Internal Type Theory

Peter Dybjer

Department of Computing Science
Chalmers University of Technology
Göteborg, Sweden

Abstract. We introduce *categories with families* as a new notion of model for a basic framework of dependent types. This notion is close to ordinary syntax and yet has a clean categorical description. We also present categories with families as a generalized algebraic theory. Then we define categories with families formally in Martin-Löf's intensional intuitionistic type theory. Finally, we discuss the *coherence problem* for these *internal categories with families*.

1 Introduction

In a previous paper [8] I introduced a general notion of *simultaneous inductive-recursive definition* in intuitionistic type theory. This notion subsumes various reflection principles and seems to pave the way for a natural development of what could be called "internal type theory", that is, the construction of models of (fragments of) type theory in type theory, and more generally, the formalization of the metatheory of type theory in type theory.

The present paper is a first investigation of such an internal type theory. We introduce *categories with families* to model a basic framework of dependent types and show how to formalize them in intensional intuitionistic type theory.

One goal is to represent Hofmann's setoid model of type theory [11, 13] in type theory. He also used a categorical notion of model of dependent types (Cartmell's categories with attributes) but worked in ordinary set-theoretic metalanguage. The setoid model of type theory can be viewed as a formalization of the standard "intuitive" model of type theory. It also justifies the rules of extensional type theory [16] and certain rules for quotient formation.

Our categorical approach to internal type theory can be contrasted to the syntactic approach by Pollack [19]. He formalized the syntax of dependent type theory in type theory and proved syntactic properties such as Church-Rosser.

The plan of the paper is as follows. In section 2 we introduce categores with families and their formalization as a generalized algebraic theory. In section 3 we show how to formalize some basic categorical notions in type theory, and then use these to define categories with families in type theory. We first give an abstract presentation and then show how to derive a system of inference rules. In section 4 we state a coherence problem for internal categories with families and propose how to solve it by constructing a model of normal forms.

2 Categories with Families

2.1 Basic Definitions

Categories with families (cwfs) are variants of Cartmell's *categories with attributes* [12, 18]. The point of the reformulation is to get a more direct link to the syntax of dependent types. In particular we avoid reference to pullbacks, which give rise to a conditional equation when formalized in a straightforward way. Cwfs can therefore directly be formalized as a generalized algebraic theory with clear similiarities to Martin-Löf's *substitution calculus* for type theory [17].

Let Fam be the category of families of sets. An *object* is a family of sets $(B(x))_{x \in A}$ and a *morphism* with source $(B(x))_{x \in A}$ and target $(B'(x'))_{x' \in A'}$ is a pair consisting of a function $f : A \to A'$ and a family of functions $g(x) : B(x) \to B'(f(x))$ indexed by $x \in A$.

The components of a cwf are named after the corresponding syntactic notions.

Definition 1. A *category with families* consists of the following four parts:

- A base *category* C. Its objects are called *contexts* and its morphisms are called *substitutions*.
- A *functor* $T : C^{op} \to Fam$. We write $T(\Gamma) = (\Gamma \vdash A)_{A \in Type(\Gamma)}$, where Γ is an object of C, and call it the family of *terms* indexed by *types* in context Γ. Moreover, if γ is a morphism of C then the two components of $T(\gamma)$ interpret substitution in types and terms respectively. We write $A[\gamma]$ for the application of the first component to a type A and $a[\gamma]$ for the application of the second component to a term a.
- A *terminal object* $[]$ of C called the *empty context*.
- A *context comprehension* operation which to an object Γ of C and a type $A \in Type(\Gamma)$ associates an object $\Gamma; A$ of C; a morphism $p : \Gamma; A \to \Gamma$ of C (the *first projection*); and a term $q \in \Gamma; A \vdash A[p]$ (the *second projection*). The following universal property holds: 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 $p \circ \theta = \gamma$ and $q[\theta] = a$.

Throughout the paper we shall freely use "polymorphic" notation for improving readability. Without this convention we should for example have included indices for the projections and written $p_{\Gamma,A}$ and $q_{\Gamma,A}$ instead of just p and q.

A basic example of a cwf is obtained by letting C be the category of sets, $Type(\Gamma)$ be the set of Γ-indexed small sets, and

$$\begin{aligned}
\Gamma \vdash A &= \prod_{\gamma \in \Gamma} A(\gamma) \\
A[\delta](\gamma) &= A(\delta(\gamma)) \\
a[\delta](\gamma) &= a(\delta(\gamma)) \\
[] &= 1 \\
\Gamma; A &= \sum_{\gamma \in \Gamma} A(\gamma)
\end{aligned}$$

Given a cwf we can recover the structure of a category with attributes. For example, the diagram

$$\begin{tikzcd}
\Delta; A[\gamma] \arrow[r, "\langle \gamma \circ p, q \rangle"] \arrow[d, "p"'] & \Gamma; A \arrow[d, "p"] \\
\Delta \arrow[r, "\gamma"'] & \Gamma
\end{tikzcd}$$

is a pullback, and terms $a \in \Gamma \vdash A$ are in one-to-one correspondence with sections $\langle id, a \rangle : \Gamma \to \Gamma; A$ of first projections $p : \Gamma; A \to \Gamma$.

Conversely, given a category with attributes we can recover the structure of a cwf. The terms can be recovered from sections of projections and substitution can be recovered from pullbacks.

Definition 2. Let (C, T) denote a cwf with base category C and functor T. A *morphism of cwfs* with source (C, T) and target (C', T') is a pair (F, σ), where $F : C \to C'$ is a functor and $\sigma : T \to T'F$ is a natural transformation, such that terminal object and context comprehension are preserved on the nose.

Definition 3. A pair of morphisms of cwfs

$$\begin{tikzcd}
(C, T) \arrow[r, shift left, "(F, \sigma)"] & (C', T') \arrow[l, shift left, "(G, \tau)"]
\end{tikzcd}$$

form an *equivalence of cwfs* iff

$$\begin{tikzcd}
C \arrow[r, shift left, "F"] & C' \arrow[l, shift left, "G"]
\end{tikzcd}$$

form an equivalence of categories with unit η and counit ϵ, such that

$$\begin{tikzcd}
T \arrow[r, "\sigma"] \arrow[d, "T\eta"'] & T'F \arrow[dl, "\tau F"] \\
TGF &
\end{tikzcd}
\qquad
\begin{tikzcd}
TG \arrow[dr, "\sigma G"'] & T' \arrow[l, "\tau"'] \arrow[d, "T'\epsilon^{-1}"] \\
& T'FG
\end{tikzcd}$$

commute.

Remark. There are alternative notions of cwf and morphism of cwfs which avoid reference to chosen structure and where preservation is up to isomorphism.

2.2 The Generalized Algebraic Theory of Categories with Families

The next step is to formalize cwfs as a *generalized algebraic theory* in the sense of Cartmell [5]. Generalized algebraic theories generalize many-sorted algebraic theories and are based on a framework of dependent types. They have four parts:

- A list of *sort symbols* with dependent typings.
- A list of *operator symbols* with dependent typings.
- A list of *equations* between well-formed *sort expressions* (not needed here).
- A list of *equations* between well-formed *terms*.

Cwfs make precise what it means to be a model of a generalized algebraic theory. Expressing the notion of a cwf as a generalized algebraic theory can hence be seen as a syntactic reflection process and a step towards internal type theory.

We present sort and operator symbols by their typing rules. As before, we use polymorphic notation and write $\delta \circ \gamma$ instead of the proper $\delta \circ_{\Theta,\Delta,\Gamma} \gamma$, etc.

Rules for the Category *C*

Sort symbols:

$$Context : Sort$$

$$\frac{\Delta, \Gamma : Context}{\Delta \to \Gamma : Sort}$$

Operator symbols:

$$\frac{\Theta, \Delta, \Gamma : Context \qquad \gamma : \Delta \to \Gamma \qquad \delta : \Theta \to \Delta}{\gamma \circ \delta : \Theta \to \Gamma}$$

$$\frac{\Gamma : Context}{id : \Gamma \to \Gamma}$$

Equations:

$$\begin{aligned}(\gamma \circ \delta) \circ \theta &= \gamma \circ (\delta \circ \theta)\\ id \circ \gamma &= \gamma\\ \gamma \circ id &= \gamma\end{aligned}$$

Rules for the Functor *T*

Sort symbols:

$$\frac{\Gamma : Context}{Type(\Gamma) : Sort}$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{\Gamma \vdash A : Sort}$$

Operator symbols:

$$\frac{\Delta, \Gamma : Context \qquad A : Type(\Gamma) \qquad \gamma : \Delta \to \Gamma}{A[\gamma] : Type(\Delta)}$$

$$\frac{\Delta, \Gamma : Context \quad A : Type(\Gamma) \quad a : \Gamma \vdash A \quad \gamma : \Delta \to \Gamma}{a[\gamma] : \Delta \vdash A[\gamma]}$$

Equations:

$$\begin{aligned} A[\gamma \circ \delta] &= A[\gamma][\delta] \\ A[id] &= A \\ a[\gamma \circ \delta] &= a[\gamma][\delta] \\ a[id] &= a \end{aligned}$$

Rules for the Terminal Object

Operator symbols:

$$[] : Context$$

$$\frac{\Gamma : Context}{<> : \Gamma \to []}$$

Equations

$$\begin{aligned} <> \circ \gamma &= <> \\ id_{[]} &= <> \end{aligned}$$

Rules for Context Comprehension

Operator symbols:

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{\Gamma; A : Context}$$

$$\frac{\Delta, \Gamma : Context \quad A : Type(\Gamma) \quad \gamma : \Delta \to \Gamma \quad a : \Delta \vdash A[\gamma]}{<\gamma, a> : \Delta \to \Gamma; A}$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{p : \Gamma; A \to \Gamma}$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{q : \Gamma; A \vdash A[p]}$$

Equations:

$$\begin{aligned} p \circ <\gamma, a> &= \gamma \\ q[<\gamma, a>] &= a \\ <\delta, a> \circ \gamma &= <\delta \circ \gamma, a[\gamma]> \\ id_{\Gamma;A} &= <p, q> \end{aligned}$$

This completes the description of the generalized algebraic theory of cwfs. Note the correspondence between sort symbols and judgement forms and between operator symbols and inference rules of a substitution calculus for dependent types. But note also that there are no sort symbols which correspond to equality judgements and no operator symbols which correspond to general equality rules. Instead general equality reasoning is inherited from the metalanguage.

Reflecting the Rules for an Arbitrary Generalized Algebraic Theory

The generalized algebraic theory of cwfs is a categorical formulation of the basic framework of dependent types. It corresponds to those rules of a standard presentation which deal with formation of contexts, with substitution, and with assumption. These rules underly both generalized algebraic theories and intuitionistic type theory.

We can reflect the rules of an arbitrary generalized algebraic theory as an extension of the generalized algebraic theory of cwfs. For example, the rules of a generalized algebraic theory which introduces a sort of natural numbers and the sort of proofs of equality of natural numbers

$$N : sort$$

$$\frac{m, n : N}{I(m, n) : sort}$$

are reflected by the rules

$$\frac{\Gamma : Context}{N : Type(\Gamma)}$$

$$\frac{\Gamma : Context \qquad m, n : \Gamma \vdash N}{I(m, n) : Type(\Gamma)}$$

We could even reflect the generalized algebraic theory of cwfs itself, and add the following operator symbols:

$$\frac{\Gamma : Context}{\text{‘}Context\text{’}(\Gamma) : Type(\Gamma)}$$

$$\frac{\Gamma : Context \qquad \text{‘}\Delta\text{’}, \text{‘}\Gamma\text{’} : \text{‘}Context\text{’}(\Gamma)}{\text{‘}\Delta\text{’}\,\text{‘}\!\to\text{’}_{\Gamma}\,\text{‘}\Gamma\text{’} : Type(\Gamma)}$$

etc. Note the difference between the sort *Context* of contexts and the operator symbol '*Context*' of reflected contexts.

Such a game of reflection is quite pointless, since ultimately we have to understand what it means to be a model of a generalized algebraic theory in terms of, say, classical set theory anyway. The main point of the paper (see section 3) is to show an alternative explanation in terms of intuitionistic type theory.

Rules for the Cartesian Product of a Family of Types

We can also extend our generalized algebraic theory with various operator symbols and equations corresponding to the inference rules of intuitionistic type theory. As an example we give the rules for Π:

Operator symbols:

$$\frac{\Gamma : Context \qquad A : Type(\Gamma) \qquad B : Type(\Gamma; A)}{\Pi(A, B) : Type(\Gamma)}$$

$$\frac{\Gamma : Context \quad A : Type(\Gamma) \quad B : Type(\Gamma; A) \quad b : \Gamma; A \vdash B}{\lambda(b) : \Gamma \vdash \Pi(A, B)}$$

$$\frac{\Gamma : Context \quad A : Type(\Gamma) \quad B : Type(\Gamma; A) \quad c : \Gamma \vdash \Pi(A, B) \quad a : \Gamma \vdash A}{app(c, a) : \Gamma \vdash B[<id, a>]}$$

Equations:

$$\begin{aligned}
\Pi(A, B)[\gamma] &= \Pi(A[\gamma], B[<\gamma \circ p, q>]) \\
\lambda(b)[\gamma] &= \lambda(b[<\gamma \circ p, q>]) \\
app(c, a)[\gamma] &= app(c[\gamma], a[\gamma]) \\
app(\lambda(b), a) &= b[<id, a>] \\
\lambda(app(c[p], q)) &= c
\end{aligned}$$

The three first of the five equations represent the laws for substitution under Π, λ, and *app*. The fourth and fifth represent β and η-conversion respectively.

It is straightforward to formalize the other rules of type theory too.

3 Internal Categories with Families

3.1 Formalizing some Basic Categorical Notions in Type Theory

The definition of a cwf refers in particular to the notions of category, functor, and the category *Fam*. Therefore we shall first formalize these notions in intuitionistic type theory, and then use them to get a notion of cwf in intuitionistic type theory – an *internal cwf*.

But before doing this we shall briefly discuss the simpler problem of how to define ordinary algebraic notions, such as monoids, in type theory.

Remark. Throughout the rest of the paper we work in intuitionistic type theory and standard mathematical terms, such as category, functor, etc., will henceforth refer to a notion in type theory, unless stated otherwise.

Monoids in Type Theory. A monoid in the ordinary sense consists of a set M, a binary composition function $\circ$ on M and an element id, such that composition is associative and id is an identity with respect to composition. A naive way to interpret this in type theory would be to say that M is a set in the sense of type theory and $\circ$ is a function in the sense of type theory, that is, an algorithm. Furthermore, we could require that associativity and identity laws are valid as intensional identities (I-sets).

But this yields a too restrictive notion. For example, one way to define a free monoid is to quotient the set of binary trees (S-expressions). But since quotienting is not an operation on sets in type theory this construction would then not yield a monoid in type theory.

So instead we let the carrier of a monoid be a *setoid* $\mathcal{M}$, that is, a set M with an equivalence relation $\sim_M$, and $\circ$ be a *(setoid-)map*, that is, a function between the underlying sets which respects the equivalence relations. Furthermore associativity and identity laws have to be valid with respect to $\sim_M$.

Formally, a setoid is a quintuple $\mathcal{A} = \langle A, \sim_A, \mathit{ref}_A, \mathit{trans}_A, \mathit{sym}_A \rangle$. When we use a calligraphic letter to stand for a setoid, the corresponding italic letter will stand for its carrier, and *ref*, *trans*, *sym* (with an italic letter as a subscript) stand for the proofs of reflexivity, transitivity, and symmetry, respectively.

Categories and Functors in Type Theory. We follow Aczel [1] and Huet and Saibi [15] and define a *category* to have a *set* of objects, but hom-*setoids*. We shall not need to refer to equality of objects. The object part of a functor is a function between the object sets and the morphism part is a family of maps between the hom-setoids, such that the functor laws are satisfied with respect to the equivalence relation in the hom-setoid.

Setoids and maps under extensional equality $\sim_{ext}$ form a category in type theory which plays the role of the category of sets in ordinary category theory.

For the description of an implementation (in Coq) of category theory along these lines we refer to Huet and Saibi [15].

Setoid-Indexed Families of Setoids in Type Theory.

Definition 4. Let $\mathcal{A}$ be a setoid. An $\mathcal{A}$-*indexed family of setoids* consists of

- a family $\mathcal{B}$ of setoids indexed by the set A;
- a *reindexing* map $\iota(P) : \mathcal{B}(x') \to \mathcal{B}(x)$ whenever $P : x \sim_A x'$.

This family is *coherent* provided

- $\iota(\mathit{ref}_A) \sim_{ext} id$ (the identity map);
- $\iota(\mathit{trans}_A(P, P')) \sim_{ext} \iota(P) \circ \iota(P')$ (composition of maps);
- $\iota(P)$ is an isomorphism with inverse $\iota(\mathit{sym}_A(P))$ in the category of setoids.

Definition 5. Let $\mathcal{B}$ be an $\mathcal{A}$-indexed family of setoids and let $\mathcal{B}'$ be an $\mathcal{A}'$-indexed family with reindexings ι and ι', respectively. A *morphism* between these two families consists of

– a map $f : \mathcal{A} \to \mathcal{A}'$;
– an A-indexed family of maps $g(x) : \mathcal{B}(x) \to \mathcal{B}'(f(x))$ for $x : \mathcal{A}$.

Such a morphism is *coherent* if the following diagram commutes in the category of setoids:

$$\begin{tikzcd}
\mathcal{B}(x') \arrow[r, "g(x')"] \arrow[d, "\iota(P)"'] & \mathcal{B}'(f(x')) \arrow[d, "\iota'(f(P))"] \\
\mathcal{B}(x) \arrow[r, "g(x)"'] & \mathcal{B}'(f(x))
\end{tikzcd}$$

whenever $P : x \sim_A x'$.

Two *morphisms* (f, g) and (f', g') are *equivalent* iff $P : f \sim_{ext} f'$ and

$$\begin{tikzcd}
\mathcal{B}(x) \arrow[r, "g(x)"] \arrow[dr, "g'(x)"'] & \mathcal{B}'(f(x)) \\
 & \mathcal{B}'(f'(x)) \arrow[u, "\iota'(P(x))"']
\end{tikzcd}$$

commutes in the category of setoids.

It is useful to note that the type-theoretic notion of a setoid is related to the category-theoretic notion of a *groupoid*. The underlying set of a setoid corresponds to the set of objects of the groupoid and the sets of equivalence proofs correspond to the homsets of a groupoid. The explicit proofs of reflexivity, transitivity, and symmetry in a setoid correspond to identity, composition, and inverse in a groupoid. Moreover, a setoid-indexed family of setoids is similar to a groupoid-indexed family of groupoids defined as a functor and the coherence conditions correspond to functoriality. Similarly, for morphisms, where the coherence condition corresponds to naturality.

Internal Cwfs. We are now ready to reinterpret Definition 1 (section 2.1) of cwfs using our type-theoretic definition of the categorical notions. For the purpose of defining internal cwfs we require that objects of Fam are coherent setoid-indexed families of setoids and that morphisms of Fam are coherent morphisms up to equivalence of morphisms as defined in the previous section.

3.2 Formal Rules for Internal Categories with Families

We present the formal rules for internal cwfs in a similar style as the generalized algebraic theory of cwfs in section 2.2. All formal rules are derived systematically from the abstract presentation of internal cwfs as outlined in the previous section.

The fundamental novelty is that among these rules there are general rules for equality reasoning, which have no counterpart among the sort and operator symbols of the generalized algebraic theory of cwfs.

So in addition to the four set constructors corresponding to the four sort symbols:

$$\mathit{Context} : \mathit{Set}$$

$$\frac{\Delta, \Gamma : \mathit{Context}}{\Delta \to \Gamma : \mathit{Set}}$$

$$\frac{\Gamma : \mathit{Context}}{\mathit{Type}(\Gamma) : \mathit{Set}}$$

$$\frac{\Gamma : \mathit{Context} \qquad A : \mathit{Type}(\Gamma)}{\Gamma \vdash A : \mathit{Set}}$$

there are three set constructors corresponding to equality judgements:

$$\frac{\Delta, \Gamma : \mathit{Context} \qquad \gamma, \gamma' : \Delta \to \Gamma}{\Delta \to \gamma \sim \gamma' \in \Gamma : \mathit{Set}}$$

$$\frac{\Gamma : \mathit{Context} \qquad A, A' : \mathit{Type}(\Gamma)}{\Gamma \vdash A \sim A' : \mathit{Set}}$$

$$\frac{\Gamma : \mathit{Context} \qquad A : \mathit{Type}(\Gamma) \qquad a, a' : \Gamma \vdash A}{\Gamma \vdash a \sim a' \in A : \mathit{Set}}$$

There is no set constructor for context equality, since our base category has a *set* and not a setoid of objects (contexts). We note the similarity to Martin-Löf's substitution calculus [17] which (unlike Ehrhard's [10], Curien's [7], and Ritter's [20]) lacks a judgement for context equality.

The element constructors in the definition can be divided into three kinds:

- Those which correspond to operator symbols of the generalized algebraic theory of cwfs, such as the rules for composition, identity, and substitution.
- General rules for equality reasoning.
- Those which correspond to the equations of the generalized algebraic theory of cwfs. These rules need to be modified to take into account explicit reasoning about equality.

We do not have space to display all these rules and shall therefore limit our discussion to the most interesting of the second and the third kind.

Among the general rules for equality there are the rules of reflexivity, transitivity, and symmetry for the three forms of equality judgement. There are also congruence rules for each operation. Of particular interest is the rule of type equality, which in the traditional formulation of type theory is written

$$\frac{\Gamma : \mathit{Context} \quad A, A' : \mathit{Type}(\Gamma) \quad \Gamma \vdash A = A' \quad \Gamma \vdash a : A'}{\Gamma \vdash a : A},$$

If we apply our definition of setoid-indexed family of setoids to the family of terms indexed by types, the reindexing function gives rise to the following rule:

$$\frac{\Gamma : Context \quad A, A' : Type(\Gamma) \quad P : \Gamma \vdash A \sim A' \quad a : \Gamma \vdash A'}{\iota(P, a) : \Gamma \vdash A}$$

Note the similarity with the typing rule in Curien's *explicit syntax* [7].

We also have equality rules for ι coming from the coherence conditions for objects of Fam:

$$\frac{\Gamma : Context \qquad A : Type(\Gamma) \qquad a : \Gamma \vdash A}{\iota ref : \Gamma \vdash \iota(ref, a) \sim a \in A}$$

$$\frac{\Gamma : Context \ A, A', A'' : Type(\Gamma) \ P : \Gamma \vdash A \sim A' \ P' : \Gamma \vdash A' \sim A'' \ a : \Gamma \vdash A''}{\iota trans : \Gamma \vdash \iota(trans(P, P'), a) \sim \iota(P, \iota(P', a)) \in A}$$

and also equalities expressing that $\iota(sym(P))$ is the inverse of $\iota(P)$.

Another rule comes from the coherence condition for morphisms of Fam and states that substitution in terms commutes with applications of type equality:

$$\frac{\Delta, \Gamma : Context \quad A, A' : Type(\Gamma) \quad P : \Gamma \vdash A \sim A' \quad a : \Gamma \vdash A' \quad \gamma : \Delta \to \Gamma}{\iota sub : \Delta \vdash \iota(P[\gamma], a[\gamma]) \sim \iota(P, a)[\gamma] \in A[\gamma]}$$

The third kinds of rules correspond to the equations of the generalized algebraic theory of cwfs. But to show the well-typedness of some of these rules we need to appeal to the type-equality law. For example, the two equations

$$A[id] = A$$
$$a[id] = a$$

are replaced by

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{Subid : \Gamma \vdash A[id] \sim A}$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma) \qquad a : \Gamma \vdash A}{subid : \Gamma \vdash a[id] \sim \iota(Subid, a) \in A[id]}$$

Note the explicit dependence on the proof $Subid$ in the second rule.

3.3 Examples of Internal Categories with Families

The Internal Cwf of Setoids. Hofmann [13] showed that the category of setoids has finite limits (and much more!). This result is unproblematic to represent in type theory. Therefore we can also construct an internal cwf of setoids as follows. Let C be an arbitrary category with finite limits. It gives rise to an internal cwf, where the category of contexts is C itself, where types in context Γ are objects of the slice category C/Γ, and where terms are sections. Substitution in types and terms is interpreted using the pullback functor. Moreover, proofs of type equality are interpreted as isomorphisms in the slice category. Reindexing is interpreted by composition with this isomorphism so that the coherence conditions follow from the laws of associativity and identity, etc.

The Internal Cwf $\mathcal{D}$ of Derivations. The simplest way to get a syntactic cwf is to turn the definition of an internal cwf in the previous section into a big mutual inductive definition of the seven families of sets corresponding to the seven forms of judgements in a substitution calculus. (To get a non-trivial cwf we should also include some basic type constructors.)

The elements of these sets are best thought of as *derivations* in a substitution calculus. Alternatively, they can be viewed as terms in explicit syntax [7].

Note also that the definition of a cwf in section 2.1 (or its presentation as a generalized algebraic theory in section 2.2) does not directly show how to generate free such categories. In contrast the definition of an internal cwf yields a straightforward construction of free internal cwfs.

An Internal Cwf from Ordinary Syntax of Dependent Types. One can also define a free cwf from a standard formulation of a substitution calculus based on *raw expressions*. Objects of the base category are pairs of raw contexts and derivations of "correct context" judgements, etc. The detailed proof that this indeed yields an internal cwf is non-trivial.

An Internal Cwf $\mathcal{N}$ of Normal Derivations. The basic idea is that a normal derivation of a term is one which is built up by the rules of thinning and of assumption. If we think of derivations as terms in an explicit syntax, then we note that such derivations correspond to de Bruijn numbers: zero comes from assuming the last variable and successor corresponds to the rule of thinning. Moreover, a normal substitution is a sequence of normal terms.

Explicitly, we have the following inductive clauses for generating normal derivations:

$$[\,] : Context$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{\Gamma ; A : Context}$$

$$\frac{\Gamma : Context}{\texttt{<>} : \Gamma \to [\,]}$$

$$\frac{\Delta, \Gamma : Context \quad A : Type(\Gamma) \quad \gamma : \Delta \to \Gamma \quad a : \Delta \vdash A[\gamma]}{\texttt{<}\gamma, a\texttt{>} : \Delta \to \Gamma ; A}$$

$$\frac{\Gamma : Context \qquad A : Type(\Gamma)}{0 : \Gamma ; A \vdash A[p]}$$

$$\frac{\Gamma : Context \qquad A, B : Type(\Gamma) \qquad a : \Gamma \vdash B}{s(a) : \Gamma ; A \vdash B[p]}$$

(To get a non-trivial cwf we should again include some basic type constructors.)

Note that the operations $\_[\_]$ and p appear in the inductive clauses and hence we have a *simultaneous inductive-recursive definition* [8] of the sets of normal derivations and the cwf-operations on them.

We can prove that we get an internal cwf $\mathcal{N}$ by interpreting equality of types, terms, and substitutions as I-equality. This result has been implemented in ALF, a proof checker for intensional intuitionistic type theory [2].

4 The Coherence Problem for Internal Cwfs

Categorical interpretations of type theory can be divided into those which interpret type equality as isomorphism, such as Seely's lccc-interpretation [21], and those which interpret it as true equality, such as Cartmell's category with attributes interpretation. In either case a coherence problem arises. In order to interpret syntax in an lccc we have to make sure that two different derivations of the same judgement have the same interpretation. A proof of this was given by Curien [7]. On the other hand interpreting syntax in categories with attributes is relatively straightforward [13]. But here a coherence problem arises when one already has an lccc and wants to construct a category with attributes. For this purpose Hofmann [12] adapted a method due to Bénabou [3] for constructing a split fibration from an arbitrary fibration.

There is also a coherence problem for internal type theory. Proofs of type equality appear in terms and it is sometimes essential to know that the term does not depend on this particular proof. Formally:

Conjecture 1 *Coherence: if* $P, P' : \Gamma \vdash A' \sim A$ *and* $a : \Gamma \vdash A$ *then* $\Gamma \vdash \iota(P, a) \sim \iota(P', a) \in A'$ *in an internal cwf.*

(Note that we do not want to refer to equality between equality proofs in an internal cwf, so we cannot simply ask whether P and P' are "equal".)

This coherence proposition can be proved as a corollary to the following:

Conjecture 2 *Normalization: there is an equivalence of internal cwfs:*

$$\begin{tikzcd} \mathcal{D} \arrow[r, shift left, "{(N,\nu)}"] & \mathcal{N} \arrow[l, shift left, "{(I,i)}"] \end{tikzcd}$$

The upper arrow is a normalizing *cwf-morphism and the lower an* inclusion of normal forms*.*

The two crucial properties of normalization are (i) that two convertible terms have identical normal forms and (ii) that a term is convertible to its normal form [6]. In our case property (i) is a consequence of the fact that equality in $\mathcal{N}$ is the basic I-equality in type theory. Property (ii) is a consequence of the equivalence of $\mathcal{D}$ and $\mathcal{N}$ and can be expressed as follows. Let

$$\begin{aligned} \widehat{\Gamma} &= (IN)(\Gamma) \\ \widehat{\gamma} &= (IN)(\gamma) \\ \widehat{A} &= (i \circ \nu)(A) \\ \widehat{a} &= (i \circ \nu)(a) \end{aligned}$$

be the normal forms of Γ, γ, A, a, respectively, in $\mathcal{D}$. Then there is an isomorphism

$$\eta_\Gamma : \Gamma \to \widehat{\Gamma}$$

which is natural

$$\Delta \to \gamma \sim \eta_\Gamma^{-1} \circ \widehat{\gamma} \circ \eta_\Delta \in \Gamma$$

and for each $A : Type(\Gamma)$, there is a proof

$$E_A : \Gamma \vdash A \sim \widehat{A}[\eta_\Gamma]$$

and for each $a : \Gamma \vdash A$,

$$\Gamma \vdash a \sim \iota(E_A, \widehat{a})[\eta_\Gamma] \in A$$

To prove that coherence follows from normalization we instantiate a to $\iota(P, a)$ and $\iota(P', a)$ respectively in the last equation. We get

$$\Gamma \vdash \iota(P, a) \sim \iota(E_A, \iota(\widehat{P, a}))[\eta_\Gamma] \in A'$$
$$\Gamma \vdash \iota(P', a) \sim \iota(E_A, \iota(\widehat{P', a}))[\eta_\Gamma] \in A'$$

Since $\iota(\widehat{P, a})$ and $\iota(\widehat{P', a})$ are I-equal the coherence proposition follows.

Unfortunately, we have only been able to give an informal sketch of the normalization proof. This is not satisfactory, since this is a proof in intuitionistic type theory which involves the manipulation of large terms in explicit syntax and it is difficult to carry out these manipulations safely by hand. As mentioned above we did manage to completely formalize the internal cwfs $\mathcal{D}$ and $\mathcal{N}$ in the proof assistant ALF, but we failed to prove their equivalence. In fact, the construction of $\mathcal{N}$ was a major undertaking which took the present version of ALF to the limits of its capability with very slow responses. We believe that the equivalence proof should be feasible with an improved proof assistant. Alternatively, a different construction of $\mathcal{N}$ might make the proof more manageable.

5 Related Work

The present paper is a revised version of a paper that appears in the proceedings of the *Joint CLICS-TYPES Workshop on Categories and Type Theory, Göteborg, January 1995* [9]. Much useful information on cwfs can also be found in the lecture notes on "Syntax and Semantics of Dependent Types" by Martin Hofmann [14]. He uses ordinary set-theoretic cwfs as the central semantic notion and gives several examples. He also discusses the relationship with other categorical notions of model for dependent types and gives a detailed proof of the equivalence of cwfs and categories with attributes.

Cwfs have also been used in recent unpublished work on "Tarski Semantics for Type Theory" by Per Martin-Löf (lecture at the meeting *Twenty-Five Years of Constructive Type Theory", Venice, October, 1995*).

The reader is also referred to the paper by Beylin and Dybjer [4] which shows how related phenomena appear in another proof of coherence in type theory.

Acknowledgements. The author is grateful for support from the ESPRIT BRA's TYPES and CLICS-II, from TFR (the Swedish Technical Research Council), and from The Isaac Newton Institute for Mathematical Sciences.

References

1. P. Aczel. Galois: a theory development project. A report on work in progress for the Turin meeting on the Representation of Logical Frameworks, 1993.
2. T. Altenkirch, V. Gaspes, B. Nordström, and B. von Sydow. A user's guide to ALF. Draft, January 1994.
3. J. Bénabou. Fibred categories and the foundation of naive category theory. *Journal of Symbolic Logic*, 50:10–37, 1985.
4. I. Beylin and P. Dybjer. Extracting a proof of coherence for monoidal categories from a proof of normalization for monoids. This volume.
5. J. Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.
6. T. Coquand and P. Dybjer. Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science*, 1996. To appear.
7. P.-L. Curien. Substitution up to isomorphism. *Fundamenta Informaticae*, 19(1,2):51–86, 1993.
8. P. Dybjer. Universes and a general notion of simultaneous inductive-recursive definition in type theory. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, 1992.
9. P. Dybjer and R. Pollack, editors. *Informal Proceedings of the CLICS-TYPES Workshop on Categories and Type Theory*, Programming Methodology Group, Göteborg University and Chalmers University of Technology, Report 85, 1995.
10. T. Ehrhard. *Une sémantique catégorique des types dépendents: Applications au Calcul des Constructions.* PhD thesis, Université Paris VII, 1988.
11. M. Hofmann. Elimination of extensionality and quotient types in Martin-Löf's type theory. In *Types for Proofs and Programs, International Workshop TYPES'93, LNCS 806*, 1994.
12. M. Hofmann. Interpretation of type theory in locally cartesian closed categories. In *Proceedings of CSL.* Springer LNCS, 1994.
13. M. Hofmann. *Extensional concepts in intensional type theory.* PhD thesis, University of Edinburgh, 1995.
14. M. Hofmann. Syntax and semantics of dependent types. In A. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation.* Cambridge University Press, 1996. To appear.
15. G. Huet and A. Saibi. Constructive category theory. In *Proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory, Göteborg*, January 1995.
16. P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science, VI, 1979*, pages 153–175. North-Holland, 1982.
17. P. Martin-Löf. Substitution calculus. Notes from a lecture given in Göteborg, November 1992.
18. A. M. Pitts. Categorical logic. In *Handbook of Logic in Computer Science.* Oxford University Press, 1997. Draft version of article to appear.
19. R. Pollack. *The Theory of Lego A Proof Checker for the Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, 1994.
20. E. Ritter. *Categorical Abstract Machines for Higher-Order Typed Lambda Calculi.* PhD thesis, Trinity College, Cambridge, September 1992.
21. R. A. G. Seely. Locally cartesian closed categories and type theory. *Proceedings of the Cambridge Philosophical Society*, 95:33–48, 1984.