View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Edinburgh Research Explorer

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Computational Adequacy in an Elementary Topos

Citation for published version:
Simpson, A 1999, Computational Adequacy in an Elementary Topos. in G Gottlob, E Grandjean & K Seyr (eds), Computer Science Logic: 12th International Workshop, CSL'98, Annual Conference of the EACSL, Brno, Czech Republic, August 24-28, 1998. Proceedings. vol. 1584, Lecture Notes in Computer Science, vol. 1584, Springer-Verlag GmbH, pp. 323-342. DOI: 10.1007/10703163_22

Digital Object Identifier (DOI):
10.1007/10703163_22

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Computer Science Logic

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 05. Apr. 2019

Simpson, A. (1999). Computational Adequacy in an Elementary Topos. In Gottlob, G., Grandjean, E., & Seyr, K. (Eds.), Computer Science Logic. (pp. 323-342). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. doi: 10.1007/10703163_22

Computational Adequacy in an Elementary Topos

Alex K. Simpson

LFCS, Division of Informatics, University of Edinburgh
JCMB, King's Buildings, Edinburgh, EH9 3JZ
<Alex.Simpson@dcs.ed.ac.uk>

Abstract. We place simple axioms on an elementary topos which suffice for it to provide a denotational model of call-by-value PCF with sum and product types. The model is synthetic in the sense that types are interpreted by their set-theoretic counterparts within the topos. The main result characterises when the model is computationally adequate with respect to the operational semantics of the programming language. We prove that computational adequacy holds if and only if the topos is 1-consistent (i.e. its internal logic validates only true Σ^0_1-sentences).

1 Introduction

One axiomatic approach to domain theory is based on axiomatizing properties of the category of *predomains* (in which objects need not have a "least" element). Typically, such a category is assumed to be bicartesian closed (although it is not really necessary to require all exponentials) with natural numbers object, allowing the denotations of simple datatypes to be determined by universal properties. It is well known that such a category cannot have a fixed-point operator [9], but crucially predomains have a *lift* monad acting on them which plays a critical role in recovering the more familiar category of *domains* in which the expected fixed-point operator resides. The lift monad also determines the subcategory of strict functions between domains, with respect to which the fixed-point is characterised by the property of *uniformity*. Further, the monad determines a category of partial functions, which is arguably the most suitable category for program semantics. The development of this viewpoint can be found in [23, 2, 18, 29, 3].

In recent years it has become apparent that many natural categories of predomains arise as full subcategories of elementary toposes. For example, the category of ω-complete partial orders and ω-continuous functions is a full reflective subcategory of the Grothendieck topos, $\mathcal{H}$, considered in [6, 5]. More generally, models of Fiore and Plotkin's axioms for domain theory also have such embeddings [4]. Other categories of predomains are found as full subcategories of realizability toposes, see [15] for examples and references. Certain such examples have been shown to account for phenomena such as effectivity [21, 7] and strong stability [19]. Work in progress by a number of researchers looks likely to establish similar embeddings for categories of games.

The wealth of examples suggests that elementary toposes provide a plausible environment for developing a unified account of the many models of domain theory. However, early axiomatizations of domain theory inside toposes, [10, 31, 27, 24], included axioms that are valid only in restricted classes of models. A general axiomatization, encompassing both parallel and sequential computation, was proposed in [15], where its consequences were worked out in detail in the specific case of realizability toposes. This axiomatization has since proved to be applicable also to: Grothendieck toposes [5, 4], models of intuitionistic Zermelo-Frænkel set theory [30], and models of intuitionistic type theory [25].

In this paper we consider the same general axiomatic approach in the setting of an elementary topos. Given an elementary topos with a natural numbers object and a distinguished dominance (determining a lift monad), we isolate a full subcategory of predomains, the *well-complete* objects, as in [15]. Under a first (now ubiquitous) axiom, this category is closed under function spaces, products and the lift functor and supports recursion on appropriate objects. To obtain closure under finite coproducts (in the topos), it is necessary to strengthen the axiom in a simple way. For the natural numbers to be a predomain, yet another simple strengthening is necessary in general, although not in the case that Markov's Principle is valid in the topos (Theorem 1).

The axioms are sufficient to enable any of the many variants of PCF [22] to be modelled in the topos. Moreover, the closure properties of predomains have the important consequence that datatypes are modelled by their set-theoretic counterparts within the internal logic of the topos. Obtaining such a set-theoretic interpretation of type constructors was one of Dana Scott's motivations for proposing synthetic domain theory. He hoped that the set-theoretic viewpoint would lead to simple and intuitive logics for reasoning about programs. Whether or not this proves to be the case, we would like to stress another motivation for the work in this paper, and for the programme of synthetic domain theory as a whole. The axioms of synthetic domain theory allow technically distinct constructions across a wide range of models to be uniformly accounted for as examples of the same set-theoretic construction (when regarded from the viewpoint of the internal logic). Thus synthetic domain theory offers a general and unifying axiomatic account of the constructions of domain theory, and one which applies across a very wide class of models.

The main goal of this paper is to exploit this axiomatic account to understand the induced interpretation of PCF-like languages in the topos. For maximum generality, we consider a call-by-value version with sum and product types. The question we consider is: When is the interpretation of the language in the topos *computationally adequate* with respect to its operational semantics [8, 32]? We obtain a complete characterisation in terms of the internal logic of the topos. We prove that computational adequacy holds if and only if the topos is 1-consistent (Theorem 2). (A topos is said to be 1-consistent if it validates only true Σ^0_1-sentences.) Thus computational adequacy is equivalent to a simple logical property whose statement makes no reference to PCF. It is to be expected that an identical result will hold for other programming languages too.

2 Partiality and Lifting

Throughout this paper we assume that $\mathcal{E}$ is an elementary topos with a natural numbers object $\mathbf{N}$, see e.g. [13, 17]. We write $\mathbf{0}$ and $\mathbf{1}$ for chosen initial and terminal objects respectively, $[0, s] : \mathbf{1} + \mathbf{N} \longrightarrow \mathbf{N}$ for the structure map of $\mathbf{N}$ as the initial algebra of the endofunctor $1 + (-)$ on the topos, and $pred : \mathbf{N} \longrightarrow \mathbf{1} + \mathbf{N}$ for its inverse. We write $\mathbf{2}$ for the object $\mathbf{1} + \mathbf{1}$, denoting the left and right injections by $-, \top : \mathbf{1} \longrightarrow \mathbf{2}$. We consider $\mathbf{2}$ as a subobject of the subobject classifier, Ω, via a distinguished mono taking $-, \top : \mathbf{1} \longrightarrow \mathbf{2}$. to the standard $-, \top : \mathbf{1} \longrightarrow \Omega$.

We shall require one other piece of primitive data, an object Σ arising as a subobject $\Sigma \rightarrowtail \Omega$ (thus Σ classifies a family of subobjects, the *Σ-subobjects*, in $\mathcal{E}$). Moreover, we require that Σ is a *dominance* in the sense of Rosolini [26, 3]. Specifically this means that $\top : \mathbf{1} \longrightarrow \Omega$ factors through the mono $\Sigma \rightarrowtail \Omega$ (i.e. all isomorphisms are Σ-subobjects) and that Σ-subobjects are closed under composition.

Because $\mathbf{2}$ and Σ are subobjects of Ω, we shall often consider them, in the internal logic, as subsets of the set of all propositions. In particular, we sometimes refer to $\mathbf{2}$ as the subset of *logically decidable* propositions, because $\mathbf{2}$ can be defined internally as $\{p \in \Omega \mid p \vee \neg p\}$.

The conditions above imply that the collection of partial maps in $\mathcal{E}$ with Σ-subobjects for domains forms a category under the usual composition of partial maps. We write $X \rightharpoonup Y$ for the object of such *Σ-partial maps* from X to Y (easily defined using the internal logic). For $f \in X \rightharpoonup Y$ and $x \in X$, we shall loosely write $f(x)$ to mean a possibly undefined element of Y. We use equality between such possibly undefined expressions in the strict sense, thus $e = e'$ means that both e and e' are defined and equal. We write $e\downarrow$ to mean that e is defined, i.e. the Σ-property $\exists y \in Y.\, e = y$ holds (such a y is necessarily unique). We use Kleene equality, $e \simeq e'$ to mean that e is defined iff e' is, in which case both are equal. The above notation is adopted for its readability. The formally inclined reader will have no problem in translating the expressions we use into an appropriate rigorous logic for partial terms (see e.g. [28]), although our partial terms will always be special ones whose definedness property is a Σ-proposition.

The operation $\mathbf{1} \rightharpoonup (-)$ determines, in the obvious way, an endofunctor on $\mathcal{E}$, the *lift* functor L. Given $e \in LX$, we write $e\downarrow$ in place of the more cumbersome $e(*)\downarrow$ (where we write $*$ for the unique element of $\mathbf{1}$), and similarly, when $e\downarrow$ we write e for the determined value $e(*) \in X$. We write $\chi : LX \longrightarrow \Sigma$ for the morphism mapping any $e \in LX$ to its definedness property $e\downarrow$, which is indeed a Σ proposition.

The lift functor carries the structure of a monad (L, η, μ) on $\mathcal{E}$. Its unit $\eta : X \longrightarrow LX$ maps any x to the unique $e \in LX$ such that $e = x$ (thus $e\downarrow$). The lift monad is a strong commutative monad satisfying some additional well-documented properties, see e.g. [26, 3, 1]. The Kleisli category, $\mathcal{E}_L$, of the lift monad is isomorphic to the category of Σ-partial maps on $\mathcal{E}$. The object Σ is isomorphic to $L\mathbf{1}$ and $X \rightharpoonup Y$ is isomorphic to the exponential $(LY)^X$. An alternative, but equivalent, development is to take the lift monad as primitive

(as in [5]) and to derive the dominance and partial function space using the above isomorphisms

One consequence of $\mathcal{E}$ having a natural numbers object, is that the functor, L, has both an initial algebra $\sigma : L\mathbf{I} \longrightarrow \mathbf{I}$ and a final coalgebra $\tau : \mathbf{F} \longrightarrow L\mathbf{F}$. Moreover, the object $\mathbf{F}$ is a retract of $\Sigma^{\mathbf{N}}$, and the unique algebra homomorphism from σ to τ^{-1} is a mono $\iota : \mathbf{I} \longmapsto \mathbf{F}$. These results were proved in 1995 by the author and Mamuka Jibladze independently. For a published account see [11].

The initial algebra of L as a functor, interacts nicely with the monad structure on L. We call a morphism $\alpha : LX \longrightarrow X$ a *monad algebra* if it is an Eilenberg-Moore algebra for the monad (L, η, μ) [16]. The morphism $\mu' = \sigma \circ \mu \circ L\sigma^{-1}$ is a monad algebra $L\mathbf{I} \longrightarrow \mathbf{I}$. We write $up : \mathbf{I} \longrightarrow \mathbf{I}$ for the composite $\sigma \circ \eta$. The result below appears as Theorem A.5 of [12], where it is attributed to Bénabou and Jibladze.

Proposition 1. *For any monad algebra $\alpha : LX \longrightarrow X$ and any morphism $f : X \longrightarrow X$, there exists a unique algebra homomorphism $h : \mu' \longrightarrow \alpha$ such that $f \circ h = h \circ up$ (in $\mathcal{E}$).*

An important example of a monad algebra is $\alpha : L(X \rightharpoonup Y) \longrightarrow (X \rightharpoonup Y)$ defined by $\alpha(e)(x) \simeq e(x)$. When referring to such monad algebras, we shall just refer to the underlying object $X \rightharpoonup Y$, always understanding the structure map to be that given above.

3 Completeness and Fixed-points

The mono $\iota : \mathbf{I} \longmapsto \mathbf{F}$ plays a fundamental role in developing a basic notion of "chain completeness" sufficient for establishing fixed-points for endomorphisms on suitable objects. For further motivation see [15]. The results in this section are, by now, standard [15, 27, 25].

Definition 1. An object X is said to be *complete* if the induced morphism $X^{\iota} : X^{\mathbf{F}} \longrightarrow X^{\mathbf{I}}$ is an isomorphism.

If X is complete and $\alpha : LX \longrightarrow X$ is a monad algebra, then, for any morphism $f : X \longrightarrow X$ let $h : \mathbf{I} \longrightarrow X$ be the unique algebra homomorphism given by Proposition 1. Because X is complete, $h : \mathbf{I} \longrightarrow X$ extends along ι to a unique morphism $\overline{h} : \mathbf{F} \longrightarrow X$. Let $\infty : \mathbf{1} \longrightarrow \mathbf{F}$ be the unique coalgebra homomorphism from $\eta : \mathbf{1} \longrightarrow L\mathbf{1}$ to $\tau : \mathbf{F} \longrightarrow L\mathbf{F}$. We write $fix_{\alpha}(f)$ for the point $\overline{h} \circ \infty$.

Proposition 2. *For any monad algebra $\alpha : LX \longrightarrow X$ with X complete, and any morphism $f : X \longrightarrow X$, it holds that $f \circ fix_{\alpha}(f) = fix_{\alpha}(f)$. (i.e. fix is a fixed-point operator.)*

Moreover, for any monad algebras $\alpha : LX \longrightarrow X$ and $\beta : LY \longrightarrow Y$ (where X and Y are complete), morphisms $f : X \longrightarrow X$ and $g : Y \longrightarrow Y$, and algebra homomorphism $h : X \longrightarrow Y$, if $h \circ f = g \circ h$ then $fix_{\beta}(g) = h \circ fix_{\alpha}(f)$. (i.e. fix is uniform.)

Because the property of Σ being a dominance can be expressed in the internal logic [26], the dominance transfers to any slice topos $\mathcal{E}/Z$. Thus the above proposition holds in any slice. Essentially, this means that the obvious internalization of the above proposition holds in the internal logic of $\mathcal{E}$.

Our main application of uniformity will be in the proof of Lemma 5 in Section 7, where it will be used in the following form. Suppose that X is complete and carries a monad algebra $\alpha : LX \longrightarrow X$. We say that a mono $m : X' \longmapsto X$ *carries a subalgebra* of α if there exists $\alpha' : LX' \longmapsto X'$ such that $m \circ \beta = \alpha \circ Lm$. One easily shows that such an α' is unique, and further that it is itself a monad algebra. (Indeed, it holds for an arbitrary monad that any subalgebra of a monad algebra is also a monad algebra.) Now suppose that X' is complete and that an endomorphism $f : X \longrightarrow X$ restricts to an endomorphism $f' : X' \longrightarrow X'$ (i.e. that $m \circ f' = f \circ m$). Then it follows from the uniformity of fixed-points that $m \circ \mathit{fix}_{\alpha'}(f') = \mathit{fix}_{\alpha}(f)$, i.e. that $\mathit{fix}_{\alpha}(f)$ factors through the subobject $m : X' \longmapsto X$.

4 Axioms for Synthetic Domain Theory

The complete objects are not, in general, closed under the lifting functor, L, [20]. As our category of predomains we take the largest full subcategory of complete objects that *is* closed under lifting, following [15].

Definition 2. An object X is said to be *well-complete* if LX is complete.

We write $\mathcal{C}$ for the full subcategory of well-complete objects of $\mathcal{E}$. We shall adopt $\mathcal{C}$ as our category of predomains.

Our axioms will all be assertions that useful objects are well-complete. We begin with the basic *completeness axiom* of [15].

Axiom 1. $\mathbf{1}$ is well-complete.

Henceforth we assume that Axiom 1 holds. As a first consequence, we obtain a helpful reformulation of the notion of well-completeness [5].

Proposition 3. *The following are equivalent.*

1. *X is well-complete.*
2. $\mathcal{E} \models \forall F' \subseteq_{\Sigma} \mathbf{F}.\ \forall f \in X^{(\mathbf{I} \cap F')}.\ \exists! \overline{f} \in X^{F'}.\ f = \overline{f} \circ \iota.$

Here, as is standard, we write $\exists!$ for the unique existence quantifier. We henceforth take the formula in statement 2 above as stating the property of X being well-complete in the internal logic.

Axiom 1 alone implies many important closure properties of well-complete objects. The proposition below is now standard, with parts appearing essentially in [15, 5, 25]. The easiest proof uses the formulation of well-completeness given by Proposition 3.2.

Proposition 4. *1. Any well-complete object is complete.*

2. *For any internal family* $\{X_j\}_{j\in J}$ *in* $\mathcal{E}$ *(given by a morphism* $X \longrightarrow I$*),*

$$\mathcal{E} \models (\forall j \in J.\ X_j \textit{ well-complete}) \longrightarrow (\textstyle\prod_{j\in J} X_j) \textit{ well-complete}$$

3. *If* X *and* Y *are well-complete then, for any* $f, g : X \longrightarrow Y$*, the equalizing object is well-complete.*
4. *If* X *is well-complete then so is* LX*.*

Externally, statement 2 above implies that $\mathcal{C}$ is closed under finite products in $\mathcal{E}$ (hence, by statement 3, under finite limits too), and is moreover an exponential ideal of $\mathcal{E}$ (i.e. If Y is well-complete then so is Y^X for any X in $\mathcal{E}$). In particular $\mathcal{C}$ is cartesian closed. Furthermore, if Y is well-complete then so is $X \rightharpoonup Y$ (because $X \rightharpoonup Y \cong (LY)^X$ and $\mathcal{C}$ is closed under lifting).

At present there is nothing to prevent taking $\Sigma = \mathbf{1}$ (for any topos $\mathcal{E}$ whatsoever), in which case $\mathbf{1}$ is the only (well-)complete object. The remaining axioms will rule out such trivial examples (except in the case of a trivial $\mathcal{E}$), and, more positively, imply that $\mathcal{C}$ is closed under useful coproducts in $\mathcal{E}$. First, we consider the basic implications between the properties we shall later assume as axioms.

Proposition 5. *Consider the statements below.*

1. $\mathbf{0}$ *is well-complete.*
2. $\mathbf{2}$ *is well-complete.*
3. $\mathbf{N}$ *is well-complete.*

Then $3 \Rightarrow 2 \Rightarrow 1$.

Proof. $\mathbf{0}$ is an equalizer of $-, \top : \mathbf{1} \longrightarrow \mathbf{2}$, and $\mathbf{2}$ is a retract of $\mathbf{N}$. The implications follow by Proposition 4(3). □

Examples in [20] show that neither of the implications of Proposition 5 can be reversed in general.

In the remainder of this section we examine further consequences of and relationships between each of the 3 properties in Proposition 5.

Proposition 6. *The following are equivalent.*

1. $\mathbf{0}$ *is well-complete.*
2. $\mathcal{E} \models - \in \Sigma$.

Proof.

1 $\Longrightarrow$ 2. Suppose $\mathbf{0}$ is well-complete. Then $L\mathbf{0}$ is complete and carries a monad algebra $\mu : LL\mathbf{0} \longrightarrow L\mathbf{0}$. Therefore the identity $\mathsf{id} : L\mathbf{0} \longrightarrow L\mathbf{0}$ has a fixed-point $\mathit{fix}(\mathsf{id}) \in L\mathbf{0}$. Because $\mathcal{E} \models \forall x \in \mathbf{0}.\ -$, we have that $\mathcal{E} \models \neg(\mathit{fix}(\mathsf{id})\downarrow)$, i.e. $\mathcal{E} \models (\mathit{fix}(\mathsf{id})\downarrow) = -$. But $\mathcal{E} \models (\mathit{fix}(\mathsf{id})\downarrow) \in \Sigma$ (because $\mathit{fix}(\mathsf{id}) \in L\mathbf{0}$). Thus indeed $\mathcal{E} \models - \in \Sigma$.

2 $\Longrightarrow$ 1. If $- \in \Sigma$ then $\mathbf{1} \cong L\mathbf{0}$. Thus $L\mathbf{0}$ is complete, hence $\mathbf{0}$ is well-complete. □

Axiom 0. $\mathbf{0}$ is well-complete.

Henceforth in this section we assume Axiom 0.

A first consequence of Axiom 0 is that, for any object X of $\mathcal{E}$, we have a point $-_{LX} \in LX$ defined as the everywhere undefined partial function in $\mathbf{1} \rightharpoonup X$. Given $f : X \longrightarrow Y$, it is clear that $Lf : LX \longrightarrow LY$ maps $-_{LX}$ to $-_{LY}$. More generally, given any algebra for the lift functor, $\alpha : LX \longrightarrow X$, define $-_\alpha \in X$ by $-_\alpha = \alpha(-_{LX})$. For any $\beta : LY \longrightarrow Y$ and any algebra homomorphism $h : \alpha \longrightarrow \beta$, it is clear that $h(-_\alpha) = -_\beta$. In the case that there is an identified (usually monad) algebra α on an object X, we often write $-_X$ for $-_\alpha$, and refer to $-_X$ as the *bottom* of X. Thus algebra homomorphisms preserve bottoms.

Another consequence of Axiom 0 is that a morphism $step : \mathbf{N} \longrightarrow \mathbf{I}$ can be defined, using the initial algebra property of $\mathbf{N}$, as the unique map making the diagram below commute.

$$\begin{tikzcd}
\mathbf{1} + \mathbf{N} \arrow[r, "1 + step"] \arrow[d, "{[0,s]}"'] & \mathbf{1} + \mathbf{I} \arrow[d, "\sigma \circ [-_{L\mathbf{I}}, \eta]"] \\
\mathbf{N} \arrow[r, "step"] & \mathbf{I}
\end{tikzcd}$$

The following technical lemma (whose straightforward proof is omitted, see e.g. [20]) will be used in the proof of Theorem 1 below. Recall, e.g. from [17], that a mono $m : X' \longrightarrow X$ is said to be $\neg\neg$-*dense* if

$$\mathcal{E} \models \forall x \in X.\ \neg\neg(\exists x' \in X'.\ m(x') = x).$$

Lemma 1. *The morphism* $step : \mathbf{N} \longrightarrow \mathbf{I}$ *is a* $\neg\neg$*-dense mono.*

Next we examine condition 2 of Proposition 5.

Proposition 7. *The following are equivalent:*

1. $\mathbf{2}$ *is well-complete.*
2. $\mathcal{C}$ *is closed under finite coproducts in* $\mathcal{E}$.

PROOF. The proof from [15] transfers to this more general setting. □

Axiom 2. $\mathbf{2}$ is well-complete.

Henceforth we asume Axiom 2. By Proposition 5 we can now drop Axiom 0. In fact, it is straightforward to show that Axiom 2 alone implies Axiom 1, so we can also drop Axiom 1. Although we do not yet know that $\mathbf{N}$ is well-complete, Axiom 2 does allow a number of conditions to be given that are equivalent to, or at least imply, the well-completeness of $\mathbf{N}$.

To state the theorem, we require further terminology. An object X of $\mathcal{E}$ is said to be $\neg\neg$-*separated* if

$$\mathcal{E} \models \forall x, y \in X.\ \neg\neg(x = y) \longrightarrow x = y.$$

This is a standard concept in topos theory with many equivalent formulations (see e.g. [17]). The following easy result depends only on Σ being a subobject of Ω containing $\top$.

Lemma 2. *The following are equivalent.*

1. *Σ is $\neg\neg$-separated.*
2. $\mathcal{E} \models \forall p \in \Sigma.\ (\neg\neg p) \longrightarrow p.$

The property of Σ being $\neg\neg$-separated has been referred to as Markov's Principle in the synthetic domain theory literature, see e.g. [25]. For certain dominances in certain toposes (e.g. the semidecidable subobject classifier in the effective topos [26, 21]) this terminology is justified because the $\neg\neg$-separation of Σ is equivalent to the standard logical property known as Markov's Principle (see statement 4 of Theorem 1 below). However, in general, it seems sensible to maintain a distinction between the two non-equivalent properties. The theorem below, which depends heavily on Axiom 2, shows that both notions have a role to play in synthetic domain theory.

Theorem 1. *Consider the following statements.*

1. *$\mathbf{N}$ is well-complete.*
2. *$pred : \mathbf{N} \longrightarrow \mathbf{1} + \mathbf{N}$ is the final coalgebra for the functor $\mathbf{1} + (-)$ on the category $\mathcal{E}_L$ of Σ-partial maps.*
3. $\mathcal{E} \models \forall P \in \mathbf{2}^{\mathbf{N}}.\ (\exists n \in \mathbf{N}.\, P(n)) \in \Sigma.$
4. *Markov's Principle holds, i.e.*
   $\mathcal{E} \models \forall P \in \mathbf{2}^{\mathbf{N}}.\ \neg\neg(\exists n \in \mathbf{N}.\, P(n)) \longrightarrow \exists n \in \mathbf{N}.\, P(n).$

Then $1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftarrow 4$. Moreover, if Σ is $\neg\neg$-separated then $3 \Rightarrow 4$.

PROOF.

1 $\Longrightarrow$ 2. Suppose that $\mathbf{N}$ is well-complete, and that $h : X \rightharpoonup 1 + X$ is any Σ-partial map. We must show that there is a unique Σ-partial map $g : X \rightharpoonup \mathbf{N}$ such that the diagram below commutes.

$$\begin{tikzcd}
\mathbf{1} + X \arrow[r, "1+g"] & \mathbf{1} + \mathbf{N} \arrow[d, "{[0,s]}"] \\
X \arrow[u, "h"] \arrow[r, "g"] & \mathbf{N}
\end{tikzcd}$$

Define $\phi : (X \rightharpoonup \mathbf{N}) \longrightarrow (X \rightharpoonup \mathbf{N})$ by:

$$\phi(f) \quad = \quad [0, s] \circ (1 + f) \circ h$$

(the composition is, of course, composition of partial maps). By the well-completeness of $\mathbf{N}$, we have that $X \rightharpoonup \mathbf{N}$ is complete and carries a monad

algebra for L. Therefore, by Proposition 2, we can define $g = \mathit{fix}(\phi)$, which, by its very definition, makes the diagram commute.
For uniqueness, suppose that g, g' are two partial maps making the diagram commute. Then, by an internal induction on n,

$$\mathcal{E} \quad \models \quad \forall n \in \mathbf{N}.\, \forall x \in X.\, g(x) = n \ \mathit{iff}\ g'(x) = n.$$

(Here we are using our conventions about possibly undefined expressions, as $g(x)$ and $g'(x)$ need not be defined.) Thus $g = g'$.

2 $\Longrightarrow$ 3. Suppose that $\mathit{pred} : \mathbf{N} \longrightarrow \mathbf{1} + \mathbf{N}$ is the final coalgebra. Consider the map $d : \mathbf{2}^{\mathbf{N}} \longrightarrow \mathbf{1} + \mathbf{2}^{\mathbf{N}}$ defined, using the internal logic, by:

$$d(P) \quad = \quad \begin{cases} \mathit{inl}(*) & \text{if } P(0) \\ \mathit{inr}(\lambda n. P(n+1)) & \text{if not } P(0) \end{cases}$$

This is a good definition, because $P(0)$ is a logically decidable proposition. Let $h : \mathbf{2}^{\mathbf{N}} \longrightarrow \mathbf{N}$ be the unique coalgebra homomorphism from d to pred. Using only the fact that h is a coalgebra homomorphism, one has, by an internal induction on n, that

$$\mathcal{E} \ \models \ \forall n \in \mathbf{N}.\, \forall P \in \mathbf{2}^{\mathbf{N}}.\, d(P) = n \ \mathit{iff}\ (P(n) \ \wedge\ \forall m < n.\, \neg P(m)). \quad (1)$$

Note that $d(P)$ need not be defined. Indeed, we claim that

$$\mathcal{E} \quad \models \quad \forall P \in \mathbf{2}^{\mathbf{N}}.\, (\exists n \in \mathbf{N}.\, P(n)) \ \mathit{iff}\ d(P)\downarrow .$$

Statement 3 follows, because $(d(P)\downarrow) \in \Sigma$ (as d is a Σ-partial map).
The claim is derived from (1) by the following internal reasoning. For the right-to-left implication, if $d(P)\downarrow$ then $d(P) = n$ for some n. But, by (1), $d(P) = n$ implies $P(n)$, thus indeed $\exists n \in \mathbf{N}.\, P(n)$. For the converse, suppose $\exists n \in \mathbf{N}.\, P(n)$. Then, because, for all n, the proposition $P(n)$ is logically decidable, there exists a least n such that $P(n)$. It follows from (1) that $d(P)$ equals this least n. Hence $d(P)\downarrow$ as required.

3 $\Longrightarrow$ 1. Consider the subobject $D \longmapsto \mathbf{2}^{\mathbf{N}}$ defined by

$$D \quad = \quad \{P \in \mathbf{2}^{\mathbf{N}} \mid \forall n \in \mathbf{N}.\, P(n) \ \longrightarrow\ \forall m < n.\, \neg P(m)\}.$$

As the proposition $\forall m < n.\, \neg P(m)$ is logically decidable, it is easy to show that the mono $D \longmapsto \mathbf{2}^{\mathbf{N}}$ is split. Hence D is a retract of $\mathbf{2}^{\mathbf{N}}$, and thus well-complete (because $\mathbf{2}^{\mathbf{N}}$ is well-complete by Axiom 2).
Consider the subobject $C \longmapsto D$ defined by

$$C \quad = \quad \{P \in D \mid \exists n \in \mathbf{N}.\, P(n)\}.$$

Assume that statement 3 of the theorem holds. Then the inclusion $C \longmapsto D$ is an equalizer of the maps $\lambda P \in D.\, (\exists n \in \mathbf{N}.\, P(n)) : D \to \Sigma$ (which is a map to Σ by the assumption) and $\lambda P \in D.\, \top$. Therefore C is well-complete. We show that $\mathbf{N}$ is isomorphic to C, hence $\mathbf{N}$ is indeed well-complete. The isomorphism from $\mathbf{N}$ to C maps n to $\lambda m \in \mathbf{N}.\, (n = m)$. Its inverse maps any $P \in C$ to the unique n such that $P(n)$. It is easily checked that the two maps are indeed mutually inverse.

4 $\Longrightarrow$ 3. First define a function $\phi : \Sigma^{\mathbf{2^N}} \longrightarrow \Sigma^{\mathbf{2^N}}$ by:

$$\phi(f)(P) \quad = \quad \begin{cases} \top & \text{if } P(0) \\ f(\lambda n.\, P(n+1)) & \text{if not } P(0). \end{cases}$$

Now $\Sigma^{\mathbf{2^N}}$ is complete and carries a monad algebra for L. So, by Proposition 2, we can define a morphism $e : \mathbf{2^N} \longrightarrow \Sigma$ by $e = \mathit{fix}(\phi)$. We claim that Markov's Principle implies:

$$\mathcal{E} \quad \models \quad \forall P \in \mathbf{2^N}.\ (\exists n \in \mathbf{N}.\, P(n)) \ \mathit{iff}\ e(P)$$

Statement 3 follows because $e(P) \in \Sigma$.
For the left-to-right implication of the claim, one proves (internally) that, for all $n \in \mathbf{N}$, $P(n)$ implies $e(P)$. This is a straightforward induction on n, using only the fixed-point property of e. Markov's Principle is not required.
To prove the right-to-left implication of the claim, let $h : \mathbf{I} \longrightarrow \Sigma^{\mathbf{2^N}}$ be the unique algebra homomorphism (given by Proposition 1) such that $\phi \circ h = h \circ \mathit{up}$. Let $\overline{h} : \mathbf{F} \longrightarrow \Sigma^{\mathbf{2^N}}$ be the unique extension of h. Then, by definition, $e = \overline{h}(\infty)$. Assume $\neg\exists n \in \mathbf{N}.P(n)$. Below we show that, for all $i \in \mathbf{I}$, $h(i)(P) = -$. It follows that the unique extension $\lambda j \in \mathbf{F}.\overline{h}(j)(P) : \mathbf{F} \longrightarrow \Sigma$ of $\lambda i \in \mathbf{I}.\, h(i)(P) : \mathbf{I} \longrightarrow \Sigma$ is the constant function $\lambda j \in \mathbf{F}. - : \mathbf{F} \longrightarrow \Sigma$. Therefore $e(P) = \overline{h}(\infty)(P) = -$, i.e. $\neg e(P)$. Thus $\neg\exists n \in \mathbf{N}.\, P(n)$ implies $\neg e(P)$, or equivalently $e(P)$ implies $\neg\neg\exists n \in \mathbf{N}.\, P(n)$. Thus, by Markov's Principle, $e(P)$ implies $\exists n \in \mathbf{N}.\, P(n)$ as required.
It remains to show that $\neg\exists n \in \mathbf{N}.\, P(n)$ implies, for all $i \in \mathbf{I}$, $h(i)(P) = -$. Assume $\neg\exists n \in \mathbf{N}.\, P(n)$. The monad algebras on $\mathbf{I}$ and $\Sigma^{\mathbf{2^N}}$ determine their bottoms: $-_{\mathbf{I}} = \mathit{step}(0)$ and $-_{\Sigma^{\mathbf{2^N}}} = \lambda P. -$. As algebra homomorphisms preserve bottoms, $h(\mathit{step}(0))(P) = -$. Now, by an easy induction on n, for all $n \in \mathbf{N}$, it holds that $h(\mathit{step}(n))(P) = -$ (because we have assumed $\forall n.\neg P(n)$). However, by Lemma 1, we have that, for all $i \in \mathbf{I}$, $\neg\neg(\exists n.\, i = \mathit{step}(n))$. Therefore, for all $i \in \mathbf{I}$, $\neg\neg(h(i)(P) = -)$, i.e. $\neg\neg\neg h(i)(P)$, i.e. $\neg h(i)(P)$, i.e. $h(i)(P) = -$.

3 $\Longrightarrow$ 4. Immediate from Lemma 2 when Σ is $\neg\neg$-separated.

□

Axiom N. $\mathbf{N}$ is well-complete.

By Proposition 5, Axiom N implies Axiom 2, so in its presence Axiom 2 may be dropped. However, in practice it might be more convenient to establish Axiom 2 directly, and then use one of the conditions of Theorem 1 to derive Axiom N. Markov's Principle is a particularly useful condition because it is independent of the definition of Σ. Indeed the proof in [15] that Axiom 2 implies Axiom N in realizability models makes implicit use of the validity of Markov's Principle in all realizability toposes. Markov's Principle is also valid in all presheaf toposes, but not in all Grothendieck toposes. The other statements in Theorem 1 also have their uses. For example, in Section 8, we make crucial use of statement 3 as a consequence of Axiom N.

We call an elementary topos $\mathcal{E}$ together with a dominance Σ satisfying Axiom N a *natural model (of synthetic domain theory)*. (The adjective "natural" is to emphasise that the natural numbers object is well-complete.) All the realizability examples considered in [15] provide natural models, as does the the model $\mathcal{H}$ from [6, 5]. Throughout the rest of this paper, unless otherwise stated, we assume that $\mathcal{E}$ and Σ together form a natural model.

5 Interpreting a Programming Language

The axioms we have are sufficient for simply-typed programming languages like PCF [22] and its variants to be modelled in $\mathcal{E}$. We exploit all the structure we have identified on well-complete objects by including sums and product types in the language. The call-by-value language we introduce is essentially equivalent to similar languages considered in e.g. [8, 32].

We shall use σ, τ, ... to range over *types*, which are given by the grammar:

$$\sigma \ ::= \ \mathbf{1} \mid \mathbf{N} \mid \sigma + \tau \mid \sigma \times \tau \mid \sigma \to \tau.$$

Assuming a countably infinite collection of variable symbols, *contexts* are finite sequences of variable-type assignments, written $x_1 : \sigma_1, \ldots, x_k : \sigma_k$. (We do not assume that all the x_i are distinct.) We use Γ, ... to range over contexts, and we write $x \in \Gamma$ to say that the variable x appears in Γ.

$$\frac{x \notin \Gamma'}{\Gamma, x:\sigma, \Gamma' \vdash x:\sigma} \qquad \frac{}{\Gamma \vdash *:\mathbf{1}} \qquad \frac{}{\Gamma \vdash 0:\mathbf{N}} \qquad \frac{\Gamma \vdash M:\mathbf{N}}{\Gamma \vdash s(M):\mathbf{N}} \qquad \frac{\Gamma \vdash M:\mathbf{N}}{\Gamma \vdash pred(M) : \mathbf{1}+\mathbf{N}}$$

$$\frac{\Gamma \vdash M:\sigma}{\Gamma \vdash inl_\tau(M) : \sigma+\tau} \qquad \frac{\Gamma \vdash M : \sigma_1+\sigma_2 \quad \Gamma, x_1:\sigma_1 \vdash N_1:\tau \quad \Gamma, x_2:\sigma_2 \vdash N_2:\tau}{\Gamma \vdash case\ M\ of\ inl(x_1).\, N_1,\ inr(x_2).\, N_2 : \tau}$$

$$\frac{\Gamma \vdash M:\tau}{\Gamma \vdash inr_\sigma(M) : \sigma+\tau} \qquad \frac{\Gamma \vdash M:\sigma \quad \Gamma \vdash N:\tau}{\Gamma \vdash (M,N) : \sigma\times\tau} \qquad \frac{\Gamma \vdash M : \sigma\times\tau}{\Gamma \vdash fst(M) : \sigma} \qquad \frac{\Gamma \vdash M : \sigma\times\tau}{\Gamma \vdash snd(M) : \tau}$$

$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x:\sigma.\, M : \sigma\to\tau} \qquad \frac{\Gamma \vdash M : \sigma\to\tau \quad \Gamma \vdash N : \tau}{\Gamma \vdash M(N) : \tau} \qquad \frac{\Gamma, f:\sigma\to\tau, x:\sigma \vdash M:\tau}{\Gamma \vdash rec\, f:\sigma\to\tau\,(x).\, M : \sigma\to\tau}$$

Fig. 1. Typing rules

The notation for terms is introduced simultaneously with their typing rules, which are presented in Fig. 1. These rules manipulate judgements $\Gamma \vdash M : \sigma$, which say that the term M has type σ in context Γ. The notions of bound and free variable occurrences are defined in the standard way. We *do not* identify terms up to alpha equivalence (as we have no reason to do so). (This simplifies the formalization of the programming language in $\mathcal{E}$.)

Observe that M has at most one type in any Γ (as just sufficient type information is included in terms for this to hold). Moreover, any typing judgement

$\Gamma \vdash M : \sigma$ has at most one derivation. If $M : \sigma$ is derivable in the empty context then we say that M is a *program* of type σ.

$$\frac{}{* \Longrightarrow *} \qquad \frac{}{0 \Longrightarrow 0} \qquad \frac{M \Longrightarrow V}{s(M) \Longrightarrow s(V)} \qquad \frac{M \Longrightarrow 0}{pred(M) \Longrightarrow inl(*)} \qquad \frac{M \Longrightarrow s(V)}{pred(M) \Longrightarrow inr(V)}$$

$$\frac{M \Longrightarrow V}{inl(M) \Longrightarrow inl(V)} \qquad \frac{M \Longrightarrow inl(U) \quad N_1[U/x_1] \Longrightarrow V}{case\ M\ of\ inl(x_1).\, N_1,\ inr(x_2).\, N_2 \Longrightarrow V}$$

$$\frac{M \Longrightarrow V}{inr(M) \Longrightarrow inr(V)} \qquad \frac{M \Longrightarrow inr(U) \quad N_2[U/x_2] \Longrightarrow V}{case\ M\ of\ inl(x_1).\, N_1,\ inr(x_2).\, N_2 \Longrightarrow V}$$

$$\frac{M \Longrightarrow U \quad N \Longrightarrow V}{(M, N) \Longrightarrow (U, V)} \qquad \frac{(M, N) \Longrightarrow (U, V)}{fst(M) \Longrightarrow U} \qquad \frac{(M, N) \Longrightarrow (U, V)}{snd(M) \Longrightarrow V}$$

$$\frac{}{\lambda x.\, M \Longrightarrow \lambda x.\, M} \qquad \frac{M \Longrightarrow \lambda x.\, L \quad N \Longrightarrow U \quad L[U/x] \Longrightarrow V}{M(N) \Longrightarrow V}$$

$$\frac{}{rec\, f\,(x).\, M \Longrightarrow \lambda x.\, M[\, rec\, f\,(x).\, M\, /\, f\,]}$$

Fig. 2. Operational semantics

The behaviour of programs is specified by the operational semantics presented in Fig. 2. The rules manipulate judgements of the form $M \Longrightarrow V$ where both M and V are programs (to ease readability, we omit type information from the terms). In the rules, we write $N[U/x]$ to mean the term N with program U substituted for all free occurrences of x in N. Because the only entities ever substituted are closed terms, the possibility of variable capture does not arise, therefore a simple textual substitution suffices.

The programs V which appear on the right-hand side of the arrow in Fig. 2 are known as *values*. It is straightforward to show that, for any program M, there exists at most one value V such that $M \Longrightarrow V$. Moreover, for any M, V, there exists at most one derivation of the judgement $M \Longrightarrow V$. These results state that program evaluation is deterministic. It is also consistent with the typing rules: if M is a program of type σ and $M \Longrightarrow V$ then V is also has type σ. (Because we use a a simple textual substitution and do not identify terms up to alpha-equivalence, this last result depends on the possibility, which we do permit, of contexts containing repeated variables.)

We now turn to the denotational semantics. Types σ are interpreted as well-complete objects $[\![\sigma]\!]$, using the full strength of Axiom N to obtain the desired closure conditions. The definition of $[\![\sigma]\!]$ is by the expected induction on types. From the viewpoint of the internal logic of $\mathcal{E}$, the interpretation gives the full set-theoretic Σ-partial type hierarchy over $\mathbf{N}$ (i.e. $[\![\mathbf{1}]\!] = \mathbf{1}$; $[\![\mathbf{N}]\!] = \mathbf{N}$; $[\![\sigma + \tau]\!] = [\![\sigma]\!] + [\![\tau]\!]$; $[\![\sigma \times \tau]\!] = [\![\sigma]\!] \times [\![\tau]\!]$; $[\![\sigma \to \tau]\!] = [\![\sigma]\!] \rightharpoonup [\![\tau]\!]$).

A context $x_1{:}\,\sigma_1, \ldots, x_n{:}\,\sigma_n$ is interpreted as a product $[\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!]$. A term M such that $\Gamma \vdash M : \sigma$ is interpreted as a Σ-partial map $[\![M]\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!]$. This map can be defined by induction on the structure of M, using the external structure of $\mathcal{C}$. However, for later purposes, it is convenient to make an equivalent definition, using the internal logic of $\mathcal{E}$. For any term M such that $\Gamma \vdash M : \sigma$ we define, by induction on the structure of M, a definite description (in the internal logic) of a partial function $(\!|M|\!) \in ([\![\Gamma]\!] \rightharpoonup [\![\sigma]\!])$. Then $(\!|M|\!)$ determines a morphism $\mathbf{1} \longrightarrow ([\![\Gamma]\!] \rightharpoonup [\![\sigma]\!])$ in $\mathcal{E}$, which in turn determines the required Σ-partial map $[\![M]\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!]$.

Most of the definition of $(\!|M|\!)$ is straightforward (indeed set-theoretic). The most interesting clause is the definition of $(\!|\mathit{rec}\, f : \sigma \to \tau\ (x).\, M|\!)$. Suppose that $\Gamma, f{:}\,\sigma \to \tau, x{:}\,\sigma \vdash M : \tau$. Then we have $(\!|M|\!) \in ([\![\Gamma]\!] \times [\![\sigma \to \tau]\!] \times [\![\sigma]\!]) \rightharpoonup [\![\tau]\!]$. So:

$$\mathcal{E} \models \forall \gamma \in [\![\Gamma]\!].\, (\lambda f \in [\![\sigma]\!] \rightharpoonup [\![\tau]\!].\, (\lambda x \in [\![\sigma]\!].\, (\!|M|\!)(\gamma, f, x))) \in ([\![\sigma]\!] \rightharpoonup [\![\tau]\!])^{([\![\sigma]\!] \rightharpoonup [\![\tau]\!])}.$$

Write $\phi_{M\gamma}$ for $(\lambda f \in [\![\sigma]\!] \rightharpoonup [\![\tau]\!].\, (\lambda x \in [\![\sigma]\!].\, (\!|M|\!)(\gamma, f, x)))$. As $[\![\sigma]\!] \rightharpoonup [\![\tau]\!]$ is well-complete, hence complete, with a monad algebra structure (see Section 2), we use Proposition 2 to define $(\!|\mathit{rec}\, f : \sigma \to \tau\ (x).\, M|\!) \in [\![\Gamma]\!] \rightharpoonup [\![\sigma \to \tau]\!]$ by:

$$(\!|\mathit{rec}\, f : \sigma \to \tau\ (x).\, M|\!)(\gamma) \ = \ \mathit{fix}(\phi_{M\gamma}).$$

Observe that $(\!|\mathit{rec}\, f : \sigma \to \tau\ (x).\, M|\!)$ is a total function from $[\![\Gamma]\!]$ to $[\![\sigma]\!] \rightharpoonup [\![\tau]\!]$.

6 Computational Adequacy

We now come to the main question addressed in this paper, the question of *computational adequacy*, relating the operational behaviour of the programming language and its denotational interpretation.

We say that a program $M : \sigma$ *converges* (notation $M\Downarrow$) if there exists V such that $M \Longrightarrow V$. For the denotational analogue, we write $[\![M]\!]\downarrow$ if the Σ-partial map $[\![M]\!] : \mathbf{1} \longrightarrow [\![\sigma]\!]$ is total. Denotational convergence corresponds to the expected property of $(\!|M|\!)$ in the internal logic of $\mathcal{E}$. We have $(\!|M|\!) \in \mathbf{1} \rightharpoonup [\![\sigma]\!]$, i.e. $(\!|M|\!) \in L[\![\sigma]\!]$, and, using our conventions for elements of lifted objects, $[\![M]\!]\downarrow$ if and only if $\mathcal{E} \models (\!|M|\!)\downarrow$. By an easy induction on operational derivations, one shows that for all programs $M : \sigma$, if $M \Longrightarrow V$ then $[\![M]\!]\downarrow$ and $[\![M]\!] = [\![V]\!]$.

Definition 3. We say that $[\![\cdot]\!]$ is *computationally adequate* if, for all programs $M : \sigma$, $[\![M]\!]\downarrow$ implies $M\Downarrow$.

As stated in the introduction, we shall obtain a complete characterisation of computational adequacy in terms of a logical property of $\mathcal{E}$. Recall that any semidecidable k-ary relation on $\mathbf{N}$ can be written in the standard form $\exists n_1 \in \mathbf{N}, \ldots, n_l \in \mathbf{N}.\, P(m_1, \ldots, m_k, n_1, \ldots, n_k)$, where P is a $(k+l)$-ary primitive recursive relation. By a natural encoding of primitive recursive predicates in the internal logic of $\mathcal{E}$ [13], one obtains the analogous formal notion of a Σ^0_1-*formula*. In particular, a Σ^0_1-*sentence* is a sentence of the form

$\exists n_1 \in \mathbf{N}, \ldots, n_l \in \mathbf{N}.\, \psi(n_1, \ldots, n_k)$, where ψ is a naturally encoded k-ary primitive recursive predicate. (In fact, because one can define a primitive recursive pairing operation, it is sufficient to consider just unary predicates ψ.) We say that $\mathcal{E}$ is *1-consistent* if, for all Σ^0_1-sentences ϕ, it holds that $\mathcal{E} \models \phi$ implies that ϕ is true in reality. The next theorem is the main result of the paper.

Theorem 2. *The following are equivalent.*

1. $[\![\cdot]\!]$ *is computationally adequate.*
2. $\mathcal{E}$ *is 1-consistent.*

To prove the theorem, we carry out, as far as possible, a proof of computational adequacy within the internal logic of $\mathcal{E}$. In fact $\mathcal{E}$ always believes itself to be computationally adequate. The property of 1-consistency is just what is needed to relate this internal belief with external reality.

First, we need to formalize the syntax of the programming language, and its operational semantics, within the internal logic. This is an exercise in Gödel numbering. One encodes the types, terms and contexts as natural numbers in a standard way, so that all convenient operations on them are primitive recursive. Moreover, the relation $\Gamma \vdash M : \sigma$ is also primitive recursive, as the term M determines the whole derivation tree, allowing a primitive recursive decision procedure. We write $\mathcal{P}_\sigma$ for the formalized set of programs of type σ, which is, via its Gödel numbering, a primitive recursive subset of $\mathbf{N}$.

For the operational semantics, the Gödel numbering is extended to encode *derivations* of evaluation judgements $M \Longrightarrow V$. The relation "π is a derivation of $M \Longrightarrow V$" is a a primitive recursive ternary relation on π, M and V. Thus the binary relation $M \Longrightarrow V$ and the unary predicate $M\Downarrow$ are both Σ^0_1.

The proposition below, is stated using the formalized operational semantics.

Proposition 8. *For all programs $M{:}\sigma$, if $[\![M]\!]\downarrow$ then $\mathcal{E} \models M\Downarrow$.*

The lengthy proof is left for Sections 7 and 8. Here we apply the result to prove Theorem 2.

The $2 \Longrightarrow 1$ implication of Theorem 2 is now immediate. If $\mathcal{E}$ is 1-consistent then we have that $[\![M]\!]\downarrow$ implies $\mathcal{E} \models M\Downarrow$, by Proposition 8, which in turn implies $M\Downarrow$, by 1-consistency.

For the converse, suppose $[\![\cdot]\!]$ is computationally adequate. We must show 1-consistency. Accordingly, let P be a primitive recursive predicate. Without loss of generality, we can assume P is unary. Using a straightforward encoding of primitive recursive predicates, we can define a program $M : \mathbf{N} \to (\mathbf{1} + \mathbf{1})$ such that, using only the fixed-point property of *fix*,

$$\mathcal{E} \models \forall n \in \mathbf{N}.\ (\!|M|\!)(n) = \mathit{inl}(*) \ \mathit{iff}\ P(n),$$

and also $M(\overline{n}) \Longrightarrow \mathit{inl}(*)$ if and only if $P(n)$ (where $\overline{n}$ is the value $s^n(0)$). Let $N : \mathbf{1}$ be the following search program.

$$(\,\mathit{rec}\, f{:}\mathbf{N}\to\mathbf{1}\,(n).\ \mathit{case}\ M(n)\ \mathit{of}\ \mathit{inl}(x).\,*,\ \mathit{inr}(y).\, f(s(n))\,)(0)$$

Then, by an internal induction on n using only the fixed-point property of *fix*,

$$\mathcal{E} \models (\exists n \in \mathbf{N}.\, P(n)) \longrightarrow (\!|N|\!)\downarrow .$$

Also, by an induction on the number of unfoldings of *rec* in the operational semantics, $N\Downarrow$ implies there exists n such that $P(n)$.

Now, to show 1-consistency, suppose that $\mathcal{E} \models \exists n \in \mathbf{N}.\, P(n)$. Then, by the implication derived above, $\mathcal{E} \models (\!|N|\!)\downarrow$, i.e. $[\![N]\!]\downarrow$. It follows, by computational adequacy, that $N\Downarrow$. Thus indeed there exists n such that $P(n)$. This completes the derivation of Theorem 2 from Proposition 8.

We end this section with some applications of Theorem 2. It is easily verified that any non-trivial Grothendieck topos is 1-consistent (indeed, it is a folk theorem that any Grothendieck topos validates all classically true sentences of elementary arithmetic). Also, any non-trivial realizability topos, $\mathcal{E}$, is 1-consistent (because the hom-set $\mathcal{E}(\mathbf{1}, \mathbf{N})$ consists of just the numerals). Therefore, by Theorem 2, any non-trivial natural model furnished by either a Grothendieck or realizability topos is computationally adequate. (A different argument for computational adequacy in the case of realizability toposes appears in [14].)

One may wonder whether in fact any non-trivial model is computationally adequate. As a negative application of Theorem 2, we show that this is not the case. To be precise, we say that a model $(\mathcal{E}, \Sigma)$ is *trivial* if $\mathcal{E}$ is equivalent to a category with a single (necessarily identity) morphism. Non-triviality alone implies many important well-behavedness properties of $\mathcal{E}$, e.g. the two morphisms $-, \top : \mathbf{1} \to \mathbf{2}$ are distinct and the numerals $\overline{n} : \mathbf{1} \to \mathbf{N}$ are all distinct. Non-triviality is obviously also a necessary condition for computational adequacy to hold.

Corollary 1. *There exist non-trivial natural models $(\mathcal{E}, \Sigma)$ such that the induced $[\![\cdot]\!]_{(\mathcal{E},\Sigma)}$ is not computationally adequate.*

PROOF. Suppose, on the contrary, that any non computationally adequate model is trivial. Let ψ be the (easily constructed) sentence in the internal logic of an elementary topos with natural numbers object and distinguished object Σ saying "Σ is a dominance and Axiom N holds". Thus $(\mathcal{E}, \Sigma) \models \psi$ if and only if $(\mathcal{E}, \Sigma)$ is a natural model of synthetic domain theory. Let ϕ be any false Σ^0_1-sentence. By Theorem 2, any model of $\psi \wedge \phi$ is not computationally adequate and hence, by the assumption, trivial. By the completeness theorem for elementary toposes with respect to their internal logic [13], this means that $\neg(\psi \wedge \phi)$ is a theorem of the internal logic. On the other hand, when ϕ is a true Σ^0_1-sentence, then $\psi \wedge \phi$ is equivalent to ψ, which is consistent as there exist non-trivial natural models. In summary, for Σ^0_1-sentences ϕ, we have that $\neg(\psi \wedge \phi)$ is a theorem in the internal logic of toposes if and only if ϕ is false. But the theorems of the internal logic are recursively enumerable. Therefore, we can recursively enumerate the false Σ^0_1-sentences (i.e. the true Π^0_1-sentences). This is impossible, contradicting the initial assumption.

□

Incidentally, it is not too difficult to give a similar direct proof of Corollary 1 (not relying on Theorem 1) using the halting problem for the programming language in place of the truth of Σ^0_1-sentences.

7 Internal Adequacy

In this section we provide the missing proof of Proposition 8. This is achieved by formalising a standard relational proof of computational adequacy, see e.g. [8, 32], internally within $\mathcal{E}$.

For each type σ we define a predicate $\preceq^\sigma \longmapsto [\![\sigma]\!] \times \mathcal{P}_\sigma$ in the internal logic of $\mathcal{E}$. The definition proceeds inductively on the structure of σ, so that the relations satisfy the (internal) equivalences below.

$$
\begin{array}{lll}
* \preceq^{\mathbf{1}} M & \textit{iff} & M \Longrightarrow * \\
n \preceq^{\mathbf{N}} M & \textit{iff} & M \Longrightarrow \overline{n} \\
\mathit{inl}(d) \preceq^{\sigma+\tau} M & \textit{iff} & M \Longrightarrow \mathit{inl}(V) \ \textit{where}\ d \preceq^\sigma V \\
\mathit{inr}(d) \preceq^{\sigma+\tau} M & \textit{iff} & M \Longrightarrow \mathit{inr}(V) \ \textit{where}\ d \preceq^\tau V \\
(d, d') \preceq^{\sigma\times\tau} M & \textit{iff} & M \Longrightarrow (U, V) \ \textit{where}\ d \preceq^\sigma U \wedge d' \preceq^\tau V \\
f \preceq^{\sigma\to\tau} M & \textit{iff} & M \Longrightarrow \lambda x.\, L \ \textit{where}\ \forall d \in [\![\sigma]\!].\, \forall N \in \mathcal{P}_\sigma. \\
 & & \qquad (d \preceq^\sigma N \wedge f(d)\downarrow) \longrightarrow f(d) \preceq^\tau L[N/x]
\end{array}
$$

These equivalences are easily translated into formal definitions, in the internal logic, of the $\preceq^\sigma$ predicates. It is a straightforward consequence of the definitions that (internally) if $M, M' \in \mathcal{P}_\sigma$ are such that both $M \Longrightarrow V$ and $M' \Longrightarrow V$ (the same V) then, for all $d \in [\![\sigma]\!]$, $d \preceq^\sigma M$ iff $d \preceq^\sigma M'$. This fact will be used in the proof of Lemma 5 below without further comment.

Define $[\![\sigma]\!]_M = \{x \in [\![\sigma]\!] \mid x \preceq^\sigma M\}$. This definition determines, for any type σ, an internal $\mathcal{P}_\sigma$-indexed family, $\{[\![\sigma]\!]_M\}_{M\in\mathcal{P}_\sigma}$, of subobjects of $[\![\sigma]\!]$.

Lemma 3. *For all types σ, τ,*

$$\mathcal{E} \models \forall M \in \mathcal{P}_{\sigma\to\tau}.\ M\Downarrow \longrightarrow ([\![\sigma \to \tau]\!]_M \ \textit{carries a subalgebra of}\ [\![\sigma]\!] \rightharpoonup [\![\tau]\!]).$$

Proof. Recall $[\![\sigma]\!] \rightharpoonup [\![\tau]\!]$ has the algebra $\alpha : L([\![\sigma]\!] \rightharpoonup [\![\tau]\!]) \longrightarrow ([\![\sigma]\!] \rightharpoonup [\![\tau]\!])$ defined by $\alpha(e)(x) \simeq e(x)$. Reasoning internally in $\mathcal{E}$, suppose that $M \in \mathcal{P}_{\sigma\to\tau}$ and $M\Downarrow$. We show that $e \in L[\![\sigma \to \tau]\!]_M$ implies $\alpha(e) \in [\![\sigma \to \tau]\!]_M$. Suppose $e \in L[\![\sigma \to \tau]\!]_M$. As $M\Downarrow$, we have that $M \Longrightarrow \lambda x.\, L$ for some $\lambda x.\, L$. We must show that, for all $d \in [\![\sigma]\!]$ and all $N \in \mathcal{P}_\sigma$, if $d \preceq^\sigma N$ and $\alpha(e)(d)\downarrow$ then $\alpha(e)(d) \preceq^\tau L[N/x]$. But if $\alpha(e)(d)\downarrow$ then $\alpha(e)(d) = e(d)$ and also $e\downarrow$, hence $e \in [\![\sigma \to \tau]\!]_M$. It follows that $d \preceq^\sigma N$ implies $e(d) \preceq^\tau L[N/x]$, i.e. $\alpha(e)(d) \preceq^\tau L[N/x]$ as required. □

Lemma 4. *For all types σ,*

$$\mathcal{E} \models \forall M \in \mathcal{P}_\sigma.\ [\![\sigma]\!]_M \ \textit{is well-complete.}$$

The proof of Lemma 4, which is surprisingly involved, is given in Section 8.

Lemma 5. *If $x_1 : \sigma_1, \ldots x_k : \sigma_k \vdash M : \tau$ then*

$$\begin{array}{l} \mathcal{E} \models \forall d_1 \in [\![\sigma_1]\!] \ldots \forall d_k \in [\![\sigma_k]\!]. \forall N_1 \in \mathcal{P}_{\sigma_1} \ldots \forall N_k \in \mathcal{P}_{\sigma_k}. \\ \qquad d_1 \preceq^{\sigma_1} N_1 \wedge \ldots \wedge d_k \preceq^{\sigma_k} N_k \wedge (\!|M|\!)(d_1, \ldots, d_k)\downarrow \\ \qquad\qquad \longrightarrow (\!|M|\!)(d_1, \ldots, d_k) \preceq^{\tau} M[N_1, \ldots, N_k \,/\, x_1, \ldots, x_k] \end{array}$$

PROOF. This is proved by (external) induction on the derivation of $\Gamma \vdash M : \tau$ (where here and below we write Γ for the context $x_1 : \sigma_1, \ldots x_k : \sigma_k$). In each case, reasoning internally in $\mathcal{E}$, we suppose that, for each j with $1 \leq j \leq k$, we have $d_j \in [\![\sigma_j]\!]$ and $N_j \in \mathcal{P}_{\sigma_j}$ such that $d_k \preceq^{\sigma_k} N_k$. Then we show that $(\!|M|\!)(\overrightarrow{d})\downarrow$ implies $(\!|M|\!)(\overrightarrow{d}) \preceq^{\tau} M[\overrightarrow{N}/\overrightarrow{x}]$ (where, of course, $\overrightarrow{d}$ abbreviates $d_1, \ldots, d_k$, and $\overrightarrow{N}$ abbreviates $N_1, \ldots, N_k$). Here we consider only the critical case in which M is $rec\, f : \sigma' \to \tau'\, (y).\, M'$ (and hence τ is $\sigma' \to \tau'$).

In this case, $\Gamma \vdash M : \tau$ is derived from $\Gamma, f : \sigma' \to \tau', y : \sigma' \vdash M' : \tau'$. Given $\overrightarrow{d}$ and $\overrightarrow{N}$ as above, it holds automatically that $(\!|M|\!)(\overrightarrow{d})\downarrow$, so we must show that $(\!|M|\!)(\overrightarrow{d}) \preceq^{\sigma' \to \tau'} M[\overrightarrow{N}/\overrightarrow{x}]$, i.e. that $(\!|M|\!)(\overrightarrow{d}) \in [\![\sigma' \to \tau']\!]_{M[\overrightarrow{N}/\overrightarrow{x}]}$. Recall that $(\!|M|\!)(\overrightarrow{d}) = \mathit{fix}(\phi_{M'\overrightarrow{d}})$, where $\phi_{M'\overrightarrow{d}}$ is the endofunction on $[\![\sigma']\!] \rightharpoonup [\![\tau']\!]$. defined by $\phi_{M'\overrightarrow{d}}(f)(e) \simeq (\!|M'|\!)(\overrightarrow{d}, f, e)$. We claim that $\phi_{M'\overrightarrow{d}}$ restricts to an endofunction on $[\![\sigma' \to \tau']\!]_{M[\overrightarrow{N}/\overrightarrow{x}]}$. By Lemmas 3 and 4, $[\![\sigma' \to \tau']\!]_{M[\overrightarrow{N}/\overrightarrow{x}]}$ carries a monad algebra structure and is (well-)complete. By (the internalized) Proposition 2 (see, in particular, the discussion immediately after), the element $\mathit{fix}(\phi_{M'\overrightarrow{d}}) \in \sigma' \rightharpoonup \tau'$ is contained in the subset $[\![\sigma' \to \tau']\!]_{M[\overrightarrow{N}/\overrightarrow{x}]}$. Thus indeed $(\!|M|\!)(\overrightarrow{d}) \in [\![\sigma' \to \tau']\!]_{M[\overrightarrow{N}/\overrightarrow{x}]}$.

It remains to prove the claim. Suppose then that $f \preceq^{\sigma' \to \tau'} M[\overrightarrow{N}/\overrightarrow{x}]$. We must show that $\phi_{M'\overrightarrow{d}}(f) \preceq^{\sigma' \to \tau'} M[\overrightarrow{N}/\overrightarrow{x}]$. As M is $rec\, f : \sigma' \to \tau'\, (y).\, M'$, we have that $M[\overrightarrow{N}/\overrightarrow{x}] \Longrightarrow \lambda y : \sigma'.\, M'[\overrightarrow{N}, M[\overrightarrow{N}/\overrightarrow{x}] \,/\, \overrightarrow{x}, f]$. Take any $e \in [\![\sigma']\!]$ and $N' \in \mathcal{P}_{\sigma'}$ such that $e \preceq^{\sigma'} N'$ and $\phi_{M'\overrightarrow{d}}(f)(e)\downarrow$. We must show that $\phi_{M'\overrightarrow{d}}(f)(e) \preceq^{\tau'} M'[\overrightarrow{N}, M[\overrightarrow{N}/\overrightarrow{x}], N' \,/\, \overrightarrow{x}, f, y]$. But $\phi_{M'\overrightarrow{d}}(f)(e) = (\!|M'|\!)(\overrightarrow{d}, f, e)$ and indeed $\phi_{M'\overrightarrow{d}}(f)(e)\downarrow \preceq^{\tau'} M'[\overrightarrow{N}, M[\overrightarrow{N}/\overrightarrow{x}], N'/\overrightarrow{x}, f, y]$, by the induction hypothesis on M'. □

Proposition 8 is an easy consequence of Lemma 5. For any program $M : \sigma$, if $[\![M]\!]\downarrow$ then, equivalently, $\mathcal{E} \models (\!|M|\!)\downarrow$. So, by Lemma 5, $\mathcal{E} \models (\!|M|\!) \preceq^{\sigma} M$. Hence, by the definition of $\preceq^{\sigma}$, indeed $\mathcal{E} \models M\Downarrow$.

One remark about Lemma 5 is that the quantification over terms is external. This is by necessity, as there are limitations on the extent to which the denotational semantics can be formalized within $\mathcal{E}$. In particular, writing $\mathcal{T}$ for the formalized set of types (as a primitive recursive subset of $\mathbf{N}$), one cannot define within the internal logic a semantic function $[\![\cdot]\!] \in \prod_{\sigma \in \mathcal{T}} \mathcal{P}_{\sigma} \rightharpoonup [\![\sigma]\!]$, because the family $\{[\![\sigma]\!]\}_{\sigma \in \mathcal{T}}$ need not live as an internal family within $\mathcal{E}$ (its usual definition involves the Axiom of Replacement from set theory).

8 The Well-completeness of $[\![\sigma]\!]_M$

It remains only to prove Lemma 4, showing (internally) the well-completeness of the sets $[\![\sigma]\!]_M$. Although the proof of Lemma 5 required only the completeness of these sets, it is necessary to establish the well-completeness in order to have a property that can be proved by induction on types.

We begin with some useful closure properties of well-complete objects. To establish these, it is convenient to work with the formulation of well-completeness given by Proposition 3.2. (We write "X *inhabited*" for "$\exists x \in X.\ \top$".)

Lemma 6.

$$\mathcal{E} \models (\exists p \in \Sigma.\ (X \textit{ inhabited} \longrightarrow p) \wedge (p \longrightarrow X \textit{ well-complete})) \\ \longrightarrow X \textit{ well-complete}$$

PROOF. We reason in the internal logic. Let $p \in \Sigma$ satisfy the assumption. Suppose $F' \subseteq_\Sigma \mathbf{F}$. Take any $f \in X^{I'}$, where $I' = \mathbf{I} \cap F'$. We show that F' inhabited implies X well-complete (using the assumed f). First , observe that I' inhabited implies X inhabited (because $i \in I'$ implies $f(i) \in X$), so the two functions $\lambda i.\top \in \Sigma^{I'}$ and $\lambda i.p \in \Sigma^{I'}$ are equal (by the assumption). Therefore $\lambda j.\top \in \Sigma^{F'}$ and $\lambda j.p \in \Sigma^{F'}$ both extend $\lambda i.\top \in \Sigma^{I'}$. By the well-completeness of Σ, there is a unique such extension, thus, for all $j \in F'$, $p = \top$, i.e. F' inhabited implies p. But p implies X is well-complete. Thus indeed F' inhabited implies X is well-complete.

We must show there is a unique function $\overline{f} \in X^{F'}$ such that $f = \overline{f} \circ \iota$. For any $j \in F'$, we have that X is well-complete so there exists a unique $\overline{f_j} \in X^{F'}$ such that $f = \overline{f_j} \circ \iota$ (the notation is to emphasise that $\overline{f_j}$ depends on the assumed element j). Therefore $\overline{f} = \lambda j \in F'.\overline{f_j}(j)$ defines a function in $X^{F'}$. It is easily checked that this is the unique function such that $f = \overline{f} \circ \iota$. □

Lemma 7. *If $\{X_j\}_{j\in J}$ is an internal family of subobjects of a well-complete object Y then*

$$\mathcal{E} \models (\forall j \in J.\ X_j \textit{ well-complete}) \longrightarrow (\textstyle\bigcap_{j\in J} X_j) \textit{ well-complete.}$$

We now turn to the proof of Lemma 4. We prove, by an (external) induction on the structure of the type σ, the required property:

$$\mathcal{E} \models \forall M \in \mathcal{P}_\sigma.\ [\![\sigma]\!]_M \text{ is well-complete.}$$

Lemma 6 will be basic to the argument. To apply it, we crucially observe that, for any Σ^0_1-sentence ϕ, it holds that $\phi \in \Sigma$. This is because any primitive-recursive predicate determines a corresponding $P \in \mathbf{2}^{\mathbf{N}}$, so any Σ^0_1-sentence, ϕ, is of the form $\exists n \in \mathbf{N}.\ P(n)$ for some $P \in \mathbf{2}^{\mathbf{N}}$, and therefore $\phi \in \Sigma$, by Theorem 1.3. We consider just one case of the induction.

When σ is $\sigma' \to \tau'$, reasoning internally, take any $M \in \mathcal{P}_{\sigma'\to\tau'}$. If $[\![\sigma' \to \tau']\!]_M$ is inhabited then there exists (a unique) $(\lambda x.\ L) \in \mathcal{P}_{\sigma'\to\tau'}$ such that $M \Longrightarrow \lambda x.\ L$. We claim that $M \Longrightarrow \lambda x.\ L$ implies $[\![\sigma' \to \tau']\!]_M$ is well-complete. From which,

it follows, by Lemma 6, that $[\![\sigma' \to \tau']\!]_M$ is well-complete as required (as the proposition $M \Longrightarrow \lambda x.\, L$ is a Σ-property).

To prove the claim, observe that if $M \Longrightarrow \lambda x.\, L$ then

$$[\![\sigma' \to \tau']\!]_M \;=\; \bigcap_{N \in \mathcal{P}_{\sigma'}} \bigcap_{d \in [\![\sigma']\!]_N} X_{Nd}$$

where

$$X_{Nd} \;=\; \{f \in [\![\sigma' \to \tau']\!] \mid f(d)\downarrow \longrightarrow\; f(d) \preceq^{\tau'} L[N/x]\}.$$

As $[\![\sigma' \to \tau']\!]$ is well-complete, it suffices, by Lemma 7, to show the well-completeness of each set X_{Nd}. However, writing i for the inclusion function from $[\![\tau']\!]_{L[N/x]}$ to $[\![\tau']\!]$, and g for the function $\lambda f.\, f(d) : [\![\sigma' \to \tau']\!] \longrightarrow L[\![\tau']\!]$, we have an evident pullback diagram:

$$\begin{tikzcd}
X_{Nd} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner"] & L[\![\tau']\!]_{L[N/x]} \arrow[d, "Li"] \\
{[\![\sigma' \to \tau']\!]} \arrow[r, "p"] & L[\![\tau']\!]
\end{tikzcd}$$

As the three vertices of the diagram being pulled back are well-complete (in particular, $L[\![\tau']\!]_{L[N/x]}$ is well-complete by the induction hypothesis on τ'), so is the pullback X_{Nd} as required. This completes the proof of Lemma 4.

Acknowledgements

This paper was written during a visit to Utrecht University supported by the NWO Pionier Project *The Geometry of Logic* led by Ieke Moerdijk. I thank Jaap van Oosten for many helpful and interesting discussions. Paul Taylor's diagram macros were used.

References

1. A. Bucalo and G. Rosolini. Lifting. In *Category Theory and Computer Science, Proceedings of CTCS '97*. Springer LNCS 1290, 1997.
2. R.L. Crole and A.M. Pitts. New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic. *Inf. and Comp.*, 98:171–210, 1992.
3. M.P. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Cambridge University Press, 1996.
4. M.P. Fiore and G.D. Plotkin. An extension of models of axiomatic domain theory to models of synthetic domain theory. In *Proceedings of CSL 96*, pages 129–149. Springer LNCS 1258, 1997.
5. M.P. Fiore and G. Rosolini. The category of cpos from a synthetic viepoint. Presented at *MFPS XIII*, 1997.

6. M.P. Fiore and G. Rosolini. Two models of synthetic domain theory. *Journal of Pure and Applied Algebra*, 116:151–162, 1997.
7. P.J. Freyd, P. Mulry, G. Rosolini, and D.S. Scott. Extensional PERs. In *Proc. of 5th Annual Symposium on Logic in Computer Science*, 1990.
8. C.A. Gunter. *Semantics of Programming Languages*. MIT Press, 1992.
9. H. Huwig and A. Poigné. A note on inconstencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73:101–112, 1990.
10. J.M.E. Hyland. First steps in synthetic domain theory. In *Category Theory, Proceedings, Como 1990*. Springer LNM 1488, 1990.
11. M. Jibladze. A presentation of the initial lift algebra. *Journal of Pure and Applied Algebra*, 116:185–198, 1997.
12. A. Joyal and I. Moerdijk. *Algebraic Set Theory*. LMS Lecture Notes, CUP, 1995.
13. J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge studies in Advanced Mathematics. Cambridge University Press, 1986.
14. J.R. Longley. *Realizability Toposes and Language Semantics*. Ph.D. thesis, Department of Computer Science, University of Edinburgh, 1995.
15. J.R. Longley and A.K. Simpson. A uniform account of domain theory in realizability models. *Math. Struct. in Comp. Sci.*, 7:469–505, 1997.
16. S. Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer Verlag, 1971.
17. S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: a First Introduction to Topos Theory*. Universitext. Springer Verlag, 1992.
18. P. S. Mulry. Categorical fixed-point semantics. *Theoretical Computer Science*, 70:85–97, 1990.
19. J. van Oosten. A combinatory algebra for sequential functionals of higher type. In *Logic Colloquium 1997*. Cambridge University Press. To appear. Currently available as Preprint 996, Dept. of Mathematics, Utrecht University, 1997.
20. J. van Oosten and A.K. Simpson. *Axioms and (Counter)examples in Synthetic Domain Theory*. Preprint 1080, Dept. of Mathematics, Utrecht University, 1998.
21. W.K.-S. Phoa. Effective domains and intrinsic structure. In *Proceedings of 5th Annual Symposium on Logic in Computer Science*, 1990.
22. G.D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.
23. G.D. Plotkin. *Denotational semantics with partial functions*. Lecture notes, CSLI Summer School, 1985.
24. B. Reus. *Program Verification in Synthetic Domain Theory*. PhD thesis, University of Munich, 1995.
25. B. Reus and Th. Streicher. General synthetic domain theory — a logical approach. In *Proceedings of CTCS '97*, pages 293–313. Springer LNCS 1290, 1997.
26. G. Rosolini. *Continuity and Effectivity in Topoi*. PhD thesis, Oxford, 1986.
27. G. Rosolini. *Notes on Synthetic Domain Theory*. Unpublished notes, Available from `ftp://ftp.disi.unige.it/`, 1995.
28. D. S. Scott. Identity and existence in intuitionistic logic. In *Applications of Sheaves*, pages 661–696. Springer LNM 753, 1979.
29. A.K. Simpson. *Recursive types in Kleisli categories*. Unpublished manuscript. Available from `ftp://ftp.dcs.ed.ac.uk/pub/als/Research/`, 1992.
30. A.K. Simpson. *Algebraic Compactness in Intuitionistic Set Theory*. Presented at *PSSL*, Edinburgh, October 1995. In preparation, 1999.
31. P. Taylor. The fixed point property in synthetic domain theory. In *Proc. of 6th Annual Symposium on Logic in Computer Science*, 1991.
32. G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993.