EINDHOVEN UNIVERSITY OF TECHNOLOGY

Some categorical properties for a model for second order lambda calculus with subtyping

Citation for published version (APA):
Poll, E. (1991). *Some categorical properties for a model for second order lambda calculus with subtyping*. (Computing science notes; Vol. 9119). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1991

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Eindhoven University of Technology

Department of Mathematics and Computing Science

Some categorical properties for a model for second order lambda calculus with subtyping

by

Erik Poll

Computing Science Note 91/19
Eindhoven, September 1991

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. F. van Neerven
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Some categorical properties for a model for second order lambda calculus with subtyping

Erik Poll *

Abstract

In this paper we answer some of the category-theoretical questions, that were raised by the construction of a model for a second order lambda calculus with subtyping in [Pol91].

Contents

*supported by the Dutch organization for scientific research (NWO).

1 Introduction

For the construction of a model for second order (or polymorphic) lambda calculus with subtyping in [Pol91] , some category-theoretical ingredients are needed. Some of these are already discussed in [tEH89b] and [BH88]; here we deal with the rest of them.

For the model construction the standard technique for solving recursive domain equations, as presented in [SP82], is used . We take the initial fixed-point of an ω-continuous functor on an ω-category (the inverse-limit construction). That this category is an ω-category and that this functor is an ω-continuous functor is proved using properties of so-called O-categories. A clear and self-contained presentation of this method can be found in [BH88].

To apply the technique in this particular case, we have to work in a functor category, i.e. a category with functors from a category A to a category B as objects. We will show how all the necessary properties of an O-category B and of functors on B can be lifted to such a functor category and to functors on this functor category.

2 Second order lambda calculus

In [Pol91] the general structure of an environment model for a second order lambda calculus with subtyping is given. It is an extension of the general structure of an environment model for second order lambda calculus as described in [BMM90] and [tEH89a]. Here we will not give all the details, but just those which are relevant for the problem that we set out to solve in this paper. We consider a somewhat simpler version than the one in [Pol91]. However, the same method can be used for any of the versions of second order lambda calculus that can be found in literature.

2.1 Syntax

Types

Let $\mathcal{V}_{type}$ be a set of type variables and B a set of type constants, or base types (e.g. *bool*,*int*or *real*). The set of types over B is given by:

$$\sigma = b \mid \alpha \mid \sigma_1 \rightarrow \sigma_2 \mid \Pi(\Lambda\alpha.\sigma)$$

where $b \in B$ and $\alpha \in \mathcal{V}_{type}$.

Terms

Let $\mathcal{V}_{term}$ be a set of term variables and $\mathcal{C}_{term}$ a set of term constants. All term constants have a specifies type, which we will write as a superscript when necessary. We first define the set of *pseudo*-terms over $\mathcal{C}_{term}$ and $\mathcal{V}_{term}$, of which the set of terms will be a subset. The set of pseudo-terms over $\mathcal{C}_{term}$ and $\mathcal{V}_{term}$ is given by:

$$M = c \mid x \mid \lambda x : \sigma.\ M) \mid M_1 M_2 \mid \Lambda\alpha.\ M) \mid M\sigma$$

where $x \in \mathcal{V}_{term}$, $c \in \mathcal{C}_{term}$ and σ a type.
So we have abstraction over *term* variables , $(\lambda x : \sigma.\ M)$, and we have abstraction over *type* variables, $(\Lambda\alpha.\ M)$, and the corresponding forms of application : of a term to a term, $M_1 M_2$, and of a term to a type, $M\sigma$.
Terms are those pseudo-terms for which a type can be derived in a context. A context is a syntactic type assignment of the form $x_0 : \sigma_0, \ldots, x_n : \sigma_n$, i.e. a partial function from $\mathcal{V}_{term}$ to the set of

types. We write $\Gamma \vdash M : \sigma$ if we can derive that in context Γ the term M has type σ, using the following rules:

$$\frac{c^\sigma \in \mathcal{C}_{term}}{\Gamma \vdash c^\sigma : \sigma} \qquad \frac{(x : \sigma) \in \Gamma}{\Gamma \vdash x : \sigma}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\ M) : \sigma \rightarrow \tau}\ (\rightarrow I) \qquad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (\rightarrow E)$$

$$\frac{\Gamma \vdash M : \tau \quad \alpha \in \mathcal{V}_{type} \quad \alpha \text{ not free in } \Gamma}{\Gamma \vdash (\Lambda\alpha.\ M) : \Pi(\Lambda\alpha.\ \tau)}\ (\Pi\ I) \qquad \frac{\Gamma \vdash M : \Pi(\Lambda\alpha.\ \tau) \quad \sigma \text{ a type}}{\Gamma \vdash M\sigma : \tau[\alpha := \sigma]}\ (\Pi\ E)$$

Subtyping

We have a relation $\leq$ on types, the subtype relation . If $\sigma \leq \tau$, we say that σ is a subtype of τ. The subtype relation will be a pre-order (i.e. reflexive and transitive). We add the following type inference rule: the *subsumption rule*

$$\frac{\Gamma \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau}(SUB)$$

All subtyping will be based on a subtype relation $\leq^B$ on the base types. For example, if *int* and *real* are base types , we could have $int \leq^B real$.

We have the following rules for deducing $\sigma \leq \tau$:

$$\frac{\sigma \leq^B \tau}{\sigma \leq \tau}(START) \qquad \frac{\sigma : *}{\sigma \leq \sigma}(REFL) \qquad \frac{\rho \leq \sigma \quad \sigma \leq \tau}{\rho \leq \tau}(TRANS)$$

$$\frac{\sigma' \leq \sigma \quad \tau \leq \tau'}{\sigma \rightarrow \tau \leq \sigma' \rightarrow \tau'}(\leq\rightarrow) \qquad \frac{\sigma \leq \tau}{\Pi(\Lambda\alpha.\sigma) \leq \Pi(\Lambda\alpha.\tau)}(\leq \Pi)$$

Note the contravariance of $\rightarrow$ with respect to the subtype relation. That $\leq$ is indeed a pre-order is of course guaranteed by the rule $(REFL)$ and $(TRANS)$. Actually, because $\leq^B$ is already transitive, the rule $(TRANS)$ is derivable.

2.2 Semantics

Let T be the set of closed type expressions.

We have to find a suitable domain for every type. Because each free type variable will be assigned a closed type expression by an environment $\eta \in \mathcal{V}_{type} \rightarrow T$, we only have to consider the closed type expressions, i.e. the elements of T. From now on, whenever we say 'a is a type' we measn 'a is a closed type expression'.

The domains will be cpos. For every $a \in T$ we have a cpo Dom_a. Terms of type a will be interpreted as elements of the cpo Dom_a. These cpos have to satisfy certain domain equations.

For function types $a \rightarrow b$ we require

$$Dom_{a\rightarrow b} \cong [Dom_a \longrightarrow Dom_b]$$

Here $[Dom_a \longrightarrow Dom_b]$ is the cpo of continuous functions from Dom_a to Dom_b, with the pointwise ordering. This isomorphism allows us to interpret terms of type $a \rightarrow b$ not only as elements of the cpo $Dom_{a\rightarrow b}$, but also, via projection, as functions from Dom_a to Dom_b.

For polymorphic types $\Pi(\Lambda\alpha.\tau)$ we require

$$Dom_{\Pi(\Lambda\alpha.\tau)} \cong \prod_{a\in T} Dom_{\tau[\alpha:=a]}$$

$\prod_{a\in T} Dom_{\tau[\alpha:=a]}$ is the cpo which is the product of all the cpos $D_{\tau[\alpha:=a]}$, with the ordering coordinatewise. Terms of type $\Pi(\Lambda\alpha.\tau)$ can then be interpreted not only as elements of $Dom_{\Pi(\Lambda\alpha.\tau)}$ but also as elements of $Dom_{\tau[\alpha:=a]}$ for all types a. Because we take the product over all types, including the type $\Pi(\Lambda\alpha.\tau)$ itself, this form of polymorphism is called *impredicative.*

Notation Instead of $\Pi(\Lambda\alpha.\tau)$ we will write Πf; instead of $\tau[\alpha := a]$ we will then write $f(a)$.

Finally, for every base type a a cpo $domain_a$ is given. We could of course simply take Dom_a equal to $domain_a$, but instead we will just require that

$$Dom_a \cong domain_a$$

So the family of cpos $Dom =< Dom_a \mid a \in T >$ should be a solution of the following recursive domain equations

$$\begin{array}{lcll} Dom_a & \cong & domain_a & \text{for all } a \in B \\ Dom_{a\to b} & \cong & [Dom_a \longrightarrow Dom_b] & \text{for all } a \to b \in T \\ Dom_{\Pi f} & \cong & \prod_{a\in T} Dom_{f(a)} & \text{for all } \Pi f \in T \end{array}$$

The associated bijections are called Φ_a,$\Phi_{a\to b}$ and $\Phi_{\Pi f}$, respectively. So

$$\begin{array}{lcll} \Phi_a & \in & Dom_a \to domain_a & \text{for all } a \in B \\ \Phi_{a\to b} & \in & Dom_{a\to b} \to [Dom_a \longrightarrow Dom_b] & \text{for all } a \to b \in T \\ \Phi_{\Pi f} & \in & Dom_{\Pi f} \to \prod_{a\in T} Dom_{f(a)} & \text{for all } \Pi f \in T \end{array}$$

These bijections are also needed for the model.

$\underline{CPO}$ is the category with cpos as objects and continuous functions as morphisms.
For the domain equations for function types we have the *function space functor* , FS,

$$FS : \underline{CPO}^{OP} \times \underline{CPO} \to \underline{CPO}$$

defined by

- if D and E are cpos, then $FS(D, E) = [D \to E]$, the cpo of continuous functions from D to E, with the ordering pointwise.
- if $f \in [D' \to D]$ and $g \in [E \to E]$, then
  $FS(f,g) \; = \; (\boldsymbol{\lambda}\xi \in [D \to E].g{\circ}\xi{\circ}f) \; \in \; [[D \to E] \to [D' \to E']]$

For the polymorphic types we have the the *generalized product functor* , GP,

$$GP : \prod_{a\in T} \underline{CPO} \to \underline{CPO}$$

$\prod_{a\in T} \underline{CPO}$ is a product category. Its objects are T-indexed families of $\underline{CPO}$-objects, and its morphisms are T-indexed families of $\underline{CPO}$-morphisms. Composition is defined coordinatewise. GP is defined by

- if $< D_a \mid a \in I >$ is a family of cpos, then $GP(< D_a \mid a \in Kind_* >) = \prod_{a\in I} D_a$, the cpo which is the product of all the cpos D_a, with the ordering coordinatewise.

- if $< f_a \mid a \in I >$ is a family of functions, where $f_a \in [D_a \to E_a]$ for all $a \in I$, then $GP(< f_a \mid a \in I >) = \lambda(< d_a \mid a \in I >) \in GP(< D_a \mid a \in I >). < f_a(d_a) \mid a \in I >$ which is a continuous function from $GP(< D_a \mid a \in I >)$ to $GP(< E_a \mid a \in I >)$.

We can now write the recursive domain equations as follows

$$\begin{array}{lcll} Dom_a & \cong & domain_a & \text{for all } a \in B \\ Dom_{a \to b} & \cong & FS(Dom_a, Dom_b) & \text{for all } a \to b \in T \\ Dom_{\Pi f} & \cong & GP(< Dom_{f(a)} \mid a \in T >) & \text{for all } \Pi f \in T \end{array}$$

Coercions

Coercion functions are used to interpret subtyping: for all $a \le b$, we need a coercion function $Coe_{a\ b}$ from Dom_a to Dom_b. The coercions between base types are given: for all $a \le^B b$ we have a function $coerce_{ab} \in domain_a \to domain_b$. For these coercions the following holds

$$\begin{array}{lcll} coerce_{aa} & = & \lambda\xi \in domain_a\ .\ \xi & \text{for all } a \in B \\ coerce_{ac} & = & coerce_{bc} \circ coerce_{ab} & \text{for all } a \le^B b \le^B c \end{array}$$

The meaning of a term is defined by induction on its type derivation. Due to the subtyping, there may be many different type derivations for a term. We want the semantics to be *coherent*, which means that we get the same meaning for a term, irrespective of the particular type derivation we choose.

For example, suppose that $int \le real$, so $real \to bool \le int \to bool$. Let f be a term of type $real \to bool$ and M a term of type int. Then f also has type $int \to bool$, and M also has type $real$. For the meaning of fM we can the either consider f as a function from *real* to *bool* and M as an argument of type *real*, or f as a function from *int* to *bool* and M as an argument of type *int*. In the former case the coercion $Coe_{int\ real}$ will be used to coerce (the meaning of) M , in the latter case $Coe_{real \to bool\ int \to bool}$ will be used to coerce (the meaning of) f. For certain $Coe_{int\ real}$ and $Coe_{real \to bool\ int \to bool}$ this will result in two different meanings for fM. To prevent this, some additional conditions have to be imposed on the coercion functions.

The family of coercion functions $Coe =< Coe_{a\ b} \mid a \le b >$ should satisfy the following *coherence conditions*

$$\begin{array}{llcll} (0) & Coe_{a\ a} & = & \lambda\xi \in Dom_a.\ \xi & \text{for all } a \in T \\ (1) & Coe_{a\ c} & = & Coe_{b\ c} \circ Coe_{a\ b} & \text{for all } a \le b \le c \\ (2) & Coe_{a\ b} & = & \Phi_b^{-1} \circ coerce_{ab} \circ \Phi_a & \text{for all } a \le^B b \\ (3) & Coe_{a \to b\ a' \to b'} & = & \Phi_{a' \to b'}^{-1} \circ FS(\ Coe_{a'\ a},\ Coe_{b\ b'}) \circ \Phi_{a \to b} & \text{for all } a \to b \le a' \to b' \\ (4) & Coe_{\Pi f\ \Pi g} & = & \Phi_{\Pi g}^{-1} \circ GP(<\ Coe_{f(a)\ g(a)} \mid a \in T >) \circ \Phi_{\Pi f} & \text{for all } \Pi f \le \Pi g \end{array}$$

In [Pol91] it is shown that the semantics is coherent if and only if the coercions satisfy these requirements.

We will now show this can be elegantly described in category-theoretical terms.

Every pre-order $(A, \le)$ can be seen as a category. The objects are the elements of A , and there is a (unique) arrow, called $x \le y$, from x to y iff $x \le y$. Because $\le$ is reflexive, there is an identity $x \le x$ for all objects x and because $\le$ is transitive, composition is always defined: $y \le z \circ x \le y$ will be $x \le z$.

Let $\mathcal{T}$ be the category corresponding with the pre-order $(T, \leq)$. Together, *Dom* and *Coe* can be seen as a *functor* from $\mathcal{T}$ to $\underline{CPO}$. *Dom* is the object part, mapping every $\mathcal{T}$-object, i.e. every element of T, to a $\underline{CPO}$-object, a cpo. *Coe* is the morphism part, mapping every $\mathcal{T}$-morphism $a \leq b$ to a continuous function from Dom_a to Dom_b. We will call this functor *Dom&Coe*.
For *Dom&Coe* to be a functor, identities and composition must be preserved. Preservation of identities and composition is equivalent with coherence conditions (0) and (1).

In the same way, $< domain_a \mid a \in B >$ and $< coerce_{ab} \mid a \leq^B b >$ form a functor from the category corresponding with the pre-order $\leq^B$ on base types to $\underline{CPO}$.

So we are looking for a *functor* $Dom\&Coe : \mathcal{T} \rightarrow \underline{CPO}$ and a family of bijections $\Phi = < \Phi_a \mid a \in T >$ such that

$$\begin{array}{lcll} Dom_a & \cong & domain_a & \text{for all } a \in B \\ Dom_{a \rightarrow b} & \cong & FS(Dom_a, Dom_b) & \\ Dom_{\Pi f} & \cong & GP(< Dom_{f(a)} \mid a \in T >) & \end{array}$$

with Φ the associated family of bijections, and

$$\begin{array}{llcll} (2) & Coe_{a\ b} & = & \Phi_b^{-1} \circ coerce_{ab} \circ \Phi_a & \text{for all } a \leq^B b \\ (3) & Coe_{a \rightarrow b\ a' \rightarrow b'} & = & \Phi_{a' \rightarrow b'}^{-1} \circ FS(\ Coe_{a'\ a},\ Coe_{b\ b'}) \circ \Phi_{a \rightarrow b} & \text{for all } a \rightarrow b \leq a' \rightarrow b' \\ (4) & Coe_{\Pi f\ \Pi g} & = & \Phi_{\Pi g}^{-1} \circ GP(<\ Coe_{f(a)\ g(a)} \mid a \in T >) \circ \Phi_{\Pi f} & \text{for all } \Pi f \leq \Pi g \end{array}$$

Any functor from $\mathcal{T}$ to $\underline{CPO}$ will satisfy conditions (0) and (1), so these can be omitted.

2.3 The solution method

In [SP82] and [BH88] a solution method is given for equations of the form

$$X \cong FX$$

where X ranges over the objects of a category K and $F : K \rightarrow K$ is an endofunctor on that category. If K is an ω-category - i.e. a category with an initial object and colimits for all ω-chains - and F is an ω-continuous functor - i.e. a functor that preserves colimits of ω-chains [1] - the method yields a fixed point, a pair (A, Φ) where $A \in Obj(K)$ and Φ is an isomorphism from FA to A in the category K.

This is the solution method we will use to to construct *Dom&Coe* and Φ. So we have to find a suitable ω-category, with functors from $\mathcal{T}$ to $\underline{CPO}$ as objects, and an ω-continuous functor on that category.

Because in general it is difficult to prove that a category is an ω-category or that a functor is ω-continuous, a special class of categories, the *O-categories* , have been introduced. For every O-category there is an associated category of embedding-projection pairs. Checking if such a category is an ω-category is relatively easy, as is proving ω-continuity of functors on these categories.
In the next section we list some properties of O-categories and functors on O-categories that appear in [BH88], that we need in sections 3 and 4.
In section 3 a suitable (functor) category is found. That this category is indeed an ω-category is proved using properties of O-categories and the associated categories of projection-embedding pairs.
In section 4 we will define a functor on this category and show that any fixed point of this functor gives us a functor *Dom&Coe* and a family of bijections Φ solving the recursive domain equations and satisfying the coherence conditions. ω-continuity is proved using so-called local continuity.

[1] Actually, such a functor should be called ω-cocontinuous.

3 O-categories

This section lists some of the definitions and results from [BH88]. All proofs can be found there, except those involving the functor GP. GP and its properties are discussed in [tEH89b].

1 definition ω-category , ω-continuous functor

- an ω-category is a category with an initial object and colimits for all ω-chains
- an ω-continuous functor is a functor that preserves colimits of ω-chains

□

3.1 O-categories

2 definition O-category
A category is an *O-category* iff

- every hom-set is a poset in which every ascending ω-chain has a l.u.b.
- composition is ω-continuous with respect to the partial order on the hom-sets

□

3 definition category of embedding-projection pairs
If B is an O-category, then the associated category of embedding-projection pairs B_{PR} is the category with

- the same objects as B, i.e. $Obj(B_{PR}) = Obj(B)$
- as morphisms embedding-projection pairs of morphisms, i.e. for $a, b \in Obj(B_{PR})$
  $(f, g) \in Hom_B(a, b)$
  $\Leftrightarrow$
  $f \in Hom_B(a, b) \wedge g \in Hom_B(b, a) \wedge f \circ g \sqsubseteq id_b \wedge g \circ f = id_a$

□

4 definition localized category
An O-category B is called *localized* if for any ω-chain Δ in B_{PR} and for any Δ-colimit $(D, <(\phi_i, \psi_i)>_{i\in\mathbb{N}})$ there exists a B-object E and a B_{PR}-morphism (f, g) from E to D such that

$$\bigsqcup_{i\geq 0}(\phi_i \circ \psi_i) = f \circ g$$

□

5 theorem initiality theorem
Let B be a localized O-category, Δ an ω-chain in B_{PR} and $(D, <(\phi, \psi)>_{i\in\mathbb{N}})$ a co-cone for Δ. Then

$$(D, <(\phi, \psi)>_{i\in\mathbb{N}}) \text{ is a co-limit for } \Delta \iff \bigsqcup_{i\geq 0}(\phi_i \circ \psi_i) = id_D$$

□

This theorem enables us to prove or disprove that a category B_{PR} is an ω-category in a simple way, provided that B is localized.

6 definition idempotent,split
Let B be a category and $b \in Obj(B)$. Then
a morphism $f \in Hom_B(b,b)$ is called an *idempotent* if $f \circ f = f$
and
a morphism $f \in Hom_B(b,b)$ is called *split* if there exist a B-object a and morphisms $g \in Hom_B(b,a)$ and $h \in Hom_B(a,b)$ such that $f = g \circ h$ and $h \circ g = id_a$.
□

Using these definitions we can give an easy method to establish that an O-category is localized.

7 theorem
If B is an O-category in which every idempotent is split, then B is localized.
□

3.2 Functors on O-categories

8 definition local monotonicity , local continuity
Let B and C be O-categories, and F a functor from B to C.
F is called *locally monotonic* (*locally continuous*) if for all $a, b \in B$, the functor F, viewed as a map from $Hom_B(a,b)$ to $Hom_C(Fa, Fb)$, is monotonic (continuous) with respect to the partial order on hom-sets.
□

Clearly any locally continuous functor is also locally monotonic.

9 definition F_{PR}
Let B and C be O-categories, and F a locally monotonic functor from B to C.
Then F_{PR} is a functor from B_{PR} to C_{PR}, defined as follows

- if $b \in Obj(B_{PR})$ then $F_{PR}(b) = F(b)$. (Remember $Obj(B_{PR}) = Obj(B_{PR})$ and $Obj(C_{PR}) = Obj(C_{PR})$)
- if $(f,g) \in Hom_{B_{PR}}(b,b')$ then $F_{PR}(f,g) = (F(f), F(g))$

Local-monotonicity of F is needed to guarantee that $(F(f), F(g))$ is an embedding-projection pair.
□

The next theorem now enables us to prove that a functor F_{PR} is ω-continuous in a relatively simple way.

10 theorem continuity theorem
Let B and C be O-categories and F a functor from B to C.
If F is locally continuous and B is localized, then $F_{PR} : B_{PR} \to C_{PR}$ is ω-continuous.
□

3.3 Some examples of O-categories and locally continuous functors

11 definition $\underline{CPO}$

$\underline{CPO}$ is the category with cpos as objects and continuous functions as morphisms

□

12 definition $\underline{CPO}_\perp$

$\underline{CPO}_\perp$ is the category with cpos as objects and strict continuous functions as morphisms

□

$\underline{CPO}_\perp$ is a subcategory of $\underline{CPO}$.

13 lemma In $\underline{CPO}$ and in $\underline{CPO}_\perp$ every idempotent is split. □

14 theorem $\underline{CPO}$ and $\underline{CPO}_\perp$ are localized O-categories . □

15 theorem $(\underline{CPO}_\perp)_{PR} = \underline{CPO}_{PR}$ □

16 theorem $\underline{CPO}_{PR}$ is an ω-category. □

Finally, we consider two ways to construct new O-categories from old ones.

17 lemma

If B is a localized O-category, so is B^{OP}. Moreover, $B_{PR} \cong (B^{OP})_{PR}$; the associated isomorphism is given by the following functor $F_1: B_{PR} \rightarrow (B^{OP})_{PR}$.

The object part of F_1 is defined by $F_1 b = b$

and the morphism part by $F_1(f,g) = (g,f)$

□

18 lemma

If A and B are localized O-categories, so is $A \times B$. Moreover, $A_{PR} \times B_{PR} \cong (A \times B)_{PR}$; the associated isomorphism is given by the following functor $F_2: (B^{OP})_{PR} \times B_{PR} \rightarrow (B^{OP} \times B)_{PR}$.

The object part of F_2 is defined by $F_2(a,b) = (a,b)$

and the morphism part by $F_2((f,f'),(g,g')) = ((f,g),(f',g'))$

□

19 lemma FS and GP as defined on page 2.2 are locally continuous. □

Because $\underline{CPO}_\perp$ is a subcategory of $\underline{CPO}$ and because FS and GP preserve strictness, we also have $FS: \underline{CPO}_\perp{}^{OP} \times \underline{CPO}_\perp \rightarrow \underline{CPO}_\perp$ and $GP: \prod_{a\in T} \underline{CPO}_\perp \rightarrow \underline{CPO}_\perp$.
Using definition 9, we get $FS_{PR}: (\underline{CPO}^{OP} \times \underline{CPO})_{PR} \rightarrow \underline{CPO}_{PR}$ defined by

$$\begin{aligned} FS_{PR}(D,E) &= FS(D,E) \\ FS_{PR}((\psi,\phi'),(\phi,\psi')) &= (FS(\psi,\phi'),FS(\phi,\psi')) \end{aligned}$$

If $((\psi,\phi),(\phi,\psi')): A \times B \rightarrow C \times D$ in $\underline{CPO}^{OP} \times \underline{CPO}$, this means that
$\phi': B \rightarrow D \quad \phi: C \rightarrow A$
$\psi': D \rightarrow B \quad \psi: A \rightarrow C$ in $\underline{CPO}$.
$GP_{PR}: (\prod_{a\in I} \underline{CPO})_{PR} \rightarrow \underline{CPO}_{PR}$ is given by

$$\begin{aligned} GP_{PR}(< D_a \mid a \in I >) &= GP(< D_a \mid a \in I >) \\ GP_{PR}(< \phi_a \mid a \in I >, < \psi_a \mid a \in I >) &= (GP(< \phi_a \mid a \in I >), GP(< \psi_a \mid a \in I >)) \end{aligned}$$

By theorem 10 FS_{PR} and GP_{PR} are ω-continuous.

20 remark

FS_{PR} is usually composed with the isomorphism between

$$(\underline{CPO}^{OP} \times \underline{CPO})_{PR} \cong \underline{CPO}_{PR}$$

given by lemma's 17 and 18, and GP_{PR} with the isomorphism between

$$(\prod \underline{CPO})_{PR} \cong \prod \underline{CPO}_{PR}$$

resulting in $FS'_{PR} : \underline{CPO}_{PR} \times \underline{CPO}_{PR} \rightarrow \underline{CPO}_{PR}$ and $GP'_{PR} : \prod_{a \in I} \underline{CPO}_{PR} \rightarrow \underline{CPO}_{PR}$ with the following definitions

$$\begin{aligned} FS'_{PR}(D, E) &= FS(D, E) \\ FS'_{PR}((\phi, \psi), (\phi', \psi')) &= (FS(\psi, \phi'), FS(\phi, \psi')) \end{aligned}$$

$$\begin{aligned} GP'_{PR}(< D_a \mid a \in I >) &= GP(< D_a \mid a \in I >) \\ GP'_{PR}(< (\phi_a, \psi_a) \mid a \in I >) &= (GP(< \phi_a \mid a \in I >), GP(< \psi_a \mid a \in I >)) \end{aligned}$$

These functors are also ω-continuous.

□

4 Functor categories

21 definition functor category $[A, B]$
If A and B are categories, then $[A, B]$ is the category with functors from A to B as objects and natural transformations between such functors as morphisms, i.e.

$$\eta \in Hom_{[A,B]}(F, G) \text{ iff } \eta : F \stackrel{\bullet}{\longrightarrow} G$$

□

As we shall see , for our purposes the notation $[A, B]$ is preferable to the more conventional notation B^A.
If A is a discrete category - i.e. the only morphisms are identities - then $[A, B]$ is simply a product category, viz. $\prod_{a \in Obj(A)} B$.

22 lemma
If B is an O-category, then $[A, B]$ is an O-category.
proof
An $[A, B]$-morphism is a natural transformation, i.e. a mapping from A-objects to B-morphisms. The ordering on $[A, B]$-morphisms is just the ordering on B-morphisms, pointwise. That $[A, B]$ is indeed an O-category is easily verified:

- every hom-set in $[A, B]$ is a poset, and every ascending chain in a hom-set has a lub, which we get by taking the pointwise lubs.
- composition of natural transformations is defined pointwise, so composition is ω-continuous with respect to the ordering on the hom-sets.

□

23 lemma
Let B be an O-category in which every idempotent is split.
Then $[A, B]$ is a localized O-category.
proof
Idempotents in $[A, B]$ are mappings from A-objects to B-idempotents. So if every idempotent in B splits, then every idempotent in $[A, B]$ splits (pointwise). If every idempotent is split in a category then it is a localized category (theorem 7) so $[A, B]$ is localized.
□

From now on, B will be an O-category, and A an arbritary category.
Because $[A, B]$ is an O-category, there is an associated category of embedding-projection pairs. By definition 3, this category is defined as follows.

24 definition $[A, B]_{PR}$
$[A, B]_{PR}$ is the category with functors from A to B as objects and projection-embedding pairs of natural transformations between such functors as morphisms,

$$(\eta, \theta) \in Hom_{[A,B]_{PR}}(F, G) \quad \text{iff} \quad \begin{array}{l} \eta : F \stackrel{\bullet}{\longrightarrow} G \\ \theta : G \stackrel{\bullet}{\longrightarrow} F \\ \theta \circ \eta = id_F \\ \eta \circ \theta \sqsubseteq id_G \end{array}$$

□

Because everything is defined pointwise,

$$\theta \circ \eta = id_F \iff \forall_{a \in Obj(A)} [\theta_a \circ \eta_a = id_{Fa}]$$
$$\eta \circ \theta \sqsubseteq id_G \iff \forall_{a \in Obj(A)} [\eta_a \circ \theta_a \sqsubseteq id_{Ga}]$$

25 lemma

Let B and $[A, B]$ be localized O-categories and suppose that B_{PR} is an ω-category. Then in $[A, B]_{PR}$ every ω-chain has a colimit.

proof

Let Δ be the following ω-chain in $[A, B]_{PR}$

$$F^0 \xrightarrow{(\Phi^0, \Psi^0)} F^1 \xrightarrow{(\Phi^1, \Psi^1)} F^2 \xrightarrow{(\Phi^2, \Psi^2)} \cdots$$

We will define a functor E from A to B. First we define its object part.
Let $a \in Obj(A)$. Then

$$F^0 a \xrightarrow{(\Phi^0_a, \Psi^0_a)} F^1 a \xrightarrow{(\Phi^1_a, \Psi^1_a)} F^2 a \xrightarrow{(\Phi^2_a, \Psi^2_a)} \cdots$$

is an ω-chain B_{PR}. B_{PR} is an ω-category, so this chain has a colimit : $(Ea, < (\phi^i_a, \psi^i_a) \mid i \in \mathbb{N} >)$.
This means that for all $i \in \mathbb{N}$

$$\begin{array}{ccc} & Ea & \\ {\scriptstyle (\phi^i_a, \psi^i_a)} \nearrow & & \nwarrow {\scriptstyle (\phi^{i+1}_a, \psi^{i+1}_a)} \\ F^i a & \xrightarrow{(\Phi^i_a, \Psi^i_a)} & F^{i+1} a \end{array}$$

and, since B is localized, $\bigsqcup \phi^i_a \circ \psi^i_a = id_{Ea}$.
We define the morphism part of $E \in Obj([A, B]_{PR})$ by

$$Ef \quad = \quad \bigsqcup \phi^i_b \circ F^i f \circ \psi^i_a \qquad \text{for } f \in Hom_A(a, b)$$

We will prove that this is defined , i.e.
(i) $\bigsqcup \phi^i_b \circ F^i f \circ \psi^i_a$ exists for all $f : a \to b$ in A
and that
(ii) $(E, < (\phi^i, \psi^i) \mid i \in \mathbb{N} >)$ is a cocone for Δ.
Once we have established (i) and (ii), then

$$\begin{array}{rl} & (E, < (\phi^i, \psi^i) \mid i \in \mathbb{N} >) \text{ is a colimit for } \Delta \\ = & \{[A, B] \text{ is localized}\} \\ & \bigsqcup \phi^i \circ \psi^i = id_E \ \wedge \ (E, < (\phi^i, \psi^i) \mid i \in \mathbb{N} >) \text{ is a cocone for } \Delta \\ = & \{ \ id_E \text{ and lubs defined pointwise}\} \\ & \forall_{a \in A} \bigsqcup \phi^i_a \circ \psi^i_a = id_{Ea} \ \wedge \ (E, < (\phi^i, \psi^i) \mid i \in \mathbb{N} >) \text{ is a cocone for } \Delta \\ = & \{ \text{ def } \phi^i \text{ and } \psi^i \text{ , (ii)}\} \\ & true \end{array}$$

and we have proved that Δ has a colimit .

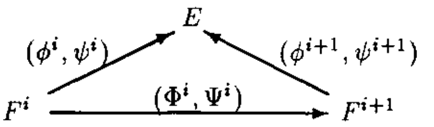
(i) To prove : $\bigsqcup \phi_b^i \circ F^i f \circ \psi_a^i$ exists for all $f : a \rightarrow b$ in A.
Because B is an O-category, a proof that $< \phi_b^i \circ F^i f \circ \psi_a^i >_{i \in \mathbb{N}}$ is an ascending chain in $Hom_B(Ea, Eb)$ suffices.

$$
\begin{array}{ll}
 & \phi_b^i \circ F^i f \circ \psi_a^i \\
= & \{ \phi_b^i = \phi_b^{i+1} \circ \Phi_b^i \ , \ \psi_a^i = \Psi_a^i \circ \psi_a^{i+1} \} \\
 & \phi_b^{i+1} \circ \Phi_b^i \circ F^i f \circ \Psi_a^i \circ \psi_a^{i+1} \\
= & \{ \Phi^i : F^i \stackrel{\bullet}{\longrightarrow} F^{i+1} \} \\
 & \phi_b^{i+1} \circ F^{i+1} f \circ \Phi_a^i \circ \Psi_a^i \circ \psi_a^{i+1} \\
\sqsubseteq & \{ \Phi_a^i \circ \Psi_a^i \sqsubseteq id_{F_a^{i+1}} \} \\
 & \phi_b^{i+1} \circ F^{i+1} f \circ \psi_a^{i+1}
\end{array}
$$

(ii) To prove : $(E, < (\phi^i, \psi^i) \mid i \in \mathbb{N} >)$ is a cocone for Δ.
We must prove that for all $i \in \mathbb{N}$

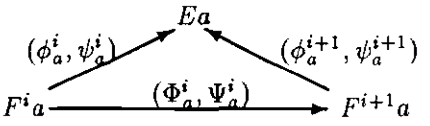
(a) $(\phi^i, \psi^i) \in Hom_{[A,B]_{PR}}(F^i, E)$

(b) $(\phi^i, \psi^i) = (\phi^{i+1}, \psi^{i+1}) \circ (\Phi^i, \Psi^i)$, i.e.



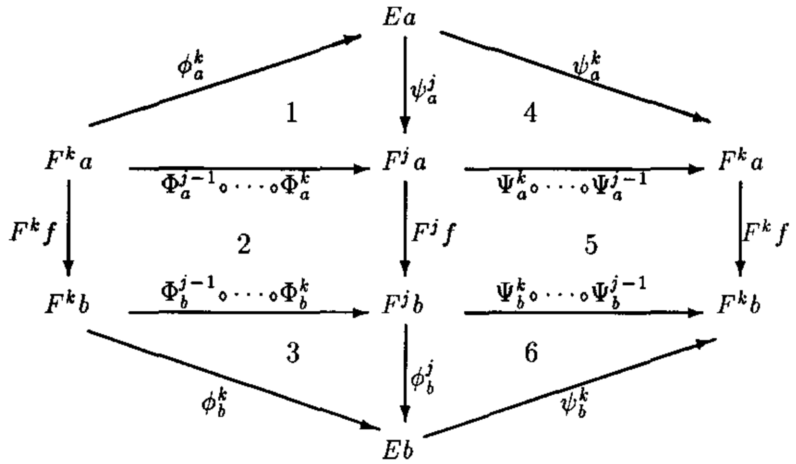
We know that for all $a \in Obj(A)$ and $i \in \mathbb{N}$

$(\phi_a^i, \psi_a^i) = (\phi_a^{i+1}, \psi_a^{i+1}) \circ (\Phi_a^i, \Psi_a^i)$, i.e.



so we know that (b) is true.
To prove (a) we only have to prove that $\phi^i : F^i \stackrel{\bullet}{\longrightarrow} E$, $\psi^i : E \stackrel{\bullet}{\longrightarrow} F^i$
since we already know that for all $a \in Obj(A)$ $\psi_a^i \circ \phi_a^i = id_{F^i a}$, $\phi_a^i \circ \psi_a^i \sqsubseteq id_{Ea}$

Suppose $k < j$ and $f : a \to b$ in A.



For all i $\Phi^i : F^i \overset{\bullet}{\longrightarrow} F^{i+1}$, so (2) commutes, and $\Psi^i : F^{i+1} \overset{\bullet}{\longrightarrow} F^i$, so (5) commutes.
$(\phi^i, \psi^i) = (\phi^{i+1}, \psi^{i+1}) \circ (\Phi^i, \Psi^i)$, so (3) and (4) commute.
Finally, (1) and (6) commute because

$$\begin{array}{llll} & \psi^j_a \circ \phi^k_a & & \psi^k_b \circ \phi^j_b \\ = & \{\phi^i \;=\; \phi^{i+1} \circ \Phi^i \text{ for all } i\ \} & = & \{\psi^i \;=\; \Psi^i \circ \psi^{i+1} \text{ for all } i\ \} \\ & \psi^j_a \circ \phi^j_a \circ \Phi^{j-1}_a \circ \cdots \circ \Phi^k_a & & \Psi^k_b \circ \cdots \circ \Psi^{j-1}_b \circ \psi^j_b \circ \phi^j_b \\ = & \{\ \psi^j_a \circ \phi^j_a \;=\; id_{F^j a}\} & = & \{\ \psi^j_b \circ \phi^j_b \;=\; id_{F^j b}\} \\ & \Phi^{j-1}_a \circ \cdots \circ \Phi^k_a & & \Psi^k_b \circ \cdots \circ \Psi^{j-1}_b \end{array}$$

Using

$$(*) \quad \begin{array}{lll} Ef & = \bigsqcup_{j \in \mathbb{N}} \phi^j_b \circ F^j f \circ \psi^j_a & \text{by definition} \\ & = \bigsqcup_{j : j > k} \phi^j_b \circ F^j f \circ \psi^j_a & \text{because } < \phi^j_b \circ F^j f \circ \psi^j_a >_{j \in \mathbb{N}} \text{ is an ascending chain} \end{array}$$

we can show that for all $j > k$

For all $j > k$

$$\begin{array}{llll} & \phi^k_b \circ F^k f & & F^k f \circ \psi^k_a \\ = & \{\text{LHS diagram}\} & = & \{\text{RHS diagram}\} \\ & \phi^j_b \circ F^j f \circ \psi^j_a \circ \phi^k_a & & \psi^k_b \circ \phi^j_b \circ F^j f \circ \psi^j_a \end{array}$$

so

$$\begin{array}{llll} & \phi^k_b \circ F^k f & & F^k f \circ \psi^k_a \\ = & & = & \\ & \left(\bigsqcup_{j : j > k} \phi^j_b \circ F^j f \circ \psi^j_a \circ \phi^k_a\right) & & \left(\bigsqcup_{j : j > k} \psi^k_b \circ \phi^j_b \circ F^j f \circ \psi^j_a\right) \\ = & & = & \\ & \left(\bigsqcup_{j : j > k} \phi^j_b \circ F^j f \circ \psi^j_a\right) \circ \phi^k_a & & \psi^k_b \circ \left(\bigsqcup_{j : j > k} \phi^j_b \circ F^j f \circ \psi^j_a\right) \\ = & \{\ (*)\ \} & = & \{\ (*)\ \} \\ & Ef \circ \phi^k_a & & \psi^k_b \circ Ef \end{array}$$

i.e.

$$\phi^k : F^k \overset{\bullet}{\longrightarrow} E \qquad\qquad \psi^k : E \overset{\bullet}{\longrightarrow} F^k$$

□

26 corollary
Let B be a O-category in which every isomorphism is split (so B is localized) . Suppose that B_{PR} is an ω-category and that $[A, B]_{PR}$ has an initial element.
Then $[A, B]_{PR}$ is an ω-category.
proof
B_{PR} is an ω-category and by lemma 23 $[A, B]$ is a localized O-category, and so by lemma 25 every ω-chain in $[A, B]_{PR}$ has a colimit.
So if $[A, B]_{PR}$ has an initial element, $[A, B]_{PR}$ is an ω-category.
□

27 corollary $[A,\ \underline{CPO_\perp}]_{PR}$ is an ω-category.
proof
In $\underline{CPO_\perp}$ every idempotent is split (lemma 13) and ($\underline{CPO_\perp})_{PR} = \underline{CPO}_{PR}$ is an ω-category.
By the previous corollary we only have to find an initial element in $[A,\ \underline{CPO}_{PR}]_{PR}$.
The obvious candidate for an initial object in $[A,\ \underline{CPO_\perp}]_{PR}$ is the constant functor which maps every A-object to the one-point cpo and every A-morphism to the only possible function between two one-point cpos. It can easily be verified that this is indeed an initial element.
□

The category $[A,\ \underline{CPO}]_{PR}$, however, is *not* an ω-category, because it does not have an initial object. The initial object of $[A,\ \underline{CPO_\perp}]_{PR}$ is of course also an $[A,\ \underline{CPO}]_{PR}$-object, but it is not initial.
We will construct the model in the category $[\mathcal{T},\ \underline{CPO_\perp}]_{PR}$. As a consequence of using $\underline{CPO_\perp}$ instead of $\underline{CPO}$ all coercions will be strict. The coercions $coerce_a\ b$ for base types a and b the also need to be strict.

5 The model construction

In the rest of this paper, the definitions of $FS : \underline{CPO_\perp}^{OP} \times \underline{CPO_\perp} \to \underline{CPO_\perp}$ and $GP : \prod_{a\in T} \underline{CPO_\perp} \to \underline{CPO_\perp}$ no longer matter. The only thing that matters is that they are locally continuous.
$\mathbb{K}$ is short for the category $[T, \underline{CPO_\perp}]_{PR}$.

28 definition $\mathbb{F} : \mathbb{K} \to \mathbb{K}$
$\mathbb{F}$ is a functor $\mathbb{K}$ to $\mathbb{K}$, so it consists of an object part, a mapping from $Obj(\mathbb{K})$ to $Obj(\mathbb{K})$, and an morphism part, a mapping from $Mor(\mathbb{K})$ to $Mor(\mathbb{K})$.
The object part of $\mathbb{F}$ is defined as follows: if $F \in Obj(\mathbb{K})$, then $\mathbb{F}F \in Obj(\mathbb{K})$, i.e. $\mathbb{F}F$ is a functor from T to $\underline{CPO_\perp}$. The object part of $\mathbb{F}F$, a mapping from $Obj(T)$ to $Obj(\underline{CPO_\perp})$, is defined by

$$\begin{array}{lcl} (\mathbb{F}F)a & = & domain_a \\ (\mathbb{F}F)a \to b & = & FS(Fa, Fb) \\ (\mathbb{F}F)\Pi f & = & GP(< F(f(a)) \mid a \in T >) \end{array}$$

and the morphism part of $\mathbb{F}F$, a mapping from $Mor(T)$ to $Mor(\underline{CPO_\perp})$, is defined by

$$\begin{array}{lcl} (\mathbb{F}F)a \leq b & = & coerce_{ab} \\ (\mathbb{F}F)a \to b \leq a' \to b' & = & FS(F\ a' \leq a, F\ b \leq b') \\ (\mathbb{F}F)\Pi f \leq \Pi g & = & GP(< F\ f(a) \leq g(a) \mid a \in T >) \end{array}$$

The morphism part of $\mathbb{F}$ is defined as follows:
if $(\eta, \theta) \in Hom_{\mathbb{K}}(F, G)$, then $\mathbb{F}(\eta, \theta) = (\eta', \theta')$, where

$$\begin{array}{lcl} (\eta'_a, \theta'_a) & = & (id_{domain_a}, id_{domain_a}) \\ (\eta'_{a\to b}, \theta'_{a\to b}) & = & (FS(\eta_a, \theta_a), FS(\eta_b, \theta_b)) \\ (\eta'_{\Pi f}, \theta'_{\Pi f}) & = & (GP(< \eta_{f(a)} \mid a \in T >), GP(< \theta_{f(a)} \mid a \in T >)) \end{array}$$

Checking $\eta' : \mathbb{F}F \stackrel{\bullet}{\longrightarrow} \mathbb{F}G$ and $\theta' : \mathbb{F}G \stackrel{\bullet}{\longrightarrow} \mathbb{F}F$ is straightforward, and it can easily be verified (coordinatewise) that $\mathbb{F}$ preserves identities and composition.
□

Note that for the coercions FS is used, which takes care of the contravariance of $\to$ with respect to the subtype relation whereas for the morphisms FS_{PR} is used:

$$(FS(\eta_a, \theta_a), FS(\eta_b, \theta_b)) = FS_{PR}((\theta_a, \eta_b), (\eta_a, \theta_b))$$

which is covariant in both arguments, so that a fixed point can be constructed.
Similarly, GP is used for the coercions , and GP_{PR} is used for the morphisms :

$$(GP(< \eta_{f(a)} \mid a \in T >), GP(< \theta_{f(a)} \mid a \in T >)) = GP_{PR}(< \eta_{f(a)} \mid a \in T >, < \theta_{f(a)} \mid a \in T >)$$

In terms of the functors FS_{PR}' and $GPpr'$, as defined in remark 20 :

$$(FS(\eta_a, \theta_a), FS(\eta_b, \theta_b)) = FS'_{PR}((\eta_a, \theta_a), (\eta_b, \theta_b))$$

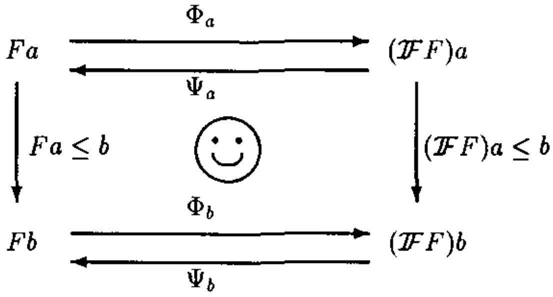
$$(GP(< \eta_{f(a)} \mid a \in T >), GP(< \theta_{f(a)} \mid a \in T >)) = GP'_{PR}(< (\eta_{f(a)}, \theta_{f(a)}) \mid a \in T >)$$

Any fixed point of $\mathbb{F}$ will solve the recursive domain equations and satisfy the conditions for the coercion functions.

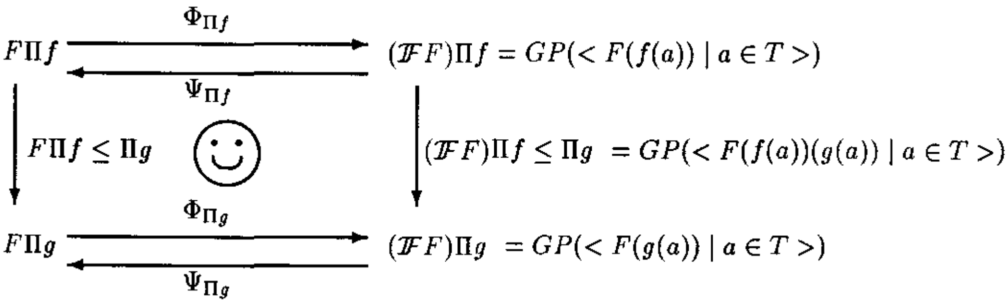
For example, let $(F, (\Phi, \Psi))$ be a fixed point of $\mathcal{F}$, i.e. (Φ, Ψ) is an isomorphism between F and $\mathcal{F}F$. This means that $\Phi : F \stackrel{\bullet}{\longrightarrow} \mathcal{F}F$ and $\Psi : \mathcal{F}F \stackrel{\bullet}{\longrightarrow} F$, such that $\Phi \circ \Psi = id_{\mathcal{F}F}$ and $\Psi \circ \Phi = id_F$. Because everything is defined pointwise, this means that for all $a \in T$

$$\begin{aligned} \Phi_a \circ \Psi_a &= id_{(\mathcal{F}F)a} \\ \Psi_a \circ \Phi_a &= id_{Fa} \end{aligned}$$

and for all $a \leq b$



Suppose $\Pi f \leq \Pi g$. Then



and

$$F\ \Pi f {\leq} \Pi g\ = \Psi_{\Pi g} \circ (\mathcal{F}F)\Pi f {\leq} \Pi g \circ \Phi_{\Pi f}\ = \Psi_{\Pi g} \circ GP(< F\ f(a) {\leq} g(a) \mid a \in T >) \circ \Phi_{\Pi f}$$

so condition (4) (see page 6) is satisfied. In the same way it can be shown that condition (2) and (3) is satisfied.

We now want to prove that $\mathcal{F}$ is an ω-continuous functor, so that by the initial fixed point lemma an initial fixed point of $\mathcal{F}$ can be constructed. For this we can use the notion of local continuity. We define the following functor.

29 definition $\mathcal{H}\ :\ [T,\ \underline{CPO_\perp}]^{OP} \times [T,\ \underline{CPO_\perp}] \to [T,\ \underline{CPO_\perp}]$

If $(F, G)\ \in Obj([T,\ \underline{CPO_\perp}]^{OP} \times [T,\ \underline{CPO_\perp}])$, so F and G are functors from T to $\underline{CPO_\perp}$, then $\mathcal{H}(F, G)$ is defined by

$$\begin{aligned} (\mathcal{H}(F,G))a &= domain_a \\ (\mathcal{H}(F,G))a \to b &= FS(Fa, Gb) \\ (\mathcal{H}(F,G))\Pi f &= GP(< G(f(a)) \mid a \in T >) \end{aligned}$$

and

$$\begin{aligned} (\mathcal{H}(F,G))a{\leq}b &= coerce_{ab} \\ (\mathcal{H}(F,G))a \to b{\leq}a' \to b' &= FS(F\ a'{\leq}a, G\ b{\leq}b') \\ (\mathcal{H}(F,G))\Pi f{\leq}\Pi g &= GP(< G\ f(a){\leq}g(a) \mid a \in T >) \end{aligned}$$

If $(\eta,\theta) \in Hom((F,G),(F',G'))$, so $\eta : F' \stackrel{\bullet}{\longrightarrow} F$ and $\theta : G \stackrel{\bullet}{\longrightarrow} G'$ then $\mathcal{H}(\eta,\theta)$ is defined by

$$\begin{array}{lcl} (\mathcal{H}(\eta,\theta))a & = & id_{domain_a} \\ (\mathcal{H}(\eta,\theta))a \to b & = & FS(\eta_a,\theta_b) \\ (\mathcal{H}(\eta,\theta))\Pi f & = & GP(<\ \theta_{f(a)}\ \mid a \in T >) \end{array}$$

Checking $\mathcal{H}(\eta,\theta) : \mathcal{H}(F,G) \stackrel{\bullet}{\longrightarrow} \mathcal{H}(F',G')$ is straightforward, and it can easily be verified (coordinatewise) that $\mathcal{H}$ preserves identities and composition.

□

30 lemma $\mathcal{H}$ is locally continuous

proof

Because the ordering on the hom-sets of $[T,\ \underline{CPO_\perp}]$ is defined coordinatewise, we can prove this coordinatewise.

Let $< (\eta^i,\theta^i) >_{i\in\mathbb{N}}$ be an ascending chain in $Hom_{[T,\underline{CPO_\perp}]^{OP}\times[T,\underline{CPO_\perp}]}((F,G),(F',G'))$, so $\eta^i : F' \stackrel{\bullet}{\longrightarrow} F$, $\theta^i : G \stackrel{\bullet}{\longrightarrow} G'$, $\eta^i \sqsubseteq \eta^{i+1}$ and $\theta^i \sqsubseteq \theta^{i+1}$.

We must prove

$$\bigsqcup \mathcal{H}(\eta^i,\theta^i) = \mathcal{H}(\bigsqcup \eta^i, \bigsqcup \theta^i)$$

which is equivalent to

$$\forall_{a\in Obj(T)} (\bigsqcup \mathcal{H}(\eta^i,\theta^i))_a = \mathcal{H}(\bigsqcup \eta^i_a, \bigsqcup \theta^i_a)$$

because lubs are take pointwise.

We distinguish three cases : a is a base type, a is a function type, and a is a polymorphic type.

For base types it is trivial:

$$(\bigsqcup \mathcal{H}(\eta^i,\theta^i))_a = id_{domain_a} = (\mathcal{H}(\bigsqcup \eta^i_a, \bigsqcup \theta^i))_a$$

For function types it follows from local continuity of FS, and for polymorphic types it follows from local continuity of GP:

$\to$-types :

$$\begin{array}{cl} & (\bigsqcup \mathcal{H}(\eta^i,\theta^i))_{a\to b} \\ = & \\ & \bigsqcup FS(\eta^i_a,\theta^i_b) \\ = & \\ & FS(\bigsqcup \eta^i_a, \bigsqcup \theta^i_b) \\ = & \\ & (\mathcal{H}(\bigsqcup \eta^i, \bigsqcup \theta^i))_{a\to b} \end{array}$$

Π-types :

$$\begin{array}{cl} & (\bigsqcup \mathcal{H}(\eta^i,\theta^i)_{\Pi f} \\ = & \\ & \bigsqcup GP(<\theta^i_{f(a)} \mid a \in Obj(T) >) \\ = & \\ & GP(< \bigsqcup \theta^i_{f(a)} \mid a \in Obj(T) >) \\ = & \\ & (\mathcal{H}(\bigsqcup \eta^i, \bigsqcup \theta^i))_{\Pi f} \end{array}$$

□

$[T,\ \underline{CPO_\perp}]$ is a localized O-category and $\mathcal{H}$ is locally continuous, so $\mathcal{H}_{PR} : ([T,\ \underline{CPO_\perp}]^{OP} \times [T,\ \underline{CPO_\perp}])_{PR} \to [T,\ \underline{CPO_\perp}]_{PR}$ is ω-continuous.

Let the functor $\mathcal{F}_1$ be the isomorphism from $[A,\ \underline{CPO_\perp}]_{PR}$ to $([A,\ \underline{CPO_\perp}]^{OP})_{PR}$, and let the functor $\mathcal{F}_2$ be the isomorphism from $([A,\ \underline{CPO_\perp}]^{OP})_{PR} \times [A,\ \underline{CPO_\perp}]_{PR}$ to $([A,\ \underline{CPO_\perp}]^{OP} \times [A,\ \underline{CPO_\perp}])_{PR}$, as defined in lemma's 17 and 18.

So the object part of $\mathcal{F}_1$ is defined by $\mathcal{F}_1 F = F$
and the morphism part by $\mathcal{F}_1(\eta,\theta) = (\theta,\eta)$

and the object part of $\mathcal{F}_2$ is defined by $\mathcal{F}_2(F,G) = (F,G)$
and the morphism part by $\mathcal{F}_2((\eta,\theta),(\phi,\psi)) = ((\eta,\phi),(\theta,\psi))$

31 definition $\Delta : [A, \underline{CPO_\perp}]_{PR} \to [A, \underline{CPO_\perp}]_{PR} \times [A, \underline{CPO_\perp}]_{PR}$

The object part of Δ is defined by $\quad \Delta F = (F, F)$
and the morphism part by $\quad \Delta(\eta, \theta) = ((\eta, \theta), (\eta, \theta))$

□

32 lemma $\mathcal{F} = \mathcal{H}_{PR} \circ \mathcal{F}_2 \circ (\mathcal{F}_1 \times I) \circ \Delta$

proof

$$
\begin{array}{ll}
 & (\mathcal{H}_{PR} \circ \mathcal{F}_2 \circ (\mathcal{F}_1 \times I) \circ \Delta) F \\
= & \{ \text{ definition } \Delta, \mathcal{F}_1, \mathcal{F}_2 \} \\
 & \mathcal{H}_{PR}(F, F) \\
= & \{ \text{ definition } PR \} \\
 & \mathcal{H}(F, F) \\
= & \{ \text{ definition } \mathcal{H}, \mathcal{F} \} \\
 & \mathcal{F} F
\end{array}
\qquad
\begin{array}{ll}
 & (\mathcal{H}_{PR} \circ \mathcal{F}_2 \circ (\mathcal{F}_1 \times I) \circ \Delta)(\eta, \theta) \\
= & \{ \text{ definition } \Delta, \mathcal{F}_1, \mathcal{F}_2 \} \\
 & \mathcal{H}_{PR}((\theta, \eta), (\eta, \theta)) \\
= & \{ \text{ definition } PR \} \\
 & (\mathcal{H}(\theta, \eta), \mathcal{H}(\eta, \theta)) \\
= & \{ \text{ definition } \mathcal{H}, \mathcal{F} \} \\
 & \mathcal{F}(\eta, \theta)
\end{array}
$$

and so $\mathcal{F} = \mathcal{H}_{PR} \circ \mathcal{F}_2 \circ (\mathcal{F}_1 \times I) \circ \Delta$

□

So

33 lemma $\mathcal{F}$ is ω-continuous

proof

$\mathcal{H}_{PR}$, $\mathcal{F}_2$, $\mathcal{F}_1$, I and Δ are ω-continuous, and hence so is $\mathcal{H}_{PR} \circ \mathcal{F}_2 \circ (\mathcal{F}_1 \times I) \circ \Delta$. So by lemma 32 $\mathcal{F}$ is ω-continuous.

□

6 Concluding remarks

It should be noted that the construction we have described is not limited to the particular set of types, subtype relation , O-category or functors that we gave in section 1.

For the functors, FS and GP in the our case, only the local continuity is essential. Instead of $\underline{CPO_\perp}$ other O-cateories can also be used, provided the conditions needed to apply corollary 26 are satisfied.

It is of course no coincidence that the same functor comes up in both the recursive domain equations for function types and the coherence condition for functions types, nor that the mixed contra/covariance of this functor exactly matches the mixed contra/covariance of the type constructor $\rightarrow$ with respect to the subtype relation.

Other type constructors, such as $\times$ (Cartesian product) , + (separated sum), $\otimes$ (smashed product) , $\oplus$ (coalesced sum) or $(\_)_\perp$ (lifting) can easily be included. All that is required is the corresponding (locally continuous) functor. In fact, FS represents the most difficult case, because it is contravariant in one argument. For example, product types of the form $a \times b$ can be made using the cartesian product functor $CP : \underline{CPO_\perp} \times \underline{CPO_\perp} \rightarrow \underline{CPO_\perp}$. The recursive domain equation for $\times$-types is

$$Dom_{a\times b} \;\; \cong \;\; CP(Dom_a, Dom_b)$$

and the coherence condition is

$$Coe_{a\times b\ a'\times b'} \;\; = \;\; \Phi_{a'\times b'} \circ CP(\ Coe_{a\ a'},\ Coe_{b\ b}) \circ \Phi_{a\times b}$$

Labelled products (records) and labelles sums (variants) (see [CW85]) can also be incorporated in the model, as well as the natural subtype relation on them.

Σ-types - also called existential types or weak sums (see [MP88]) can also be added, using the generalized sum functor (see [tEH89b]), as well as bounded Π- and Σ-types. The subtype relation on types can be extended accordingly. In [Pol91] the results described in this paper are also used to construct models for second order lambda calculus with recursive types and subtyping.

Acknowledgements

I would like to thank Huub the Eikelder and Jaap van der Woude for their comments on preliminary versions of this paper.

References

[BH88] R. Bos and C. Hemerik. An introduction to the category-theoretic solution of recursive domain equations. Technical Report 15, Eindhoven University of Technology, 1988.

[BMM90] Kim B. Bruce, Albert R. Meyer, and John C. Mitchell. The semantics of second-order lambda calculus. *Information and Computation*, 85:76–134, 1990.

[CW85] Luca Cardelli and Peter Wegner. On understanding types, data abstraction and polymorphism. *Computing Surveys*, 17(4):471–522, 1985.

[MP88] John C. Mitchell and Gordon D. Plotkin. Abstract types have existential type. *ACM Trans. on Prog. Lang. and Syst.*, 10(3):470–502, 1988.

[Pol91] Erik Poll. Cpo-models for second order lambda calculus with recursive types and subtyping. Computing Science Note 07, Eindhoven University of Technology, 1991.

[SP82] J.C. Smyth and G.D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11:761–783, 1982.

[tEH89a] H. ten Eikelder and C. Hemerik. The construction of a cpo model for second order lamba calculus with recursion. In *Procs. CSN'89 Computing Science in the Netherlands*, pages 131–148, 1989.

[tEH89b] H. ten Eikelder and C. Hemerik. Some category-theoretical properties related to a model for a polymorphic lambda calculus. Computing Science Note 03, Eindhoven University of Technology, 1989.

In this series appeared:

89/1	E.Zs.Lepoeter-Molnar	Reconstruction of a 3-D surface from its normal vectors.
89/2	R.H. Mak P.Struik	A systolic design for dynamic programming.
89/3	H.M.M. Ten Eikelder C. Hemerik	Some category theoretical properties related to a model for a polymorphic lambda-calculus.
89/4	J.Zwiers W.P. de Roever	Compositionality and modularity in process specification and design: A trace-state based approach.
89/5	Wei Chen T.Verhoeff J.T.Udding	Networks of Communicating Processes and their (De-)Composition.
89/6	T.Verhoeff	Characterizations of Delay-Insensitive Communication Protocols.
89/7	P.Struik	A systematic design of a parallel program for Dirichlet convolution.
89/8	E.H.L.Aarts A.E.Eiben K.M. van Hee	A general theory of genetic algorithms.
89/9	K.M. van Hee P.M.P. Rambags	Discrete event systems: Dynamic versus static topology.
89/10	S.Ramesh	A new efficient implementation of CSP with output guards.
89/11	S.Ramesh	Algebraic specification and implementation of infinite processes.
89/12	A.T.M.Aerts K.M. van Hee	A concise formal framework for data modeling.
89/13	A.T.M.Aerts K.M. van Hee M.W.H. Hesen	A program generator for simulated annealing problems.
89/14	H.C.Haesen	ELDA, data manipulatie taal.
89/15	J.S.C.P. van der Woude	Optimal segmentations.
89/16	A.T.M.Aerts K.M. van Hee	Towards a framework for comparing data models.
89/17	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra.

90/1	W.P.de Roever- H.Barringer- C.Courcoubetis-D.Gabbay R.Gerth-B.Jonsson-A.Pnueli M.Reed-J.Sifakis-J.Vytopil P.Wolper	Formal methods and tools for the development of distributed and real time systems, p. 17.
90/2	K.M. van Hee P.M.P. Rambags	Dynamic process creation in high-level Petri nets, pp. 19.
90/3	R. Gerth	Foundations of Compositional Program Refinement - safety properties - , p. 38.
90/4	A. Peeters	Decomposition of delay-insensitive circuits, p. 25.
90/5	J.A. Brzozowski J.C. Ebergen	On the delay-sensitivity of gate networks, p. 23.
90/6	A.J.J.M. Marcelis	Typed inference systems : a reference document, p. 17.
90/7	A.J.J.M. Marcelis	A logic for one-pass, one-attributed grammars, p. 14.
90/8	M.B. Josephs	Receptive Process Theory, p. 16.
90/9	A.T.M. Aerts P.M.E. De Bra K.M. van Hee	Combining the functional and the relational model, p. 15.
90/10	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra, p. 30. (Revised version of CSNotes 89/17).
90/11	P. America F.S. de Boer	A proof system for process creation, p. 84.
90/12	P.America F.S. de Boer	A proof theory for a sequential version of POOL, p. 110.
90/13	K.R. Apt F.S. de Boer E.R. Olderog	Proving termination of Parallel Programs, p. 7.
90/14	F.S. de Boer	A proof system for the language POOL, p. 70.
90/15	F.S. de Boer	Compositionality in the temporal logic of concurrent systems, p. 17.
90/16	F.S. de Boer C. Palamidessi	A fully abstract model for concurrent logic languages, p. p. 23.
90/17	F.S. de Boer C. Palamidessi	On the asynchronous nature of communication in logic languages: a fully abstract model based on sequences, p. 29.

90/18	J.Coenen E.v.d.Sluis E.v.d.Velden	Design and implementation aspects of remote procedure calls, p. 15.
90/19	M.M. de Brouwer P.A.C. Verkoulen	Two Case Studies in ExSpect, p. 24.
90/20	M.Rem	The Nature of Delay-Insensitive Computing, p.18.
90/21	K.M. van Hee P.A.C. Verkoulen	Data, Process and Behaviour Modelling in an integrated specification framework, p. 37.
91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.

91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Somê categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. .
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.