Relative Completeness for Logics of Functional Programs

Bernhard Reus [1] and Thomas Streicher [2]

1 University of Sussex,
Brighton BN1 9QH, UK
bernhard@sussex.ac.uk

2 TU Darmstadt,
64289 Darmstadt, Germany
streicher@mathematik.tu-darmstadt.de

Abstract

We prove a relative completeness result for a logic of functional programs extending D. Scott's LCF. For such a logic, contrary to results for Hoare logic, it does not make sense to ask whether it is complete relative to the full theory of its first-order part, since the first order part does not determine uniquely the model at higher-order types. Therefore, one has to fix a model and choose an appropriate data theory w.r.t. which the logic is relatively complete. We establish relative completeness for two models: for the Scott model we use the theory of Baire Space as data theory, and for the effective Scott model we take first-order arithmetic. In both cases we need to extend traditional LCF in order to capture a sufficient amount of domain theory.

1998 ACM Subject Classification D.2.4 Program Verification; D.3.1 Formal Definitions and Theory; F.3.1 Specifying and Verifying and Reasoning about Programs; F.3.2 Semantics of Programming Languages

Keywords and phrases Completeness, Program Logics, LCF

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.470

1 Introduction

Program logics play an important role in Computer Science to complement testing. A program logic allows one to prove that a program satisfies a given specification. Seminal work has been done in the late sixties by Hoare on axiomatic semantics for stateful programs [8]. Since then many calculi have been developed for all kinds of programming languages and meanwhile mechanizations of these logics in numerous verification tools exist.

Two properties of a program logic are of particular interest. *Soundness* states that any property one can prove of a program in the calculus is actually valid. *Completeness* states the converse, namely that any valid property can also be derived. In an ideal world, a formal calculus for a program logic would be both, sound *and* complete, thus faithfully and completely reflecting the semantics of programs and correctness assertions, also called specifications.

Due to Gödel's Incompleteness Theorem it is hopeless to look for absolutely complete program logics since for any (sufficiently expressive) formal system S one can find a correctness assertion G_S which is true but cannot be derived in S. Nevertheless one might ask whether the axioms of some program logic $\mathcal{L}$ are sufficient for proving all true correctness assertions relative to some complete data theory $\mathcal{T}$, i.e. whether $\mathcal{L}$ is *complete relative* to $\mathcal{T}$.

In [6] this problem was considered for the case where $\mathcal{L}$ is the Hoare logic for a basic imperative language which can store and manipulate objects of a data structure and $\mathcal{T}$ is the complete first order theory of this data structure. Obviously, the logic $\mathcal{L}$ is complete relative to $\mathcal{T}$ provided that for every program P and postcondition B

(a) the *weakest liberal precondition* $wlp(P)(B)$ is expressible in $\mathcal{T}$ and
(b) $\{wlp(P)(B)\}P\{B\}$ is provable in $\mathcal{L}$

because by definition $\{A\}P\{B\}$ is equivalent to $A \Rightarrow wlp(P)(B)$ and $\mathcal{L}$ can derive $\{A\}P\{B\}$ from $A \Rightarrow wlp(P)(B)$ and $\{wlp(P)(B)\}P\{B\}$ via the consequence rule. In [6] it was shown that (b) holds under the assumption of (a), i.e. that $\mathcal{T}$ is *expressive* w.r.t $\mathcal{L}$. In practice, expressivity is ensured by the first order definability of $[\![P]\!]$, the semantics of P: if R is a first order relation expressing $[\![P]\!]$ then $wlp(P)(B)(s)$ can be expressed as $\forall s'.\ R(s,s') \Rightarrow B(s')$. A typical example is obtained by taking for $\mathcal{T}$ the set of all *true* first order sentences of arithmetic since for all programs P its input/output relation $[\![P]\!]$ is recursively enumerable and thus expressible by a formula of first order arithmetic.

To the best of our knowledge, the question of relative completeness for logics of *functional* programming languages has not been investigated so far[1] though it has been suggested in [13].

Historically, the first logic for a functional programming language was Dana Scott's LCF introduced in [26]. This is a (many-sorted) predicate logic whose terms are PCF programs as studied in [20] and many subsequent publications. There is some pragmatic evidence that most correctness assertions about PCF programs can be proved within LCF. But there are quite easy correctness assertions which can neither be proved nor disproved in LCF. Let, for example, $E(f)$ be the purely equational specification of the "parallel or" function then LCF proves neither $\exists f.E(f)$ nor its negation. The reason simply is that the former holds in the Scott model but its negation holds in the fully abstract model (cf. [17]) where "parallel or" does not exist (see [20]). Notice, however, that these two models are not different w.r.t. the data type `nat` of natural numbers (and the type `nat`→`nat` of unary functions on `nat`) but they do differ at higher types and, actually, already at type `nat`→`nat`→`nat`, the type of "parallel or". Accordingly, it does not make sense to ask whether LCF is relatively complete w.r.t. the *full theory of its first order part* since the latter – unlike for the basic imperative language considered in [6] – does not fully determine the (higher type part of the) model.

Thus, the right question is whether "natural" models for PCF can have nice axiomatizations $\mathcal{L}$ which are complete relative to a full data theory $\mathcal{T}$. Though "natural" is somewhat subjective one may want to consider the following three kinds of models:

(1) the Scott model and its effective variant (for an introduction see e.g. [27])
(2) the observably sequential algorithms model and its effective variant (see the original paper [4] or more modern adaptations like [10, 14, 15])
(3) fully abstract models like Milner's model (see [17, 27]) or its sequentially realised submodel $\mathcal{F}$ (see [18]).

The models in (1) allow one to interpret PCF^{++}, i.e. PCF extended with a parallel or and a continuous existential quantifier as in [20]. The models in (2) allow one to interpret SPCF, an extension of PCF with error elements and a `catch`-construct which allows one to observe sequentiality (see [4, 15]). In both cases all types σ appear as definable retracts of type `nat`→`nat`. Thus, it seems plausible that one can axiomatize these models by adding some "obvious" axioms to LCF (of course, depending on the kind of model) and show completeness

[1] A notable exception is [9] where a different form of completeness for a functional language with state is proven that is weaker (see Conclusions).

of the ensuing logics, relative to the complete theory $\mathcal{T}$ of the total strict elements of $\texttt{nat}{\to}\texttt{nat}$. In case of effective variants of these models it is, however, possible to instantiate $\mathcal{T}$ by the set of all arithmetic truths because it suffices to consider the computable elements of every type which can be encoded by natural numbers (see e.g. [20]).

In this paper we will perform the task for (1) in Sections 2 and 3, respectively. In the final section we will discuss the cases (2) and (3) and extensions to models with higher order references.

2 The Scott Model

Let $\mathcal{D}$ be the Scott model of PCF as introduced in [26]. In *loc.cit.* one finds a program logic LCF suitable for reasoning about elementary properties of PCF programs (see also Sect. 3.3 of [27] for a quick recap of LCF). However, the axioms of LCF are so general that they hold in all cpo-enriched order extensional models of PCF. The aim of this paper is to extend LCF to a logic $\mathcal{L}$ for which $\mathcal{D}$ is a model and which is complete relative to the complete theory $\mathcal{T}$ of Baire space $\mathbb{N}^{\mathbb{N}}$ (considered as a subset of the interpretation of $\texttt{nat}{\to}\texttt{nat}$ in the Scott model). This theory $\mathcal{T}$ will be modeled after the theory **EL** (short for *Elementary Analysis*) of [28] which is "an extension of Heyting arithmetic with variables and quantifiers for number-theoretic functions". Our theory $\mathcal{T}$ differs from **EL** in two respects: it is based on classical logic and formulated in a sublanguage of $\mathcal{L}$ which refers only to the strict total elements of $\texttt{nat}$ and $\texttt{nat}{\to}\texttt{nat}$. In order to stay within the realm of $\mathbb{N}^{\mathbb{N}}$, general recursion is not available in the language of $\mathcal{T}$ though primitive recursion is. But this is not a real restriction since all inhabited r.e. sets can be enumerated by a primitive recursive function. This fact will be used subsequently without further mention.

As shown in Plotkin's paper [21] every coherently complete countably algebraic domain appears as retract of $[\mathbb{N}_\perp{\to}\mathbb{N}_\perp]$, the interpretation of $\texttt{nat}{\to}\texttt{nat}$ in $\mathcal{D}$. By inspection of the proof in [21] one sees that $\texttt{nat}{\to}\texttt{nat}$ contains all PCF types as computable retracts of $\texttt{nat}{\to}\texttt{nat}$. Thus, from results in [20] it follows that for every PCF type σ there exist PCF^{++} programs $\mathsf{e}_\sigma : \sigma \to \texttt{nat} \to \texttt{nat}$ and $\mathsf{p}_\sigma : (\texttt{nat} \to \texttt{nat}) \to \sigma$ such that $\mathsf{p}_\sigma \circ \mathsf{e}_\sigma$ is the identity on D_σ (the interpretation of type σ in $\mathcal{D}$). By PCF^{++} we denote the extension of PCF with the "parallel or" operation $\texttt{por} : \texttt{nat} \to \texttt{nat} \to \texttt{nat}$ and Plotkin's continuous existential quantifier $\exists : (\texttt{nat} \to \texttt{nat}) \to \texttt{nat}$. As shown in [20] (see also final chapter of [27]) all computable elements of the Scott model arise as denotations of PCF^{++}-terms. Recall that an element $d \in D_\sigma$ is computable if, and only if, the set of codes of compact approximations to d is recursively enumerable by some (prim. rec.) function α_d. In the language $\mathcal{T}$ we can refer to elements of D_σ in terms of sequences $\alpha \in \mathbb{N}^{\mathbb{N}}$ which enumerate codes of compact approximations to d. See [3, 2] for further information on representations of domains and more general spaces via Baire space and its connection to function realizability.

We will define a program logic $\mathcal{L}$ which axiomatizes the Scott model sufficiently well. Our aim is to show that $\mathcal{L}$ is complete relative to $\mathcal{T}$. For this purpose it suffices that for every $\mathcal{L}$-predicate P on objects of type σ

(A) there is a $\mathcal{T}$-predicate $\widetilde{P}$ such that $P(M)$ is equivalent to $\widetilde{P}(\alpha_M)$ for all closed PCF terms M of type σ and
(B) $\mathcal{L}$ proves that $P(M) \leftrightarrow \widetilde{P}(\alpha_M)$

where α_M enumerates the codes of compact approximations to (the interpretation of) M. Condition (A) is analogous to the expressivity condition (a) in Cook's original proof since (A) requires that every specification P formulated in the program logic $\mathcal{L}$ can be expressed equivalently by a predicate $\widetilde{P}$ in the "data theory" $\mathcal{T}$. Condition (B) is analogous to condition (b) in Cook's original proof since (B) requires that the program logic is strong enough to prove this equivalence.

2.1 The program logic $\mathcal{L}$

The logic $\mathcal{L}$ is similar to the language of LCF as introduced in [26] in the respect that its base types are the types of PCF. However, terms of type σ will be all PCF^{++}-terms. The only base predicates are the inequality relations $\sqsubseteq_\sigma$ on type σ. Equality on σ can be expressed as $x \sqsubseteq_\sigma y \wedge y \sqsubseteq_\sigma x$. In contrast to the original LCF, our $\mathcal{L}$ will not be based on classical first order but rather on *classical higher order predicate logic.*

The usual axioms of LCF are sufficient for performing most simple verification exercises but do not capture the deeper domain theoretic structure of the Scott model. From our axiomatization one can derive all the axioms of traditional LCF but we also will axiomatize a reasonable part of domain theory à la Scott. This was done also in [22] and was machine checked within HOL (a "synthetic" intuitionistic version has been developed and verified in [23] using LEGO). Unlike those formalizations, however, we will also have to speak about compact elements. In order to do that we do not need to extend the term language but we need to state continuity and similar properties in terms of compact elements.

First of all we postulate that all relations $\sqsubseteq_\sigma$ are partial orders. Furthermore, using higher order logic we can state that all types σ are complete partial orders w.r.t. $\sqsubseteq_\sigma$.

(1) every type σ is a directed complete partial order (dcpo)

Furthermore, we require that

(2) all f of type $\sigma{\to}\tau$ are Scott continuous

The next two axioms characterize the order and suprema in function spaces

(3) for all f, g of type $\sigma{\to}\tau$ we have $f \sqsubseteq_{\sigma\to\tau} g$ iff $\forall x{:}\sigma.\, f(x) \sqsubseteq_\tau g(x)$
(4) for all directed subsets F of $\sigma{\to}\tau$ we have $\forall x{:}\sigma.\ \big(\bigsqcup F\big)(x) = \bigsqcup_{f\in F} f(x)$

where $\bigsqcup F$ is the supremum of F in $\sigma{\to}\tau$ whose existence follows from axiom (1). The following axioms (5–9) are standard:

(5) $\lambda x{:}\sigma.M_1 \sqsubseteq_{\sigma\to\tau} \lambda x{:}\sigma.M_2 \Leftrightarrow \forall x{:}\sigma.\, M_1 \sqsubseteq_\tau M_2$
(6) $(\lambda x{:}\sigma.M)(N) =_\tau M[N/x]$
(7) $\lambda x{:}\sigma.M(x) =_{\sigma\to\tau} M$ provided x is not free in M
(8) $\mathsf{fix}_\sigma(M) =_\sigma M(\mathsf{fix}_\sigma(M))$
(9) $\forall x{:}\sigma.\ M(x) \sqsubseteq_\sigma x \Rightarrow \mathsf{fix}_\sigma(M) \sqsubseteq_\sigma x$ provided x is not free in M

Thus, for $\Omega_\sigma \equiv \mathsf{fix}_\sigma(\lambda x{:}\sigma.x)$ we can show that $\forall x{:}\sigma.\Omega_\sigma \sqsubseteq_\sigma x$. Further, we postulate axiom

(10) forall f of type $\sigma{\to}\sigma$ we have $\mathsf{fix}_\sigma(f) = \bigsqcup_{n\in\omega} f^n(\Omega_\sigma)$.

from which one can derive fixpoint induction and computational induction as usual.

Using the defined predicate $N(x) \equiv x \neq \Omega_{\mathtt{nat}}$ we can state the following axioms about `nat`.

(11) $\neg\,\mathsf{succ}(x) = 0$
(12) $\forall x, y{:}\mathtt{nat}.\ N(x) \wedge N(y) \wedge \mathsf{succ}(x) = \mathsf{succ}(y) \Rightarrow x = y$

(13) $P(0) \wedge \big(\forall x{:}\mathtt{nat}.\, N(x) \wedge P(x) \Rightarrow P(\mathsf{succ}(x))\big) \Rightarrow \forall x{:}\mathtt{nat}.\, N(x) \Rightarrow P(x)$
(14) $N(0)$
(15) $\forall x{:}\mathtt{nat}.\, N(x) \Leftrightarrow N(\mathsf{succ}(x))$
(16) $\mathsf{pred}(0) = 0$
(17) $\forall x{:}\mathtt{nat}.\, \mathsf{pred}(\mathsf{succ}(x)) = x$
(18) $\mathsf{ifz}(\Omega_{\mathtt{nat}}, x, y) = \Omega_{\mathtt{nat}}$
(19) $\mathsf{ifz}(0, x, y) = x$
(20) $\forall z{:}\mathtt{nat}.\, N(z) \Rightarrow \mathsf{ifz}(\mathsf{succ}(z), x, y) = y.$

We have to add axioms for the extra PCF^{++} constants por and $\exists$.

(21) $\forall x, y{:}\mathtt{nat}.\ \mathsf{por}(x, y) = \Omega_{\mathtt{nat}} \vee \mathsf{por}(x, y) = 0 \vee \mathsf{por}(x, y) = 1$
(22) $\forall x, y{:}\mathtt{nat}.\ \mathsf{por}(x, y) = 0 \leftrightarrow (x = 0 \vee y = 0)$
(23) $\forall x, y{:}\mathtt{nat}.\ \mathsf{por}(x, y) = 1 \leftrightarrow (x = 1 \wedge y = 1)$
(24) $\forall f{:}\mathtt{nat}{\to}\mathtt{nat}.\ \exists(f) = \Omega_{\mathtt{nat}} \vee \exists(f) = 0 \vee \exists(f) = 1$
(25) $\forall f{:}\mathtt{nat}{\to}\mathtt{nat}.\ \exists(f) = 0 \leftrightarrow \exists x : \mathtt{nat}.N(x) \wedge f(x) = 0$
(26) $\forall f{:}\mathtt{nat}{\to}\mathtt{nat}.\ \exists(f) = 1 \leftrightarrow f(\Omega) = 1$

Though $\mathcal{L}$ is sufficient for expressing “ordinary” correctness proofs it is not clear how to formalize basic domain theory in this language. For this purpose one has to speak about compact elements. For every type σ one can define in PCF^{++} a strict function $\varepsilon^\sigma : \mathtt{nat} \to \sigma$ enumerating the compact elements of D_σ in such a way that $(D_\sigma, \varepsilon^\sigma)$ is an effectively given domain, see [20, 27]. We often write $x \in \varepsilon^\sigma_n$ instead of $\varepsilon^\sigma_n \sqsubseteq x$. The ε^σ are chosen in such a way that

(27) $x \in \varepsilon^{\mathtt{nat}}_0$ and $\forall x{:}\mathtt{nat}.\ x \in \varepsilon^{\mathtt{nat}}_{n+1} \Leftrightarrow x = n$
(28) $f \in \varepsilon^{\sigma\to\tau}_{[\langle n_1,m_1\rangle,\ldots,\langle n_k,m_k\rangle]} \Leftrightarrow \bigwedge_{i=1}^{k} f(\varepsilon^\sigma_{n_i}) \in \varepsilon^\tau_{m_i}$

where $\langle -, - \rangle$ refers to some primitive recursive coding of pairs and $[n_1, \ldots, n_k]$ is a code for the finite set $\{n_1, \ldots, n_k\}$. In order to relate ε^σ to $\sqsubseteq_\sigma$ we postulate the axiom

(29) $x \sqsubseteq_\sigma y \Leftrightarrow \forall n{:}\mathbb{N}.\ x \in \varepsilon^\sigma_n \Rightarrow y \in \varepsilon^\sigma_n$

Continuity of all functions in $\sigma{\to}\tau$ is expressed by the axiom

(30) $\forall f{:}\sigma{\to}\tau.\forall x{:}\sigma.\forall n{:}\mathbb{N}.\ f(x) \in \varepsilon^\tau_n \Rightarrow \exists m{:}\mathbb{N}.\ x \in \varepsilon^\sigma_m \wedge \forall x{:}\sigma.\ x \in \varepsilon^\sigma_m \Rightarrow f(x) \in \varepsilon^\tau_n$

In presence of higher order logic it is clear that the above axioms are sufficient for deriving the usual theorems of domain theory à la Scott (see e.g. other axiomatizations of domain theory like Holcf [22]). However, these axioms are not irredundant[2] but sufficient for their purpose.

2.2 $\mathcal{T}$ as a sublanguage of $\mathcal{L}$

There is an obvious translation from the language of **EL** into the language of $\mathcal{L}$ whose image we denote by $\mathcal{T}$. The type of natural numbers in **EL** will be interpreted in $\mathcal{L}$ as the subset of $\mathtt{nat}$ as given by the predicate N. The type of sequences in **EL** will be interpreted in $\mathcal{L}$ as the subset B of strict and total elements of $\mathtt{nat}{\to}\mathtt{nat}$. We write $\forall n{:}\mathbb{N}. \cdots$ as an abbreviation for $\forall n{:}\mathtt{nat}.N(n) \Rightarrow \cdots$ and $\forall \alpha{:}\mathbb{N}^{\mathbb{N}}. \cdots$ as an abbreviation for $\forall \alpha{:}\mathtt{nat}{\to}\mathtt{nat}.B(\alpha) \Rightarrow \cdots$.

The basic operations of **EL** are interpreted by their corresponding basic operations in $\mathcal{L}$. The primitive recursor of **EL** is implemented in terms of the fixpoint operator of $\mathcal{L}$.

[2] For instance, axioms (8) and (9) follow from(10).

2.3 Reducing $\mathcal{L}$ to $\mathcal{T}$

From [21] it follows that there are PCF^{++} terms

$$\begin{aligned}&\mathsf{p}_{\to} : (\mathtt{nat}{\to}\mathtt{nat}) \to (\mathtt{nat}{\to}\mathtt{nat}) \to \mathtt{nat}{\to}\mathtt{nat} \qquad \text{and}\\ &\mathsf{e}_{\to} : ((\mathtt{nat}{\to}\mathtt{nat}) \to \mathtt{nat}{\to}\mathtt{nat}) \to \mathtt{nat} \to \mathtt{nat}\end{aligned}$$

such that $\mathsf{p}_{\to} \circ \mathsf{e}_{\to}$ is the identity on $\mathtt{nat}{\to}\mathtt{nat}$ and this is provable in $\mathcal{L}$. We may exhibit $\mathtt{nat}$ as a retract of $\mathtt{nat}{\to}\mathtt{nat}$ by putting $\mathsf{p}_{\mathtt{nat}}(f) = f(0)$ and $\mathsf{e}_{\mathtt{nat}}(x)(y) = x$. For function types $\sigma{\to}\tau$ we define

$$\mathsf{e}_{\sigma\to\tau}(g) = \mathsf{e}_{\to}(\mathsf{e}_\tau \circ g \circ \mathsf{p}_\sigma) \qquad \mathsf{p}_{\sigma\to\tau}(f) = \mathsf{p}_\tau \circ \mathsf{p}_{\to}(f) \circ \mathsf{e}_\sigma$$

exhibiting $\sigma{\to}\tau$ as a retract of $\mathtt{nat}{\to}\mathtt{nat}$ provably in $\mathcal{L}$.[3]

However, in general for a computable f of type $\mathtt{nat}{\to}\mathtt{nat}$ there will not exist a total recursive α with $\mathsf{p}_\sigma(f) = \mathsf{p}_\sigma(\alpha)$. Thus, it seems appropriate to consider in addition a PCF definable map $\mathsf{r} : (\mathtt{nat}{\to}\mathtt{nat}) \to (\mathtt{nat}{\to}\mathtt{nat})$ which turns a sequence of codes of compact elements of $\mathtt{nat}{\to}\mathtt{nat}$ into the supremum of the coded elements in $\mathtt{nat}{\to}\mathtt{nat}$, i.e. $\mathsf{r}(\alpha) = \bigsqcup_n \varepsilon^{\mathtt{nat}\to\mathtt{nat}}_{\alpha(n)}$, provided this supremum exists. Obviously, the restriction of r to $\mathbb{N}^{\mathbb{N}}$ is still surjective on $\mathtt{nat}{\to}\mathtt{nat}$ and, moreover, the corresponding statement $\forall f{:}\mathtt{nat}{\to}\mathtt{nat}.\exists\alpha{:}\mathbb{N}^{\mathbb{N}}.\, f = \mathsf{r}(\alpha)$ is provable in $\mathcal{L}$. Thus, for every type σ one can prove in $\mathcal{L}$ that $\forall x{:}\sigma.\exists\alpha{:}\mathbb{N}^{\mathbb{N}}.\, x = \widetilde{\mathsf{p}}_\sigma(\alpha)$ where $\widetilde{\mathsf{p}}_\sigma = \mathsf{p}_\sigma \circ \mathsf{r}$. Moreover, one can show that $\widetilde{\mathsf{p}}_\sigma$ restricted to $\mathbb{N}^{\mathbb{N}}$ is an *admissible* representation of D_σ in the sense of [2]. This means that for every (computable) continuous map f from a subset R of $\mathbb{N}^{\mathbb{N}}$ to D_σ there exists a (computable) continuous map $\phi : R \to \mathbb{N}^{\mathbb{N}}$ realizing f, i.e. $f = \widetilde{\mathsf{p}}_\sigma \circ \phi$.

Based on these facts one may replace quantifications of the form $Qx{:}\sigma.A(x)$ (where Q stands for $\forall$ or $\exists$) by $Q\alpha{:}\mathbb{N}^{\mathbb{N}}.A(\widetilde{\mathsf{p}}_\sigma(\alpha))$. Formulas of the latter form are not yet in the fragment $\mathcal{T}$ since $\widetilde{\mathsf{p}}_\sigma$ is not a term of $\mathcal{T}$. Thus, we have to replace atomic formulas $\widetilde{\mathsf{p}}_\sigma(\alpha) \sqsubseteq_\sigma \widetilde{\mathsf{p}}_\sigma(\beta)$ by $\mathcal{L}$-provably equivalent formulas in the language of $\mathcal{T}$. For this purpose we first replace $\widetilde{\mathsf{p}}_\sigma(\alpha) \sqsubseteq_\sigma \widetilde{\mathsf{p}}_\sigma(\beta)$ by $\forall n{:}\mathbb{N}.\widetilde{\mathsf{p}}_\sigma(\alpha) \in \varepsilon^\sigma_n \Rightarrow \widetilde{\mathsf{p}}_\sigma(\beta) \in \varepsilon^\sigma_n$. Thus, it suffices to replace formulas of the form $\widetilde{\mathsf{p}}_\sigma(\alpha) \in \varepsilon^\sigma_n$ by $\mathcal{L}$-provably equivalent formulas $R_\sigma(\alpha, n)$ in the language of $\mathcal{T}$. This is achieved by the following lemma.

▶ **Lemma 1.** *For every* PCF *type* σ *there is a* $\mathcal{T}$*-predicate* $R_\sigma(\alpha, n)$ *such that*

$$(\dagger) \qquad R_\sigma(\alpha, n) \Leftrightarrow \widetilde{\mathsf{p}}_\sigma(\alpha) \in \varepsilon^\sigma_n$$

is provable in $\mathcal{L}$.

Proof. For base type $\mathtt{nat}$, we get by definition of $\varepsilon^{\mathtt{nat}}$ and r that

$$\begin{aligned}\widetilde{\mathsf{p}}_{\mathtt{nat}}(\alpha) \in \varepsilon^{\mathtt{nat}}_n &\iff n = 0 \vee \mathsf{p}_{\mathtt{nat}}(\bigsqcup_k \varepsilon^{\mathtt{nat}\to\mathtt{nat}}_{\alpha(k)}) = n{-}1\\ &\iff n = 0 \vee (\bigsqcup_k \varepsilon^{\mathtt{nat}\to\mathtt{nat}}_{\alpha(k)})(0) = n{-}1\\ &\iff n = 0 \vee \exists k{:}\mathtt{nat}.\,(\varepsilon^{\mathtt{nat}\to\mathtt{nat}}_{\alpha(k)})(0) = n{-}1\\ &\iff n = 0 \vee \exists k{:}\mathtt{nat}.\,\langle 1, n\rangle \in \alpha(k)\end{aligned}$$

which is a $\mathcal{T}$-predicate. Therefore, we can set $R_{\mathtt{nat}}(\alpha, n) \equiv n = 0 \vee \exists k{:}\mathtt{nat}.\,\langle 1, n\rangle \in \alpha(k)$. Suppose as induction hypothesis that we have achieved our goal for σ and τ already. Now

[3] In [21] Plotkin shows that all coherently complete countably algebraic cpo's arise as retracts of $\mathtt{nat}{\to}\mathtt{nat}$. But all PCF types get interpreted as coherently complete countably algebraic cpo's in the Scott model.

we can prove in $\mathcal{L}$ that

$$\begin{aligned}
\widetilde{\mathsf{p}}_{\sigma\to\tau}(\alpha) \in \varepsilon^{\sigma\to\tau}_{[\langle n_1,m_1\rangle,\dots,\langle n_k,m_k\rangle]} &\iff \bigwedge_{i=1}^{k} \widetilde{\mathsf{p}}_{\sigma\to\tau}(\alpha)(\varepsilon^{\sigma}_{n_i}) \in \varepsilon^{\tau}_{m_i} \\
&\iff \bigwedge_{i=1}^{k} \mathsf{p}_{\tau}(\mathsf{p}_{\to}(\mathsf{r}(\alpha))(\mathsf{e}_{\sigma}(\varepsilon^{\sigma}_{n_i}))) \in \varepsilon^{\tau}_{m_i} \\
&\iff \bigwedge_{i=1}^{k} \widetilde{\mathsf{p}}_{\tau}(\Phi_{\sigma,\tau}(\alpha,n_i)) \in \varepsilon^{\tau}_{m_i} \\
&\iff \bigwedge_{i=1}^{k} R_{\tau}(\Phi_{\sigma,\tau}(\alpha,n_i),m_i)
\end{aligned}$$

where $\Phi_{\sigma,\tau}(\alpha,n)$ is a term in $\mathcal{T}$ of type $\mathbb{N}^{\mathbb{N}}$ such that $\mathcal{L}$ proves $\mathsf{p}_{\tau}(\mathsf{p}_{\to}(\mathsf{r}(\alpha))(\mathsf{e}_{\sigma}(\varepsilon^{\sigma}_{n}))) = \widetilde{\mathsf{p}}_{\tau}(\Phi_{\sigma,\tau}(\alpha,n))$. The term $\Phi_{\sigma,\tau}$ exists because $\widetilde{\mathsf{p}}_{\tau}$ is an admissible representation of D_{τ} and every code of an r.e. set can effectively be transformed into a code of a primitive recursive function enumerating it.

According to the equivalence established above, we may now define $R_{\sigma\to\tau}$ inductively as

$$R_{\sigma\to\tau}(\alpha,[\langle n_1,m_1\rangle,\dots,\langle n_k,m_k\rangle]) \equiv \bigwedge_{i=1}^{k} R_{\tau}(\Phi_{\sigma,\tau}(\alpha,n_i),m_i)$$

which can be expressed in the language of $\mathcal{T}$ (see [28]). ◀

Now we may associate with the base predicate $\sqsubseteq_{\sigma}$ of $\mathcal{L}$ the $\mathcal{T}$-predicate $\widetilde{\sqsubseteq}_{\sigma}$ defined as

$$\alpha \mathrel{\widetilde{\sqsubseteq}_{\sigma}} \beta \equiv \forall n{:}\mathbb{N}.\ R_{\sigma}(\alpha,n) \Rightarrow R_{\sigma}(\beta,n)$$

and, accordingly, we have

$$\alpha \mathrel{\widetilde{=}_{\sigma}} \beta \equiv \forall n{:}\mathbb{N}.\ R_{\sigma}(\alpha,n) \Leftrightarrow R_{\sigma}(\beta,n)\ .$$

For $\mathcal{L}$-predicates P we define their translation to a $\mathcal{T}$-predicate $\widetilde{P}$ by replacing $\sqsubseteq$ by $\widetilde{\sqsubseteq}$, leaving propositional connectives unchanged and replacing quantification over σ by quantification over $\mathbb{N}^{\mathbb{N}}$.

It remains to explain how one translates $\mathcal{L}$-terms to $\mathcal{T}$. For this purpose notice that Scott domains form a full subcategory of $\mathbf{Mod}(K_2)$, the modest sets in the (function) realizability topos over the second Kleene algebra K_2 as shown e.g. in [2]. As already mentioned above for the domain D_{σ} an admissible representation is provided by the restriction of $\widetilde{\mathsf{p}}_{\sigma}$ to $\mathbb{N}^{\mathbb{N}}$, the underlying set of K_2. Thus, for every PCF^{++} term $x_1{:}\sigma_1,\dots,x_k{:}\sigma_k \vdash t : \tau$ one can find a primitive recursive neighbourhood function α_t in $\mathbb{N}^{\mathbb{N}}$ such that $\mathcal{L}$ proves $\forall\beta_1,\dots,\beta_k{:}\mathbb{N}^{\mathbb{N}}.\,\widetilde{\mathsf{p}}_{\tau}(\alpha_t(\beta_1|\dots|\beta_k)) = t[\widetilde{\mathsf{p}}_{\sigma_1}(\beta_1),\dots,\widetilde{\mathsf{p}}_{\sigma_k}(\beta_k)/x_1,\dots,x_k]$. Accordingly, we may translate the term t to the $\mathcal{T}$-term $\alpha_t(\beta_1|\dots|\beta_k)$ (where juxtaposition denotes application in K_2 and $(\beta_1|\dots|\beta_k)$ is a code for the respective k-tuple in $\mathbb{N}^{\mathbb{N}}$).

Thus, in summary, we have shown our first main result:

▶ **Theorem 2.** *For every sentence A of $\mathcal{L}$ there is a sentence $\widetilde{A}$ in the fragment $\mathcal{T}$ such that $\mathcal{L}$ proves $A \Leftrightarrow \widetilde{A}$. Thus $\mathcal{L}$ is complete relative to $\mathcal{T}$, the set of all true sentences of* **EL**.

3 The Effective Scott Model

For the effective Scott model [20] it should be possible to find an axiomatization $\mathcal{L}_e$ which is complete relative to the set $\mathcal{T}_e$ of all true arithmetic sentences. However, in this model the interpretation of types will not be cpo's anymore because not all directed suprema exist. For this reason we replace axioms (1-4) of subsection 2.1 by the following ones where the terms ε^{σ} are the same as in the respective subsection.

(1) $\forall f, g{:}\sigma{\to}\tau . f \sqsubseteq_{\sigma\to\tau} g \iff \big(\forall x, y : \sigma . f(x) \sqsubseteq g(y)\big)$
(2) for all x in σ the set $\{n : \mathtt{nat} \mid N(n) \wedge n \in \varepsilon^\sigma_x\}$ is r.e.
(3) for every r.e. set A if the subset $\{\varepsilon^\sigma_n \mid n \in A\}$ is directed then it has a supremum in σ
(4) the suprema of (3) are pointwise in case of function types.

The logic $\mathcal{L}_e$ of the effective Scott model is given by these four axioms and the axiom (5-30) of subsection 2.1.

The system $\mathcal{T}_e$ is the set of all true arithmetic sentences formulated in the obvious sublanguage of $\mathcal{L}_e$. We assume that $\mathcal{T}_e$ contains constants for all primitive recursive functions on natural numbers.

For reducing $\mathcal{L}_e$ to $\mathcal{T}_e$ we proceed essentially as in subsection 2.3. The main difference is that instead of the map $\mathsf{r} : (\mathtt{nat}{\to}\mathtt{nat}) \to \mathtt{nat}{\to}\mathtt{nat}$ used there we consider a map $\mathsf{r} : \mathtt{nat} \to (\mathtt{nat}{\to}\mathtt{nat})$ which sends a code of a recursive enumeration of compact elements in type $\mathtt{nat}{\to}\mathtt{nat}$ to its supremum provided the elements given in the enumeration are consistent. As in subsection 2.3 we define $\widetilde{\mathsf{p}}_\sigma$ as $\mathsf{p}_\sigma \circ \mathsf{r} : \mathtt{nat} \to \sigma$ which is easily seen to be an *admissible numbering* of the computable elements of the effectively given domain $(D_\sigma, \varepsilon^\sigma)$ (see last chapter of [27]).

In analogy to Lemma 1 we have

▶ **Lemma 3.** *For every* PCF *type σ there is a $\mathcal{T}_e$-predicate $R_\sigma(\ell, n)$ such that*

$$(\dagger) \qquad R_\sigma(\ell, n) \Leftrightarrow \widetilde{\mathsf{p}}_\sigma(\ell) \in \varepsilon^\sigma_n$$

is provable in $\mathcal{L}_e$.

Proof. The claim is evident for base type $\mathtt{nat}$. Suppose as induction hypothesis that we have achieved our goal for σ and τ already.

Now, completely analogously to the proof of Lemma 1 we can prove in $\mathcal{L}_e$ that

$$\widetilde{\mathsf{p}}_{\sigma\to\tau}(\ell) \in \varepsilon^{\sigma\to\tau}_{[\langle n_1, m_1\rangle, \ldots, \langle n_k, m_k\rangle]} \quad \iff \quad \bigwedge_{i=1}^{k} R_\tau(\Phi_{\sigma,\tau}(\ell, n_i), m_i)$$

where $\Phi_{\sigma,\tau}(\ell, n)$ is a term in $\mathcal{T}_e$ such that $\mathcal{L}_e$ proves $\mathsf{p}_\tau(\mathsf{p}_\to(\mathsf{r}(\ell))(\mathsf{e}_\sigma(\varepsilon^\sigma_n))) = \widetilde{\mathsf{p}}_\tau(\Phi_{\sigma,\tau}(\ell, n))$. The term $\Phi_{\sigma,\tau}$ exists because $\widetilde{\mathsf{p}}_\tau$ is an admissible numbering of the the computable elements of the effectively given domain D_τ.

According to the above, we may define $R_{\sigma\to\tau}$ inductively as

$$R_{\sigma\to\tau}(\ell, [\langle n_1, m_1\rangle, \ldots, \langle n_k, m_k\rangle]) \equiv \bigwedge_{i=1}^{k} R_\tau(\Phi_{\sigma,\tau}(\ell, n_i), m_i)$$

which can be expressed in the language of $\mathcal{T}_e$ (see [28]). ◀

Now in analogy with subsection 2.3 we may associate with the base predicate $\sqsubseteq_\sigma$ of $\mathcal{L}_e$ the $\mathcal{T}_e$-predicate $\widetilde{\sqsubseteq}_\sigma$ defined as

$$n \mathrel{\widetilde{\sqsubseteq}_\sigma} m \equiv \forall k{:}\mathbb{N}.\ R_\sigma(n, k) \Rightarrow R_\tau(m, k)$$

and, accordingly, we have

$$n \mathrel{\widetilde{=}_\sigma} m \equiv \forall k{:}\mathbb{N}.\ R_\sigma(n, k) \Leftrightarrow R_\tau(m, k)$$

For $\mathcal{L}_e$-predicates P we define their translation to a $\mathcal{T}_e$-predicate $\widetilde{P}$ by replacing $\sqsubseteq$ by $\widetilde{\sqsubseteq}$, leaving propositional connectives unchanged and replacing quantification over σ by quantification over $\mathbb{N}$.

As before, it remains to say how one translates $\mathcal{L}_e$-terms to $\mathcal{T}_e$. For this purpose notice that effective Scott domains form a full subcategory of $\mathbf{Mod}(K_1)$, the modest sets in the (number) realizability topos (aka *effective topos*) over the first Kleene algebra K_1 as shown e.g. in [16]. Actually, for the effective domain D_σ an admissible numbering is provided by the restriction of $\widetilde{\mathsf{p}}_\sigma$ to $\mathbb{N}$, the underlying set of K_1. For every PCF^{++} term $x_1{:}\sigma_1, \ldots, x_k{:}\sigma_k \vdash t : \tau$ one can find a primitive recursive function f_t such that $\mathcal{L}_e$ proves $\forall m_1, \ldots, m_k{:}\mathbb{N}.\, \widetilde{\mathsf{p}}_\tau(f_t(m_1, \ldots, m_k)) = t[\widetilde{\mathsf{p}}_{\sigma_1}(m_1), \ldots, \widetilde{\mathsf{p}}_{\sigma_k}(m_k)/x_1, \ldots, x_k]$. Accordingly, we may translate the term t to the $\mathcal{T}_e$-term $f_t(m_1, \ldots, m_k)$.

Thus, in summary, we have our second main result

▶ **Theorem 4.** *For every sentence A of $\mathcal{L}_e$ there is a sentence $\widetilde{A}$ in the fragment $\mathcal{T}_e$ such that $\mathcal{L}_e$ proves $A \Leftrightarrow \widetilde{A}$. Thus $\mathcal{L}_e$ is complete relative to $\mathcal{T}_e$, the set of all true sentences of arithmetic.*

4 Conclusions and Directions for Future Work

We have proved relative completeness of an extension of LCF axiomatising the Scott model with respect to the full theory of Baire space $\mathbb{N}^\mathbb{N}$. Similarly, an extension of the effective Scott model has been shown to be relative complete w.r.t. all true sentences of first order arithmetic. To the best of our knowledge these are original results.

As mentioned in the Introduction, one could now go on and attempt similar relative completeness proofs for axiomatisations of other models of PCF. For the observably sequential algorithms model [4] one can exploit the fact that it admits a universal type `nat`$\to$`nat` as shown in [13] and its axiomatization could follow the ideas on Locally Boolean Domains presented in [10].

For fully abstract models of PCF, unfortunately, the methods of our paper cannot be applied. From Prop. 7.6 of [11] it follows that the fully abstract games model for PCF does not admit a universal type. This, however, does not entail that there does not exist a universal type in the model $\mathcal{F}$ which is obtained from the games model by taking the quotient modulo observational equivalence. The reason is that the quotient may create new retractions in $\mathcal{F}$. But even if there existed a universal PCF type in $\mathcal{F}$ there would still remain the problem that by Loader's result [12] the decisive predicates on compact elements are not effective and thus it would not be obvious how to axiomatise them.

In any case, the model $\mathcal{F}$ is particularly ill-behaved as shown in [18]. Firstly, not all functionals in the model preserve suprema of ω-chains. Secondly, many "finitary" objects are not compact in the sense of domain theory. However, the situation is much more satisfactory when looking at $\mathcal{F}$ from the point of view of "operationally based domain theory" as in [7, 25] where instead of order-theoretic suprema of ascending chains one considers limits of so-called "ω-chains" (cf. [19, 24]). Thus, it may be worthwhile to axiomatize $\mathcal{F}$ despite the problems discussed above. For instance, one could try to formulate the games model of PCF within the fragment **EL** (which allows one to speak about arenas and strategies which can be represented by functions on $\mathbb{N}$) and to consider the logical relation between the games model and $\mathcal{F}$ itself specifying which elements are realized by which strategies. This way one obtains representations of PCF types which, though different from the ones considered in this paper, can still be used for reformulating correctness assertions in terms of **EL**.

Another open problem is to find relatively complete logics for languages with higher order store as in [1]. In [11] a language λ_{co} has been exhibited that is universal for a games model $\mathcal{G}_!$ where $\mathcal{G}$ is the category of affine sequential algorithms on Curien-Lamarche games and ! is the *repetetive* exponential on $\mathcal{G}$. The reason for this universality is that there is a simple universal type $\mathtt{nat}^{\mathtt{nat}} \to \mathtt{nat}^{\mathtt{nat}}$ whose computable elements can all be denoted by λ_{co} terms. Alas, the model $\mathcal{G}_!$ is more restrictive than the games model considered in [1] and thus we do not know whether the latter contains a universal type. At first sight the paper [9] seems to address this problem but the assertion language used there is too strong in the sense that it refers to higher order objects and thus contains already all correctness assertions. The main achievement of *loc.cit.* is rather that the program logic characterises programs up to observational equivalence.

Moreover, program specifications in *loc.cit.* have to be formulated as Hoare triples which cannot be combined by logical connectives and quantifiers. This, however, would be desirable since it allows one to avoid the problems with verification of higher-order local procedures as described in [5].

Acknowledgements This research has been supported by the EPSRC Research Grant "Relative Completeness for Logics of Functional Programs" (EPSRC EP/I01456X/1). We would like to thank Martin Berger for discussions on the completeness results of [9] and the anonymous referees for their suggestions and comments.

References

1 S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In Proc. of the 13th Symp. on Logic in Computer Science, pp. 334–344, IEEE Press, Washington, 1997.

2 I. Battenfeld, M. Schröder, and A. Simpson. A Convenient Category of Domains, Electronic Notes in Theoretical Computer Science, vol. 172, pp. 69–99, 2007.

3 A. Bauer. Realizability as Connection between Constructive and Computable Mathematics. Proc. of CCA 2005 - Second International Conference on Computability and Complexity in Analysis, pp. 378–379, 2005. Long version electronically available from `math.andrej.com/data/c2c.pdf`.

4 R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. Inf. Comput., 111(2):297–401, 1994.

5 E.M. Clarke. Programming Language Constructs for Which it is Impossible to Obtain Good Hoare-like Axiom Systems. Journal of the Association for Computing Machinery, Vol. 26, No. l, pp. 129-147, 1979.

6 S. Cook. Soundness and completeness of an axiom system for program verification. SIAM Journal on Computing 7:70–90, 1978.

7 M. Escardó and W. Kin Ho. Operational domain theory and topology of sequential programming languages, Inf. Comput. 207(3): 411-437, 2009.

8 C.A.R. Hoare. An axiomatic basis for computer programming. Comm. ACM 12:576–583, 1969.

9 K. Honda, N. Yoshida, and M. Berger. An Observationally Complete Program Logic for Imperative Higher Order Functions. In Proc. of the 20th Symp. on Logic in Computer Science, pp. 270–279, IEEE Press, Washington, 2005.

10 J. Laird. Locally boolean domains. Theor. Comput. Sci., 342(1):132–148, 2005.

11 J. Laird. Functional Programs as Coroutines: A Semantic Analysis. Logical Methods in Computer Science, to appear.

12 R. Loader. Finitary PCF is not decidable. Theor. Comput. Sci. 266(1-2): 341-364, 2001.

13 J. Longley. Universal types and what they are good for. In GQ. Zhang, J. Lawson, Y.-M. Liu and M.-K. Luo (editors), Domain theory, logic and computation: Proc. of the 2nd International Symposium on Domain Theory, Semantic Structures in Computation 3, pp. 25-63, Kluwer, 2003.

14 T. Löw. Locally Boolean Domains and Universal Models for Infinitary Sequential Languages. PhD thesis, TU Darmstadt, 2006.

15 T. Löw and Th. Streicher. Universality results for models in locally boolean domains. In Z. Ésik, editor, Proc. of the 20th Int. Workshop Computer Science Logic, volume 4207 of Lecture Notes in Computer Science, pp. 456–470. Springer, 2006.

16 D.C. McCarty. Realizability and Recursive Mathematics. PhD Thesis, Oxford, 1984.

17 R. Milner. Fully Abstract Models of Typed λ-Calculi. Theor. Comput. Sci. 4, pp. 1-22,1977.

18 D. Normann and V.Yu. Sazonov. The extensional ordering of the sequential functionals. preprint, 2010.

19 J. van Oosten and A.K. Simpson. Axioms and (counter) examples in synthetic domain theory. Ann. Pure Appl. Logic, 104(1-3):233–278, 2000.

20 G. Plotkin. LCF considered as a programming language. TCS 5, pp. 223-255, 1977.

21 G. Plotkin. $\mathbb{T}^\omega$ as a universal domain. J. Comput. System Sci. 17, no. 2, pp. 209–236, 1978.

22 F. Regensburger. HOLCF: Higher Order Logic of Computable Functions, Proc. of the 8th International Workshop on Higher Order Logic Theorem Proving and Its Applications, pp. 293–307, vol. 957 of Lecture Notes in Computer Science, Springer, 1995.

23 B. Reus. Formalizing Synthetic Domain Theory. Journal of Automated Reasoning, 23:411–444, 1999.

24 B. Reus and Th. Streicher. General Synthetic Domain Theory – a logical approach. Math. Struct. in Comp. Sci., 9:177–223, 1999.

25 A. Rohr. A Universal Realizability Model for Sequential Computation. PhD Thesis, TU Darmstadt, 2002.

26 D. S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. Unpublished paper from 1969 reprinted in Böhm Festschrift, Theor. Comput. Sci. 121, No. 1-2, pp. 411–440, 1993.

27 Th. Streicher. Domain-theoretic Foundations of Functional Programming. World Scientific, 2006.

28 A. Troelstra and D. van Dalen. Constructivism in Mathematics. Two volumes, North Holland, 1988.