Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 218 (2008) 21–32
www.elsevier.com/locate/entcs

The Interpretation of Intuitionistic Type Theory in Locally Cartesian Closed Categories – an Intuitionistic Perspective

to Phil Scott on the occasion of his 60th birthday year

Alexandre Buisse[1]

Programming, Logic and Semantics Group
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 København S

Peter Dybjer[2]

Department of Computer Science and Engineering
Chalmers University of Technology
Rännvägen 6, S-41296 Göteborg

Abstract

We give an intuitionistic view of Seely's interpretation of Martin-Löf's intuitionistic type theory in locally cartesian closed categories. The idea is to use Martin-Löf type theory itself as metalanguage, and E-categories, the appropriate notion of categories when working in this metalanguage. As an E-categorical substitute for the formal system of Martin-Löf type theory we use E-categories with families (E-cwfs). These come in two flavours: groupoid-style E-cwfs and proof-irrelevant E-cwfs. We then analyze Seely's interpretation as consisting of three parts. The first part is purely categorical: the interpretation of groupoid-style E-cwfs in E-locally cartesian closed categories. (The key part of this interpretation has been type-checked in the Coq system.) The second is a coherence problem which relates groupoid-style E-cwfs with proof-irrelevant ones. The third is a purely syntactic problem: that proof-irrelevant E-cwfs are equivalent to traditional lambda calculus based formulations of Martin-Löf type theory.

Keywords: Martin-Löf intuitionistic type theory, locally cartesian closed category, E-category, Coq

1 Introduction

In this lecture we draw together two of Phil Scott's interests: the relationship between type theory and category theory on the one hand and constructive category

[1] Email: abui@itu.dk
[2] Email: peterd@cs.chalmers.se

doi:10.1016/j.entcs.2008.10.003

theory on the other. Let us begin with a quotation from the book *Introduction to higher order categorical logic* by Lambek and Scott [10]:

> We also claim that intuitionistic type theories and toposes are closely related, in as much as there is a pair of adjoint functors between their respective categories. This is worked out in Part II. The relationship between Martin-Löf type theories and locally cartesian closed categories was established too recently (by Robert Seely) to be treated here.

We shall here discuss Seely's interpretation from an intuitionistic perspective. Our key idea is to work in a constructive metalanguage. In fact we shall use Martin-Löf type theory itself as a metalanguage! We shall use the same mindset as in the paper *Normalization and the Yoneda Embedding* by Čubrić, Dybjer, and Scott [4]. In that paper we used the constructive notion of a *P-category* where each hom-set is equipped with a partial equivalence relation. We showed how the decision problem for equality in cartesian closed categories follows more or less directly from a constructive reading of a few well-known facts about presheaf categories including the Yoneda lemma. This provides a categorical and constructive alternative to the traditional solution, where equality in cartesian closed categories is decided by using the normalization and Church-Rosser properties of the simply typed lambda calculus.

Note also that Martin-Löf called his theory "intuitionistic type theory", although it is quite different from the "usual" intuitionistic type theory of Lambek and Scott which is an intuitionistic version of type theory in the tradition of Russell and Church. Unlike the usual type theory Martin-Löf type theory is a *programming language*. It is based on the Curry-Howard *identification* of propositions and types, and the notion of *dependent type* is primitive. When we talk about "intuitionistic type theory" in this paper we henceforth always mean Martin-Löf's intuitionistic type theory.

Seely's interpretation is described in the paper *Locally cartesian closed categories and type theory* [15]. His main result states that the following two categories are equivalent:

- the category of "Martin-Löf theories" with types $\prod_{x\in A} B[x], \sum_{x\in A} B[x]$, and $\mathrm{I}(a, b)$, where the rules for the identity type I are those of the *extensional* intuitionistic type theory of Martin-Löf [12,13].
- the category of locally cartesian closed categories.

Close scrutiny of Seely's proof however reveals some issues in need of further clarification. These issues were discussed by Curien in his paper *Substitution up to isomorphism* [5]. Curien proposes a way

> ... to solve a difficulty arising from a mismatch between syntax and semantics: in locally cartesian closed categories, substitution is modelled by pullbacks (more generally pseudo-functors), that is, only up to isomorphism, unless split fibrational hypotheses are imposed. ... but not all semantics do satify them, and in particular not the general description of the interpretation in an arbitrary locally cartesian

closed category. In the general case, we have to show that the isomorphisms between types arising from substitution are *coherent* in a sense familiar to category theorists.

Due to this coherence problem at the level of types, we are led to:

- switch to a syntax where substitutions are explicitly present (in traditional presentations substitution is a meta-operation, defined by induction);
- include type equality judgements in this modified syntax: we consider here only equalities describing the stepwise performance as substitution.

...

To our knowledge, the work presented here is the first solution to this problem, which, until very recently, had not even been clearly identified, mainly due to an emphasis on interesting mathematical models rather than on syntactic issues.

Curien proceeded to show that it is possible to interpret type equality as isomorphism in lcccs, and solve the coherence problem, for intuitionistic type theory with Π-types.

Somewhat later, Hofmann [7] showed how to construct a model of dependent type theory (category with attributes) with Π-types, Σ-types, and (extensional) identity types from a locally cartesian closed category by using a construction of Bénabou [2].

In this talk we shall revisit the Seely-Curien-Hofmann interpretation from an intuitionistic perspective. Seely, Curien and Hofmann of course all worked with the usual notion of category and with the usual classical (informal set-theoretic) metalanguage. We shall show how we get a new perspective on the problem if we work in an intuitionistic metalanguage, In particular, we shall explain why we are naturally led to interpret equality of types as isomorphism of objects in a category; why constructions of intuitionistic category theory nevertheless come with a choice (of pullbacks for example); and why the intuitionistic perspective helps us to understand the construction of a term models of intuitionistic type theory. Moreover, we point out that Seely's abstract fact about an equivalence of categories becomes a computer program. We write a "compiler" between two "programming languages": the language of intuitionistic type theory and the language of lcccs. Metamathematics has become metaprogramming!

Our approach will be based on the notion of an *E-category*. This is the standard notion of a category in the constructive sense. An E-category is just like a P-category, but hom-sets are equipped with equivalence relations rather than partial equivalence relations.

In the remainder of the paper we shall

- use intuitionistic type theory itself as metalanguage; in fact, we only need the very core of intuitionistic type theory, *the "logical framework"* with Π, Σ, and a universe;
- use the notion of an E-locally cartesian closed category (E-lccc);
- introduce the notion of an E-category with families (E-cwf) as a categorical substitute for the formal system of intuitionistic type theory;
- show two alternative definitions: groupoid-style and proof-irrelevant E-cwfs;

- outline the proof that *any E-lccc is a groupoid style E-cwf with* Π, Σ, *and* I*-types*
- show some code which suggests how this result is implemented in the Coq-system;
- introduce the coherence problem of relating groupoid-style and proof-irrelevant E-cwfs.

The key part of the proof that any E-lccc is a groupoid style E-cwf has been implemented in the Coq system by the first author. He has constructed the E-category of (groupoid style) E-families **EFam** and also shown how to construct an E-functor $T : \mathcal{C} \to \mathbf{EFam}$ whenever $\mathcal{C}$ has finite limits. We plan to present the details of the Coq-implementation in a forthcoming publication.

Acknowledgements

These notes are based on a lecture given by the second author in the Special Session to honour Phil Scott on the occasion of his 60th birthday year. It was held in conjunction with MFPS XXIV in Philadelphia in May 2008. The second author is grateful for the many things Phil Scott has taught him about the connections between logic, types, and categories, and for very enjoyable collaboration. He would also like to thank Rick Blute and Andre Scedrov for organizing the Special Session and for inviting him to contribute to it.

2 E-locally cartesian closed categories

In Martin-Löf type theory a *set* is the same as a *data type* in a programming language. However, many sets (the real numbers, the rationals, the carrier of the free group, etc) come equipped with a notion of "equality" which is not the intrinsic identity of objects of the data type. Since quotient formation is not a constructive operation on sets, constructive mathematicians instead work with sets which are equipped with an equivalence relation. We shall follow the terminology of Čubrić, Dybjer, and Scott [4] and call them "E-sets", although they are usually called "setoids" in the type-theoretic community. Martin-Löf has proposed to call them "extensional sets". Bishop simply called them "sets", and used the term "preset" for the underlying representing data type.

E-sets and E-functions

An *E-set* (setoid, Bishop set, extensional set) is a set with an equivalence relation. Type-theoretically, an equivalence relation is a quadruple: a relation together with proofs of reflexivity, transitivity, and symmetry. Hence an E-set is a quintuple, a set together with the four components of the equivalence relation. In Coq we use records to represent tuples and the type of E-sets is thus defined as follows, bearing in mind that "sets" in the sense of Martin-Löf are implemented as "types" in Coq.

```
Record ESet : Type := {
  carrier :> Type;
  eq      :  carrier → carrier → Type;
  refl    :  ∀x     : carrier, eq x x;
```

```
  trans   :  ∀x y z : carrier, eq x y → eq y z → eq x z;
  sym     :  ∀x y   : carrier, eq x y → eq y x
}.
Notation "x ≡ y" := (eq x y) (at level 70).
```

We would like to remark that this definition uses a form of universe polymorphism. The level of the two instances of `Type` is implicit and will be determined by the context in which the E-set is used. By choosing the levels appropriately we can both get a notion of "small" E-set and various levels of "large" E-sets. Note that the type of `ESet` must be one level higher in the universe hierarchy than the type of `carrier`.

Moroever, the sign `:>` signifies a coercion which allows us to use the same name for an E-set and its carrier set.

An *E-function* (setoid map, Bishop function, extensional function) preserves the equivalence relation:

```
Record EFun (A B : ESet) : Type := {
  func :> A → B;
  pres :  ∀x y : A, x ≡ y → func x ≡ func y
}.
```

E-categories

As already mentioned, the constructive notion of category has E-homsets. However, we do not include a notion of equality of objects as part of the structure of an E-category. As category-theorists often point out, the moral notion of equality of objects is isomorphism; we do not need another distinct, primitive notion of equality of objects. Our Coq implementation is as follows:

```
Record ECat : Type := {
  ob    :  Type;
  hom   :> ob → ob → ESet;
  id    :  ∀A:ob, hom A A;
  comp  :  ∀A B C:ob, hom B C ⇒ hom A B ⇒ hom A C;
  idL   :  ∀(A B:ob) (f:hom A B), comp _ _ _ (id _) f ≡ f;
  idR   :  ∀(A B:ob) (f:hom A B), comp _ _ _ f (id _) ≡ f;
  assoc :  ∀(A B C D:ob) (f:hom C D) (g:hom B C) (h:hom A B),
           comp _ _ _ (comp _ _ _ f g) h ≡
           comp _ _ _ f (comp _ _ _ g h)
}.
```

The ⇒-notation expresses that composition is a binary E-function on E-homsets.

We have used a coercion which allows us to use the notation $\mathcal{C}$ `A B` for the E-set of arrows between `A` and `B` in the E-category $\mathcal{C}$.

As for the definition of E-set, the definition of E-category can be instantiated to yield various notions of small and large category.

E-pullbacks

It is quite straightforard to formalize the basic E-categorical notions, see Huet and Saibi [9]. For example, the constructive notion of E-pullback is implemented by the following Coq code which states that an E-pullback is a function which maps a triple of objects `A, B, C` and pair of arrows `f, g` (with appropriate sources and target) to a quintuple consisting of the object `D` (the apex), the projection arrows `h, k`, and the proofs `sq` and `un` (of the commutativity of the pullback square and of the universal property, respectively):

```
Record EPullback (𝒞 : ECat) (A B C : ob 𝒞)
(f : 𝒞 B A) (g : 𝒞 C A) : Type := {
    D  : ob 𝒞;
    h  : 𝒞 D B;
    k  : 𝒞 D C;
    sq : f ∘ h ≡ g ∘ k;
    un : ...
}.
```

Since it is a constructive function an E-pullback always comes with a computable choice. We essentially have an instance of the type-theoretic axiom of choice, which expresses (in Coq-notation) how to construct a choice function `f` from a set `A` to an `A`-indexed family of sets `B`:

```
(∀ x : A, ∃ y : B x, C x y) →
(∃ f : (∀ x : A, B x), ∀ x : A, C x (f x))
```

The validity of this axiom is a direct consequence of the constructive meaning of the logical constants following the Brouwer-Heyting-Kolmogorov (and Curry-Howard-Martin-Löf) interpretation.

Note however that the "extensional" axiom of choice [14] is not valid. If `A` and `B` are E-sets, there is no reason why the choice function `f` should preserve the equivalence relation. But does the E-pullback come with an extensional choice? This is only a meaningful question if we equip the set of objects of the category with an equivalence relation. If this is isomorphism, then the answer is yes, E-pullbacks map equal arrows to isomorphic objects.

E-locally cartesian closed categories

We can now define the notion of an E-locally cartesian closed category as an E-category $\mathcal{C}$ such that all E-slice categories $\mathcal{C}/A$ are E-cccs for all objects A. The objects of the E-slice category $\mathcal{C}/A$ are arrows of $\mathcal{C}$ with target A. The arrows of $\mathcal{C}/A$ are commuting triangles, formalized type-theoretically as pairs of arrows and proofs that the triangle commutes.

Note that since there is no primitive notion of equality of objects in $\mathcal{C}/A$, the equality of arrows in $\mathcal{C}$ is not passed on to these objects. However, we can prove that equal arrows of $\mathcal{C}$ become isomorphic objects of $\mathcal{C}/A$.

Since it is straightfoward to define the notion of E-cartesian closure, we can define the notion of E-lccc as follows:

```
Record ELCCC : Type  := {
    𝒞  :> ECat;
    ccc : ∀ A : ob 𝒞, ECCC (𝒞/A)
 }.
```

3 E-categories with families

What is Martin-Löf's intuitionistic type theory?

Having completed the E-categorical definition of lcccs, we now ask ourself how to formalize Martin-Löf type theory. And since Martin-Löf type theory is also our metalanguage, the question actually is how to formalize it in itself!

Before we address this question we need to ask ourselves exactly where to find a precise definition of Martin-Löf type theory. Looking through the literature it becomes apparent that it is not clear that there is a canonical definition. When writing down the syntax and inference rules for intuitionistic type theory, we have to make some choices. Should we use typed lambda calculus a la Church or a la Curry? Is the rule of substitution primitive or derived? Is the substitution operation explicit (a constructor of syntax) or implicit (an operation on the metalevel)? How are variables represented, with names or de Bruijn indices or de Bruijn levels? Are universes formulated a la Russell or a la Tarski? Etc. Of course, we believe that there is a number of equivalent formulations, but it may not be so easy to prove this rigorously. And how do we make sure that we do not forget any inference rules? The lack of a canonical definition is somewhat disturbing.

We here propose to use an abstract algebraic characterization of intuitionistic type theory as the initial category with families (cwfs) with extra structure [6,8,3,1]. This is a notion defined up to isomorphism. Cwfs provide the "minimal algebraization" of intuitionistic type theory: substitution is made explicit and variables are replaced by projections. However, dependent types are not modelled by fibrations as in lcccs and many other categorical notions of model of dependent types.

Note that cwfs are similar to indexed categories. However, cwfs match the syntactic structure of dependent type theory better, whereas indexed categories are closer to the syntactic structure of predicate logic.

Categories with families (cwfs)

A category with families consists of

- C, a *category of contexts*. Its objects are called *contexts* and its morphisms are called *substitutions*.
- $T : C^{op} \to \mathbf{Fam}$, a functor where the
  object part maps a context Γ to the family of sets of *terms* $\{a \mid \Gamma \vdash a : A\}$ indexed by the set of *types* $\{A \mid \Gamma \vdash A\ type\}$ in Γ.
  arrow part maps a substitution γ to a pair of functions which perform substitution of γ in types and terms respectively. We write $A[\gamma]$ for *substitution* of γ in a type A and $a[\gamma]$ for *substitution* of γ in the term a.

- A *terminal object* [ ] of C called the *empty context*. The unique arrow into [ ] is the *empty substitution*.
- A *context comprehension* operation which to an object Γ of C and a type A in Γ associates four components
  context extension: an object $\Gamma; A$ of C;
  weakening: a morphism $\mathrm{p}_{\Gamma,A} : \Gamma; A \to \Gamma$ of C - the *first projection*
  assumption: a term $\mathrm{q}_{\Gamma,A} \in \Gamma; A \vdash A[\mathrm{p}_{\Gamma,A}]$ - the *second projection*
  substitution extension: for each object Δ in C, morphism $\gamma : \Delta \to \Gamma$, and term $a \in \Delta \vdash A[\gamma]$, there is a unique morphism $\theta = \langle \gamma, a \rangle : \Delta \to \Gamma; A$, such that $\mathrm{p}_{\Gamma,A} \circ \theta = \gamma$ and $\mathrm{q}_{\Gamma,A}[\theta] = a$. This is the *universal property* of context comprehension.

Context comprehension in categories with families is similar to Lawvere's comprehension schema in hyperdoctrines [11].

E-cwfs and the E-category of E-families

It is clear how to understand the above definition of cwf if we base it on the usual (set-theoretic) notions of category, functor, etc. But how are cwfs understood constructively as "E-cwfs"? It should consist of

- an E-category $\mathcal{C}$.
- an E-functor $T : \mathcal{C} \to \mathbf{EFam}$.
- an E-terminal object.
- an E-context comprehension.

The crucial question is how to define the E-category of E-families **EFam**, since the other E-categorical notions are clear. It turns out that there are two interesting alternatives: *groupoid style* E-families and *proof-irrelevant* E-families.

The first alternative uses the analogy between E-sets and *groupoids*. As already mentioned an E-set is a quintuple consisting of a set, a relation, and proofs of reflexivity, transitivity, and symmetry. If we equate all proofs we get a groupoid, where the carrier becomes the set of objects, the proofs that two objects are related become arrows, the proofs of reflexivity become an identity arrows, transitivity proofs become composition arrows, and symmetry proofs become inverse arrows in a groupoid.

Hence an E-set indexed E-family should be analalogous to a groupoid-indexed family of groupoids. These are isomorphism-preserving functors from a groupoid $\mathcal{A}$ to the category of groupoids:

$$B : \mathcal{A} \to \mathbf{Groupoid}$$

We write down the resulting definition in ordinary mathematical notation, since the Coq-code is somewhat lengthy.

If A is an E-set, then an *A-indexed family of E-sets* consists of

- a family B of E-sets indexed by the carrier set of A;
- a *reindexing* map $\iota(p) : B(x') \to B(x)$ whenever $p : x \equiv_A x'$.

such that

- $\iota(\mathrm{refl}) \equiv_{ext} \mathrm{id}$ (the identity map);
- $\iota(\mathrm{trans}(p, p')) \equiv_{ext} \iota(p) \circ \iota(p')$ (composition of maps);
- $\iota(p)$ is an E-bijection with inverse $\iota(\mathrm{sym}(p))$.

Here, $\equiv_{ext}$ refers to extensional equality of E-functions, that is, functions which map equivalent elements of the domain to equivalent elements of the codomain.

Let B be an A-indexed family of E-sets and let B' be an A'-indexed family with reindexings ι and ι', respectively. A *morphism* between these two families consists of

- an E-function $f : A \to A'$;
- an A-indexed family of E-functions $g(x) : B(x) \to B'(f(x))$ for $x : A$.
- which is natural in x:

$$\begin{array}{ccc} B(x') & \overset{g(x')}{\longrightarrow} & B'(f(x')) \\ {\scriptstyle \iota(p)}\downarrow & & \downarrow{\scriptstyle \iota'(f(p))} \\ B(x) & \underset{g(x)}{\longrightarrow} & B'(f(x)) \end{array}$$

whenever $p : x \equiv_A x'$.

There is an obvious definition of equivalence of morphisms of E-families.

The first author has implement the E-category `EFam` in Coq, but we have to postpone showing the details of this implementation to a forthcoming publication. Given this definition and definitions of E-functors, E-terminal objects, and E-context comprehension, the code for E-cwfs can be given as the following Coq-record:

```
Record ECwf : Type  := {
    C :> ECat;
    T  : EFunctor C EFam;
    te : ETerminal C;
    cc : EContextComprehension C T
 }.
```

Cwfs only capture the most basic structure of dependent types, but it is easy to add extra structure for interpreting Π-types, Σ-types, and I-types [6,1]:

```
Record ECwfPiI : Type  := {
    C   :> ECwf;
    pi    : EPi C;
    sigma : ESigma C
    i     : EI C
 }.
```

We have ended up with an iterated record structure. For example, the first component of an E-cwf is an E-category which itself is a record, and the second component of an E-category, the family of E-homsets, is a binary record-valued function.

E-cwfs as a flat record

Our iterated record structure can however be flattened (in the sense of functional programming). If we reorder and rename the components we see that this flattened record bears a strong similarity with the structure of the inference rules for the judgements of Martin-Löf type theory. The first seven components of the flattened record codify the seven forms of judgement of a substitution calculus in the style of Martin-Löf:

```
Record FlatECwf =  := {
   Ctxt : Type;
   Hom  : Ctxt -> Ctxt -> Type;
   EHom : ∀ G D : Ctxt, Hom G D → Hom G D → Type;
   Ty   : Ctxt -> Type;
   ETy  : ∀ G : Ctxt, Ty G → Ty G → Type;
   Tm   : Ctxt -> Ty -> Type;
   ETm  : ∀ G : Ctxt, ∀ A : Ty G, Tm G A → Tm G A → Type;
   ...
   inference rules
   ...
 }.
```

Note that there is no judgement for equality of contexts, since our category of context does not have an equality of objects.

The remaining components of the flattened record correspond to the inference rules of a substitution calculus for dependent types. We only give one example of an inference rule: the type equality rule (conversion rule). It comes from the reindexing map of E-families:

```
 iota : ∀ G : Ctxt, ∀ A A' : Ty G,
        ETy G A A' -> Tm G A' -> Tm G A
```

E-cwfs as a flat record resembles Curien's [5] explicit substitution calculus for dependent types with explicit witnesses of type equalities. We can view it is a systematic reconstruction of Curien's syntactic calculus, where we have relied on E-categorical structures.

4 Seely's interpretation, intuitionistically

E-cwfs from E-categories with finite limits

We can now prove an E-categorical version of Seely's theorem. As in Seely [15] we get the E-cwf structure (with Π, Σ, and extensional identity types) from an E-lccc $\mathcal{C}$ in the following way:

- The base E-category is $\mathcal{C}$.
- A type in a context Γ is an object of the slice E-category $\mathcal{C}/\Gamma$. Equality of types is *isomorphism* in the slice E-category.
- A term of type A in context Γ is a section of A. Equality of terms is inherited from equality of arrows in the base E-category.
- Substitution in types is obtained from the E-pullback construction. We can here verify the laws of groupoid-style E-cwfs.
- Etc, essentially following Seely, but with explicit treatment of inference rules of (flattened) E-cwfs relating to the interpretation of type equalities as isomorphisms in $\mathcal{C}$.

We have implemented the key part in Coq, the construction of the groupoid style E-category `EFam` and the E-functor `T : EFunctor` $\mathcal{C}$ `EFam`. To prove this result, and more generally to construct the groupoid style E-cwf structure it suffices for $\mathcal{C}$ to be an E-category with finite limits. The details of this implementation are planned to appear in a forthcoming publication.

The coherence problem

Have we now finished our constructive version of Seely's theorem? No, although we have argued that the flattened version of the E-cwf record has a close correspondence to Curien's calculus of explicit substitution we need to relate this to the "usual" syntax. The usual syntax however corresponds to *proof irrelevant* E-cwfs, the second alternative mentioned above. This is because proofs of type-equalities do not matter in the usual inference system.

We define an E-cwf to be proof irrelevant iff the following principle holds.

```
coh : ∀ G : Ctxt, ∀ A A' : Ty G,
      ∀ p p' : ETy G A A', ∀ a : Tm G A,
      ETm (iota G A A' p a) (iota G A A' p' a)
```

The coherence problem is to relate groupoid style E-cwfs and proof irrelevant ones. Curien solved a similar coherence problem by a process of cut-elimination. We expect that it is possible to provide an E-categorical version of Curien's proof, but have to leave this as a conjecture for future work. It is not clear to us whether Hofmann's use of the Bénabou construction [7] can be transfered to our constructive setting.

References

[1] A. Abel, T. Coquand, and P. Dybjer. On the algebraic foundation of proof assistants for intuitionistic type theory. In *FLOPS*, pages 3–13, 2008.

[2] J. Bénabou. Fibred categories and the foundation of naive category theory. *Journal of Symbolic Logic*, 50:10–37, 1985.

[3] A. Buisse and P. Dybjer. Towards formalizing categorical models of type theory in type theory. *Electr. Notes Theor. Comput. Sci.*, 196:137–151, 2008.

[4] D. Čubrić, P. Dybjer, and P. Scott. Normalization and the Yoneda embedding. *Mathematical Structures in Computer Science*, 8:153–192, 1998.

[5] P.-L. Curien. Substitution up to isomorphism. *Fundamenta Informaticae*, 19(1,2):51–86, 1993.

[6] P. Dybjer. Internal type theory. In *TYPES '95, Types for Proofs and Programs*, number 1158 in Lecture Notes in Computer Science, pages 120–134. Springer, 1996.

[7] M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In L. Pacholski and J. Tiuryn, editors, *CSL*, volume 933 of *Lecture Notes in Computer Science*. Springer, 1994.

[8] M. Hofmann. Syntax and semantics of dependent types. In A. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*. Cambridge University Press, 1996.

[9] G. Huet and A. Saibi. Constructive category theory. In *Proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory, Göteborg*, January 1995.

[10] J. Lambek and P. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

[11] F. W. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. In A. Heller, editor, *Applications of Categorical Algebra, Proceedings of Symposia in Pure Mathematics*. AMS, 1970.

[12] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science, VI, 1979*, pages 153–175. North-Holland, 1982.

[13] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[14] P. Martin-Löf. 100 years of Zermelo's axiom of choice: what was the problem with it? *The Computer Journal*, 49(3):343–350, 2006.

[15] R. Seely. Locally cartesian closed categories and type theory. *Math. Proc. Camb. Phil. Soc.*, 95(33), 1984.