New Foundations for Fixpoint Computations

Roy L. Crole* Andrew M. Pitts†
Computer Laboratory, University of Cambridge CB2 3QG, England

Abstract

This paper introduces a new higher-order typed constructive predicate logic for fixpoint computations, which exploits the categorical semantics of computations introduced by Moggi [8] and contains a strong version of Martin-Löf's 'iteration type' [11]. The type system enforces a separation of computations from values. The logic contains a novel form of fixpoint induction and can express partial and total correctness statements about evaluation of computations to values. The constructive nature of the logic is witnessed by strong metalogical properties which are proved using a category-theoretic version of the 'logical relations' method.

1 Computation types

It is well known that primitive recursion at higher types can be given a categorical characterisation in terms of Lawvere's concept of *natural number object* [6]. We show that a similar characterisation can be given for general recursion via fixpoint operators of higher types, in terms of a new concept—that of a *fixpoint object* in a suitably structured category. This notion was partly inspired by contemplation of Martin-Löf's non-standard 'iteration type' in his domain theoretic interpretation of type theory [11]. However, the key ingredient which allows the formulation of the concept of fixpoint object is the treatment of computations using *monads* introduced by Moggi [8], where there is a distinction between the elements of a type α and computations of elements of that type—the latter being grouped into a new type $T(\alpha)$. Moggi's computational meta-language $\lambda\mathrm{ML}_T$ [10], contains the following formation rules:

$$\frac{\alpha \text{ type}}{T(\alpha) \text{ type}}$$

$$\frac{M \in \alpha}{\mathsf{Val}(M) \in T(\alpha)}$$

$$\frac{E \in T(\alpha) \qquad \begin{array}{c}[x \in \alpha] \\ F(x) \in T(\beta)\end{array}}{\mathsf{Let}\, x \Leftarrow E.F(x) \in T(\beta)}$$

Note These rules, and the others which appear in this paper, are presented in natural deduction style, with discharged hypotheses enclosed in square brackets. Since there are several unfamiliar variable binding operations in the syntax, we will also adopt Martin-Löf's theory of expressions and arities—a $\beta\eta$-lambda calculus over a single ground type of expressions with abstraction denoted $(x)e$, application denoted $f(e)$ and a multiple application such as $(f(e))(e')$ abbreviated to $f(e, e')$; see [12], for example. Finally, it should be noted that our syntax is a slight variant of Moggi's.

Intuitively, $\mathsf{Val}(M)$ is the value M regarded as a trivial computation which immediately evaluates to itself; and $\mathsf{Let}\, x \Leftarrow E.F(x)$ denotes the computation which firstly tries to evaluate E to some value $x \in \alpha$ and then proceeds to evaluate

*Research supported by a SERC Research Studentship.

†Research supported by the CLICS project (EEC ESPRIT BR nr 3003).

$F(x)$. These intended meanings are captured by three equational axioms:

$$\begin{aligned} \mathsf{Let}\, x \Leftarrow \mathsf{Val}(M).F(x) &= F(M) \\ \mathsf{Let}\, x \Leftarrow E.\mathsf{Val}(x) &= E \\ \mathsf{Let}\, y \Leftarrow (\mathsf{Let}\, x \Leftarrow E.F(x)).G(y) &= \\ \mathsf{Let}\, x \Leftarrow E.\mathsf{Let}\, y \Leftarrow F(x).G(y) & \end{aligned}$$

In addition, $\lambda\mathrm{ML}_T$ extends the simply typed lambda calculus: there are function types $\alpha \to \beta$ with lambda abstractions $\lambda x \in \alpha.F(x)$ and applications $(M\,N)$ satisfying the usual β and η equalities. The system also contains product types $\alpha \times \beta$ with (surjective) pairing $\langle M, N\rangle$ and projections $\mathsf{Fst}(M)$, $\mathsf{Snd}(M)$; and it contains a type *unit* with unique element $\langle\rangle \in \mathit{unit}$.

The categorical counterpart of this basic metalanguage is the notion of a 'cartesian closed category equipped with a strong monad T' [8, section 2]. We shall refer to such structures as *let cartesian closed categories*, or just *let-ccc's*. Such categories can be presented very simply using a language of categorical combinators for $\lambda\mathrm{ML}_T$, extending Curien's ccc combinators for the simply typed lambda calculus [2].

Definition 1.1 The *let-categorical combinators* are defined by the following grammar:

$$\begin{aligned} F \quad ::= \quad & \mathsf{Id} \mid F; F \mid \langle\rangle \mid \langle F, F\rangle \mid \mathsf{Fst} \mid \mathsf{Snd} \\ & \mid \mathsf{Cur}(F) \mid \mathsf{App} \mid \eta \mid \mathsf{Lift}(F). \end{aligned}$$

A typing statement for these combinators takes the form $F : \alpha \to \beta$; an equality statement takes the form $F = F' : \alpha \to \beta$. The rules for deducing these statements are those for ccc's (see [6]) augmented by the following rules (where $F \times G$ abbreviates $\langle \mathsf{Fst}; F, \mathsf{Snd}; G\rangle$):

- $\eta : \alpha \to T(\alpha)$.
- If $F : \alpha \times \beta \to T(\gamma)$, then $\mathsf{Lift}(F) : \alpha \times T(\beta) \to T(\gamma)$ (and $\mathsf{Lift}(F) = \mathsf{Lift}(F')$ if $F = F'$).
- If $F : \alpha \to \beta$ and $G : \beta \times \gamma \to T(\delta)$, then $(F \times \mathsf{Id}); \mathsf{Lift}(G) = \mathsf{Lift}((F \times \mathsf{Id}); G) : \alpha \times T(\gamma) \to T(\delta)$.
- If $F : \alpha \times \beta \to T(\gamma)$, then $(\mathsf{Id} \times \eta); \mathsf{Lift}(F) = F : \alpha \times \beta \to T(\gamma)$.
- $\mathsf{Lift}(\mathsf{Snd}; \eta) = \mathsf{Snd} : \alpha \times T(\beta) \to T(\beta)$.
- If $F : \alpha \times \beta \to T(\gamma)$ and $G : \alpha \times \gamma \to T(\delta)$, then $\mathsf{Lift}(\langle \mathsf{Fst}, F\rangle; \mathsf{Lift}(G)) = \langle \mathsf{Fst}, \mathsf{Lift}(F)\rangle; \mathsf{Lift}(G) : \alpha \times T(\beta) \to T(\delta)$.

Proposition 1.1 *There are mutually inverse translations between the above combinators, $F : \alpha \to \beta$, and $\lambda\mathrm{ML}_T$-terms, G, satisfying $G(x) \in \beta\ [x \in \alpha]$.*

Combining this proposition with the fact that equational $\lambda\mathrm{ML}_T$ theories correspond to let-ccc's [10], one concludes that the categorical combinators give a simple presentation of this variety of category; in particular, the action of T on morphisms, the monad multiplication and the monad strength are all definable from the combinators. In terms of the $\lambda\mathrm{ML}_T$ language, the action of T on a morphism $F : \alpha \to \beta$ yields

$$\begin{aligned} T(F) \quad \stackrel{\mathrm{def}}{=} \quad & \lambda e \in T(\alpha).\mathsf{Let}\, x \Leftarrow e.\mathsf{Val}(F\,x) \\ & : T(\alpha) \to T(\beta). \end{aligned}$$

Similarly, the monad multiplication is defined by

$$\begin{aligned} \mu_\alpha \quad \stackrel{\mathrm{def}}{=} \quad & \lambda e' \in T(T\alpha).\mathsf{Let}\, e \Leftarrow e'.e \\ & : T^2(\alpha) \to T(\alpha) \end{aligned}$$

and the monad strength is defined by

$$\begin{aligned} t_{\alpha,\beta} \stackrel{\mathrm{def}}{=} & \\ & \lambda z \in \alpha \times T\beta.\mathsf{Let}\, y \Leftarrow \mathsf{Snd}(z).\mathsf{Val}(\langle \mathsf{Fst}(z), y\rangle) \\ & : \alpha \times T(\beta) \to T(\alpha \times \beta). \end{aligned}$$

A concept which is intimately bound up with the correspondence between the metalanguage $\lambda\mathrm{ML}_T$ and let-ccc's is *functional completeness*, as used by Lambek and Scott [6] in the context of ordinary cartesian closed categories. Here one must consider polynomial categories over let-ccc's and show that they are functionally complete in a suitable sense.

Definition 1.2 Let $\mathcal{C}$ be a let-ccc and let α be an object of $\mathcal{C}$. The *polynomial category* $\mathcal{C}[X]$ is specified by the following data:

- The objects of $\mathcal{C}[X]$ are just the objects of $\mathcal{C}$.

- The morphisms of $\mathcal{C}[X]$ are equivalence classes of the let-categorical combinators generated by the morphisms of $\mathcal{C}$, together with an indeterminate global element $X : \mathit{unit} \to \alpha$. The equivalence relation is that generated by the equality rules in definition 1.1 and the existing identities in $\mathcal{C}$—so that there is a canonical inclusion of let-categories $i : \mathcal{C} \hookrightarrow \mathcal{C}[X]$.

Theorem 1.1 (Functional Completeness) *Let $\mathcal{C}[X]$ be as above. For every morphism $U : \beta \to \gamma$ in $\mathcal{C}[X]$ there is a unique morphism $\Lambda(U) : \alpha \times \beta \to \gamma$ in $\mathcal{C}$ such that $\langle(\langle\rangle; X), \mathsf{Id}\rangle; \Lambda(U) = U$. (In particular, each global element of γ in $\mathcal{C}[X]$ is of the form $X; F$ for some unique morphism $F : \alpha \to \gamma$ in $\mathcal{C}$.)*

It is possible to prove the functional completeness result simply by calculation with the categorical combinators. Then one may derive the correspondence between the metalanguage and the category theory using the recipe described for cartesian closed categories by Lambek and Scott [6]. We make the comment that when specifying the metalanguage terms generated by a given category, terms $\mathsf{Let}\, x \Leftarrow E.U$ in the metalanguage are instances of morphisms $\langle \mathsf{Id}, E\rangle; \mathsf{Lift}(\langle \mathsf{Snd}, \mathsf{Fst}\rangle; \Lambda(U))$. When constructing the category from the metalanguage, all morphisms in the category are formed in the way one would expect.

2 The fixpoint type

Definition 2.1 In a let-ccc, a *fixpoint object* is specified by the following data:

- An initial algebra $\sigma : T(\mathit{fix}) \to \mathit{fix}$ for the functor $T(\_)$. Thus for any $F : T(\alpha) \to \alpha$ there is a unique $\mathsf{It}_\alpha(F) : \mathit{fix} \to \alpha$ with $(\sigma; \mathsf{It}_\alpha(F)) = (T(\mathsf{It}_\alpha(F)); F) : T(\mathit{fix}) \to \alpha$.
- A global element $\omega : \mathit{unit} \to T(\mathit{fix})$ which is the equalizer of $\sigma; \eta$ and the identity on $T(\mathit{fix})$. In other words ω is the unique fixed point of $\sigma; \eta : T(\mathit{fix}) \to T(\mathit{fix})$: for any $F : \alpha \to T(\mathit{fix})$, $F = F; (\sigma; \eta)$ if and only if $F = \langle\rangle; \omega$.

The usual category-theoretic considerations imply that the structure $\mathit{fix}, \sigma, \omega$ is determined uniquely up to isomorphism, within the given let-ccc, by the above properties. One should also note that σ, being the structure morphism for the intial algebra of an endofunctor, is itself an isomorphism. (This fact will be used later in Proposition 3.1.) Using the relation between syntax and category theory discussed in section 1, one can translate the definition of a fixpoint object into a corresponding extension of the metalanguage $\lambda\mathrm{ML}_T$. This entails adding a new type *fix*, together with the term-forming and equality rules shown in Figure 1. The last rule in this figure, which expresses the uniqueness of $\mathsf{It}_\alpha(F)$, will be redundant in the full FIX logic of section 3: it is derivable from the induction rule for *fix* introduced in that section.

Fixpoint objects are so called because they enable one to define fixpoint combinators at all types of the form $\alpha \to T(\beta)$.[1]

Proposition 2.1 (Definability of fixpoint combinators) *In the presence of a fixpoint object, one may define a combinator*

$$\mathsf{Y}_{\alpha,\beta} \in ((\alpha \to T\beta) \to \alpha \to T\beta) \to \alpha \to T\beta$$

which satisfies $(\mathsf{Y}_{\alpha,\beta}\ F) = (F\ (\mathsf{Y}_{\alpha,\beta}\ F))$ for all $F \in (\alpha \to T\beta) \to \alpha \to T\beta$. Indeed, defining

$$\mathsf{Y}_{\alpha,\beta} = (F)\mathsf{It}_{\alpha \to T\beta}(\hat{F}, \sigma(\omega))$$

where $\hat{F}$ is $(e)(\lambda x \in \alpha.\mathsf{Let}\, f \Leftarrow e.F(x, f))$, then

$$\frac{[f \in \alpha \to T\beta, x \in \alpha] \quad F(f, x) \in T\beta}{\mathsf{Y}_{\alpha,\beta}(F) \in \alpha \to T\beta}$$

and

$$\frac{[f \in \alpha \to T\beta, x \in \alpha] \quad F(f, x) \in T\beta \qquad M \in \alpha}{(\mathsf{Y}_{\alpha,\beta}(F)\ M) = F(\mathsf{Y}_{\alpha,\beta}(F), M)}$$

are derived rules.

[1] In Moggi's computational semantics, a program of type β with a parameter of type α has a term of type $\alpha \to T(\beta)$ for its denotation; thus to interpret recursively defined programs, fixpoint combinators are only needed at such types.

$$\frac{}{\omega \in T(\mathit{fix})} \qquad \frac{E \in T(\mathit{fix})}{\sigma(E) \in \mathit{fix}} \qquad \frac{\begin{array}{c}[x \in T(\alpha)] \\ F(x) \in \alpha \quad N \in \mathit{fix}\end{array}}{\mathsf{It}_\alpha(F, N) \in \alpha}$$

$$\frac{}{\omega = \mathsf{Val}(\sigma(\omega))} \qquad \frac{E = \mathsf{Val}(\sigma(E))}{E = \omega} \qquad \frac{\begin{array}{c}[x \in T(\alpha)] \\ F(x) \in \alpha \quad E \in T(\mathit{fix})\end{array}}{\mathsf{It}_\alpha(F, \sigma(E)) = F(\mathsf{Let}\, n \Leftarrow E.\mathsf{Val}(\mathsf{It}_\alpha(F, n)))}$$

$$\frac{\begin{array}{ccc}[x \in T(\alpha)] & [n \in \mathit{fix}] & [e \in T(\mathit{fix})] \\ F(x) \in \alpha & G(n) \in \alpha & G(\sigma(e)) = F(\mathsf{Let}\, n \Leftarrow e.\mathsf{Val}(G(n))) \quad N \in \mathit{fix}\end{array}}{G(N) = \mathsf{It}_\alpha(F, N)}$$

Figure 1: Rules for *fix*.

In order to extend the correspondence between category theory and metalanguage, discussed in the previous section, to include fixpoint objects and types, one needs the following result about fixpoint objects in polynomial categories.

Proposition 2.2 *Let $\mathcal{C}$ be a let-ccc with a fixpoint object. Then $\mathcal{C}[X]$ also has a fixpoint object, given by $i(\mathit{fix})$, where $i : \mathcal{C} \hookrightarrow \mathcal{C}[X]$ is the canonical inclusion functor.*

Proof The main idea for the proof is sketched. Suppose that $F : T(\beta) \to \beta$ is a morphism in the polynomial category. By the functional completeness result, the theorem will be true if we can find a unique morphism $\mathsf{It}(F)$, which satisfies $\Lambda(\sigma; \mathsf{It}(F)) = \Lambda(\mathsf{Lift}(\mathsf{It}(F); \eta); F)$, which on expansion of the definitions yields $(\mathsf{Id} \times \sigma); \Lambda(\mathsf{It}(F)) = \langle \mathsf{Fst}, J; \mathsf{Lift}(I; \Lambda(\mathsf{It}(F)); \eta)\rangle; \Lambda(F)$, where I and J are certain canonical isomorphisms. This equality holds in $\mathcal{C}$; using cartesian closedness and the fixpoint object, we obtain the result.

□

Adding a fixpoint type, coproduct types $\alpha + \beta$ and a natural number type *nat* to the metalanguage $\lambda\mathrm{ML}_T$, we arrive at a system which extends Gödel's system T [4, Chapter 7] but which also admits sound translations of Plotkin's PCF [14] (with either a call-by-value or a call-by-name operational semantics [9, section 5]). A domain-theoretic model of this system is provided by the following category.

Definition 2.2 Let $\mathcal{C}po$ denote the ccc whose objects are posets possessing joins of countably infinite chains, and whose morphisms are Scott-continuous functions, i.e. monotonic functions preserving joins of countably infinite chains. The objects of $\mathcal{C}po$ are not required to possess a least element; we will refer to them as *bottomless cpos* in this paper.

The operation of adjoining a least element to $D \in \mathcal{C}po$ to give the lifted cpo $D_\perp = \{[d] \mid d \in D\} \cup \{\perp\}$ gives a strong monad on $\mathcal{C}po$. In this case $\mathcal{C}po$ possesses a fixpoint object, namely the cpo

$$\Omega = \{0 \sqsubset 1 \sqsubset \ldots \sqsubset \top\},$$

with $\sigma : \Omega_\perp \to \Omega$ the continuous function sending $\perp$ to 0, $[n]$ to $n + 1$ and $[\top]$ to $\top$; and with $\omega = [\top] \in \Omega_\perp$. In this case, the combinators $\mathsf{Y}_{\alpha,\beta}$ defined above are indeed the usual least fixed point operators.

3 The FIX logical system

The definition of an initial algebra $\sigma : T(\Omega) \to \Omega$ for a functor $T(\_)$ contains both an existence and

a uniqueness part. The uniqueness part amounts to a form of induction principle, namely:

Initial T-algebra Induction Principle. (Plotkin, Lehmann-Smyth [7, section 5.2].) To show that a subobject $i : S \hookrightarrow \Omega$ is the whole of Ω, it suffices to show that the composition $T(i);\sigma : T(S) \to \Omega$ factors through $i : S \hookrightarrow \Omega$.

When the functor is $(\_) + 1$ on the category of sets, the initial algebra is the natural numbers and the above statement is equivalent to the usual principle of mathematical induction. What about when the functor is lifting on $\mathcal{C}po$? Restricting attention to subobjects of domains which are specified by *inclusive* subsets (those subsets of a cpo which are closed under taking joins of countable chains), we can use the fact that whenever $i : S \hookrightarrow \Omega$ is an inclusive subset of the cpo $\Omega = \{0 \sqsubset 1 \sqsubset \ldots \sqsubset \top\}$, then $(i)_\perp : S_\perp \to \Omega_\perp$ is just the inclusive subset of $\Omega_\perp$ given by $\{e \in \Omega_\perp \mid \forall d \in \Omega.[d] = e \supset d \in S\}$. Then the initiality property of Ω yields the following form of the induction principle, with $S \subseteq \Omega$ inclusive:

$$\frac{\forall e \in \Omega_\perp.(\forall d \in \Omega.[d] = e \supset d \in S) \supset \sigma(e) \in S}{\forall d \in \Omega.d \in S}$$

Just as least fixed points are definable using the universal property of the initial $(\_)_\perp$-algebra Ω, so is Scott's induction principle for least fixed points [16] derivable from the above rule.

In order to formulate this induction principle for a fixpoint object within the metalanguage, we introduce a constructive logic, called FIX, of properties of terms over the metalanguage. Thus there are strong connections between FIX and the traditional 'axiomatic domain theory' of LCF [13] and to Plotkin's approach to denotational semantics using partial continuous functions [15]. However, our logic is inherently more constructive, since it is based on the notion of *evaluation* of a (possibly non-terminating) computation to a value, rather than on non-termination and on information ordering between (possibly partial) computations.

Definition of the FIX logic: The FIX propositions form a fragment of first-order intuitionistic predicate calculus [3] with equality, conjunction and universal quantification (over elements of a given type), together with the following predicate constructors which implicitly contain forms of implication, disjunction and existential quantification.[2]

- Given a proposition $\Phi(x)$ about $x \in \alpha$ and a term $E \in T(\alpha)$, there is a proposition $\forall x \Leftarrow E.\Phi(x)$ whose intended meaning is '$\forall x \in \alpha.(\mathsf{Val}(x) = E \supset \Phi(x))$', together with natural deduction rules which capture this intended meaning.

- Given a proposition $\Phi(x)$ about $x \in \alpha$ and a term $E \in T(\alpha)$, there is a proposition $\exists x \Leftarrow E.\Phi(x)$ whose intended meaning is '$\exists x \in \alpha.(\mathsf{Val}(x) = E \;\&\; \Phi(x))$', together with natural deduction rules which capture this intended meaning.

- Given propositions $\Phi(x)$ and $\Psi(y)$ about $x \in \alpha$ and $y \in \beta$, and a term $E \in \alpha + \beta$, there is a proposition $(\Phi + \Psi)(E)$ whose intended meaning is '$\exists x \in \alpha.(\mathsf{Inl}(x) = E \;\&\; \Phi(x)) \vee \exists y \in \beta.(\mathsf{Inr}(y) = E \;\&\; \Psi(y))$', together with natural deduction rules which capture this intended meaning. (Inl and Inr are the coproduct insertions.)

There are two further rules for the 'bounded' quantifiers which make them (first-order) examples of Moggi's concept of a *T-modal operator* [10, Definition 4.8]:

$$\frac{\Phi(M)}{\forall x \Leftarrow \mathsf{Val}(M).\Phi(x)}$$

and

$$\frac{\forall x \Leftarrow E.\forall y \Leftarrow F(x).\Psi(y)}{\forall y \Leftarrow (\mathsf{Let}\, x \Leftarrow E.F(x)).\Psi(y)}.$$

In fact, modulo the other rules, the first of the above rules is equivalent to imposing Moggi's 'mono condition' on the monad, i.e.

$$\frac{\mathsf{Val}(M) = \mathsf{Val}(M')}{M = M'},$$

[2] The necessity of restricting implication, disjunction and existential quantification is discussed in Remark 3.1.

$$\frac{\Phi(\mathsf{0}) \qquad [n \in \mathit{nat}, \Phi(n)] \quad \Phi(\mathsf{Suc}(n)) \qquad N \in \mathit{nat}}{\Phi(N)} \qquad \frac{[e \in T(\mathit{fix}), \forall n \Leftarrow e.\Phi(n)] \quad \Phi(\sigma(e)) \qquad N \in \mathit{fix}}{\Phi(N)}$$

Figure 2: Rules for *nat* and *fix* induction.

and the second rule is equivalent to

$$\frac{(\mathsf{Let}\, x \Leftarrow E.F(x)) = \mathsf{Val}(N)}{\exists x \Leftarrow E.(F(x) = \mathsf{Val}(N))}.$$

The definition of the FIX logic is completed by induction rules for the natural number type *nat* and for the fixpoint type *fix*, as shown in Figure 2.

Theorem 3.1 (Consistency)
Bottomless cpos, Scott-continuous functions and inclusive predicates form a model of the FIX *logical system.*

Remark 3.1 The induction rule for *nat* is just the usual principle of mathematical induction. The induction rule for *fix* can be rendered informally as: *to prove that a property* $\Phi(n)$ *holds of all elements* n *in fix, it is sufficient to prove for all computations* e *of an element of fix that* $\Phi(\sigma(e))$ *holds if whenever* e *evaluates to a value, that value satisfies* Φ. This principle is consistent (by Theorem 3.1), but only because the FIX propositions have limited forms. In fact, *extending the* FIX *logic with unrestricted intuitionistic negation, implication or existential quantification renders it inconsistent.* A proof of this can be obtained by mimicking within our framework the proof that inclusive subsets of cpos are not in general closed under these operations. Here is the proof for the case of intuitionistic implication (and negation):

Proposition 3.1 *Extending the* FIX *logic with intuitionistic implication renders the system inconsistent.*

Proof Since FIX contains falsity (false), adding implication ($\Phi \supset \Psi$) means that one also has negation ($\neg\Phi \equiv (\Phi \supset \mathsf{false})$). So consider the proposition $\Phi(n) \equiv \neg(\sigma(\omega) = n)$ about $n \in \mathit{fix}$. (In the cpo model, the denotation of this proposition would have to be the largest inclusive subset of Ω not containing $\top$—but no such subset exists.)

Now this $\Phi(n)$ satisfies the hypotheses of the induction principle for *fix* in Figure 2. For if $\forall n \Leftarrow e.\neg(\sigma(\omega) = n)$ holds then $\neg(\omega = e)$, since otherwise we could deduce $\forall n \Leftarrow \omega.\neg(\sigma(\omega) = n)$, which is false because $\mathsf{Val}(n) = \omega$ holds for $n = \sigma(\omega)$. However, as was noted after Definition 2.1, σ is provably a bijection: so from $\neg(\omega = e)$ we deduce $\neg(\sigma(\omega) = \sigma(e))$, i.e. $\Phi(\sigma(e))$, as required.

So the induction principle for *fix* entails that $\Phi(n)$ holds of all $n \in \mathit{fix}$, and in particular of $\sigma(\omega)$, which is a contradiction.

□

The above proof gives weight to the feeling that the FIX logic resembles a calculus of 'formally inclusive' predicates. We next state metatheorems about our logic of fixpoint computations which witness its constructive nature and suggest its potential as a programming logic.

Theorem 3.2 ('Existence Property') *If* E *is a closed term of type* $T(\alpha)$*, then* $\exists x \Leftarrow E.\Phi(x)$ *is provable in* FIX *if and only if there is a closed term* M *of type* α *for which* $\Phi(M)$ *and* $E = \mathsf{Val}(M)$ *are provable. (In other words, a formal proof that* E *evaluates to a value satisfying* Φ *necessitates the existence of a term denoting that value.)*

Theorem 3.3 ('Disjunction Property') *If* E *is a closed term of coproduct type* $\alpha + \beta$*,* Φ *and* Ψ *are properties of* α *and* β *and* $(\Phi + \Psi)(E)$ *is provable in* FIX*, then either* $E = \mathsf{Inl}(M)$ *and* $\Phi(M)$ *are provable for some closed term* M *of*

type α, or $E = \mathsf{Inr}(N)$ and $\Psi(N)$ are provable for some closed term N of type β.

The Existence Property enables one to produce closed terms of type *nat* from a computation of a number (i.e. a closed term of type $T(nat)$) together with a proof that the computation converges. There remains the possibility that a closed term of type *nat* is not a value, i.e. a standard numeral. In other words, the strong universal property of *fix* looks as though it might create 'non-standard' natural numbers and hence mix the 'total' world of primitive recursive functions with the 'partial' world of unrestricted fixpoint computations at T-types. However, this is not so:

Theorem 3.4 (Standardness of *nat*) *Every closed term of type nat in the logic* FIX *is provably equal to a standard numeral* $\mathsf{Suc}^n(\mathsf{O})$.

The method of proving these theorems is described in the next section.

4 Glueing and Logical Relations

Before further discussion of the main issues of this section, we shall describe briefly two results which form a bridge between previous work and new ideas involving logical relations.

Lemma 4.1 *Let $\Gamma : \mathcal{D} \to \mathcal{C}$ be a functor preserving finite products from a let-ccc to a ccc with finite limits. Then the arrow category $(\mathcal{C} \downarrow \Gamma)$, which is called the* glued *category and denoted $\mathcal{G}l(\Gamma)$, is also a let-ccc. Additionally, the second projection functor $P_2 : \mathcal{G}l(\Gamma) \to \mathcal{D}$ preserves the let-ccc structure.*

The objects of $\mathcal{G}l(\Gamma)$ are morphisms in $\mathcal{C}$ of the form $F : \alpha \to \Gamma(\delta)$; the strong monad structure on $\mathcal{G}l(\Gamma)$ is given on objects by sending $F : \alpha \to \Gamma(\delta)$ to $F;\Gamma(\eta) : \alpha \to T(\delta)$. Indeed, the strong monad acts on the 'second object coordinate' throughout the proof; details of the calculation are omitted. Using this lemma and a suitable version of Freyd's glueing argument (see [6, p 250] and [5]), one obtains the following corollary:

Corollary 4.1 *Let $\mathcal{C}$ be a category, and* F$\mathcal{C}$ *the freely generated let-ccc. Then the canonical functor $\mathcal{C} \to$ F$\mathcal{C}$ is full and faithful. Thus a closed term of ground type in the $\lambda\mathrm{ML}_T$ metalanguage (over some ground signature) always converts to a ground term.*

Our original aim was to consider proofs of Theorems 3.2, 3.3 and 3.4 in the manner that, for example, glueing may be used to prove disjunction and explicit definability for intuitionistic predicate logic. It soon became clear that such a naïve approach will not work in the presence of a fixpoint type, and so the following method was developed. We exploit the fact that FIX theories correspond to the following categorical structure:

- A cartesian closed category $\mathcal{C}$, which has finite coproducts and a natural number object. (The objects and morphisms of $\mathcal{C}$ are used to model the FIX types and terms.)
- A strong monad T on $\mathcal{C}$ whose unit components $\eta_\alpha : \alpha \to T(\alpha)$ are all monomorphisms, and for which a fixpoint object exists.
- A $\mathcal{C}$-indexed poset, $\mathcal{P}_\mathcal{C}$, which is used to model the FIX propositions, and hence which is closed under a certain number of completeness and adjointness conditions corresponding to the FIX logical rules for the propositional connectives and quantifier forms. (For lack of space, these conditions will not be given here; however, they are fairly standard ones from categorical logic, adapted to the particular forms of proposition occurring in the FIX logic.)

Call such a structure a FIX-*hyperdoctrine*; a *morphism* of FIX-hyperdoctrines is specified by a functor and an indexed monotone function, preserving the structure mentioned above. The pure FIX logic (regarded as the empty theory, with no extra-logical axioms) corresponds to such a FIX-hyperdoctrine $(\mathcal{F}, \mathcal{P}_\mathcal{F})$, which is initial amongst all such. The category $\mathcal{C}po$ together with the $\mathcal{C}po$-indexed poset assigning to each cpo its poset of inclusive subsets, forms a FIX-hyperdoctrine $(\mathcal{C}po, \mathcal{P}_\mathcal{I})$. Hence there is

a (unique) FIX-hyperdoctrine morphism $[\![\_]\!] : (\mathcal{F}, \mathcal{P}_{\mathcal{F}}) \to (\mathcal{C}po, \mathcal{P}_{\mathcal{I}})$, which assigns cpo meanings to the FIX types, cpo morphisms to the FIX terms and inclusive predicate meanings to the FIX propositions.

Let $\Gamma : \mathcal{F} \to \mathcal{C}po$ denote the functor which assigns to each object $\alpha \in \mathcal{F}$ its set $\Gamma(\alpha)$ of global elements equipped with the *discrete* partial order. We construct a new category $\mathcal{L}r(\Gamma)$, by the following 'logical relations' construction:

- An object of $\mathcal{L}r(\Gamma)$ is a triple $(D, \trianglelefteq, \alpha)$, where $D \in \mathcal{C}po$, $\alpha \in \mathcal{F}$ and $\trianglelefteq$ is an inclusive subset of $D \times \Gamma(\alpha)$.

- A morphism $(D, \trianglelefteq, \alpha) \to (D', \trianglelefteq', \alpha')$ in $\mathcal{L}r(\Gamma)$ is a pair (ϕ, F), where $\phi : D \to D'$ in $\mathcal{C}po$, $F : \alpha \to \alpha'$ in $\mathcal{F}$, satisfying the following condition:

  $$\forall d \in D, M \in \Gamma\alpha.(d \trianglelefteq M \supset \phi(d) \trianglelefteq' (M; F)).$$

There is a strong monad on $\mathcal{L}r(\Gamma)$ which sends an object $(D, \trianglelefteq, \alpha)$ to the object $(D_\perp, \trianglelefteq', T\alpha)$, where for all $e \in D_\perp$ and $E \in \Gamma(T\alpha)$, $e \trianglelefteq' E$ if and only if $\forall d \in D.[d] = e \supset \exists M \in \Gamma(\alpha).(d \trianglelefteq M \ \& \ M; \eta_\alpha = E)$.The rest of the strong monad structure is specified 'componentwise' from $\mathcal{F}$ and $\mathcal{C}po$. There is also a $\mathcal{L}r(\Gamma)$-indexed poset whose fibre over an object $(D, \trianglelefteq, \alpha)$ consists of triples $(S, \trianglelefteq', \Phi)$, with S an inclusive subset of D, $\Phi \in \mathcal{P}_{\mathcal{F}}(\alpha)$ and $\trianglelefteq'$ an inclusive subset of $S \times \{M \in \Gamma(\alpha) \mid \Phi(M) \text{ is provable}\}$. These triples are ordered componentwise.

The above construction produces a new FIX-hyperdoctrine $(\mathcal{L}r(\Gamma), \mathcal{P}_{\mathcal{L}r(\Gamma)})$. It is easy to see that there are FIX-hyperdoctrine morphisms $P_1 : (\mathcal{L}r(\Gamma), \mathcal{P}_{\mathcal{L}r(\Gamma)}) \to (\mathcal{C}po, \mathcal{P}_{\mathcal{I}})$ and $P_2 : (\mathcal{L}r(\Gamma), \mathcal{P}_{\mathcal{L}r(\Gamma)}) \to (\mathcal{F}, \mathcal{P}_{\mathcal{F}})$ given by first and second projections. Then the initiality of $(\mathcal{F}, \mathcal{P}_{\mathcal{F}})$ implies there is a FIX-hyperdoctrine morphism $I : (\mathcal{F}, \mathcal{P}_{\mathcal{F}}) \to (\mathcal{L}r(\Gamma), \mathcal{P}_{\mathcal{L}r(\Gamma)})$ whose composition with P_2 is the identity and whose composition with P_1 is the morphism $[\![\_]\!] : (\mathcal{F}, \mathcal{P}_{\mathcal{F}}) \to (\mathcal{C}po, \mathcal{P}_{\mathcal{I}})$ mentioned above. Theorems 3.2, 3.3 and 3.4 follow by examining the particular form of the $\trianglelefteq$ relations obtained when I is applied to the natural number object, to coproducts and to predicates formed using the bounded existential quantifier.

Note that the functor Γ preserves almost none of the structure of a FIX-hyperdoctrine and yet magically $(\mathcal{L}r(\Gamma), \mathcal{P}_{\mathcal{L}r(\Gamma)})$ is a FIX-hyperdoctrine, allowing us to exploit the initiality property of $(\mathcal{F}, \mathcal{P}_{\mathcal{F}})$. This is similar magic to Freyd's categorical glueing argument. Our category-theoretic method is related to logical relations in the same way that Freyd's glueing construction is related to realizability.

5 Concluding remarks and further directions

(i) The Existence Property expresses a *formal adequacy* of the FIX logic for the metalanguage. A corollary of the above proof is the *model-theoretic adequacy* of $\mathcal{C}po$ for the metalanguage: given a closed term E of type $T(\alpha)$, it is provably equal to a value $\mathsf{Val}(M)$ (M a closed term of type α) if and only if its $\mathcal{C}po$ interpretation $[\![E]\!] \in [\![T(\alpha)]\!] = [\![\alpha]\!]_\perp$ is not $\perp$.

(ii) The predicate $x \Leftarrow e \ [x \in \alpha, e \in T(\alpha)]$ of *evaluation* is implicit in FIX, but is treated in a very 'extensional' way as equivalent to $\mathsf{Val}(x) = e$. It is possible to envisage a weaker logic than FIX (and a corresponding kind of categorical structure) in which $x \Leftarrow e \ [x \in \alpha, e \in T(\alpha)]$ is an atomic predicate satisfying

$$\frac{}{M \Leftarrow \mathsf{Val}(M)} \qquad \frac{M \Leftarrow E \quad N \Leftarrow F(M)}{N \Leftarrow (\mathsf{Let}\, x \Leftarrow E.F(x))}$$

and in which there are modified rules for the bounded quantifiers.

(iii) FIX is not an 'integrated' logic—proofs of propositions are external to the system. Undoubtedly something to aim for is a system combining features of FIX with those of the Calculus of Constructions [1], obtaining both the 'terms-as-computations' and 'terms-as-proofs' paradigms in a single (consistent!) system.

(iv) FIX establishes a novel approach to fixed point equations at the level of functions. We plan to investigate whether this approach extends to the practically important level (for the semantics of programming languages) of fixed point equations for *types*.

References

[1] T.Coquand and G.Huet, *The Calculus of Constructions*, Information and Computation 76 (1988) 95–120.

[2] P.-L.Curien, *Categorical Combinators, Sequential Algorithms and Functional Programming* (Pitman, London, 1986).

[3] M.Dummett, *Elements of Intuitionism* (Oxford University Press, 1977).

[4] J.-Y.Girard, *Proofs and Types* (Cambridge University Press, 1989) (translated and with appendices by P.Taylor and Y.Lafont).

[5] Y.Lafont, *Logiques, Catégories et Machines*, Ph.D. thesis, Univ. Paris VII, 1988.

[6] J.Lambek and P.J.Scott, *Introduction to Higher Order Categorical Logic*, Cambridge Studies in Advanced Mathematics 7 (Cambridge University Press, 1986).

[7] D.J.Lehmann and M.B.Smyth, *Algebraic Specification of Data Types: A Synthetic Approach*, Math. Systems Theory 14(1981) 97–139.

[8] E.Moggi, *Computational lambda-calculus and monads*, Proc. 4th Annual Symposium on Logic in Computer Science, Asilomar CA (IEEE Computer Society Press, Washington, 1989), pp 14–23.

[9] E.Moggi, *Computational lambda-calculus and monads*, LFCS Technical Report 88-66, University of Edinburgh, 1988.

[10] E.Moggi, *Notions of Computations and Monads*, preprint, 1989.

[11] P.Martin-Löf, *Notes on the domain theoretic interpretation of type theory*, Proc. Workshop on Semantics of Programming Languages, Chalmers Univ. (1983).

[12] B.Nordström, *Martin-Löf's Type Theory as a Programming Logic*, PMG Report 27, Chalmers University, 1986.

[13] L.C.Paulson, *Logic and Computation* (Cambridge University Press, 1987).

[14] G.D.Plotkin, *LCF considered as a programming language*, Theoretical Computer Science 5(1977) 223–255.

[15] G.D.Plotkin, *Denotational semantics with partial functions*, unpublished lecture notes from CSLI Summer School (1985).

[16] D.S.Scott, *A type-theoretic alternative to CUCH, ISWIM, OWHY*, unpublished manuscript, University of Oxford, 1969.