Technical Report

UCAM-CL-TR-107
ISSN 1476-2986

Number 107

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

DI-domains as a model of polymorphism

Thierry Coquand, Carl Gunter, Glynn Winskel

May 1987

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

DI-DOMAINS AS A MODEL OF POLYMORPHISM

May 18, 1987

Thierry Coquand, Carl Gunter and Glynn Winskel

Computer Laboratory, University of Cambridge, Cambridge CB2 3QG, England

1 Introduction.

The polymorphic λ-calculus was discovered by Girard [5] and later rediscovered by Reynolds [13]. As was the case with the simple untyped λ-calculus, the syntax of the calculus was, at first, understood better than its semantics. A model for the polymorphic calculus was first presented by McCracken [9] based on the cpo of closures over the algebraic lattice of subsets of the numbers. A similar technique can be used [1] to build models for the polymorphic calculus using finitary projection models such as the ones described by Scott [14] and Gunter [7]. More recently still there has been progress in saying what a model of the polymorphic calculus is in general. As with the simple untyped calculus, this can be done through the use of environment models [3] or categorically [15].

In this paper we investigate a model construction recently described by Girard [6]. This model differs from the models of McCracken, Scott, *etc.* in that the types are interpreted (quite pleasingly) as *domains* rather than closures or finitary projections on a universal domain. The construction is carried out over an interesting cartesian closed category of algebraic cpo's called *qualitative domains* which satisfy a very strong finiteness property. Our objective in this paper is two-fold. First, we would like to generalize Girard's construction to a larger category called *dI-domains* which was introduced by Berry [2]. The dI-domains possess many of the virtues of the qualitative domains. In addition, the dI-domains are closed under the *separated sum* and *lifting* operators from denotational semantics and this is *not* true of the qualitative domains. We intend to demonstrate that our generalized construction can be used to do denotational semantics in the ordinary way, but with the added feature of type polymorphism with the "types as domains" interpretation. For example, we will be able to interpret data types such as trees ($T \cong T + T$) and S-expressions ($S \cong \mathit{Atoms} + (S \times S)$) in the way they are ordinarily interpreted in the Scott-Strachey theory. Other useful types based on the lift operation (such as the solution to the domain equation $X \cong X_\perp$) will also be available with our approach. As with the qualitative domains we will also be able to obtain solutions for equations (such as $L \cong \mathit{Atoms} + L \to L$) with higher types. Our second objective is to show how Girard's construction (and our generalization) can be done *abstractly*. An ultimate result might carry out these constructions for "qualitative categories" and "dI-categories". For the purposes of this paper, however, we will (usually) restrict ourselves to posets. We also give a *representational* description of our own construction using the notion of a *prime event structure* which was introduced by Nielsen, Plotkin and Winskel [10] and Winskel [17].

The paper is divided into four sections. In the second section we describe background definitions for dI-domains, event structures, *etc.* and demonstrate some basic properties. The third section gives the basic model construction in abstract and representational styles. In the fourth section we discuss the calculus we seek to model which we call the *polymorphic fixedpoint calculus.* Due to limitations of space we have been forced to omit many details in order to convey the spirit of the construction as directly as possible.

We would like to accord significant credit to Jean-Yves Girard and Gérard Berry for the ideas of this paper. In fact, the idea of developing a theory which includes a separated sum was suggested by Girard in Annex B of [6] (although the specific choice of dI-domains is our own). We also received valuable assistance and encouragement from Martin Hyland, Eugenio Moggi and Pino Rosolini.

2 dI-domains and event structures.

A poset $\langle D, \sqsubseteq \rangle$ having a least element $\perp$ is said to be *complete* (and we say that D is a *cpo*) if every directed subset $M \subseteq D$ has a least upper bound $\bigsqcup D$. A monotone function $f : D \to E$ between cpo's D and E is *continuous* if $f(\bigsqcup M) = \bigsqcup f(M)$ for any directed $M \subseteq D$. A point x of a cpo D is said to be *isolated* if, for every directed collection $M \subseteq D$ such that $x \sqsubseteq \bigsqcup M$, there is a $y \in M$ such that $x \sqsubseteq y$. Let $\mathbf{B}_D$ denote the collection of isolated elements of D. The cpo D is *algebraic* if, for every $x \in D$, the set $M = \{x_0 \in \mathbf{B}_D \mid x_0 \sqsubseteq x\}$ is directed and $x = \bigsqcup M$. We will just call algebraic cpo's *domains.* A cpo D is *bounded complete* if every bounded subset of D has a least upper bound. In particular, if a pair $\{x, y\}$ is bounded then we will write $x \uparrow y$. If $x \uparrow y$ then $\{x, y\}$ has a least upper bound which we write as $x \sqcup y$. In a bounded complete cpo any pair $\{x, y\}$ has a greatest lower bound which we write as $x \sqcap y$. We will say that a point $x \in D$ is *very finite* if there are only finitely many points $y \sqsubseteq x$.

Definition: A *dI-domain* is a bounded complete domain D which satisfies

- *axiom d:* for every $x, y, z \in D$, if $y \uparrow z$ then $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ and
- *axiom I:* every isolated point is very finite. ▮

The dI-domains were introduced in Berry's thesis [2], where he made the discovery that they could be made into a cartesian closed category by choosing appropriate continuous functions as morphisms. At first sight this is surprising because until then the only cartesian closed category of domains known were those in which exponentiation was the Scott function space, consisting of all continuous functions ordered pointwise, and this construction certainly leads to domains failing axiom I. Trying to solve the full-abstraction problem for typed λ-calculi, in order to capture certain operational features in denotational semantics, Berry was led to the definition of stable functions and the stable order between them. The stable order does not relate functions in a solely pointwise fashion but also takes into account the manner in which functions compute. For this reason stable functions on dI-domains possess a function space different from Scott's, one which obeys axiom I.

Definition: Let D, E be dI-domains. A function $f : D \to E$ is *stable* iff it is continuous and satisfies

$$x \uparrow y \Rightarrow f(x \sqcap y) = f(x) \sqcap f(y). \quad ▮$$

Definition: We define **DI** to be the category with objects the dI-domains and morphisms the stable functions under the usual function composition. ▮

Theorem 1 *The category* **DI** *is cartesian closed; products are formed as cartesian products ordered coordinatewise and the function space of dI-domains D and E consists of the set of stable functions $f : D \to E$ ordered by the stable ordering i.e. for stable functions $f, g : D \to E$ we put $f \sqsubseteq g$ iff*

$$\forall x \in D.\ f(x) \sqsubseteq g(x) \textit{ and}$$
$$\forall x, y \in D.\ x \sqsubseteq y \Rightarrow f(x) = f(y) \sqcap g(x). \quad ▮$$

A great deal of the usual style of denotational semantics can be done in this category including the solving of recursive domain equations involving for example product, sum and function space. The solving of domain equations depends on a more restricted definition of embedding than is usual; embeddings must be *rigid* in the sense of Kahn and Plotkin [8] so that dI-domains are closed under direct limits.

Definition: Let D and E be domains. Let $f : D \to E$ be a continuous function. Say f is a *rigid embedding* iff there is a continuous function $g : E \to D$, called a *rigid projection*, such that $g \circ f = id$ and and $f \circ g \sqsubseteq id$. ∎

The rigid embeddings in this definition correspond to embeddings in the sense of Smyth and Plotkin [16]. We have added the modifier "rigid" to emphasize the fact that the condition $f \circ g \sqsubseteq id$ is being taken with respect to the *stable ordering* on functions. The following lemma should help clarify the significance of this assumption.

Lemma 2 *Let D and E be domains and $f : D \to E$ a continuous function. Then f is a rigid embedding iff there is a continuous function $g : E \to D$ such that*

$$g \circ f(d) = d \textit{ for all } d \in D \textit{ and}$$
$$f \circ g(c) \sqsubseteq c \textit{ for all } c \in E \textit{ and}$$
$$c \sqsubseteq f(d) \Rightarrow f \circ g(c) = c. \ \blacksquare$$

Instead of showing directly that dI-domains have direct limits of rigid embeddings we shall work with a representation of dI-domains by prime event structures. The represention shows clearly the link with Girard's qualitative domains; prime event structures are like qualitative domains but with an extra partial order structure.

Definition: Define a *(prime) event structure* to be a structure $E = (E, Con, \leq)$ consisting of a set E, which are partially ordered by $\leq$, and a predicate Con on finite subsets of E, the *consistency relation*, which satisfy

$$\{e' \mid e' \leq e\} \text{ is finite,}$$
$$\{e\} \in Con,$$
$$Y \subseteq X \in Con \Rightarrow Y \in Con,$$
$$X \in Con \ \& \ \exists e' \in X.\ e \leq e' \Rightarrow X \cup \{e\} \in Con$$

for all $e \in E$, and finite subsets X, Y of E.

Define its *consistent left-closed subsets*, $\mathcal{L}(E)$, to consist of those subsets $x \subseteq E$ which are

- *consistent:* $\forall X \subseteq x.\ X \in Con$ and
- *left-closed:* $\forall e, e'.\ e' \leq e \in x \Rightarrow e' \in x$.

In particular, define $\lceil e \rceil = \{e' \in E \mid e' \leq e\}$. ∎

Remark: Event structures often appear as a basic model of parallel processes when the set E is thought of as a set of event occurrences, the partial order $\leq$ as a relation of causal dependency, and the consistency relation as expressing what events can occur together. Then the configurations, the consistent left-closed sets of events, are thought of as states.

The configurations of an event structure form a dI-domain when ordered by inclusion. In this domain the configurations $\lceil e \rceil$, for an event e, are characterized as special kinds of isolated elements, the complete primes.

Definition: Let D be a bounded complete cpo. A *complete prime* of D is an element p such that

$$p \sqsubseteq \bigsqcup X \Rightarrow \exists x \in X.\ p \sqsubseteq x. \quad \blacksquare$$

Theorem 3 *Let E be a event structure. Then $(\mathcal{L}(E), \subseteq)$ is a dI-domain. The domain $(\mathcal{L}(E), \subseteq)$ has as complete primes those elements of the form $\lceil e \rceil$ for $e \in E$.* $\blacksquare$

Conversely, as we have indicated, any dI-domain is associated with an event structure in which the events are its complete primes.

Definition: Let D be a dI-domain. Define $\mathsf{Pr}(D) = (P, Con, \leq)$, where P consists of the complete primes of D,

$$p \leq p' \iff p \sqsubseteq p',$$

for $p, p' \in P$, and

$$X \in Con \iff X \text{ is bounded}$$

for a finite subset X of P. $\blacksquare$

Theorem 4 *Let D be a dI-domain. Then $\mathsf{Pr}(D)$ is a event structure, with $\phi : D \cong (\mathcal{L}\mathsf{Pr}(D), \subseteq)$ giving an isomorphism of partial orders where $\phi(d) = \{p \sqsubseteq d \mid p \text{ is a complete prime}\}$ with inverse $\theta : \mathcal{L}\mathsf{Pr}(D) \rightarrow D$ given by $\theta(x) = \bigsqcup x$.* $\blacksquare$

Rigid embeddings between dI-domains are represented by embeddings between event structures which reduce, in the case where the embeddings are inclusions, to a substructure relation between event structures.

Definition: Let $E_0 = (E_0, Con_0, \leq_0)$ and $E_1 = (E_1, Con_1, \leq_1)$ be event structures. An *embedding* of E_0 in E_1 is a 1-1 total function $f : E_0 \rightarrow E_1$ on events such that

$$X \in Con_0 \iff f(X) \in Con_1$$
$$f(\lceil e \rceil) = \lceil f(e) \rceil,$$

for any $X \subseteq E_0$ and $e \in E_0$. When $E_0 \subseteq E_1$ and the inclusion map $\iota : E_0 \hookrightarrow E_1$ is an embedding we write $E_0 \trianglelefteq E_1$, and say E_0 is a *substructure* of E_1.

We shall use $\mathbf{E}$ for the category of event structures with embeddings. ▮

Proposition 5 *Let $f : E_0 \to E_1$ be an embedding between event structures. Then the function*

$$f^L : (\mathcal{L}(E_0), \subseteq) \to (\mathcal{L}(E_1), \subseteq)$$

given by $f^L(x) = f(x)$, is a rigid embedding with projection

$$f^R : (\mathcal{L}(E_1), \subseteq) \to (\mathcal{L}(E_0), \subseteq)$$

given by $f^R(y) = f^{-1}(y)$.

A rigid embedding $h : D \to E$ between dI-domains restricts to an embedding $h' : \mathsf{Pr}(D) \to \mathsf{Pr}(E)$ between event structures, where $h'(p) = h(p)$ for complete primes p of D. ▮

Using the event structure representation it is now easy to see two properties of dI-domains which are crucial to development that follows.

Proposition 6 *The category $\mathbf{E}$ of event structures with embeddings has direct limits and pullbacks. The category $\mathbf{DI}^L$ of dI-domains and rigid embeddings is equivalent to $\mathbf{E}$ and so has direct limits and pullbacks.*

Proof: It is sufficient to consider a family of event structures $\{E_i \mid i \in I\}$ indexed by some directed set $(I, \leq)$ so that $i \leq j \Rightarrow E_i \trianglelefteq E_j$. With the understanding that each E_i has the form $(E_i, Con_i, \leq_i)$, form the union $E = (\bigcup_{i\in I} E_i, \bigcup_{i\in I} Con_i, \bigcup_{i\in I} \leq_i)$. With the inclusion maps $E_i \hookrightarrow E$, this forms a direct limit.

In showing the existence of pullbacks it suffices to consider embeddings which are inclusions. Let $f : Y \trianglelefteq X$ and $g : Z \trianglelefteq X$. The intersection $W = (Y \cap Z, Con_Y \cap Con_Z, \leq_Y \cap \leq_Z)$ is an event structure for which the inclusions $f' : W \trianglelefteq Z$ and $g' : W \trianglelefteq Y$ form a pullback of f and g. ▮

Following well-known lines (see *e.g.* [16] or [6]) we can make function space, product and sum into functors on event structures with embeddings, or equivalently dI-domains with rigid embeddings. These functors can then be shown to have the property that they preserve direct limits and pullbacks, a property important in what follows. Moreover, this category satisfies properties [16] sufficient for solving recursive domain equations.

3 Modelling polymorphism.

In this section we show how Girard's results in [6], on types in the polymorphic λ-calculus, generalize to dI-domains. We show this in two ways. We first give an elementary proof using the representation of dI-domains by the equivalent category of event structures. The proofs are then straightforward generalizations of Girard's in [6]; we need only take account of the extra partial order structure present in event structures but absent in qualitative domains. Following this we give a more abstract proof, less directly linked to Girard's constructions, but informative and useful, we hope, as a step in understanding polymorphism more abstractly.

Two central notions in [6] are that of *variable type* and *objects of a variable type*. Adopting the idea for event structures, a variable type is a functor $T : \mathbf{E} \to \mathbf{E}$ which preserves direct limits and pullbacks. An *object of* T is defined to be a function t on event structures such that $t_X \in T(X)$, for all event structures X, and $t_X = f^R(t_Y)$ for all embeddings $f : X \to Y$ of event structures. Ordered pointwise, *i.e.* taking

$$t \sqsubseteq t' \text{ iff } t_X \sqsubseteq t'_X \text{ for all event structures } X$$

these form a partial order. In fact this partial order can be represented as the configurations of an event structure, and so forms a dI-domain, and this is the key to a treatment of universal types in polymorphic λ-calculus.

As in [6] we use the notion of a trace of such a functor.

Theorem 7 *Let $T : \mathbf{E} \to \mathbf{E}$ be a functor on the category of event structures with embeddings which preserves direct limits and pullbacks. Let X be a event structure and let e be an event of $T(X)$. Then there is a finite event structure X_0, an event e_0 of $T(X_0)$ and an embedding $f : X_0 \to X$ such that $e = T(f)(e_0)$ and for any event structure X', an embedding $f' : X' \to X$ and e' an event of $T(X')$ such that $T(f')(e') = e$ there is a unique embedding $h : X_0 \to X'$ such that*

$$e' = T(h)(e_0) \text{ and } f = f' \circ h.$$

Proof: The proof is that given by Girard in [6] but in the wider context of event structures rather than qualitative domains. ▮

Definition: Let $T : \mathbf{E} \to \mathbf{E}$ be a functor which preserves direct limits and pullbacks. A *trace* of T is defined to be a set A of pairs (X, e) with X a finite event structure and e an event of $T(X)$

with the property that for any event structure X' and event e' of $T(X')$ there is a unique (X_0, e_0) in A and an embedding $h : X_0 \to X'$ such that $e' = T(h)(e_0)$. ▮

By Theorem 7, we know that a trace as defined in above always exists. Choosing one particular trace we can define an event structure associated with a functor T on event structures preserving direct limits and pullbacks. It is slightly simpler to first define when a subset of a trace is inconsistent.

Definition: Let $T : \mathbf{E} \to \mathbf{E}$ be a functor which preserves direct limits and pullbacks. Let A be a trace of T. Say a subset of A is *inconsistent* when it includes a finite subset $\{(X_i, e_i) \mid i \in I\}$ for which there are embeddings $f_i : X_i \to X$, for $i \in I$, into some event structure X, so that

$$\{T(f_i)(e_i) \mid i \in I\} \notin Con_{T(X)},$$

where $Con_{T(X)}$ is the consistency predicate in $T(X)$. Now, define $\mathcal{E}(T) = (E, Con, \leq)$ where

- $E = \{(X, e) \in A \mid \{(X, e)\}$ is not inconsistent$\}$,
- Con consists of those finite subsets $\{(X_i, e_i) \mid i \in I\}$ of E which are not inconsistent, and
- $\leq$ is a binary relation on E given by

$$(X', e') \leq (X, e) \text{ iff } T(f)(e') \leq_{T(X)} e$$

  for some embedding $f : X' \to X$ into the event structure $X = (X, Con_X, \leq_X)$. ▮

Thus the structure $\mathcal{E}(T)$ is constructed out of the the "self-consistent" elements of a trace of T, those elements (X, e) for which there are no two embeddings $f_1, f_2 : X \to Y$ for which $\{T(f_1)(e), T(f_2)(e)\}$ is not consistent in $T(Y)$. It is unique to within isomorphism by the properties of a trace. It is an event structure, to verify which we shall use the following lemma.

Lemma 8 *For embeddings $f : X \to Y$ and $g : X \to Z$ of event structures there is an event structure W and embeddings $f' : Z \to W$ and $g' : Y \to W$ such that $g' \circ f = f' \circ g$.*

Proof: Assume $X = (X, Con_X, \leq_X)$, $Y = (Y, Con_Y, \leq_Y)$ and $Z = (Z, Con_Z, \leq_Z)$. It suffices to consider the case where $f : X \trianglelefteq Y$ and $g : X \trianglelefteq Z$, that is when the embeddings are inclusions, and where we further assume that $X = Y \cap Z$. Then we define

$$W = (Y \cup Z, Con_Y \cup Con_Z, \leq_Y \cup \leq_Z),$$

the union of Y and Z. Taking f' and g' to be the inclusions $f' : Z \trianglelefteq W$ and $g' : Y \trianglelefteq W$ fulfils the requirements of the lemma. ▮

Lemma 9 *Let $T : \mathbf{E} \to \mathbf{E}$ be a functor which preserves direct limits and pullbacks. The structure $\mathcal{E}(T)$ defined above is an event structure.*

Proof: The only difficulty comes in showing that $\mathcal{E}(T)$ satisfies the property

$$\{(X_i, e_i) \mid i \in I\} \in Con \ \&\ (X'_j, e'_j) \leq (X_j, e_j), \text{ for } j \in I, \Rightarrow \{(X_i, e_i) \mid i \in I\} \cup \{(X'_j, e'_j)\} \in Con.$$

Suppose otherwise, *i.e.* that $\{(X_i, e_i) \mid i \in I\} \in Con$ & $(X'_j, e'_j) \leq (X_j, e_j)$, for $j \in I$, while $\{(X_i, e_i) \mid i \in I\} \cup \{(X'_j, e'_j)\}$ is inconsistent. Then there is a subset $K \subseteq I$ with embeddings $f_k : X_k \to X$ and $f'_j : X'_j \to X$ into some event structure X so that $\{T(f_k)(e_k) \mid k \in K\} \cup \{T(f'_j)(e'_j)\}$ is not consistent in $T(X)$. Also there is an embedding $g : X'_j \to X_j$ such that $T(g)(e'_j) \leq_{X^\cdot} e_j$. By lemma 8, there are embeddings $g' : X \to W$ and $f_j : X_j \to W$, for some event structure W, for which $g' \circ f'_j = f_j \circ g$. However this yields embeddings $g' \circ f_k : X_k \to W$, for $k \in K$, and $f_j : X_j \to W$ for which $\{T(g' \circ f_k)(e_k) \mid k \in K\} \cup \{T(f_j)(e_j)\} \notin Con_{T(W)}$. This contradicts the consistency of $\{(X_i, e_i) \mid i \in I\}$. Hence the property is proved. ▮

We now show how the event structure associated with a variable type T has a domain of configurations isomorphic to the partial order of objects of type T.

Theorem 10 *Let $T : \mathbf{E} \to \mathbf{E}$ be a functor which preserves direct limits and pullbacks. Let $\mathcal{E}(T)$ have the form $(E, Con, \leq)$. There is an isomorphism between the domain of configurations $\mathcal{L}(\mathcal{E}(T))$, ordered by inclusion, and the objects of T, ordered pointwise; the isomorphism is determined as follows:*

1. *An object t of variable type T determines a configuration a of $\mathcal{E}(T)$ where*

$$a = \{(X, e) \in E \mid e \in t_X\}.$$

2. *A configuration a of $\mathcal{E}(T)$ determines an object t of variable type T which acts so*

$$t_X = \{T(f)(e) \mid \exists X_0.\ (X_0, e) \in a \ \&\ f : X_0 \to X \text{ is an embedding}\}$$

for all event structures X.

Proof: Again the proof more or less follows Girard's in [6]; the additional partial order structure causes no real difficulties. ▮

Now we give the more abstract proof.

Definition: Let $\mathbf{C}$ be a category, we shall call $\mathbf{C}$ a *finitary category* if, and only if, $\mathbf{C}$ has the following properties

1. $\mathbf{C}$ has pull backs
2. there exists a *set* S of objects of $\mathbf{C}$ such that every object of $\mathbf{C}$ is the direct limits of objects in S. ∎

The intuition is that all objects are approximable by objects which are finite in a very strong sense. Note the following fundamental property:

Proposition 11 *The category* $\mathbf{DI}^L$ *is finitary. Furthermore, the product of two finitary categories is finitary.* ∎

Definition: Let $\mathbf{C}$ be a category. Let F be a functor from $\mathbf{C}$ to $\mathbf{DI}^L$. We say F is a variable type (or sometimes a *stable functor*) when it stable. We say that a family (t_X), such that $t_X \in F(X)$ for all objects X in $\mathbf{C}$, is an *object of variable type* F (or sometimes a *uniform family of* F) if, and only if, for every pair of objects X and Y in $\mathbf{C}$, and every morphism $f \in \mathbf{C}(X, Y)$, we have $F(f)^R(t_Y) = t_X$. We shall write $\Pi(F)$ for the collection of all objects of variable type F. ∎

Note that the collection of uniform families of a given functor from $\mathbf{C}$ into $\mathbf{DI}^L$ is in general a class and not a set, if $\mathbf{C}$ is a large category. However, we shall see that this difficulty does not really happen in the cases we consider. Note the following presentation of objects of variable type.

Proposition 12 *Let* $\mathbf{C}$ *be a category. Let* F *be a functor from* $\mathbf{C}$ *to* $\mathbf{DI}^L$. *Then a family* (t_X) *such that* $t_X \in F(X)$ *for all objects* X *in* $\mathbf{C}$ *is a uniform family of* F *if, and only if, for every pair of objects* X *and* Y *in* $\mathbf{C}$, *and every morphism* $f \in \mathbf{C}(X, Y)$, *we have*

$$\forall p \in \mathsf{Pr}(F(X)).\ p \sqsubseteq t_X \Leftrightarrow F(f)^L(p) \sqsubseteq t_Y. \ ∎$$

We can now state Girard's discovery in our framework:

Theorem 13 *Let* $\mathbf{C}$ *be a finitary category, and* F *a functor from* $\mathbf{C}$ *to* $\mathbf{DI}^L$ *which stable. Then* $\Pi(F)$ *is a set, and it is a dI-domain for the pointwise ordering.* ∎

Proof: That $\Pi(F)$ is a set comes directly from the fact that $\mathbf{C}$ is set-generated and that F preserves directed limits. The verification of the fact that it is a bounded complete cpo uses directly our reformulation of what is a uniform family of F. As infs and sups are computed pointwise, the distributivity axiom is verified. Finally, the hard point is to find a basis with very finite and isolated elements. We shall give only the construction. Let (t_X) be a given object of variable type F, and A an element of S. For each very finite element $a \sqsubseteq t_A$ of $F(A)$, we shall show how to build a very finite and isolated object (u_X), such that $u_A = a$ and (u_X) is less than or equal to (t_X) for the pointwise ordering (which corresponds to the usual construction of step functions).

Define first the relation $(X, x) \sqsubseteq (Y, y)$ if, and only if, X, Y are objects of $\mathbf{C}$, and $x \in F(X)$, $y \in F(Y)$, and there exists a morphism $f \in \mathbf{C}(X, Y)$ such that $F(f)^L(x) \sqsubseteq y$. This is a transitive relation. We note that, for every object Y, the following subset of $F(Y)$

$$\{F(f)^L(x) \mid (X, x) \sqsubseteq (A, a) \text{ and } f \in \mathbf{C}(X, Y)\}$$

is bounded by t_Y. Since $F(Y)$ is bounded complete, we can define

$$u_Y = \bigsqcup \{F(f)^L(x) \mid (X, x) \sqsubseteq (A, a) \text{ and } f \in \mathbf{C}(X, Y)\}.$$

Then, we can check that this family has all wanted properties. (Note: the fact that F preserves pull-backs is needed to show that the family (u_X) is, in fact, an object of variable type F.) ∎

The importance of this proof is that it gives an insight as to how this model can be extended to a model of Fω. This is the remarkable closure property of dI-domains which will allow us to interpret the abstraction relatively to *types* (types as parameters). The key point is the Proposition 3. Note that in the case where the category $\mathbf{C}$ is the poset of natural numbers, then the construction is the usual inverse limit construction! However, it is important to emphasize that, in general, $\Pi(F)$ is *not* the limit of the functor F.

We will apply these constructions to provide a semantics for the calculus which is described in the following section. For now we shall only give an outline of how to give an interpretation of second-order calculus. This is already done in Girard's paper [6]. Since dI-domains form a cartesian closed category (with stable functions), it is well-known how to interpret the usual application and abstraction. Closed types will be interpreted as dI-domains and, more generally, a type with n variables will be interpreted as a functor from $(\mathbf{DI}^L)^n$ to $\mathbf{DI}^L$ which is stable. Closed terms will be interpreted as elements of the domains and, more generally, a term which depends on type variables will be interpreted as a uniform family of the functor associated to its types. One interesting feature is that the interpretation is extensional.

4 The polymorphic fixedpoint calculus.

In this section we will describe the syntax of a calculus which we wish to interpret using the constructions set out in the previous sections. It is a fragment of the language of McCracken [9] and is closely related to Fairbairn's programming language Ponder [11]. We call this language the *(pure) polymorphic fixedpoint calculus.* Its types have the following abstract syntax:

$$\sigma \ ::= \ \sigma_1 \to \sigma_2 \ \mid \ \alpha \ \mid \ \mu\alpha.\ \sigma \ \mid \ \Pi\alpha.\ \sigma,$$

where α is a type variable, and it has the following terms:

$$M ::= x \mid \lambda x : \sigma.\ M \mid M_1(M_2) \mid \Lambda\alpha.\ M \mid M\{\sigma\} \mid \mathsf{intro}^{\mu\alpha.\ \sigma}(M) \mid \mathsf{elim}^{\mu\alpha.\ \sigma}(M) \mid \mu x : \sigma.\ M$$

where x is a variable.

4.1 Typing and equational rules.

A closed term will be assigned a unique type by a system of typing rules. Typing sequents have the form $H \vdash_\Sigma M : \sigma$ where H is a (possibly empty) list of hypotheses of the form $x : \sigma$. We assume that a list H has no repetitions of variables x. Axioms are given by the scheme:

$$H_1,\ x : \sigma,\ H_2 \vdash_\Sigma x : \sigma$$

where x does not appear in H_1 or H_2 and Σ is a (possibly empty) set of type variables. There are the following rules for introducing and eliminating function types:

$$\frac{H,\ x : \sigma_1 \vdash_\Sigma M : \sigma_2}{H \vdash_\Sigma \lambda x : \sigma_1.\ M : \sigma_1 \to \sigma_2} \qquad \frac{H \vdash_\Sigma M_1 : \sigma_1 \to \sigma_2 \qquad H \vdash_\Sigma M_2 : \sigma_1}{H \vdash_\Sigma M_1(M_2) : \sigma_2}$$

There are the following rules for introducting and eliminating Π:

$$\frac{H \vdash_{\Sigma,\ \alpha} M : \sigma}{H \vdash_\Sigma \Lambda\alpha.\ M : \Pi\alpha.\ \sigma} \qquad \frac{H \vdash_\Sigma M : \Pi\alpha.\ \sigma_1}{H \vdash_\Sigma M\{\sigma_2\} : [\sigma_2/\alpha]\sigma_1}$$

where the first rule is subject to the condition that α is not free in the type of any free term variable of M and $[\sigma_2/\alpha]\sigma_1$ is the expression that results from substituting σ_2 for α in σ_1 (where bound variables in σ_1 are renamed to avoid capturing free variables of σ_2). Recursion can occur both at the level of types and at the level of terms. The rule for typing a recursive term is

$$\frac{H,\ x : \sigma \vdash_\Sigma M : \sigma}{H \vdash_\Sigma \mu x : \sigma.\ M : \sigma}$$

Introduction and elimination for recursive types use the operators intro and elim respectively.

$$\frac{H \vdash_\Sigma M : [(\mu\alpha.\ \sigma)/\alpha]\sigma}{H \vdash_\Sigma \mathsf{intro}^{\mu\alpha.\ \sigma}(M) : \mu\alpha.\ \sigma} \qquad \frac{H \vdash_\Sigma M : \mu\alpha.\ \sigma}{H \vdash_\Sigma \mathsf{elim}^{\mu\alpha.\ \sigma}(M) : [(\mu\alpha.\ \sigma)/\alpha]\sigma}$$

There is a collection of equations which an interpretation of the polymorphic fixedpoint calculus must satisfy. We assume throughout that all terms are type-sensible. First of all, there are the basic equality rules:

$$H \vdash_\Sigma M = M \qquad \frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma M_2 = M_1} \qquad \frac{H \vdash_\Sigma M_1 = M_2 \qquad H \vdash_\Sigma M_2 = M_3}{H \vdash_\Sigma M_1 = M_3}$$

and also rules for application:

$$\frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma M_1(M_3) = M_2(M_3)} \qquad \frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma M_3(M_1) = M_3(M_2)}$$

The calculus must also satisfy the β-rule:

$$H \vdash_\Sigma (\lambda x : \sigma.\ M_1)(M_2) = [M_2/x]M_1$$

and the ξ-rule

$$\frac{H,\ x : \sigma \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma \lambda x : \sigma.\ M_1 = \lambda x : \sigma.\ M_2}$$

Finally, we also require the η-rule:

$$\lambda x : \sigma.\ M(x) = M$$

(where x does not appear free in M). For type application we have, of course, the following:

$$\frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma M_1\{\sigma\} = M_2\{\sigma\}}$$

We also have the β-rule

$$H \vdash_\Sigma (\Lambda\alpha.\ M)\{\sigma\} = [\sigma/\alpha]M$$

the ξ-rule

$$\frac{H \vdash_{\Sigma,\ \alpha} M_1 = M_2}{H \vdash_\Sigma \Lambda\alpha.\ M_1 = \Lambda\alpha.\ M_2}$$

and the η-rule

$$\Lambda\alpha.\ M\{\alpha\} = M$$

(where α does not appear free in M). There are some basic rules for the recursive type operators:

$$\frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma \mathsf{intro}^{\mu\alpha.\ \sigma}(M_1) = \mathsf{intro}^{\mu\alpha.\ \sigma}(M_2)} \qquad \frac{H \vdash_\Sigma M_1 = M_2}{H \vdash_\Sigma \mathsf{elim}^{\mu\alpha.\ \sigma}(M_1) = \mathsf{elim}^{\mu\alpha.\ \sigma}(M_2)}$$

and a pair of equations asserting an isomorphism:

$$H,\ x : [(\mu\alpha.\ \sigma)/\alpha]\sigma \vdash_\Sigma \mathsf{intro}^{\mu\alpha.\ \sigma}(\mathsf{elim}^{\mu\alpha.\ \sigma}(x)) = x$$
$$H,\ x : \mu\alpha.\ \sigma \vdash_\Sigma \mathsf{elim}^{\mu\alpha.\ \sigma}(\mathsf{intro}^{\mu\alpha.\ \sigma}(x)) = x$$

4.2 An extended calculus.

For the purposes of denotational semantics, it is useful to have the additional type constructors of product $\times$ and sum $+$. We interpret the $+$ operator as the usual separated sum of denotational semantics. Given dI-domains $D_1, \ldots, D_n$, the sum $+(D_1, \ldots, D_n)$ is defined to be the disjoint sum of the domains D_i together with a new element $\perp$ which is taken to be the least element of $+(D_1, \ldots, D_n)$. We emphasize the point which we mentioned in the Introduction that this construction is *not possible* over the category of qualitative domains. We therefore propose the an extension of the pure language which will include sums. The extended language has the following types:

$$\sigma \ ::= \ \sigma_1 \times \sigma_2 \ | \ \sigma_1 + \sigma_2 \ | \ \sigma_1 \to \sigma_2 \ | \ \alpha \ | \ \mu\alpha.\, \sigma \ | \ \Pi\alpha.\, \sigma.$$

The terms of the extended language are given as follows:

$$\begin{aligned} M \ ::= \ & \langle M_1, M_2 \rangle \ | \ \mathsf{proj}_i(M) \\ & \mathsf{in}_1^{\sigma_1,\sigma_2}(M) \ | \ \mathsf{in}_2^{\sigma_1,\sigma_2}(M) \ | \ \mathsf{cases}\ M\ \mathsf{of}\ x_1 : \sigma_1.\, M_1,\ x_2 : \sigma_2.\, M_2 \ | \\ & x \ | \ \lambda x : \sigma.\, M \ | \ M_2(M_1) \ | \\ & \Lambda\alpha.\, M \ | \ M\{\sigma\} \ | \\ & \mathsf{intro}^{\mu\alpha.\,\sigma}(M) \ | \ \mathsf{elim}^{\mu\alpha.\,\sigma}(M) \ | \\ & \mu x : \sigma.\, M \end{aligned}$$

Typing rules for the extended language are the same as those for the pure language together with the rules for products:

$$\frac{H \vdash_\Sigma M_1 : \sigma_1 \qquad H \vdash_\Sigma M_2 : \sigma_2}{\langle M_1, M_2 \rangle : \times(\sigma_1, \sigma_2)} \qquad\qquad \frac{M : \sigma_1 \times \sigma_2}{\mathsf{proj}_i(M) : \sigma_i}$$

and for sums:

$$\frac{M : \sigma_i}{\mathsf{in}_i^{\sigma_1,\sigma_2}(M) : \sigma_1 + \sigma_2} \qquad\qquad \frac{M : \sigma_1 + \sigma_2 \qquad H,\ x_1 : \sigma_1 \vdash_\Sigma M_1 : \sigma \qquad H,\ x_2 : \sigma_2 \vdash_\Sigma M_2 : \sigma}{\mathsf{cases}\ M\ \mathsf{of}\ x_1 : \sigma_1.\, M_1,\ x_2 : \sigma_2.\, M_2 : \sigma}$$

$(i = 1, 2)$. There are also equations to be satisfied by the new constructs. For the product, there are some basic equations

$$\frac{H \vdash_\Sigma M_2 = M_2' \qquad H \vdash_\Sigma M_2 = M_2'}{H \vdash_\Sigma \langle M_1, M_2 \rangle = \langle M_1', M_2' \rangle} \qquad\qquad \frac{H \vdash_\Sigma M = M'}{H \vdash_\Sigma \mathsf{proj}_i(M) = \mathsf{proj}_i(M')}$$

$(i = 1, 2)$ and equations

$$H \vdash_\Sigma \langle \mathsf{proj}_1(M), \mathsf{proj}_2(M) \rangle = M$$

$$H \vdash_\Sigma \mathsf{proj}_i(\langle M_1, M_2 \rangle) = M_i$$

($i = 1,2$) which assert that the operator $\times$ is to be interpreted as a categorical product. For the sum we also require some basic equations:

$$\frac{H \vdash_\Sigma M = M'}{H \vdash_\Sigma \mathsf{proj}_i(M) = \mathsf{proj}_i(M')}$$

$$\frac{H,\ x_1 : \sigma_1 \vdash_\Sigma M_1 = M_1' \qquad H,\ x_2 : \sigma_2 \vdash_\Sigma M_2 = M_2' \qquad H \vdash_\Sigma M = M'}{H \vdash_\Sigma \mathsf{cases}\ M\ \mathsf{of}\ x_1 : \sigma_1.\ M_1,\ x_2 : \sigma_2.\ M_2 = \mathsf{cases}\ M'\ \mathsf{of}\ x_1 : \sigma_1.\ M_1',\ x_2 : \sigma_2.\ M_2'}$$

and a pair of equations

$$H,\ x_i : \sigma_i \vdash_\Sigma \mathsf{cases}\ \mathsf{in}_i(x_i)\ \mathsf{of}\ x_1 : \sigma_1.\ M_1,\ x_2 : \sigma_2.\ M_2 = M_i.$$

($i = 1,2$). However, we do *not* have the following additional equations

$$H,\ x_i : \sigma_i \vdash_\Sigma \mathsf{in}_i(\mathsf{cases}\ x_i\ \mathsf{of}\ x_1 : \sigma_1.\ M_1,\ x_2 : \sigma_2.\ M_2) = M_i$$

($i = 1,2$) which would assert that $+$ is a categorical coproduct. Surprisingly, this last equation is satisfied only by a trivial interpretation of the calculus! Under our interpretation, however, these equations are *almost* true, in the sense that they *do* hold when $x_i \neq \perp$.

5 Semantics of the polymorphic lambda calculus

5.1 Some preliminary results

We now describe our model. We omit the semantics for the recursion and the extended language and concentrate on the polymorphic lambda calculus. We give, by structural induction, the denotations $[\![\Sigma \vdash \sigma]\!]$ and $[\![H \vdash_\Sigma M : \sigma]\!]$. For this, we need first some general results.

Definition: Let F be a continuous and stable functor from **Dom** to **Dom**. A continuous stable family with respect to F is a family (t_X) indexed over dI-domains X such that

- $t_X \in F(X)$ for all X,
- if $f \in \mathbf{Dom}(X, Y)$, then $F(f)^L(t_X) \sqsubseteq t_Y$ (monotonicity),
- if $f_i \in \mathbf{Dom}(X_i, X)$ is such that X is the directed colimit of the system (X_i, f_i), then t_X is the sup of the directed family $F(f_i)^L(t_{X_i})$,
- if $u_1 \in \mathbf{Dom}(X_3, X_1)$ and $u_2 \in \mathbf{Dom}(X_3, X_2)$ define a pull-back of $f_1 \in \mathbf{Dom}(X_1, X)$ and $f_2 \in \mathbf{Dom}(X_2, X)$ and we write $f_3 = f_1 \circ u_1 = f_2 \circ u_2$, then $F(f_3)^L(t_{X_3}) = F(f_1)^L(t_{X_1}) \wedge F(f_2)^L(t_{X_2})$ (stability).

Remark that the two last conditions express that (t_X) commutes with directed colimits and pull-backs. It is convenient for checking that these properties hold for a given family to use the characterisation of directed colimit of [16]: if X is the directed colimit of the system (X_i, f_i) then $f^L \circ f^R$ is the sup of the directed family $(F(f_i)^L(t_{X^\cdot}))$, and the corresponding characterisation of pull-backs: with the notation of the definition,we have $f_3^L \circ f_3^R = f_1^L \circ f_1^R \wedge f_2^L \circ f_2^R$. ∎

To check the next proposition, the following lemma which is derived from a more general theorem due to Eugenio Moggi, is convenient.

Lemma 14 *Let F be a continuous and stable functor from* **Dom** *to* **Dom**. *An indexed family (t_X) is continuous stable if, and only if, it is uniform.* ∎

We will also call such a family a continuous stable section (or just section) of the functor F. With this result, it is possible to generalize the definition of $\Pi(F)$.

Proposition 15 *Let* **C** *be an arbitrary category, and F a continuous and stable functor from* **C** $\times$ **Dom** *to* **Dom**. *Define $\Pi(F)$, functor from* **C** *to* **Dom**, *so that, for A object of* **C**, *$\Pi(F)(A)$ is $\Pi(F(A,\_))$ as defined above*[1], *and if $f \in \mathbf{C}(A,B)$, then $\Pi(f)^L((t_X)) = (F(f, id_X)^L(t_X))$, and $\Pi(f)^R((u_X)) = (F(f, id_X)^R(u_X))$. Then $\Pi(F)$ is then a continuous and stable functor.* ∎

Indeed, it is straightforward to check, for instance, that $(F(f, id_X)^L(t_X))$ is a continuous stable family in X (and, from Moggi's result, we deduce the somewhat surprising fact that this family is also uniform). We will use the continuous and stable functors $\Rightarrow$ and $\times$, from **Dom** $\times$ **Dom** to **Dom**. If F and G are two functors from the same category **C** to **Dom**, we write $F \Rightarrow G$ for $\Rightarrow \circ \langle F, G\rangle$, and $F \times G$ for $\times \circ \langle F, G\rangle$. The same method may be applied to prove the next result.

Proposition 16 *Let F, G and H be three continuous and stable functors from the same category* **C** *into* **Dom**, *and K a continuous stable functor from* **C** $\times$ **Dom** *to* **Dom**. *We can define the operators* app, App, curry *and* Curry *on sections and continuous stable functors.*

- *if $t = (t_X)$ is a section for $F \Rightarrow (G \Rightarrow H)$ and $u = (u_X)$ a section for $F \Rightarrow G$, then* $\mathsf{app}(t, u)$ *is the section $(\lambda x.\ t_X(u_X(x)))$ of the functor $F \Rightarrow H$,*
- *if $t = (t_X)$ is a section for $F \times G \Rightarrow H$, then* $\mathsf{curry}(t)$ *is the section $(\lambda x.\ \lambda y.\ t_X(x, y))$ of the functor $F \Rightarrow (G \Rightarrow H)$,*
- *if $t = (t_X)$ a section of the functor $F \Rightarrow \Pi(K)$, then* $\mathsf{App}(t, G)$ *is the section of the functor $F \Rightarrow (K \circ \langle Id, G\rangle)$ defined as the family $(\lambda x.\ (t_X(x))_{G(X)})$,*

[1] Indeed, $F(A,\_)$ is a continuous and stable functor from **Dom** to **Dom**.

- *if $t = (t_X)$ is a section of the functor K, then* $\mathsf{Curry}(t)$ *is the section of the functor* $\Pi(K)$, *from* **C** *to* **Dom**, *defined by* $\mathsf{Curry}(t)_X = (t_{(X,Y)})$. ▮

5.2 The model

As was said before, the meaning $[\![\Sigma \vdash \sigma]\!]$ and $[\![H \vdash_\Sigma M : \sigma]\!]$ is given by structural induction.

First, we define $[\![\Sigma \vdash \sigma]\!]$ of a type (we'll write $[\![\sigma]\!]$ whenever Σ is clear enough). It is a a continuous and stable functor from $(\mathbf{Dom})^\Sigma$ to **Dom** (in particular, if Σ is empty, we see that the semantic of a type is a dI-domain). The definition is by case on σ

- $\sigma \equiv \alpha$, then $[\![\Sigma \vdash \sigma]\!]$ is the α'th projection functor,
- $\sigma \equiv \sigma_1 \Rightarrow \sigma_2$, then $[\![\sigma]\!]$ is $[\![\sigma_1]\!] \Rightarrow [\![\sigma_2]\!]$ as defined above,
- $\sigma \equiv \Pi\alpha.\ \sigma_1$, then we have $\Sigma, \alpha \vdash \sigma_1$, and $[\![\sigma_1]\!]$ is a functor from $(\mathbf{Dom})^\Sigma \times \mathbf{Dom}$ to **Dom**. We take $[\![\sigma]\!] = \Pi([\![\sigma_1]\!])$ as defined above.

Next, we define the meaning of $H \vdash_\Sigma M : \sigma$ (we'll write $[\![M]\!]$ if Σ and H are clear enough). For this, we remark that if $H = x_1 : \sigma_1, \ldots, x_n : \sigma_n$, then we have $\Sigma \vdash \sigma_1, \ldots, \Sigma \vdash \sigma_n$. Furthermore, we know that $\Sigma \vdash \sigma$. We thus have $\Sigma \vdash \sigma_1 \times \cdots \times \sigma_n \Rightarrow \sigma$. Hence, $[\![\sigma_1 \times \cdots \times \sigma_n \Rightarrow \sigma]\!]$ is a continuous and stable functor from $(\mathbf{Dom})^\Sigma$ to **Dom**. The denotation $[\![M]\!]$ is then a continuous and stable section (or, equivalently, a uniform section) of the functor $[\![\sigma_1 \times \cdots \times \sigma_n \Rightarrow \sigma]\!]$. The definition of $[\![M]\!]$ is by cases on M.

- $M \equiv x_i$, then σ is σ_i and $[\![M]\!]$ is the uniform family of i'th projections,
- $[\![M_1(M_2)]\!] = \mathsf{app}([\![M_1]\!], [\![M_2]\!])$,
- $[\![\lambda x : \tau.\ M_1]\!] = \mathsf{curry}([\![M]\!])$,
- $[\![M_1\{\tau\}]\!] = \mathsf{App}([\![M_1]\!], [\![\tau]\!])$,
- $[\![\Lambda(M)]\!] = \mathsf{Curry}([\![M]\!])$.

An alternative description of this kind of model, using the categorical presentation of Seely [15], is given in [4].

References

[1] Amadio, R., Bruce, K. B., Longo, G., *The finitary projection model for second order lambda calculus and solutions to higher order domain equations.* In: **Logic in Computer Science**, edited by A. Meyer, IEEE Computer Society Press, 1986, pp. 122–130.

[2] Berry, G., *Stable models of typed λ-calculi.* In: **Fifth International Colloquium on Automata, Languages and Programs**, Springer-Verlag, **Lecture Notes in Computer Science**, vol. 62, 1978, pp. 72–89.

[3] Bruce, K. and Meyer, A., *The semantics of polymorphic lambda-calculus.* In: **Semantics of Data Types**, edited by G. Kahn, D.B. MacQueen and G. Plotkin, **Lecture Notes in Computer Science**, vol. 173, Springer-Verlag, 1984, pp. 131–144.

[4] Coquand, T., and Ehrhard, T., *An equational presentation of higher-order logic.* Manuscript, 1987.

[5] Girard, J. Y., **Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.** Thèse d'Etat, Université Paris VII, 1972.

[6] Girard, J. Y., *The system F of variable types, fifteen years later.* Manuscript, 1985, 48 pp.

[7] Gunter, C. A., *Universal profinite domains.* **Information and Computing**, vol. 72 (1987), pp. 1–30.

[8] Kahn, G., and Plotkin, G., *Domaines concrets.* Rapport IRIA Laboria, no. 336, 1978.

[9] McCracken, N., **An Investigation of a Programming Language with a Polymorphic Type Structure**, Doctoral Dissertation, Syracuse University, 1979.

[10] Nielsen, M., Plotkin, G., Winskel, G., *Petri nets, event structures and domains.* **Theoretical Computer Science**, vol. 13, 1981.

[11] Fairbairn, J., *Design and implementation of a simple typed language based on the lambda-calculus.* University of Cambridge Computer Laboratory Technical Report, no. 75, 1985, 107pp.

[12] Reynolds, J. C., *Polymorphism is not set-theoretic.* In: **Semantics of Data Types**, edited by G. Kahn, D.B. MacQueen and G. Plotkin, **Lecture Notes in Computer Science**, vol. 173, Springer-Verlag, 1984, pp. 145–156.

[13] Reynolds, J. C., *Towards a theory of type structures.* In: **Colloque sur la Programmation**, Springer-Verlag, **Lecture Notes in Computer Science** 19, 1974, pp. 408–425.

[14] Scott, D. S., *Some ordered sets in computer science.* In: **Ordered Sets**, edited by I. Rival., D. Reidel Publishing Company, 1981, pp. 677–718.

[15] Seely, R., *Categorical semantics for higher order polymorphic lambda calculus.* Manuscript, 1986, 33pp.

[16] Smyth, M. B. and Plotkin, G. D, *The category-theoretic solution of recursive domain equations.* **SIAM Journal of Computing**, vol. 11 (1982), pp. 761–783.

[17] Winskel, G., **Events in Computation.** Doctoral Dissertation, University of Edinburgh, 1980.

[18] Winskel, G., *Event structures.* University of Cambridge Computer Laboratory Technical Report, no. 95, 1986, 69pp.