse limit construction, i.e. $\hat{0}_i$ has the elements in D_i as fixed points. Now a theorem by Sanchis[4] in [4] on page 353, section 4, tells us that the domain defined by the retract $\hat{0}_\infty$ above will be isomorphic to the inverse limit of the domains defined by the $\hat{0}_i$:s, using as connecting maps (corresponding to the maps defined above)

$$\begin{array}{rcll} \phi'_0(x) & = & \hat{0}_1(x) & x \in D_0 \\ \psi'_0(x') & = & \hat{0}_0(x') & x' \in D_1 \\ \\ \phi'_{n+1}(x) & = & \hat{0}_{n+2}(x) & x \in D_{n+1} \\ \psi'_{n+1}(x') & = & \hat{0}_{n+1}(x') & x' \in D_{n+2}, \end{array}$$

so what we have to prove is that these maps are the same as the original connecting maps ϕ_i and ψ_i above.
We encode the domain constraints above by retracts, and so we want to prove

$$\begin{array}{rcl} \hat{0}_1(\hat{0}(x)) & = & \lambda y.\hat{0}(x) \\ \hat{0}_0(\hat{0}_1(x')) & = & \hat{0}_1(x')(\perp) \\ \\ \hat{0}_{n+2}(\hat{0}_{n+1}(x)) & = & \hat{0}_{n+1} \circ \hat{0}_{n+1}(x) \circ \hat{0}_n \circ \hat{0}_{n+1} \\ \hat{0}_{n+1}(\hat{0}_{n+2}(x')) & = & \hat{0}_n \circ \hat{0}_{n+2}(x') \circ \hat{0}_{n+1} \circ \hat{0}_n \end{array}$$

[3] Using the language *LAMBDA*. Note that 0 stands for the *set* $\{0\}$ in *LAMBDA*.
[4] A similar result is stated by Scott in [7], but with no proof included.

for all $x, x' \in P\omega$. We first prove the basis cases. First suppose $0 \in x$.

$$\begin{array}{rcl} \hat{0}_1(\hat{0}(x)) & = & \hat{0}_1(0) \\ & = & \hat{0} \circ (0,0) \circ \hat{0} \\ & = & \lambda x.\hat{0}(0) \\ & = & \lambda x.\hat{0}(x) \end{array}$$

Now suppose $0 \notin x$.

$$\begin{array}{rcl} \hat{0}_1(\hat{0}(x)) & = & \hat{0}_1(\perp) \\ & = & \hat{0} \circ \perp \circ \hat{0} \\ & = & \lambda x.\hat{0}(\perp) \\ & = & \lambda x.\hat{0}(x) \end{array}$$

Suppose $0 \in x'$.

$$\begin{array}{rcll} \hat{0}_0(\hat{0}_1(x')) & = & \hat{0}(0) & \\ & = & 0 & \\ \hat{0}_1(x')(\perp) & = & \hat{0}(x'(\hat{0}(\perp))) & \\ & = & \hat{0}(x'(\perp)) & \\ & = & \hat{0}(0) & 0 = (0,0) \in x' \\ & = & 0 & \end{array}$$

Suppose $0 \notin x'$.

$$\begin{array}{rcl} \hat{0}_0(\hat{0}_1(x')) & = & \hat{0}(\lambda u.\hat{0}(x'(\hat{0}(u)))) \\ & = & \hat{0}(\{(n,m) | m \in \hat{0}(x'(\hat{0}(e_n)))\}) \end{array}$$

By definition 2.21, this can only be $\neq \perp$ if

$$0 \in \hat{0}(x'(\hat{0}(\emptyset))),$$

and since $\hat{0}$ is strict and $0 \notin x'$,

$$\hat{0}_0(\hat{0}_1(x')) = \perp.$$

We also have

$$\begin{array}{rcl} \hat{0}_1(x')(\perp) & = & \hat{0}(x'(\hat{0}(\perp))) \\ & = & \hat{0}(x'(\perp)) \\ & = & \hat{0}(\perp) \\ & = & \perp. \end{array}$$

We now prove the remaining equalities.

$$\begin{array}{rcll} \hat{0}_{n+2}(\hat{0}_{n+1}(x)) & = & \hat{0}_{n+1} \circ \hat{0}_{n+1}(x) \circ \hat{0}_{n+1} & \\ & = & \hat{0}_{n+1} \circ \hat{0}_n \circ x \circ \hat{0}_n \circ \hat{0}_{n+1} & \\ & = & \hat{0}_{n+1} \circ \hat{0}_n \circ x \circ \hat{0}_n \circ \hat{0}_n \circ \hat{0}_{n+1} & \hat{0}_n \text{ is a retract} \\ & = & \hat{0}_{n+1} \circ \hat{0}_{n+1}(x) \circ \hat{0}_n \circ \hat{0}_{n+1} & \end{array}$$

$$\begin{array}{rcll} \hat{0}_{n+1}(\hat{0}_{n+2}(x')) & = & \hat{0}_n \circ \hat{0}_{n+2}(x') \circ \hat{0}_n & \\ & = & \hat{0}_n \circ \hat{0}_{n+1} \circ x' \circ \hat{0}_{n+1} \circ \hat{0}_n & \\ & = & \hat{0}_n \circ \hat{0}_{n+1} \circ x' \circ \hat{0}_{n+1} \circ \hat{0}_{n+1} \circ \hat{0}_n & \hat{0}_{n+1} \text{ is a retract} \\ & = & \hat{0}_n \circ \hat{0}_{n+2}(x') \circ \hat{0}_{n+1} \circ \hat{0}_n & \end{array}$$

■

Chapter 3

Domain Structure

In this chapter, we will investigate the structure of the "lower part" of the lattice domain D. We will prove that there are no "second least" elements, and also exhibit a procedure for deciding equality between basis elements of the domain.

3.1 Basic Properties

Here we show some properties of the reflexive domain D that will be used in later proofs.

Definition 3.1 *Let* $int : \omega \times \omega \to \omega$ *be defined by*

$$\begin{array}{l} int(0,a) = int(a,0) = a \\ int((s_1,t_1),(s_2,t_2)) = (s, int(t_1,t_2)) \quad \textit{where } e_s = e_{s_1} \cup e_{s_2}. \end{array}$$

■

By remark 1.2, the function is well defined.

Remark 3.2 Note that the second case in the definition coincides with the first when $(s_i,t_i) = 0$ for every $i \in \{1,2\}$. ■

This function is of course meant to calculate intersections, confirmed in the following theorem.

Theorem 3.3

$$\forall a,b \in \omega : d(a) \cap d(b) = d(int(a,b))$$

Proof. Induction on $a+b$.
Set $a = (s_1,t_1)$ and $b = (s_2,t_2)$. If $t_1 < (s_1,t_1)$ or $t_2 < (s_2,t_2)$ then $t_1 + t_2 < a+b$, and by induction we get

$$\begin{array}{lll} & d(s_1,t_1) \cap d(s_2,t_2) = & \\ = & \{(n,m) | m \in d(t_1) \wedge e_{s_1} \subseteq d(e_n)\} \cap & \text{lem. 2.3} \\ & \{(n,m) | m \in d(t_2) \wedge e_{s_2} \subseteq d(e_n)\} & \\ = & \{(n,m) | m \in d(t_1) \cap d(t_2) \wedge e_{s_1} \cup e_{s_2} \subseteq d(e_n)\} & \\ = & \{(n,m) | m \in d(int(t_1,t_2)) \wedge e_s \subseteq d(e_n)\}, \text{ where } e_s = e_{s_1} \cup e_{s_2} & \\ = & d(s, int(t_1,t_2)) & \text{lem. 2.3} \\ = & d(int((s_1,t_1),(s_2,t_2))). & \text{def. 3.1} \end{array}$$

Otherwise, we have

$$t_1 = 0 \lor t_2 = 0.$$

If $t_1 = 0$, we get

$$\begin{array}{lll}
& d(s_1, t_1) \cap d(s_2, t_2) = d(s_1, 0) \cap d(s_2, t_2) = & \\
= & \{(n, m) | m \in d(0) \land e_{s_1} \subseteq d(e_n)\} \cap & \\
& \{(n, m) | m \in d(t_2) \land e_{s_2} \subseteq d(e_n)\} & \\
= & \{(n, m) | m \in d(t_2) \land e_s \subseteq d(e_n)\},\ \text{where } e_s = e_{s_1} \cup e_{s_2} & \text{lem. 2.11} \\
= & d(s, t_2) & \text{lem. 2.3} \\
= & d(s, \mathit{int}(0, t_2)) & \\
= & d(s, \mathit{int}(t_1, t_2)) & \text{since } t_1 = 0 \\
= & d(\mathit{int}((s_1, t_1), (s_2, t_2))), & \text{def. 3.1}
\end{array}$$

and the same argument applies when $t_2 = 0$. ∎

Lemma 3.4

$$(\forall n : e_s \subseteq d(e_n) \Leftrightarrow e_t \subseteq d(e_n)) \Leftrightarrow d(e_s) = d(e_t)$$

Proof.
$\Rightarrow$) Suppose

$$\forall n : e_s \subseteq d(e_n) \Leftrightarrow e_t \subseteq d(e_n).$$

Specialise to $n = s$, yielding

$$e_s \subseteq d(e_s) \Leftrightarrow e_t \subseteq d(e_s). \tag{3.1}$$

Since d is a finitary closure by corollary 2.13, we have

$$e_s \subseteq d(e_s),$$

and by (3.1),

$$e_t \subseteq d(e_s).$$

d being a retract, we get

$$d(e_t) \subseteq d(d(e_s)) = d(e_s),$$

and similarly, setting $n = t$, we get

$$d(e_s) \subseteq d(e_t),$$

whence

$$d(e_s) = d(e_t)$$

follows.
$\Leftarrow$) Suppose

$$d(e_s) = d(e_t). \tag{3.2}$$

Take any n and suppose

$$e_s \subseteq d(e_n).$$

Since d is a finitary closure, we have as before

$$d(e_s) \subseteq d(e_n),$$

and by (3.2) we get

$$d(e_t) \subseteq d(e_n). \tag{3.3}$$

Since d is a finitary closure,

$$e_t \subseteq d(e_t),$$

and by (3.3),

$$e_t \subseteq d(e_n).$$

The converse follows analogously. ■

Theorem 3.5

$$d(s_1, t_1) = d(s_2, t_2) \Leftrightarrow d(e_{s_1}) = d(e_{s_2}) \wedge d(t_1) = d(t_2)$$

Proof. By lemma 2.3, we get

$$\begin{array}{rcll} d(s_1,t_1) = d(s_2,t_2) & \Leftrightarrow & \{(n,m) | m \in d(t_1) \wedge e_{s_1} \subseteq d(e_n)\} = & \\ & & \{(n,m) | m \in d(t_2) \wedge e_{s_2} \subseteq d(e_n)\} & \\ & \Leftrightarrow & d(t_1) = d(t_2) \wedge & \\ & & \forall n : (e_{s_1} \subseteq d(e_n) \Leftrightarrow e_{s_2} \subseteq d(e_n)) & \\ & \Leftrightarrow & d(t_1) = d(t_2) \wedge d(e_{s_1}) = d(e_{s_2}) & \text{lem. 3.4} \end{array}$$

■

3.2 Second Least Elements

In [3], Jonsson shows that there exists a unique *second greatest element* in D, which is in fact $\mathcal{Z}_+$, the set of all positive integers, and states some alternatives as to the possibilities of the existence of *second least elements.*
Soon, we shall show that there cannot exist any second least elements. To be strict, first we have to define the notion of a set of second least elements. This is a slightly modified version of the definition from [3].

Definition 3.6 *A set of second least elements of a domain $\mathcal{D}$ is a subset $\mathcal{S}$ of D satisfying the following conditions.*

1. $\forall f \in \mathcal{D}, s \in \mathcal{S} : f \subset s \Rightarrow f = \bot$
2. $\forall f \in \mathcal{D} \exists s \in \mathcal{S} : f \neq \bot \Rightarrow s \subseteq f$

■

Lemma 3.7 *If there exists a set $\mathcal{S}$ of second least elements of D, then for each $s \in \mathcal{S}$ we must have $s = d(k)$ for some $k \in \omega$.*

Proof. Suppose $s \in \mathcal{S}$. If $s = d(u), u \in P\omega$, then by monotonicity, we must have $d(k) \subseteq d(u)$ for every $k \in u$, and since d is a finitary closure by corollary 2.13, $k \in d(k) \neq \bot$ for those k. Thus, since $s \in \mathcal{S}$, we must have $d(u) = d(k)$ for every $k \in u$. ■

Definition 3.8 *Let $below : \omega \to \omega$ be defined as follows.*

$$\begin{array}{lcll} below(0) & = & (1,0) = 1 & \\ below((a,r)) & = & (1, below(r)) & \text{for } (a,r) > 0 \end{array}$$

■

By remark 1.2, the function is well defined.

Lemma 3.9

$$\forall x \in \omega : d(below(x)) \subset d(x)$$

Proof. Induction on x.
For $x = 0$, we have

$$d(x) = d(0) = \top$$

by lemma 2.11, and

$$\begin{array}{rcll} d(below(x)) & = & d(below(0)) & \\ & = & d(1,0) & \\ & = & \{(n,m) | m \in d(0) \wedge 0 \in d(e_n)\} & \text{lem. 2.3} \\ & = & \{(n,m) | 0 \in e_n\} & \text{lem. 2.11} \\ & = & \{(2n+1, m)\} & \\ & \subset & \{(n,m)\} & \\ & = & \top & \\ & = & d(0) & \\ & = & d(x). & \end{array}$$

Now set $x = (a,r)$ with $x > 0$. By the coding properties, we have $r < x$. Thus, by induction, we know

$$d(below(r)) \subset d(r). \tag{3.4}$$

Now,

$$\begin{array}{rcll} d(x) & = & d(a,r) & \\ & = & \{(n,m) | m \in d(r) \wedge e_a \subseteq d(e_n)\} & \text{lem. 2.3} \\ & \supseteq & \{(n,m) | m \in d(r) \wedge \top = d(e_n)\} & \\ & = & \{(n,m) | m \in d(r) \wedge 0 \in d(e_n)\} & \text{lem. 2.11} \\ & \supset & \{(n,m) | m \in d(below(r)) \wedge 0 \in d(e_n)\} & \text{by (3.4)} \\ & = & d(1, below(r)) & \text{lem. 2.3} \\ & = & d(below(a,r)) & \\ & = & d(below(x)). & \end{array}$$

■

Theorem 3.10 *There is no set $\mathcal{S}$ of second least elements of D.*

Proof. Suppose that $\mathcal{S}$ is a set of second least elements of D. Since D is nonempty, we may take an $f \in D$ such that $f \neq \bot$. Then by definition 3.6, there exists an $s \in \mathcal{S}$, such that $s \subseteq f$. Fix such an s. By lemma 3.7, we must have $s = d(k)$ for some $k \in \omega$. But by lemma 3.9, $d(below(k)) \subset d(k)$, and since d is a finitary closure by corollary 2.13, $d(below(k)) \neq \bot$, contradicting definition 3.6. ■

3.3 Relations Between Basis Elements of the Domain

For computational purposes, one would like to have a finite, canonical representation of the basis elements of the domain. Here we exhibit such a representation, together with procedures for deciding equality and subset relations between basis elements.
We will see the advantage of the domain D over Scott's D_∞, since we use the classification of basis elements into singleton sets and other finite sets. It is not easy to discern which those elements are in D_∞.
Now, we will define some auxiliary functions, in order to obtain an algorithm for deciding whether $d(e_s) = d(e_t)$ for basis elements e_s and e_t.

Definition 3.11 *Let* $clean : \omega \to \omega$, $purify : P_F\omega \to P_F\omega$ *and* $subset : \omega \times \omega \to \{\mathcal{T}, \mathcal{F}\}$ *be defined by the following mutual recursion.*

$$\begin{array}{lcl} clean(0) & = & 0 \\ clean((s,r)) & = & (t, clean(r)) \text{ where } e_t = purify(e_s) \\ \\ purify(e_s) & = & \{clean(x) | x \in e_s \wedge \\ & & \neg \exists w \in e_s : (clean(x) \neq clean(w) \wedge subset(x,w))\} \\ \\ subset(x,y) & \Leftrightarrow & clean(int(x,y)) = clean(x) \end{array}$$

■

Remark 3.12 As in remark 3.2, in the case when $s = r = 0$, the first and the second defining clauses for *clean* coincide, so there is no need to impose any restrictions on s nor r in the second clause. ■

Notation 3.13 *As in notation 2.2, for convenience, we write* $clean((s,t))$ *as* $clean(s,t)$*, and later,* $Q(a,b)$ *for* $Q((a,b))$*.* ■

Discussion 3.14 The intuition behind these definitions is as follows:
$clean(x)$ is meant to generate a canonical representative for $d(\{x\})$, thus being a finite canonical representation for the elements of the reflexive domain on the form $d(\{x\})$. A desired property of *clean* is of course that

$$clean(x) = clean(y) \Leftrightarrow d(x) = d(y).$$

purify is the generalisation of *clean* to arbitrary basis elements, which should satisfy

$$purify(e_s) = purify(e_t) \Leftrightarrow d(e_s) = d(e_t).$$

The definition of *purify* should be understood like this: *clean* every element in e_s; then remove “redundant” elements from the resultning set, i.e. elements that do not affect the value of $d(e_s)$.
$subset(x,y)$ is supposed to express the property that

$$d(x) \subseteq d(y)$$

for $x, y \in \omega$.
The corresponding subset relation on basis elements will not be explicitly defined, since it is not used in the definitions, but it can be stated in terms of *purify*, as will later be shown. ■

All these desired properties will be proved in due course, but in order to make the work easier, we need an induction principle.

Lemma 3.15 *If P and Q are predicates, on $P_F\omega$ and ω respectively, such that*

$$\begin{array}{l} Q(0) \\ \forall a, r : P(e_a) \wedge Q(r) \Rightarrow Q(a, r) \\ \forall s : ((\forall x < s : Q(x)) \Rightarrow P(e_s)) \end{array}$$

are satisfied, then

$$\forall x : ((\forall s < x : P(e_s)) \Rightarrow Q(x)).$$

Proof. Assume that P and Q satisfies the preconditions.
Induction on x.
Suppose $x = 0$. Then we know that $Q(x)$ holds.
Now suppose, for any $x = (a, r) > 0$, that

$$\forall s < (a, r) : P(e_s). \tag{3.5}$$

We know

$$r < (a, r) = x;$$

thus we know by induction that

$$(\forall s < r : P(e_s)) \Rightarrow Q(r), \tag{3.6}$$

and by (3.5), we have

$$\forall s < r : P(e_s),$$

and this combined with (3.6) yields $Q(r)$ for this r.
Furthermore, we have

$$a < (a, r) \vee a = 1.$$

In the first case, by (3.5), we know $P(e_a)$. If $a = 1$, then since we know $Q(0)$, by the third precondition, $P(e_a)$ holds. We now have

$$P(e_a) \wedge Q(r),$$

and by the second precondition, we conclude

$$Q(a, r).$$

■

Theorem 3.16 *If P and Q are predicates, on $P_F\omega$ and ω respectively, such that*

$$\begin{array}{l} Q(0) \\ \forall a, r : P(e_a) \wedge Q(r) \Rightarrow Q(a, r) \\ \forall s : ((\forall x < s : Q(x)) \Rightarrow P(e_s)) \end{array}$$

are satisfied, then

$$(\forall x : Q(x)) \wedge (\forall s : P(e_s)).$$

Proof. First, we show

$$\forall s : P(e_s)$$

by induction on s.
$P(e_0)$ holds trivially by the third precondition.
Now suppose $s > 0$. By induction, we know

$$\forall a < s : P(e_a), \tag{3.7}$$

and since (3.7) implies

$$\forall x \leq s \forall a < x : P(e_a),$$

by lemma 3.15 we have

$$\forall x \leq s : Q(x).$$

By the third precondition, we easily get $P(e_s)$.
We now show

$$\forall x : Q(x),$$

by induction on x.
For $x = 0$, we already know $Q(x)$ by the first precondition.
For $x = (a, r) > 0$, we have

$$r < (a, r).$$

Then, by induction, we know $Q(r)$. We already have shown $P(e_a)$ for all a, and thus, by the second precondition, we have $Q(a, r)$. ∎

Soon, we will use this induction principle, with

$$\begin{array}{lcll} Q(x) & \Leftrightarrow & d(clean(x)) = d(x) & \text{and} \\ P(e_s) & \Leftrightarrow & d(purify(e_s)) = d(e_s). & \end{array}$$

First, we show that the "simple" properties of Q and P are satisfied by the above substitutions.

Lemma 3.17

$$d(clean(0)) = d(0)$$

and

$$d(purify(e_a)) = d(e_a) \wedge d(clean(r)) = d(r) \Rightarrow d(clean(a, r)) = d(a, r).$$

Proof. By definition 3.11,

$$d(clean(0)) = d(0).$$

Also by definition 3.11,

$$d(clean(a, r)) = d(t, clean(r)) \text{ where } e_t = purify(e_a),$$

and by theorem 3.5, we get

$$d(purify(e_a)) = d(e_a) \wedge d(clean(r)) = d(r) \Rightarrow d(clean(a, r)) = d(a, r).$$

∎

To apply theorem 3.16, it remains to show

$$\forall s : ((\forall a < s : d(clean(a)) = d(a)) \Rightarrow d(purify(e_s)) = d(e_s)), \tag{3.8}$$

but for this we will need a few lemmas. In fact, we will first prove a weaker result, where we do not use the *subset* predicate in the definition of *purify*, since the proof will be a lot more readable. Instead of the *subset* predicate, we will use the intended meaning of $subset(x, y)$, i.e. $d(x) \subseteq d(y)$, together with the intended meaning of *clean* in *purify*, yielding $d(x) \subset d(y)$. With that simplification, we get the following modification of definition 3.11, with the original functions primed. Later, we will prove that the primed and unprimed functions are the same.

Definition 3.18 *Let* $clean' : \omega \to \omega$ *and* $purify' : P_F\omega \to P_F\omega$ *be defined by the following mutual recursion.*

$$\begin{aligned} clean'(0) &= 0 \\ clean'(s, r) &= (t, clean'(r)) \text{ where } e_t = purify'(e_s) \\ \\ purify'(e_s) &= \{clean'(x) | x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\} \end{aligned}$$

■

Lemma 3.19 *The properties stated in lemma 3.17 hold for* $clean'$ *and* $purify'$ *respectively.*

Proof. In the proof of lemma 3.17, the property of *purify* using *subset* is not used. ■

Restating (3.8) in terms of the primed functions, now we of course want to show the following.

Lemma 3.20

$$\forall s : (\forall a < s : d(clean'(a)) = d(a)) \Rightarrow d(purify'(e_s)) = d(e_s)$$

Proof. Suppose, for some s, that

$$\forall a < s : d(clean'(a)) = d(a). \tag{3.9}$$

We get

$$\begin{array}{lll} & d(purify'(e_s)) = & \\ = & d(\{clean'(x) | x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\}) & \text{def. 3.18} \\ = & \{d(clean'(x)) | x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\} & \text{th. 2.10} \\ = & \{d(x) | x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\} & \text{by (3.9), since } x < s \\ = & d(\{x | x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\}) & \text{th. 2.10} \\ \subseteq & d(e_s). & \end{array}$$

We now have to show the converse, i.e. that

$$d(e_s) \subseteq d(purify'(e_s)).$$

Pick an arbitrary $x \in d(e_s)$. Then

$$\exists y \in e_s : x \in d(y)$$

since d is additive. Choose a maximal y among those y:s, i.e. a y such that

$$\neg\exists w \in e_s : d(y) \subset d(w). \tag{3.10}$$

This is possible, since e_s is a finite set. If we knew that

$$clean'(y) \in purify'(e_s),$$

then since $y < s$, by (3.9) and by monotonicity we would have

$$x \in d(y) = d(clean'(y)) \subseteq d(purify'(e_s)).$$

Therefore, suppose

$$clean'(y) \notin purify'(e_s).$$

$$\begin{array}{lll} & clean'(y) \notin purify'(e_s) & \\ \Leftrightarrow & \neg \exists z \in e_s : (clean'(z) = clean'(y) \wedge \neg \exists w \in e_s : d(z) \subset d(w)) & \text{def. 3.18} \\ \Leftrightarrow & \forall z \in e_s : (clean'(z) \neq clean'(y) \vee \exists w \in e_s : d(z) \subset d(w)) & \\ \Rightarrow & clean'(y) \neq clean'(y) \vee \exists w \in e_s : d(y) \subset d(w) & \text{set } z = y \\ \Rightarrow & \exists w \in e_s : d(y) \subset d(w), & \end{array}$$

which contradicts (3.10). ■

Theorem 3.21

$$\forall a : (d(a) = d(clean'(a))) \wedge \forall s : (d(purify'(e_s)) = d(e_s))$$

Proof. Using theorem 3.16 with

$$\begin{array}{rcll} Q(x) & \Leftrightarrow & d(clean'(x)) = d(x) & \text{and} \\ P(e_s) & \Leftrightarrow & d(purify'(e_s)) = d(e_s), & \end{array}$$

by lemma 3.19 and lemma 3.20, the result follows. ■

Before proving the same results for the unprimed functions, we need to prove a few properties of the primed functions.

Lemma 3.22 *Suppose that, for some fixed $s \in \omega$,*

$$\forall x, y < s : d(x) = d(y) \Rightarrow clean'(x) = clean'(y) \tag{3.11}$$

holds. Then also

$$\forall a, b < s : d(e_a) = d(e_b) \Leftrightarrow purify'(e_a) = purify'(e_b)$$

holds.

Proof.
$\Leftarrow$) Suppose

$$purify'(e_a) = purify'(e_b).$$

By theorem 3.21, we have

$$d(e_a) = d(purify'(e_a)) = d(purify'(e_b)) = d(e_b).$$

$\Rightarrow$) The case when $s = 0$ is trivial. Therefore suppose, for some fixed $s > 0$ and $a, b < s$, that

$$d(e_a) = d(e_b)$$

and without loss of generality, that

$$x \in purify'(e_a)$$

but

$$x \notin purify'(e_b). \tag{3.12}$$

Now,

$$\begin{array}{rcll} x & \in & purify'(e_a) & \\ & \subseteq & d(purify'(e_a)) & \text{cor. 2.13} \\ & = & d(e_a) & \text{th. 3.21} \\ & = & d(e_b) & \\ & = & d(purify'(e_b)). & \text{th. 3.21} \end{array}$$

By lemma 2.16 we then have

$$\exists y \in purify'(e_b) : x \in d(y),$$

and furthermore, that

$$x \neq y \tag{3.13}$$

for such y:s, by (3.12). Fix such a y. By a similar argument, we also for this y get

$$\exists z \in purify'(e_a) : y \in d(z).$$

Fix such a z. Since we now have

$$x \in d(y) \wedge y \in d(z),$$

since d is retract and by monotonicity, we get

$$d(x) \subseteq d(y) \subseteq d(z).$$

If we could show

$$d(x) \subset d(y)$$

and thus

$$d(x) \subset d(z),$$

we would get a contradiction by the following argument.

$$\begin{array}{lll} x \in purify'(e_a) & & \\ \quad \Rightarrow & \exists x' \in e_a : x = clean'(x') \wedge & \\ & \neg\exists w \in e_a : d(x') \subset d(w) & \text{def. 3.18} \\ \quad \Rightarrow & \neg\exists w \in e_a : d(x) \subset d(w) & \text{th. 3.21} \\ \quad \Rightarrow & \neg\exists z' \in e_a : clean'(z') = z \wedge d(x) \subset d(z') & \\ \quad \Rightarrow & \neg\exists z' \in e_a : clean'(z') = z \wedge d(x) \subset d(z) & \text{th. 3.21} \\ \quad \Rightarrow & \neg(d(x) \subset d(z)) & z \in purify'(e_a) \end{array}$$

Since we know

$$d(x) \subseteq d(y),$$

it suffices to show

$$d(x) \neq d(y).$$

For the sake of contradiction, suppose

$$d(x) = d(y). \tag{3.14}$$

We know

$$x = clean'(x') \wedge y = clean'(y')$$

for some $x' \in e_a$ and $y' \in e_b$, and also

$$x', y' < max(a, b) < s. \tag{3.15}$$

By theorem 3.21 and (3.14), we have

$$d(x') = d(clean'(x')) = d(x) = d(y) = d(clean'(y')) = d(y'),$$

and using (3.11), we get

$$x = y,$$

contradicting (3.13). Thus we conclude

$$d(x) \subset d(y)$$

contradicting (3.14), and the result follows. ■

Lemma 3.23

$$\forall x, y : d(x) = d(y) \Leftrightarrow clean'(x) = clean'(y)$$

Proof.
$\Leftarrow$) Suppose

$$clean'(x) = clean'(y).$$

By theorem 3.21, we have

$$d(x) = d(clean'(x)) = d(clean'(y)) = d(y).$$

$\Rightarrow$) We will show

$$\forall s \forall x, y < s : d(x) = d(y) \Rightarrow clean'(x) = clean'(y),$$

from which the result follows, since we may choose e.g.

$$s = max(x, y) + 1.$$

Induction on s.
The case when $s = 0$ is trivial. Now suppose $s > 0$ and for some x, y that

$$x, y < s \wedge d(x) = d(y).$$

Also set

$$x = (a, r), y = (b, t).$$

We thus have to prove

$$clean'(a, r) = clean'(b, t).$$

By theorem 3.5, and since

$$d(a, r) = d(b, t),$$

we know

$$d(e_a) = d(e_b) \wedge d(r) = d(t), \tag{3.16}$$

and by definition 3.18 that

$$\begin{aligned} & clean'(a, r) = clean'(b, t) \\ \Leftrightarrow \quad & purify'(e_a) = purify'(e_b) \wedge clean'(r) = clean'(t). \end{aligned} \tag{3.17}$$

If $r < (a, r)$ and $t < (b, t)$, then since

$$r, t < max(x, y) < s,$$

by induction, we get

$$clean'(r) = clean'(t).$$

Otherwise,

$$r = 0 \vee t = 0,$$

and since we thus by lemma 2.11 and (3.16) must have

$$d(r) = d(t) = \top,$$

and by lemma 2.15 we get

$$r = t = 0,$$

and by definition 3.18,

$$clean'(r) = clean'(t) = 0.$$

For the induction, it remains to show

$$purify'(e_a) = purify'(e_b).$$

Consider the case when

$$a < (a, r) < s \wedge b < (b, t) < s. \tag{3.18}$$

Then

$$a, b < max(x, y) < s,$$

and setting

$$t = max(x, y),$$

by induction, we have

$$\forall x, y < t : d(x) = d(y) \Rightarrow clean'(x) = clean'(y).$$

Thus the premise to lemma 3.22 is satisfied setting s in (3.11) to t. Thus we get

$$\forall a, b < t : d(e_a) = d(e_b) \Rightarrow purify'(e_a) = purify'(e_b).$$

By (3.16), and since

$$a, b < t$$

we get

$$purify'(e_a) = purify'(e_b).$$

If (3.18) does not hold, then we must have

$$a \leq 1 \vee b \leq 1.$$

If $a = 0$, then we must have $b = 0$, since by lemma 2.7, d is strict, and by corollary 2.13, d is a finitary closure. Further,

$$a = b = 0 \Rightarrow purify'(e_a) = purify'(e_b) = \emptyset,$$

by definition 3.18.
If $a = 1$, then $e_a = \{0\}$, and from definition 3.18, we get

$$purify'(e_a) = \{0\},$$

and since

$$d(e_a) = d(0) = \top,$$

by (3.16) and lemma 2.15,

$$0 \in e_b,$$

but since we know

$$b \leq 1,$$

we must have

$$b = 1.$$

By definition 3.18, we then must have

$$purify'(e_b) = \{0\}.$$

Now, (3.17) can be applied, and we have

$$clean'(x) = clean'(y).$$

■

Corollary 3.24

$$\forall a, b : d(e_a) = d(e_b) \Leftrightarrow purify'(e_a) = purify'(e_b)$$

Proof. The premise to lemma 3.22 is satisfied for all $s \in \omega$, by lemma 3.23. ■

Now, we are about to prove the "unprimed analogue" of theorem 3.21. We will again use the induction principle from theorem 3.16, but now with

$$\begin{array}{lcll} Q(x) & \Leftrightarrow & clean(x) = clean'(x) & \text{and} \\ P(e_s) & \Leftrightarrow & purify(e_s) = purify'(e_s). & \end{array}$$

Lemma 3.25

$$\begin{array}{l} clean(0) = clean'(0) \\ purify(e_a) = purify'(e_a) \wedge clean(r) = clean'(r) \Rightarrow \\ \qquad clean(a, r) = clean'(a, r) \end{array}$$

Proof. By definition 3.11 and definition 3.18,

$$clean(0) = clean'(0)$$

trivially holds.
Suppose

$$purify(e_a) = purify'(e_a) \wedge clean(r) = clean'(r). \tag{3.19}$$

We have

$$\begin{array}{rcll} clean(a, r) & = & (t, clean(r)), e_t = purify(e_a) & \\ & = & (t', clean'(r)), e_{t'} = purify'(e_a) & \text{by (3.19)} \\ & = & clean'(a, r). & \end{array}$$

■

Lemma 3.26

$$\forall s : (\forall x < s : clean(x) = clean'(x)) \Rightarrow purify(e_s) = purify'(e_s)$$

Proof. In the proof, we will use the abbreviations c and p for *clean* and *purify* respectively, and c' and p' for the primed functions[1].
Indeed, we will prove the stronger result that

$$\begin{array}{ll} \forall s : & ((\forall x < s : (c(x) = c'(x)) \Rightarrow p(e_s) = p'(e_s)) \wedge \\ & (\forall x, y < s : c(int(x, y)) = c'(int(x, y)))), \end{array} \tag{3.20}$$

from which the result follows.
Induction on s. The case when $s = 0$ is trivial.
Suppose that $s > 0$ and

$$\forall x < s : c(x) = c'(x). \tag{3.21}$$

We have

[1] Since the formulae tend to consume all available page space.

$$\begin{aligned}
p(e_s) = \{c(x)|x \in e_s \wedge \neg\exists w \in e_s : (c(x) \neq c(w) \wedge subset(x,w))\} = & \\
= \quad \{c(x)|x \in e_s \wedge \neg\exists w \in e_s : (c(x) \neq c(w) \wedge c(x) = c(int(x,w)))\} & \\
\text{def. 3.11} & \\
= \quad \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (c'(x) \neq c'(w) \wedge c'(x) = c(int(x,w)))\} & \\
\text{by (3.21), since } x, w < s. &
\end{aligned}$$

Furthermore,

$$\begin{aligned}
p'(e_s) &= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : d(x) \subset d(w)\} = \\
&= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (d(x) \neq d(w) \wedge d(x) \subseteq d(w))\} \\
&= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (d(x) \neq d(w) \wedge d(x) = d(x) \cap d(w))\} \\
&= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (d(x) \neq d(w) \wedge d(x) = d(int(x,w)))\} && \text{th. 3.3} \\
&= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (d(x) \neq d(w) \wedge d(x) = d(int(x,w)))\} && \text{th. 3.3} \\
&= \{c'(x)|x \in e_s \wedge \neg\exists w \in e_s : (c'(x) \neq c'(w) \wedge c'(x) = c'(int(x,w)))\} && \text{lem. 3.23,}
\end{aligned}$$

and thus it is enough to show that

$$\forall x, y < s : c(int(x,y)) = c'(int(x,y)) \tag{3.22}$$

to know that

$$p'(e_s) = p(e_s).$$

But if we show (3.22), then we will also have shown (3.20), and the proof would be complete.
Set

$$x = (a, r), y = (b, t)$$

for some

$$x, y < s.$$

We expand the *int*-expression one step.

$$\begin{aligned}
int(x,y) &= int((a,r),(b,t)) \\
&= (u, int(r,t)) \text{ where } e_u = e_a \cup e_b.
\end{aligned}$$

By definition 3.18, we have

$$c'(int(x,y)) = (v', c'(int(r,t))) \text{ where } e_{v'} = p'(e_a \cup e_b)$$

and by definition 3.11

$$c(int(x,y)) = (v, c(int(r,t))) \text{ where } e_v = p(e_a \cup e_b).$$

Thus, we have to show

$$c(int(r,t)) = c'(int(r,t)) \wedge p'(e_a \cup e_b) = p(e_a \cup e_b).$$

We now have

$$r, t < max(x,y) < s \vee r = t = 0,$$

and when

$$r = t = 0,$$

then

$$int(r,t) = 0$$

and by definition 3.11 and definition 3.18,

$$c(int(r,t)) = c(0) = 0 = c'(0) = c'(int(r,t)).$$

In the other case, when
$$r, t < max(x, y) < s,$$
by induction, we know
$$c(int(r, t)) = c'(int(r, t)).$$
So, it remains to show
$$p'(e_a \cup e_b) = p(e_a \cup e_b).$$
Set, as before
$$e_i = e_a \cup e_b.$$
By induction, we may now use the fact that
$$\forall w, z \in e_i : c(int(w, z)) = c'(int(w, z)),$$
since for these w, z we would have
$$w, z < max(a, b) \leq max(x, y) < s.$$
We will also use (3.21) in the proof.
$$\begin{aligned}
p(e_i) &= \{c(x) | x \in e_i \wedge \neg\exists w \in e_i : c(x) \neq c(w) \wedge c(x) = c(int(x, w))\} \\
&= \{c'(x) | x \in e_i \wedge \neg\exists w \in e_i : c'(x) \neq c'(w) \wedge c'(x) = c'(int(x, w))\} \\
&= \{c'(x) | x \in e_i \wedge \neg\exists w \in e_i : d(x) \neq d(w) \wedge d(x) = d(int(x, w))\} \\
&\qquad\qquad\qquad\qquad \text{lem. 3.23} \\
&= \{c'(x) | x \in e_i \wedge \neg\exists w \in e_i : d(x) \neq d(w) \wedge d(x) = d(x) \cap d(w)\} \\
&\qquad\qquad\qquad\qquad \text{th. 3.3} \\
&= \{c'(x) | x \in e_i \wedge \neg\exists w \in e_i : d(x) \subset d(w)\} \\
&= p'(e_i),
\end{aligned}$$
and we are done. ∎

Theorem 3.27
$$\forall x, s \in \omega : purify(e_s) = purify'(e_s) \wedge clean(x) = clean'(x)$$

Proof. By lemma 3.25 and lemma 3.26, we may use theorem 3.16 with
$$\begin{aligned}
Q(x) &\Leftrightarrow clean(x) = clean'(x) \qquad \text{and} \\
P(e_s) &\Leftrightarrow purify(e_s) = purify'(e_s),
\end{aligned}$$
and the result follows. ∎

Now, we have exhibited an algorithm for deciding when $d(e_s) = d(e_t)$, using the functions *clean* and *purify*. It is an algorithm, since the sets involved are finite, and since the functions in definition 3.11 are obviously recursive. However, some standard results need to be proved, including those carrying over from the results of the primed functions.

Corollary 3.28
$$\forall x : d(x) = d(clean(x))$$

Proof. By theorem 3.21 and theorem 3.27. ∎

Corollary 3.29
$$\forall s : d(e_s) = d(purify(e_s))$$

Proof. By theorem 3.21 and theorem 3.27. ∎

Corollary 3.30

$$\forall x, y : d(x) = d(y) \Leftrightarrow clean(x) = clean(y)$$

Proof. By lemma 3.23 and theorem 3.27. ■

Corollary 3.31

$$\forall s, t : d(e_s) = d(e_t) \Leftrightarrow purify(e_a) = clean(e_t)$$

Proof. By corollary 3.24 and theorem 3.27. ■

Corollary 3.32

$$\forall x, y : subset(x, y) \Leftrightarrow d(x) \subseteq d(y)$$

Proof.

$$\begin{array}{lcll} subset(x,y) & \Leftrightarrow & clean(x) = clean(int(x,y)) & \\ & \Leftrightarrow & d(x) = d(int(x,y)) & \text{cor. 3.30} \\ & \Leftrightarrow & d(x) = d(x) \cap d(y) & \text{th. 3.3} \\ & \Leftrightarrow & d(x) \subseteq d(y) & \end{array}$$

■

Corollary 3.33

$$clean(clean(x)) = clean(x)$$

Proof. Suppose

$$clean(clean(x)) \neq clean(x).$$

Then

$$\begin{array}{lcll} d(x) & = & d(clean(x)) & \text{cor. 3.28} \\ & \neq & d(clean(clean(x))) & \text{cor. 3.30} \\ & = & d(clean(x)) & \text{cor. 3.28} \\ & = & d(x) & \text{cor. 3.28,} \end{array}$$

which is impossible. ■

Corollary 3.34

$$purify(purify(e_s)) = purify(e_s)$$

Proof. Proved like corollary 3.33, using corollary 3.29 and corollary 3.31. ■

Now that we have a method for deciding whether

$$d(e_s) = d(e_t),$$

of course we should use it for deciding whether

$$d(e_s) \subseteq d(e_t).$$

Corollary 3.35

$$d(e_s) \subseteq d(e_t) \Leftrightarrow purify(e_t) = purify(e_s \cup e_t)$$

Proof.

$$\begin{array}{lcl} d(e_s) \subseteq d(e_t) & \Leftrightarrow & d(e_t) = d(e_s \cup e_t) \\ & \Leftrightarrow & purify(e_t) = purify(e_s \cup e_t) \end{array}$$

■

Chapter 4

Conclusions

4.1 Summary

We have found that the reflexive domain in $P\omega$ is a good domain, since its structure is made more explicit than in the D_∞ models, and since it is also isomorphic to one of those models, which are already known to have nice properties.
Also, we saw that this domain is unique, in the sense that it is the only nontrivial solution to the retract equation

$$d = d \circ\!\!\rightarrow d,$$

but this does not necessarily mean that it is the only domain as a continuous lattice with a countable basis, isomorphic to its space of continuous functions, that can be embedded in $P\omega$.
We also exhibited a finite, canonical representation of the basis elements of the reflexive domain.

4.2 Future Work

Several questions about the domain have arised.

- Is this domain also unique in the sense that it is the only nontrivial domain as a continuous lattice with a countable basis that is isomorphic to its (continuous) function space?
- Is it possible to characterise the uniqueness result in category theory, or perhaps in purely topological terms?
- If we use the retract constructions of domains for denotational semantics of programming languages, will we also get a corresponding uniqueness?
- Can we use *clean* and *purify* to prove equivalences between λ-expressions? Here, one may try to find a parallel with Wadsworth's *approximate normal forms* in [10].
- Can we extract other properties of the domain using Jonsson's reflexive retract δ (in [3]), that by our result defines the same domain as d defines?

Bibliography

[1] James Dugundji. *Topology*. Allyn and Bacon, Inc., Boston, 1966.

[2] R. Hindley and G. Longo. Lambda Calculus Models and Extensionality. *Z. Math. Logik Grundlag. Math.*, 26, 1980.

[3] Peter Jonsson. Alternative Constructions of Reflexive Scott Domains. Master's thesis, Linköping University, 1993.

[4] L.E. Sanchis. Reflexive Domains. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 1980.

[5] Harold Schellinx. Isomorphisms and Nonisomorphisms of Graph Models. *The Journal of Symbolic Logic*, 56(1), 1991.

[6] D.S. Scott. Lattice-Theoretic Models for the λ-calculus. *Princeton Univ., Princeton, N.J., Unpublished*, 1969.

[7] D.S. Scott. Data Types as Lattices. *SIAM Journal on Computing*, 5(3), 1976.

[8] D.S. Scott. Relating Theories of the λ-Calculus. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 1980.

[9] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, 1977.

[10] Christopher P. Wadsworth. The Relation Between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda Calculus. *SIAM Journal on Computing*, 5(3), 1976.