Generalizing Domain Theory

Michael Mislove*

Tulane University, New Orleans, LA 70118, USA
e-mail: mwm@math.tulane.edu
WWW home page: http://www.math.tulane.edu/mislove.html

Abstract. Domain theory began in an attempt to provide mathematical models for high-level programming languages, an area where it has proved to be particularly useful. It is perhaps the most widely-used method for devising semantic models for such languages. This paper is a survey of some generalizations of domain theory that have arisen in efforts to solve related problems. In each case, a description is given of the problem and of the solution generalizing domain theory it inspired. The problems range from the relation of domain theory to other approaches for providing semantic models, particularly in process algebra, to issues surrounding the notion of a computational model, an approach inspired by the recent work of Abbas Edalat.

1 The Basics – How Domain Theory Began

This section is a brief outline of some of the "basic ingredients" of domain theory and the applications that inspired them.

1.1 *In the beginning...*

Domain theory began in an attempt by DANA SCOTT to find mathematical models for high-level programming languages. Upon his arrival in Oxford in the mid 1960s, Scott found CHRISTOPHER STRACHEY and his colleagues at the Programming Research Group using the untyped lambda calculus of Church and Curry as a model for programming, something Scott found disturbing because he regarded it as a "formal and unmotivated" notation (cf. [11]). He thus set out to find alternative models for Strachey and his colleagues to use. Because programs can call other programs, and indeed, can even call themselves, Scott was led to consider objects X in some category or other which satisfy the property that they contain a copy of their space of selfmaps in the category. Of course, Cantor's Lemma implies the only such objects in the category of sets and functions are *degenerate* (i.e., they consist of a single point), and so no such objects can be found there. But the reals have only as many continuous selfmaps as there are real numbers (because of their having a dense, countable subset on which all continuous selfmaps are completely determined), so it is potentially

* This work partially supported by the US Office of Naval Research

possible to find such objects X among topological spaces. While attempting to find appropriate models of partially defined maps, Scott realized there had to be T_0 spaces isomorphic to their space of continuous selfmaps, and he then constructed such an object in the category of algebraic lattices and so-called *Scott continuous* maps.

In the years since Scott constructed the first model of the untyped lambda calculus, the nature of the construction has become much better understood, and it now is realized that very little of the machinery that is available in the category of algebraic lattices and Scott continuous maps actually is necessary for the construction. In fact, it now is understood that only *directed complete partial orders* are needed to carry out the construction. Remarkably, despite this much better understanding, the only known models of the calculus are within the category of such partial orders and Scott continuous functions. We now describe this setting.

1.2 Directed Complete Partial Orders

Let's switch gears for a moment, and consider what we need to model a recursive process. If we are working within some language – let's not worry about typing issues – and we are confronted with a term $\mathtt{rec}\, x.f(x)$, then the operational rule which allows us to understand this process is given by

$$\mathtt{rec}\, x.f(x) \mapsto f[(\mathtt{rec}\, x.f(x))/x].$$

This *unwinding* of recursion allows us to deduce that the recursive process $\mathtt{rec}\, x.f(x)$ actually should be a *fixed point* for the body of the recursion. In any case, if we are to model programs as functions, then the unwinding rule tells us the functions we are interested in must have fixed points. In fact, it would be nice if those fixed points were *canonical* in some sense, so that their choice is not arbitrary. This is what domain theory offers us.

To begin, a *partial order* (or *poset*) is a non-empty set P equipped with a reflexive, symmetric and transitive relation, usually denoted $\sqsubseteq$. A simple example is to take any non-empty set X and equip it with the *discrete order*, where $x \sqsubseteq y \Leftrightarrow x = y$. A subset $D \subseteq P$ of such a set is *directed* if every finite subset of D has an upper bound *in* D. In our example, the only directed subsets are the singleton sets. Finally, a partial order P is *directed complete* if every directed subset has a least upper bound in P. These are called *dcpos*. Clearly, discrete orders satisfy this condition, since the only directed subsets are singleton sets. A directed complete partial order which has a least element $\bot$ (i.e., one which is below all other elements) is sometimes called a *cpo*.

What is important about cpos is that monotone[1] selfmaps have *least fixed points*:

Theorem 1 (Tarski). *A monotone mapping $f: P \to P$ of a cpo has a least fixed point, namely,*

$$\mathrm{FIX}\, f = \sqcup_{\alpha \in \mathrm{Ord}} f^{\alpha}(\bot). \qquad \square$$

[1] A map $f: P \to Q$ is *monotone* if $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

(Here, Ord stands for the class of ordinals.) Thus, a good place to seek models for recursion is within the category of cpos and monotone mappings. But, if we require our functions to preserve sups of directed sets, we can do better.

Definition 1. *A mapping $f: P \to Q$ between dcpos is* Scott continuous *if f is monotone*[2] *and f preserves sups of directed sets:*

$$(\forall D \subseteq P \text{ directed})\ f(\sqcup D) = \sqcup f(D).$$

Corollary 1 (Scott). *A Scott continuous selfmap $f: P \to P$ on a cpo has its least fixed point given by*

$$\mathrm{FIX}\, f = \sqcup_{n \in \mathbb{N}} f^n(\bot). \qquad \square$$

The category of directed complete partial orders and Scott continuous maps has many desirable properties – it is Cartesian closed, for example. We denote the family of continuous maps between (d)cpos P and Q by $[P \to Q]$; this space becomes a dcpo when endowed with the *pointwise order*, in which

$$f \sqsubseteq g \quad \Leftrightarrow \quad f(x) \sqsubseteq g(x)\ (\forall x \in P).$$

The full subcategory whose objects are cpos also is Cartesian closed, and it is within these categories where one can find ample support for constructing denotational models of programming languages. We even can conclude more here. Since the least fixed point of a monotone or continuous selfmap always exists, assigning it as the meaning of a recursive process is in some sense canonical. In fact,

Theorem 2. *The least fixed point operator* $\mathrm{Y}: [P \to P] \to P$ *by* $\mathrm{Y}f = \mathrm{FIX}\, f$ *is continuous.* $\square$

The implication here is that one has continuous fixed point operators of all orders, so modeling recursion at higher types can be done in the same way it is at the start. There is even a *transfer principle* available; it tells us that "fixed points are preserved" by certain operators:

Proposition 1. *Let $f: P \to P$ and $g: Q \to Q$ be continuous selfmaps of cpos P and Q, and let $h: P \to Q$ also be a continuous strict*[3] *map satisfying $g \circ h = h \circ f$. Then* $\mathrm{FIX}\, g = h(\mathrm{FIX}\, f)$. $\square$

The categories DCPO and CPO of directed complete partial orders (with least element in the second case) and Scott continuous maps thus enjoy several appealing properties. In addition to Cartesian closure, they also are closed under (arbitrary) products and direct sums. In addition, there is a closely-related adjunction. If $\mathsf{CPO}_!$ denotes the category of cpos and strict Scott continuous maps, then the forgetful functor into DCPO has the *lift functor* as left adjoint; this functor adds a (new) least element to a dcpo P and extends a continuous mapping to the lifted domains to be strict.

---

[2] This hypothesis is simply to guarantee that the image of a directed set in P is directed in Q.

[3] By *strict*, we mean h takes the least element of P to the least element of Q.

1.3 The Myhill-Sheperdson Theorem

The results described so far make an appealing, if somewhat abstract case for using domain theory to build models for programming languages – well, at least for modeling recursion. We now describe a result which puts more substance to this claim.

Perhaps the most natural place to start to model programs is over the natural numbers. In order to invoke a domain-theoretic setting, we can endow $\mathbb{N}$ with the discrete order, and clearly we have a dcpo. Our interest is in using domain theory to model computable functions. Functions on $\mathbb{N}$ are mappings $f: \mathbb{N} \to \mathbb{N}$, so we want to *start* with the cpo $[\mathbb{N} \to \mathbb{N}]$. This is not quite right, either. Church's thesis says the *partial recursives* are the computable functions, and so we should consider partial mappings $f: \mathbb{N} \rightharpoonup \mathbb{N}$. Now, the family of such mappings – $[\mathbb{N} \rightharpoonup \mathbb{N}]$ – is a cpo under the *extensional ordering:*

$$f \sqsubseteq g \quad \Leftrightarrow \quad \mathrm{dom}\, f \subseteq \mathrm{dom}\, g \;\&\; g|_{\mathrm{dom}\, f} = f.$$

Here, a directed family of partial mappings has for its supremum the union of the family. Two convenient facts are that any function from $\mathbb{N}$ to itself is monotone – even continuous – with respect to the discrete order, and the extensional order on the space of mappings between two discretely ordered sets is in fact the pointwise order. Thus, the partial mappings on $\mathbb{N}$ with the extensional order are just the partial mappings endowed with the pointwise order from the discrete order on $\mathbb{N}$.

But how do we distinguish the partial recursives from arbitrary partial self-maps of $\mathbb{N}$? A simple and *very* well-worn example shows how. Consider the factorial function

$$\mathrm{Fac}(n) = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot \mathrm{Fac}(n-1) & \text{otherwise.} \end{cases}$$

This leads us to define a *functional* $F: [\mathbb{N} \rightharpoonup \mathbb{N}] \to [\mathbb{N} \rightharpoonup \mathbb{N}]$ by

$$F(f)(m) = \begin{cases} 1 & \text{if } m = 0 \\ m \cdot f(m-1) & \text{if } m > 0 \;\&\; f(m-1) \text{ defined.} \end{cases}$$

It is easy to show that this functional is continuous (it only needs to preserve increasing unions of partial functions), and that its least fixed point is the factorial. What is harder is the fact that the effective structure (in the sense of recursion theory) on $\mathbb{N}$ can be extended to one on $[\mathbb{N} \rightharpoonup \mathbb{N}] \to [\mathbb{N} \rightharpoonup \mathbb{N}]$ (using ideas from the next section – see [26] for details), and F can be shown to be effective with respect to this structure. This means F's restriction to the partial recursives leaves them invariant; i.e., $F(g)$ is partial recursive if g is. If we let $[\mathbb{N} \rightharpoonup \mathbb{N}]_k$ denote the computable mappings on the natural numbers (i.e., the partial recursives), the following says every partial recursive arises exactly in this way:

Theorem 3 (Myhill-Sheperdson). *The effective operators on* $[\mathbb{N} \rightharpoonup \mathbb{N}]_k$ *are exactly the restrictions of the the effective continuous functionals* $G: [\mathbb{N} \rightharpoonup \mathbb{N}] \to [\mathbb{N} \rightharpoonup \mathbb{N}]$ *to* $[\mathbb{N} \rightharpoonup \mathbb{N}]_k$. □

1.4 Algebraicity and Continuity

A second component of domain theory – apart from the ease with which one can model recursion – is that of *approximation.* The idea is illustrated by the Myhill-Sheperdson Theorem. For any continuous functional $G:[\mathbb{N} \rightharpoonup \mathbb{N}] \to [\mathbb{N} \rightharpoonup \mathbb{N}]$, the least fixed point $\mathrm{FIX}\, G = \bigsqcup_{n \in \mathbb{N}} G^n(\emptyset)$, since $\emptyset$ is the least partial function. In the case of G is effective, $G^n(\emptyset)$ is a finite function. The finite functions play a special role in $[\mathbb{N} \rightharpoonup \mathbb{N}]$: they are the *compact* elements.

Definition 2. *An element $k \in P$ in a dcpo is* compact *if $k \sqsubseteq \bigsqcup D$ implies $(\exists d \in D)\, k \sqsubseteq d$ for all directed subsets $D \subseteq P$. The set of compact elements of P is denoted $K(P)$, and, for each $x \in P$ the set of compact elements below x is denoted $K(x)$.*

Lastly, P is algebraic *if $K(x)$ is directed and $x = \bigsqcup K(x)$ for every $x \in P$.*

Since any partial mapping $f: \mathbb{N} \rightharpoonup \mathbb{N}$ satisfies

$$f = \bigsqcup \{ f|_X \mid X \subseteq \mathrm{dom}\, f \text{ finite} \},$$

$[\mathbb{N} \rightharpoonup \mathbb{N}]$ is algebraic.

The compact elements of an algebraic dcpo completely determine the dcpo, since each element of the dcpo is the directed supremum of the compact elements below it. This association can be made more precise. Indeed, if we call a subset $I \subseteq Q$ of a partially ordered set an *ideal* if $I = {\downarrow} I$ is a lower set which also is directed, then we have the association $x \mapsto K(x): P \to \mathrm{Idl} K(P)$ which sends each element of P to the ideal of compact elements below it. This association is an isomorphism, where the inverse mapping simply sends an ideal to its supremum (which exists because P is a dcpo). Hence, $P \simeq \mathrm{Idl} K(P)$ for each algebraic dcpo P. This gives rise to an adjunction between the category ALG of algebraic dcpos and Scott continuous maps and the category POS of posets and monotone mappings. The right adjoint is the forgetful functor from ALG to POS, and the left adjoint is the ideal functor, which sends a partially ordered set to its family of ideals ordered under inclusion. One of the important consequences of this adjunction is that each continuous mapping $f: P \to Q$ between algebraic dcpos is completely determined by the restriction of f to the compact elements of P, and, conversely, each monotone mapping $f: K(P) \to Q$ from the compact elements of P to any dcpo Q extends to a unique continuous map from P to Q.

All of this has an important extension. The motivating example is the unit interval, which has 0 as its only compact element. Yet there is a clear notion of approximation here: if $x < y$, then for a directed set to have its supremum above y, some element of the directed set must be above x.

Definition 3. *The elements $x, y \in P$ in a dcpo satisfy $x \ll y$ (read "x is* relatively compact in y*") if $y \sqsubseteq \bigsqcup D$ implies there is some $d \in D$ with $x \sqsubseteq d$, for all directed subsets D of P. The set of elements relatively compact in y is denoted $\Downarrow y$, and the dcpo P is called* continuous *if $\Downarrow y$ is directed and $y = \bigsqcup \Downarrow y$ or all $y \in P$.*

An adjunction similar to the one between ALG and POS is available for continuous dcpos. It involves the notion of an *abstract basis* originally due to SMYTH [21].

Definition 4. *An* abstract basis *is a non-empty set X equipped with a transitive relation $\prec$ which satisfies the* interpolation property*:*

$$(\forall y \in X)(\forall M \subseteq X \text{ finite})\ M \prec y \quad \Rightarrow \quad (\exists x \in X)\ M \prec x \prec y.$$

A function $f: X \to Y$ between abstract bases is ideal *if $x \prec y$ in X implies that $\{z \in Y \mid z \prec f(x)\} \subseteq \{z \in Q \mid z \prec f(y)\}$.*

For example, in any continuous dcpo P, the pair $(P, \ll)$ is an abstract basis. Any abstract basis satisfies the property that the family of ideals (defined just as in the partially ordered set case) is a continuous dcpo under the inclusion order. The notion of an ideal mapping is designed precisely to capture those functions between abstract bases which extend to continuous mappings between their ideal completions. The following result generalizes the situation for algebraic domains.

Theorem 4 ([17]). *The functor which associates to a continuous dcpo P the abstract basis $(P, \ll)$ and to a continuous mapping $f: P \to Q$ the ideal mapping $f(I) = \{y \in Q \mid (\exists z \in I)\, y \ll f(z)\}$ is right adjoint to the ideal functor which associates to an abstract basis its ideal completion and to an ideal mapping the associated continuous mapping on the space of ideals.* □

Notes: This completes our rather cursory outline of domain theory. We have left out far more than we have included, but our intention is to provide only the barest of introductions to motivate the generalizations that we describe below.

We have not made specific reference to any result. Except for the last results on continuous dcpos (which can be found in [2] for the most part), most of this is folklore now, and can be found in many places. Again, [2] is an excellent source for referencing most of these results. The last theorem, however, appears only in [17]. A survey of a number of the ideas presented here can be found in [18].

2 Continuous Posets

A rather successful approach to modeling concurrent computation was devised by the members of the Programming Research Group at Oxford using the language CSP. We briefly outline this approach below, with an eye toward finding the relationship between the CSP models and more standard ones from domain theory. In endeavoring to understand this relationship, it became clear that one of the fundamental principles of domain theory had to be relaxed in order to describe the CSP models in purely domain-theoretic terms. That fundamental property of dcpos is that they are *directed complete*: all directed subsets have least upper bounds. This property is crucial in assuring that all continuous self-maps have (least) fixed points. But it turns out that describing the CSP models in domain-theoretic terms requires relaxing this condition in order to relate the models to the world of domains. The model we focus on for this discussion is the *failures model* for CSP, which we now describe.

2.1 CSP and the Failures Model

CSP is a process algebra for reasoning about concurrent processes. It was originally devised by C. A. R. HOARE and the first, definitive model for the language was presented in [3]. This is the so-called *failures model*, which models a process in terms of the communication events it can participate in (the *traces* of the process) together with the events it may refuse to participate in after a given trace (the so-called *refusals*). A syntax for CSP suitable for our purposes is given by the following BNF-like production rules:

$$P ::= STOP \mid SKIP \mid a \rightarrow P \mid P \setminus a \mid P;P \mid P_A\|_B P \mid P \Box P \mid P \sqcap P \mid x \mid \mu x.P$$

In this syntax, *STOP* denotes immediate abnormal termination, while *SKIP* denotes immediate normal termination. The actions a range over a set Σ of atomic actions which denote communication events between processes; $a \rightarrow P$ is a process which first wishes to participate in the action a and then to act like process P. $P \setminus a$ is the process P with all occurrences of the action a hidden from the environment (but they still occur, and as soon as they are offered). $P;P$ is the sequential composition of the two component processes; $P_A\|_B P$ is the process which has the two components synchronize on all actions in $A \cap B$ $(A, B \subseteq \Sigma)$, but either branch is free to perform actions not in the intersection whenever it wishes. $P \Box P$ is the *external choice* of the two processes, in which the environment is allowed to decide which branch will be chosen on the first action only, while $P \sqcap P$ is the *internal choice* of the branches, in which the machine decides. The term x denotes a process variable, and the last term is recursion.

The failures model for CSP as presented, e.g., in [3] gives a model for this language based of pairs (s, X), where $s \in \Sigma^* \cup \Sigma^* \surd$ is a finite sequence of actions, possibly ending in the normal termination event $\surd \notin \Sigma$, and $X \subseteq \Sigma$ is a set of *refusals* – events which the process may refuse to participate in after execution of s. The second component is needed in the model in order to distinguish internal and external choice. The failures model $\mathcal{FM}$ interprets each process as a set of such pairs, and the sets F that qualify to represent a process in CSP must satisfy the following conditions:

1. $\emptyset \neq F$.
2. $(s, X) \in F$ and t a prefix of s imply $(t, \emptyset) \in F$.
3. $(s, X) \in F$ and $Y \subseteq X$ imply $(s, Y) \in F$.
4. $(s, X) \in F$ and $(s\langle c \rangle, \emptyset) \notin F$ for all $c \in Y \subseteq A$ finite imply $(s, X \cup Y) \in F$.

The sets satisfying these conditions are called the *failures model* for CSP; it is shown in [3] that they form a complete inf-semilattice. This structure is used as the basis for showing this family of sets can be endowed with operations corresponding to each of the CSP operators. This allows an interpretation of CSP in the set of subsets of $(\Sigma^* \cup \Sigma^* \surd) \times \mathcal{P}(\Sigma)$ satisfying 1) – 4) – i.e., it provides the ingredients to show the family $\mathcal{FM}$ is a denotational model for CSP.

The order on the failures model is *reverse containment* on sets. So, the smaller the set, the higher it is in the order. Because the inf-operation is used to model nondeterminism, the order on the model is the order of nondeterminism – the higher a set, the more deterministic the process it represents. In fact, the maximal elements of the model are the deterministic processes. These have the property that they cannot be refined by any other process.

The order on the model also is used to model recursion, just as in the case of cpos. All the operators from CSP are modeled by operations on the model that are continuous with respect to reverse inclusion, and so Scott's corollary to Tarski's Theorem implies that each of the recursive processes can be modeled as the least fixed point of a continuous operator on the model.

2.2 The Failures Model as Closed Sets

All of the above indicates that the failures model is a cpo (the least element of the model is the set $CHAOS = \{(s, X) \mid s \in \Sigma^* \cup \Sigma^*\surd \ \&\ X \subseteq \Sigma\}$). But the construction is far from "standard", and it is unclear what relationship this model has to languages other than CSP. The work in [15] resulted from an effort to better understand this relationship. The analysis relies on a closer scrutiny of the properties that define failures sets.

The first three conditions imply that the sets F that qualify as process meanings are lower sets from some related partial order, and it was this idea that led to a realization that the conditions listed actually describe certain closed sets from a partial order. By *closed*, we mean closed with respect to the Scott topology, which we now define. But notice that we relax the situation somewhat, and consider *any* partial order, not just ones that are directed complete. This is because the partial order that gives rise to the failures model is not directed complete.

Definition 5. *Let P be a partially ordered set. A subset $U \subseteq P$ is Scott open if*

1. $U = {\uparrow} U = \{y \in P \mid (\exists x \in U)\ x \sqsubseteq y\}$ *is an upper set in P, and*
2. $\sqcup D \in U \implies D \cap U \neq \emptyset$ *for all directed subsets D of P.*

It is routine to show that the Scott open sets on any partial order are closed under finite intersections and arbitrary unions, so they do indeed form a topology. This topology is always T_0, which means distinct points can be separated by some open set (containing exactly one of them), but the topology is Hausdorff if and only if the partial order is the discrete order. What is more, the functions we defined earlier as being Scott continuous are in fact exactly those that are continuous with respect to this topology.

The Scott closed sets are those whose complements are Scott open, and since we have a description of the latter, we can derive the following characterization of the former.

Proposition 2. *$X \subseteq P$ is Scott closed if and only if*

1. $X = {\downarrow} X$ *is a lower set in P, and*

2. $D \subseteq X$ directed implies $\sqcup D \in X$. □

Notice that a corollary of this result is that the closure of a point $x \in P$ is $\downarrow x$, the principal lower set x defines. This is what makes Scott-closed sets an appealing model for concurrent computation – in the traces setting, they naturally include the history of a process since they are lower sets.

As with any topological space, the family of Scott closed sets forms a complete Brouwerian lattice under containment (cf. [11]). But in the case of an algebraic or continuous poset[4], we can say a lot more. Indeed, in this case, the family of Scott-closed sets forms a completely distributive, hence continuous lattice. If the underlying poset P is algebraic, then the family of Scott-closed sets is in fact completely distributive and algebraic, which in turn imply it forms a *complete ring of sets.* Finally, the relation $X \ll Y$ on the family of Scott-closed sets is completely determined by that of P, and the compact elements in the Scott-closed sets are exactly the closed sets generated by finite sets of compact elements of P.

In particular, the family of non-empty Scott-closed subsets of a continuous (resp., algebraic) dcpo is a continuous (resp., algebraic) dcpo semilattice (under union) whose relative compactness relation $\ll$ is completely determined by that of the underlying poset. Moreover, in the case P is algebraic, this family is an algebraic dcpo under *reverse* containment as well, and the compact elements here are the sets of the form $P \setminus (\uparrow F)$ as $F \subseteq K(P)$ ranges over the non-empty finite sets of compact elements of P.

What all this has to do with CSP and the failures model is explained by the following:

Example 1. Consider the set $P_F = \{(s, X) \mid s \in \Sigma^* \cup \Sigma^*\surd \ \&\ X \subseteq \Sigma\}$. We define a partial order of P_F by

$$(s, X) \sqsubseteq (t, Y) \iff (s < t \ \&\ X = \emptyset) \ \vee \ (s = t \ \&\ X \subseteq Y).$$

It is routine to show this is a partial order, and it also is easy to see that the pairs (s, X) with X finite are compact in this order. Hence P_F is an algebraic poset.

Theorem 5 ([15]). *The failures model consists of Scott-closed sets from the algebraic poset P_F, and this family is closed in the family of all Scott-closed sets under filtered intersections. Each of the operators from CSP gives rise to an operation on all the Scott-closed sets that is continuous with respect to usual containment, and all but the hiding operator give rise to operations that are continuous with respect reverse containment.* □

The point to note here is that it is the hiding operator that "causes all the problems" with the failures model. More to the point, the approach adopted with

[4] By a *continuous poset* we mean a partial order P in which $\Uparrow y$ is directed and $y = \sqcup \Uparrow y$ for all $y \in P$; P is an *algebraic poset* if $K(y)$ is directed and $y = \sqcup K(y)$ for all $y \in P$. The point is that we no longer require P to be directed complete.

the failures model was to use the order of nondeterminism to model a certain type of partial correctness – namely, that deadlock or divergence is catastrophic. That decision required a model in which all the operators are continuous with respect to reverse set containment, and since hiding is the only operation which doesn't satisfy this property on all Scott-closed sets, it is the reason for the condition 4) in the definition of the sets that comprise the model. In other words, if one were to seek a model for CSP without hiding, then all the Scott closed sets of the poset P_F could be used.

3 Local Cpos

From the outset, computation has viewed sequential composition as the "most primitive" operation. When the issue of modeling concurrent computation arose, the reaction was to devise models for nondeterminism using subset-like constructions, and then to model parallel composition in terms of sequential composition and nondeterministic choice. As described in [12], three distinct models for nondeterministic choice emerged in domain theory – the so-called *power domains.* These three constructs were first defined in terms of ideal completions of three distinct orders that can be defined on the finite subsets of the set $K(P)$ of compact elements of the underlying domain P. This works for any algebraic dcpo, but more restrictive domains allow for alternative descriptions of these constructions. As described in [22], *coherent domains* (i.e., those algebraic cpos for which the intersection of any finite family of Scott compact upper sets is again compact in the Scott topology) allow the three power domains to be described in completely topological terms:

- The *lower power domain* is the family of non-empty Scott-closed subsets of the underlying domain equipped with the usual order, and with union as the nondeterministic choice operation. This family is the free sup-semilatiice cpo over P.
- the *upper power domain* is the family of non-empty Scott compact upper sets from P, again with union as the operation, but this time under the reverse containment order. This family is the free inf-semilattice cpo over P.
- the *convex power domain* is the family of non-empty order-convex subsets $X = \uparrow X \cap \downarrow X$ of P whose lower set $\downarrow X$ is Scott closed and whose upper set $\uparrow X$ is Scott compact. The operation is the convex hull of the union:

  $$(X, Y) \mapsto \downarrow(X \cup Y) \cap \uparrow(X \cup Y),$$

  and the order is the *Egli-Milner order*:

  $$X \sqsubseteq Y \;\Leftrightarrow\; X \subseteq \downarrow Y \;\&\; Y \subseteq \uparrow X.$$

  This family is the free semilattice cpo over P.

Each of these constructions produces a coherent domain from an underlying coherent domain; these are the more-or-less standard constructions for modeling

nondeterministic choice within domain theory. Each construct allows operations (such as sequential composition) defined on the underlying domain P to be extended to the power domain. But, these extended operations all distribute over the nondeterministic choice operation, and so modeling *bisimulation* requires the additional step of solving a *domain equation* defined in terms of the convex power domain (cf. [1]).

All of the above applies to bounded nondeterminism, but *unbounded* nondeterminism also is useful, especially for specification. For example, consider a process-algebraic setting in which one wants to specify a process that can participate in any finite number of a given action, say a, but which is not supposed to participate in infinitely many a's. This requires distinguishing the process $\sqcap_{n\in\mathbb{N}}\ (a^n \rightarrow STOP)$ from the process $(\sqcap_{n\in\mathbb{N}}\ (a^n \rightarrow STOP)) \sqcap a^\infty$. But, these two processes must be identified in any of the models described above, and so we have to generalize domain theory and power domains in order to allow these processes to be distinguished.

In [24], an approach to modeling unbounded nondeterminism in CSP was presented. This approach added a new component to the meaning of each process – the *infinite traces* that a process could execute. By actually listing these traces, it became possible to distinguish a process that could execute an infinite trace from one which couldn't. But the resulting model was no longer a dcpo. Moreover, some selfmaps of the model no longer were continuous, and some of those that were didn't have any fixed points, let alone least ones. The point is that the new model was not a dcpo. The question then became how to make sure all the processes that could be meanings of recursive CSP processes in this setting actually had well-defined meanings. In other words, the question became one of how to assure that the recursive terms from CSP had meanings given by least fixed points in this new model.

The solution that was found was quite inventive. It amounted to using the fact that the model $\mathcal{U}$ for unbounded nondeterminism naturally contained a model for CSP with bounded nondeterminism – i.e., a copy of the failures-divergences model $\mathcal{FD}$ [4]. This was obtained by sending each CSP process to its meaning in $\mathcal{FD}$ together with those infinite traces that the process could execute, and this gave an embedding of $\mathcal{FD}$ within $\mathcal{U}$ "at the top": any element of $\mathcal{U}$ is the infimum of those elements in $\mathcal{FD}$ that are above it. This provided a cpo "at the top" of $\mathcal{U}$, which in turn proved crucial for deriving the results that were needed to show that $\mathcal{U}$ actually could serve as a model for unbounded nondeterminism in CSP.

The heart of the proof presented in [24] amounts to showing that each of the operations on $\mathcal{U}$ from CSP has a corresponding operation on $\mathcal{FD}$ from CSP with bounded nondeterminism that "dominates" it in the pointwise order. In terms of selfmaps of the model, the dominating operation leaves $\mathcal{FD}$ invariant (as it sits in $\mathcal{U}$). As a result, each term from CSP with bounded nondeterminism has a least fixed point on this submodel, and this fixed point is a *pre-fixed point* for any corresponding term on $\mathcal{U}$ that is dominated by the original term from CSP. But a pre-fixed point for a monotone mapping is all that is necessary to assure

the mapping has a least fixed point *provided each element of the model satisfies the property that its lower set is a cpo.* This indeed is the case, and this is how it is shown that each term from CSP with unbounded nondeterminism actually has a least fixed point on $\mathcal{U}$. We now present a more general description of these results that also is more precise.

Inspired by the work in [24] and by related work in [25] on unbounded nondeterminism for Timed CSP, an effort was made to find an underlying mathematical principle for the results that were obtained in these two papers. The resulting principle turned out to be remarkably simple. It hinged on two main ideas:

- the notion of a *local cpo*, and
- a *dominated fixed point theorem.*

Definition 6. *A partial order P is a* local cpo *if $\downarrow x$ is a cpo for each $x \in P$.*

Clearly any cpo is a local cpo, but there are local cpos which are not directed complete. For example, consider $(\mathbb{N}, \leq)$ the natural numbers in the usual order – the lower set of each point is finite, but $\mathbb{N}$ has no upper bound. This is not exactly the example we have in mind for modeling unbounded nondeterminism, however.

The dominated fixed point theorem can then be stated as follows:

Theorem 6 (Dominated Fixed Point Theorem [19]). *Let P be a local cpo and E a space for which there is a mapping $\iota: E \to P$. Suppose that $f: P \to P$ is monotone and satisfies the property that there is some mapping $F: E \to E$ with $f \circ \iota \sqsubseteq \iota \circ F$. If F has a fixed point in E, then f has a least fixed point in P.* □

The proof of this result is straightforward. One only has to note that a fixed point $x = F(x)$ for F satisfies $\iota(x)$ is a *pre-fixed point* for f: $f(\iota(x)) \sqsubseteq \iota(F(x)) = \iota(x)$ by the hypothesis of the Theorem. Thus, $f: \downarrow x \to \downarrow x$, and this set is a cpo as P is a local cpo. Hence f has a least fixed point by Tarski's Theorem.

In [19] it is shown how this result provides the common mathematical underpinning for the models for unbounded nondeterminism in Timed and untimed CSP. In the former case, the space E is one of the metric space models for Timed CSP with bounded nondeterminism devised by REED and ROSCOE [23], and in the later, the space E is the failures-divergences model for untimed CSP with bounded nondeterminism. One result of [19] was the internalization of the fixed point theory for recursive process meanings in each model; in the first approach devised by Roscoe for untimed CSP, an operational model for unbounded nondeterminism and a congruence theorem were used to justify the existence of meanings for each recursive process; of course, this still is needed to validate that the fixed point meanings defined in the model are the operationally correct ones. Another result of [19] was the realization that the work done in [24] to show that each process meaning in the model is the infimum of meanings in that lie in the subspace E (which is a cpo) is not needed. It is enough to know that each mapping for which a least fixed point is required has a dominating mapping on E in the sense of the Dominated Fixed Point Theorem.

As outlined above, for coherent domains, the three power domains are each describable in topological terms. But more generally, they can be defined for any algebraic dcpo in terms of the family of non-empty finite subsets of the set of compact elements of the underlying dcpo. For example, the lower power domain is the ideal completion of the family of non-empty subsets of $K(P)$ under the quasiorder $F \sqsubseteq G \Leftrightarrow F \subseteq \downarrow G$. Similarly, the upper power domain is the ideal completion of the same family, but endowed with the quasiorder $F \sqsubseteq G \Leftrightarrow G \subseteq \uparrow F$. In both of these cases, union defines a monotone operation which extends to the ideal completions to define the meaning of nondeterministic choice. Finally, the convex power domain is the ideal completion of the same family, this time ordered by the common refinement of these two quasiorders.

In [16], an attempt was made to develop a general theory for modeling unbounded nondeterminism in a domain-theoretic setting based on the results just described. In fact, the goal of that work was to devise analogues for each of the power domains for unbounded nondeterminism. The point of departure was the assumption that the underlying model for sequential composition – P – embeds in the model for unbounded nondeterminism so that elements of P are "free" with respect to unbounded nondeterminism. More precisely, the underlying assumption is that $a \not\sqsubseteq \sqcap X$ if $a \notin \downarrow X$ for *any* subset $X \subseteq P$. This assumption is what is required if one wants to distinguish processes such as $\sqcap_{n \in \mathbb{N}} (a^n \rightarrow STOP)$ from $(\sqcap_{n \in \mathbb{N}} (a^n \rightarrow STOP)) \sqcap a^\infty$. We now describe the results obtained.

First, it was found that *there is no analogue to the lower power domain.* The reason is that the order of nondeterminism ($x \sqsubseteq y \Leftrightarrow x \sqcap y = x$) corresponds to the order used to model recursion as least fixed points in any analogue to the lower power domain, so any element that dominates all of the terms $a^n \rightarrow STOP$ also must dominate a^∞.

On the other hand, it was shown that there is an analogue to the upper power domain. This is possible because, in the setting of the upper power domain, the order of nondeterminism is opposite to the order of recursion. The model in question is defined simply as the family of all non-empty upper sets $\{X \mid \emptyset \neq X = \uparrow X \subseteq P\}$ of the underlying domain P with union as the operation. It was shown in [16] that one could construct a Cartesian closed category of local cpos and monotone mappings having least fixed points (via the Dominated Fixed Point Theorem) which is closed under this construction of an *unbounded upper power space.* By the way, this is the abstract analogue of the model devised to model unbounded nondeterminism for untimed and Timed CSP.

Finally, an open question is whether there is an analogue for the convex power domain in this setting. In [16] an example is provided which shows that the analogue for the upper power space just described will not work: it is shown there that the family of all non-empty order-convex subsets of the underlying domain P is not a local cpo in general. (Unfortunately, more is claimed there – that there is no such model – but that claim remains unsettled.) It would be nice to know if this family can be completed into a local cpo which then could serve as the desired model for unbounded nondeterminism.

Readers familiar with PLOTKIN's work on countable nondeterminism [20] may wonder about the relationship between that work and what has been described here from [16]. Plotkin's approach was to weaken the continuity properties of the maps under consideration – instead of being continuous, they are only $\aleph_1$-continuous (so that they preserve sups of directed sets of less than $\aleph_1$-cardinality). Plotkin shows there is a free object supporting countable sums within the category of $\aleph_1$-complete objects and $\aleph_1$-continuous maps. This is not at odds with our results, since we studied objects which are not assumed to be directed complete for any cardinality of directed subsets, and the maps we consider are only monotone, and do not satisfy any stronger continuity properties. Our approach is justified by the work in [24, 25] which shows that these hypotheses are as strong as can be invoked, at least in the CSP setting.

4 Computational Models

So far the generalizations we have described have been inspired by work in process algebra. In this section, we focus on another area of application of domain theory – models of computation. In the early and mid1990s, ABBAS EDALAT began producing a number of striking applications of domain theory to areas of mathematics and computation. These began with an application showing how domain theory could provide a simpler approach to modeling fractals and iterated functions systems [5], even providing new algorithms for computing these objects. There followed applications to neural networks [6], and then to integration [7]. This last was notable because it showed how domain theory could be used to devise a new approach to Riemann integration in which the focus shifted from varying the function being integrated to varying measures which approximate Riemann measure, thus allowing domain theory to define the integral. Most recently, Edalat has continued his work by developing real PCF, which contains a real numbers datatype, along with efficient algorithms for exact computations in this datatype using continued fractions [10].

In all of this work, an emerging theme has been modeling topological spaces in domain theory, thus allowing the approximation theory of domains to be applied to problems in this setting. A focal point then becomes the question of which topological spaces admit computational (i.e., domain-theoretic) models. The precise statement is:

> *Which topological spaces can be embedded as the set of maximal elements in a domain?*

An initial answer was provided by LAWSON [13] who showed that any Polish space (complete, separable metric space) can be so represented. Shortly thereafter, EDALAT and HECKMANN [9] produced the *formal ball model* which shows that any metric space can be embedded as the space of maximal elements in a continuous poset. The model is the family of all pairs $\{(x,r) \mid x \in X \ \& \ r \geq 0\}$ under the order $(x,r) \sqsubseteq (y,s) \iff d(x,y) \leq r - s$. Moreover, they show that the model is a continuous poset whose completion (as described in Section 2)

has the completion of the metric space as its space of maximal elements. Both of these results focus on domains which satisfy the property that the Scott topology is *weak at the top* [8], and indeed under this assumption, the maximal elements of the underlying domain form a separable metric space.

We now outline some results that are due to KEYE MARTIN, a PhD student at Tulane, which provide an alternative approach to these and related results. They all will be contained in [14].

To begin, Martin begins with the notion of a *measurement* on a domain.

Definition 7. *Let P be a continuous poset. A* measurement *on P is a Scott-continuous mapping $\mu\colon P \to ([0,\infty),\geq)$ satisfying*

1. $\mu^{-1}(0) = \mathrm{MAX}(P)$, *and*
2. *μ induces the Scott topology near the top of P:*

$$(\forall x \in \mathrm{MAX}(P))(\forall U \subseteq P \text{ open})\ x \in U \ \Rightarrow\ (\exists \epsilon > 0)\ \downarrow x \cap \mu^{-1}([0,\epsilon)) \subseteq U.$$

Numerous examples are available here, including:

1. The space $\mathbb{IR}$ of compact intervals of real numbers, ordered by reverse inclusion, and with length as the measurement.
2. The family $\mathrm{LIST}(A)$ of lists over a set A, again with length of the list as the measurement.

In both of these cases – and in most others – the measurement actually induces the Scott topology on the whole domain, not just near the top.

Theorem 7 (Martin [14]). *Let (P,μ) be a continuous poset with a measurement, and suppose that μ satisfies:*

$$(\forall x, y \in P)\ x \uparrow y \ \Rightarrow\ (\exists z \in P)\ z \sqsubseteq x, y\ \&\ \mu(z) \leq 2 \cdot \max\{\mu(x), \mu(y)\}.$$

Then $\mathrm{MAX}(P)$ *is metrizable.* □

Notice that the result makes no mention of the weak topology – it holds for any continuous poset with measurement.

The converse of this result follows from Edalat's and Heckmann's result about the formal ball model [9], since that model has a measurement, the function $(x,r) \mapsto r$.

4.1 Modeling Algorithms

The inspiration for Martin's results was the intuition that two of the most common algorithms had something domain-theoretic in common. Those algorithms are:

1. The *bisection algorithm* which seeks a root for a continuous selfmap $f\colon \mathbb{R} \to \mathbb{R}$ on an interval $[a,b] \subseteq \mathbb{R}$. It proceeds by testing whether the function changes sign, first on the left half of the interval and then on the right, and recursively subdivides the interval. The algorithm can be viewed as a partial mapping $\mathrm{split}_f\colon \mathbb{IR} \rightharpoonup \mathbb{IR}$. Note that split_f is not monotone, let alone continuous.

2. Any of the searching algorithms on LIST(A), the domain of lists over a set A. Here again, these algorithms give rise to partial selfmaps of LIST(A) that are not generally monotone.

These examples inspired the following:

Definition 8. *Let P be a continuous poset. A partial mapping $f: P \rightharpoonup P$ is a* splitting *if $x \sqsubseteq f(x)$ $(\forall x \in \mathrm{dom}(f))$.*

Theorem 8 (Martin [14]). *Let $f: P \to P$ be a partial selfmap on a continuous dcpo P with measurement μ. If $\mu \circ f: P \rightharpoonup [0, \infty)$ is continuous, and if f is a splitting, then $\sqcup_{n \in \mathbb{N}} f^n(x)$ is a fixed point for f $(\forall x \in \mathrm{dom}(f))$.* □

A corollary of this result is that any continuous selfmap $f: \mathbb{R} \to \mathbb{R}$ has a root on any interval $[a, b]$ for which $\mathrm{split}_f([a, b]) \subseteq [a, b]$. Similarly, many of the familiar searching algorithms can be built up from splittings on the domain of lists to which the same result can be applied to do correctness proofs. Thus, the theory of continuous posets with measurements and splittings provides a common environment to model both the "discrete" algorithms from searching and the continuous algorithms such as the bisection algorithm.

4.2 Derivatives and Rates of Convergence

Since the setting of continuous posets with measurements includes the interval domain $\mathbb{IR}$, we can use this setting to generalize some results from numerical analysis.

Definition 9. *Let $f: P \rightharpoonup P$ be a partial mapping on a continuous poset P with measurement μ. If $p \in \mathrm{MAX}(P) \setminus K(P)$, then we define the* derivative of f at p *by*

$$\frac{\mathrm{d}f}{\mathrm{d}\mu}(p) = \lim_{x \to p} \frac{\mu(f(x)) - \mu(f(p))}{\mu(x) - \mu(p)}.$$

For example, for a continuous selfmap $f: \mathbb{R} \to \mathbb{R}$, if f changes sign on the interval $[a, b]$, then the above definition says that split_f has derivative $\frac{1}{2}$ at $[a, b]$, in keeping with the fact that the mapping splits the interval in half on each iteration.

The following shows this definition is sensible.

Theorem 9 (Martin [14]). *If $f: \mathbb{R} \to \mathbb{R}$ is differentiable at x, then*

$$\frac{\mathrm{d}F}{\mathrm{d}\mu}(\{p\}) = |f'(p)|,$$

where $F: \mathbb{IR} \rightharpoonup \mathbb{IR}$ is $F([a, b]) = f([a, b])$ and $\mu([a, b]) = b - a$. Conversely, if f is locally monotone and F has a derivative at $\{p\}$, then so does f and the above equation holds. □

This result shows that the following theorem generalizes results from numerical analysis.

Proposition 3 (Martin [14]). *Let $f: P \rightharpoonup P$ be a partial mapping on a continuous poset P with measurement μ. Suppose that $\lim_{n>0} f^n(x) = r \in \mathrm{MAX}(P)$ is a fixed point for f. Then $\lim_{n>0} \dfrac{\mu(f^{n+1}(x))}{\mu(f^n(x))} = \dfrac{\mathrm{d}f}{\mathrm{d}\mu}(r)$.* □

We also can use $\dfrac{\mathrm{d}f}{\mathrm{d}\mu}(r)$ to give an estimate of how fast $f^n(x)$ converges to a fixed point $r \in \mathrm{MAX}(P)$. If $\lim_n \mu(f^n(x)) = 0$, then for any given $\epsilon > 0$ there is some n for which $\mu(f^m(r)) < \epsilon$, for $m \geq n$. Now $\dfrac{\mathrm{d}f}{\mathrm{d}\mu}(r)$ can be used to give an estimate for the number of iterations of f for which this inequality actually holds – i.e., it provides an estimate for the number of iterations required to obtain the answer to within "ϵ accuracy."

5 Summary

We have given three generalizations of domain theory along with outlines of the problems that inspired those generalizations. The applications range from process algebra to models of computation, and include novel ideas that generalize some of the basic tenets of the original theory. Namely, they include

- Relaxing the assumption that the objects under study are directed complete, but retain the structure of continuity. The result is a theory that helps explain how the models for CSP relate to standard domain-theoretic constructions, and also makes clear that the hiding operator from CSP is the one operation that requires using a subfamily of the poset of non-empty Scott closed sets.
- Relaxing the condition of directed completeness and continuity to consider local cpos and monotone maps. The theory developed provides a general setting for modeling unbounded nondeterminism, and includes using cpos "at the top" of such objects to generate (least) fixed point theorems to assure that process meanings are well-defined.
- Considering continuous posets and mapping which are not monotone in order to model examples from computation, but which include the notion of a measurement. The theory provides a rich setting for devising computational models that encompass both the continuous approach =and the discrete approach represented by list searching algorithms. In this setting, it also is possible to generalize standard results from numerical analysis.

We believe these applications only serve to scratch the surface in terms of the potential applications for domain theory, and indeed many existing results are not mentioned here. This rather cursory survey is meant only to pique the reader's interest, and to provide some examples which we believe make a convincing case that domain theory is a rich theory whose potential applications range far from the setting that inspired it.

References

1. Abramsky, S. A domain equation for bisimulation. Information and Computation **92** (1991), 161–218.
2. Abramsky, S., Jung, A. Domain Theory. in: Handbook of Computer Science and Logic, Volume **3** (1994), Clarendon Press
3. Brookes, S. D., Hoare, C. A. R., Roscoe, A. W. A theory of communicating sequential processes. Journal ACM **31** (1984), 560–599.
4. Brookes, S. D., Roscoe, A. W. An improved failures model for communicating processes. Lecture Notes in Computer Science **197** (1985) 281–305.
5. Edalat, A. Dynamical systems, measures and fractals via domain theory. Information and Computation **120** (1995), 32–48.
6. Edalat, A. Domain theory in learning processes. Electronic Notes in Theoretical Computer Science **1** (1995), URL: `http://www.elsevier.com/locate/entcs/volume1.html`.
7. Edalat, A. Domain theory and integration. Theoretical Computer Science **151** (1995), 163–193.
8. Edalat, A. When Scott is weak at the top. Mathematical Structures in Computer Science, to appear.
9. Edalat, A., Heckmann, R. A computational model for metric spaces. Theoretical Computer Science, to appear.
10. Edalat, A., Potts, P. A new representation for exact real numbers. Electronic Notes in Theoretical Computer Science **6** (1997), URL: `http://www.elsevier.com/locate/entcs/volume6.html`.
11. Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J., Mislove, M., Scott, D. "A Compendium of Continuous Lattices." Springer-Verlag, Berlin, Heidelberg, New York (1980) 326pp.
12. Hennessy, M., Plotkin. G. Full abstraction for a simple parallel programming language. Lecture Notes in Computer Science **74** (1979) Springer-Verlag.
13. Lawson, J. Spaces of maximal points, Mathematical Structures in Computer Science, to appear.
14. Martin, K. Ph.D. thesis, Tulane University, in preparation.
15. Mislove, M. Algebraic posets, algebraic cpo's and models of concurrency. in: Topology and Category Theory in Computer Science. G. M. Reed, A. W. Roscoe and R. Wachter, editors, Clarendon Press (1991), 75–111.
16. Mislove. M. Denotational models for unbounded nondeterminism. Electronic Notes in Theoretical Computer Science **1** (1995), URL: `http://www.elsevier.com/locate/entcs/volume1.html`
17. Mislove, M. Using duality to solve domain equations. Electronic Notes in Theoretical Computer Science **6** (1997), URL: `http://www.elsevier.nl/locate/entcs/volume6.html`.
18. Mislove, M. Topology, domain theory and theoretical computer science. Topology and Its Applications, to appear.
19. Mislove. M., Roscoe, A. W., Schneider, S. A. Fixed points without completeness. Theoretical Computer Science **138** (1995), 273–314.
20. Plotkin, G. D. A powerdomain for countable nondeterminism. Lecture Notes in Computer Science **140** (1982).
21. Smyth, M. Effectively given domains. Theoretical Computer Science **5** (1977) 257–274.

22. Smyth, M. Power domains and predicate transformers: a topological view. Lecture Notes in Computer Science **154** (1983) Springer-Verlag, 662–675.
23. Reed, G. M., Roscoe, A. W. Metric spaces as models for real-time concurrency. Lecture Notes in Mathematics **298** (1988), 331–343.
24. Roscoe, A. W., Barrett, G. Unbounded nondeterminism in CSP. Lecture Notes in Computer Science **442** (1990).
25. Schneider, S. A. An operational semantics for timed CSP. Information and Computation **116** (1995).
26. Stoltenberg-Hansen, A., Lindström, I., Griffor, E. B. "Mathematical Theory of Domains." Cambridge Tracts in Theoretical Computer Science **22** (1994), Cambridge University Press, 349pp.