Variants of categories with families and their formalization (work in progress)

Erik Palmgren
Stockholm University
Department of Mathematics

Type theory and formalization of mathematics
Workshop, Göteborg, December 11, 2014

Overview of this talk

(I) General goal and some obstacles

(II) Categories with families - some variants

(III) Formalizing models of type theory in type theory

I. General goal and some obstacles

A longstanding goal: A constructive model theory of depedent type theory, more specifically Martin-Löf style type theories (MLTT).

- usable model-theoretic methods for obtaining effective results about theories and their relations, or to interpret richer type theories constructively.
- implement syntax and models in type theory, to be able to formally check correctness, facilitate reasoning and extract algorithms.

I. General goal and some obstacles

A longstanding goal: A constructive model theory of depedent type theory, more specifically Martin-Löf style type theories (MLTT).

- usable model-theoretic methods for obtaining effective results about theories and their relations, or to interpret richer type theories constructively.
- implement syntax and models in type theory, to be able to formally check correctness, facilitate reasoning and extract algorithms.

Obstacles:

- usual constructivity or predicativity issues.
- MLTT is a priori not as flexible as set theory as a meta-theory, and introduces coherence problems in the sense of category theory (Dybjer 1995). (What can MLTT + Univalence Axiom do to ameliorate this?)

Two kinds of models

Kind (α): Interpretations of A=MLTT + Ax in B=MLTT*: one judgements in A are translated to possible several judgements, possibly of different kinds, in B, preferable so that definitional equalities in A are translated to definitional equalities in B.

Kind (β): The model of A is a structure in B. In general MLTT* has to be a stronger theory than MLTT to succeed (by Gödel's theorem).

II. Categories with families

One standard framework for interpreting dependent type theory is a category with attributes (Cartmell) or equivalently category with families (Dybjer).

II. Categories with families

One standard framework for interpreting dependent type theory is a category with attributes (Cartmell) or equivalently category with families (Dybjer).

- This modeling usually takes place in set theory.

II. Categories with families

One standard framework for interpreting dependent type theory is a category with attributes (Cartmell) or equivalently category with families (Dybjer).

- This modeling usually takes place in set theory.
- However, it is of interest to do the modeling in type theory itself, e.g. for the purpose of formal verification, and for foundational reasons. (Dybjer, *Internal Type Theory*, 1995 and onwards).

Definition

1. A category with families (cwf) consists of the data
   (a) A category $\mathcal{C}$ with a terminal object 1.
   This is the called the category of contexts and substitutions.

Definition

1. A category with families (cwf) consists of the data
   (a) A category $\mathcal{C}$ with a terminal object 1.
   This is the called the category of contexts and substitutions.
   (b) A functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$.
   This functor is intended to assign to each context Γ a set $\mathrm{Ty}(\Gamma)$ of types in the context and tells how substitutions act on these types. For $f : B \longrightarrow \Gamma$ and $A \in \mathrm{Ty}(\Gamma)$ we write

$$A\{f\} \text{ for } \mathrm{Ty}(f)(A).$$

Definition

1. A category with families (cwf) consists of the data

(a) A category $\mathcal{C}$ with a terminal object 1.
This is the called the category of contexts and substitutions.

(b) A functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$.
This functor is intended to assign to each context Γ a set $\mathrm{Ty}(\Gamma)$ of types in the context and tells how substitutions act on these types. For $f : B \longrightarrow \Gamma$ and $A \in \mathrm{Ty}(\Gamma)$ we write

$$A\{f\} \text{ for } \mathrm{Ty}(f)(A).$$

(c) For each $A \in \mathrm{Ty}(\Gamma)$, an object $\Gamma.A$ in $\mathcal{C}$ and a morphism

$$\mathsf{p}(A) = \mathsf{p}_\Gamma(A) : \Gamma.A \longrightarrow \Gamma \text{ in } \mathcal{C}.$$

This tells that each context can be extended by a type in the context, and that there is a projection from the extended context to the original one.

(d) For each $A \in \mathrm{Ty}(\Gamma)$, there is a set $\mathrm{Tm}(\Gamma, A)$ — thought of as the terms of type A. For $f : \Delta \longrightarrow \Gamma$ there is a function $\mathrm{Tm}(f) : \mathrm{Tm}(\Gamma, A) \to \mathrm{Tm}(\Delta, A\{f\})$, where we write $a\{f\}$ for $\mathrm{Tm}(f)(a)$. It should satisfy the following

- $a\{1_\Gamma\} = a$ for $a \in \mathrm{Tm}(\Gamma, A)$
- $a\{f \circ g\} = a\{f\}\{g\}$ for $a \in \mathrm{Tm}(\Gamma, A)$

(d) For each $A \in \mathrm{Ty}(\Gamma)$, there is a set $\mathrm{Tm}(\Gamma, A)$ — thought of as the terms of type A. For $f : \Delta \longrightarrow \Gamma$ there is a function $\mathrm{Tm}(f) : \mathrm{Tm}(\Gamma, A) \to \mathrm{Tm}(\Delta, A\{f\})$, where we write $a\{f\}$ for $\mathrm{Tm}(f)(a)$. It should satisfy the following
  - $a\{1_\Gamma\} = a$ for $a \in \mathrm{Tm}(\Gamma, A)$
  - $a\{f \circ g\} = a\{f\}\{g\}$ for $a \in \mathrm{Tm}(\Gamma, A)$

(e) For each $A \in \mathrm{Ty}(\Delta)$ there is an element $\mathrm{v}_A = \mathrm{v}_{\Gamma,A} \in \mathrm{Tm}(\Delta.A, A\{\mathrm{p}(A)\})$.

(d) For each $A \in \mathrm{Ty}(\Gamma)$, there is a set $\mathrm{Tm}(\Gamma, A)$ — thought of as the terms of type A. For $f : \Delta \longrightarrow \Gamma$ there is a function $\mathrm{Tm}(f) : \mathrm{Tm}(\Gamma, A) \to \mathrm{Tm}(\Delta, A\{f\})$, where we write $a\{f\}$ for $\mathrm{Tm}(f)(a)$. It should satisfy the following
- $a\{1_\Gamma\} = a$ for $a \in \mathrm{Tm}(\Gamma, A)$
- $a\{f \circ g\} = a\{f\}\{g\}$ for $a \in \mathrm{Tm}(\Gamma, A)$

(e) For each $A \in \mathrm{Ty}(\Delta)$ there is an element $\mathrm{v}_A = \mathrm{v}_{\Gamma,A} \in \mathrm{Tm}(\Delta.A, A\{\mathrm{p}(A)\})$.

(f) For any morphism $f : \Gamma \longrightarrow \Delta$ and $a \in \mathrm{Tm}(\Gamma, A\{f\})$, there is

$$\langle f, a\rangle_A : \Gamma \longrightarrow \Delta.A.$$

This construction should satisfy
- $\mathrm{p}(A) \circ \langle f, a\rangle_A = f$,
- $\mathrm{v}_A\{\langle f, a\rangle_A\} = a$,
- $\langle \mathrm{p}(A) \circ h, \mathrm{v}_A\{h\}\rangle_A = h$ for any $h : \Gamma \longrightarrow \Delta.A$.

and moreover for any $g : \Theta \longrightarrow \Gamma$,

$$\langle f, a\rangle_A \circ g = \langle f \circ g, a\{g\}\rangle_A \tag{1}$$

$\square$

Cwfs with named variables (projections)

Let V be an infinite discrete set. The set V is considered as the stock of variables. Let V^d denote the set of finite sequences

$$\mathbf{x} = x_1, x_2, \ldots, x_n = ((\cdots(((), x_1), x_2), \cdots), x_n)$$

from V, where $x_i \neq x_j$ whenever $i \neq j$.

Definition A *category with families and named variables (cwfn)* consists of the following data

(a1) [as before] A category $\mathcal{C}$ with a terminal object $\top$.

(a2) A function $\mathrm{n} : \mathrm{Ob}(\mathcal{C}) \longrightarrow V^d$ assigning each context a list of distinct variables (intended to be the names of projections). It is required that $\mathrm{n}(\top) = ()$, the empty list. Define the fresh variables for Γ:

$$\mathrm{Fr}(\Gamma) = \{x \in V : x \text{ is not in } \mathrm{n}(\Gamma)\}.$$

(b) [as before] Ty is a functor $\mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$.

(c) For each $A \in \mathrm{Ty}(\Gamma)$, and each $x \in \mathrm{Fr}(\Gamma)$, an object $\mathrm{Ext}(\Gamma, x, A)$ in $\mathcal{C}$, — briefly written $\Gamma.(x : A)$ — and a morphism $\mathrm{p}(x : A) = \mathrm{p}_\Gamma(x : A) : \Gamma.(x : A) \longrightarrow \Gamma$. It is required that

$$\mathrm{n}(\mathrm{Ext}(\Gamma, x, A)) = (\mathrm{n}(\Gamma), x).$$

(b) [as before] Ty is a functor $\mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$.

(c) For each $A \in \mathrm{Ty}(\Gamma)$, and each $x \in \mathrm{Fr}(\Gamma)$, an object $\mathrm{Ext}(\Gamma, x, A)$ in $\mathcal{C}$, — briefly written $\Gamma.(x : A)$ — and a morphism $\mathrm{p}(x : A) = \mathrm{p}_\Gamma(x : A) : \Gamma.(x : A) \longrightarrow \Gamma$. It is required that

$$\mathrm{n}(\mathrm{Ext}(\Gamma, x, A)) = (\mathrm{n}(\Gamma), x).$$

(d) [as before] For each $A \in \mathrm{Ty}(\Gamma)$, there is a set $\mathrm{Tm}(\Gamma, A)$. It should be such that for $f : \Delta \longrightarrow \Gamma$ there is a function $\mathrm{Tm}(f) : \mathrm{Tm}(\Gamma, A) \to \mathrm{Tm}(\Delta, A\{f\})$, where we write $a\{f\}$ for $\mathrm{Tm}(f)(a)$. It should satisfy the following

- $a\{1_\Gamma\} = a$ for $a \in \mathrm{Tm}(\Gamma, A)$
- $a\{f \circ g\} = a\{f\}\{g\}$ for $a \in \mathrm{Tm}(\Gamma, A)$

(e) For each $A \in \mathrm{Ty}(\Delta)$, and each $x \in \mathrm{Fr}(\Delta)$, there is an element

$$\mathrm{v}_{(x:A)} = \mathrm{v}_{\Delta,(x:A)} \in \mathrm{Tm}(\Delta.(x : A), A\{\mathrm{p}(x : A)\}).$$

(e) For each $A \in \mathrm{Ty}(\Delta)$, and each $x \in \mathrm{Fr}(\Delta)$, there is an element

$$\mathrm{v}_{(x:A)} = \mathrm{v}_{\Delta,(x:A)} \in \mathrm{Tm}(\Delta.(x:A), A\{\mathrm{p}(x:A)\}).$$

(f) For any morphism $f : \Gamma \longrightarrow \Delta$, $A \in \mathrm{Ty}(\Delta)$, and each $x \in \mathrm{Fr}(\Delta)$, and $a \in \mathrm{Tm}(\Gamma, A\{f\})$, there is

$$\langle f, a\rangle_{(x:A)} : \Gamma \longrightarrow \Delta.(x:A).$$

We use the suggestive notation $\langle f, x := a\rangle_A$ for this morphism. It should satisfy

- $\mathrm{p}(x:A) \circ \langle f, x := a\rangle_A = f$,
- $\mathrm{v}_{(x:A)}\{\langle f, x := a\rangle_A\} = a$,
- $\langle \mathrm{p}(x:A) \circ h, x := \mathrm{v}_{(x:A)}\{h\}\rangle_A = h$ for any $h : \Gamma \longrightarrow \Delta.(x:A)$.

and moreover for any $g : \Theta \longrightarrow \Gamma$,

$$\langle f, x := a\rangle_A \circ g = \langle f \circ g, x := a\{g\}\rangle_A \qquad (2)$$

□

We define for $f : \Theta \longrightarrow \Gamma$, $A \in \mathrm{Ty}(\Gamma)$, $y \in \mathrm{Fr}(\Theta)$ and $x \in \mathrm{Fr}(\Gamma)$, a morphism $\mathrm{q}(f, A, x, y) : \Theta.(y : A\{f\}) \longrightarrow \Gamma.(x : A)$ by

$$\mathrm{q}(f, A, x, y) =_{\mathrm{def}} \langle f \circ \mathrm{p}(y : A\{f\}), x := \mathrm{v}_{(y:A\{f\})}\rangle_A.$$

The following is a pullback square, satisfying well-known functoriality conditions in the parameter f:

$$\begin{tikzcd}
\Theta.(y : A\{f\}) \arrow[r, "\mathrm{q}(f,A,x,y)"] \arrow[d, "\mathrm{p}(y:A\{f\})"'] & \Gamma.(x : A) \arrow[d, "\mathrm{p}(x:A)"] \\
\Theta \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

Π-type construction in a cwfn

A cwfn *supports* Π*-types* if for $A \in \mathrm{Ty}(\Gamma)$, $x \in \mathrm{Fr}(\Gamma)$, and $B \in \mathrm{Ty}(\Gamma.(x : A))$ there is a type $\Pi_\Gamma(x, A, B) \in \mathrm{Ty}(\Gamma)$, and moreover for every $b \in \mathrm{Tm}(\Gamma.(x : A), B)$ there is an element $\lambda_{\Gamma,A,B}(x, b) \in \mathrm{Tm}(\Gamma, \Pi_\Gamma(x, A, B))$, and furthermore for any $c \in \mathrm{Tm}(\Gamma, \Pi_\Gamma(x, A, B))$ and any $a \in \mathrm{Tm}(\Gamma, A)$ there is an element $\mathrm{App}_{\Gamma,x,A,B}(c, a) \in \mathrm{Tm}(\Gamma, B\{\langle 1_\Gamma, x := a\rangle_A\})$, such that the following equations hold for any $f : \Theta \longrightarrow \Gamma$:

(β-conv) $\mathrm{App}_{\Gamma,A,B}(\lambda_{\Gamma,A,B}(x, b), a) = b\{\langle 1_\Gamma, x := a\rangle_A\}$,

(Π-subst) $\Pi_\Gamma(x, A, B)\{f\} = \Pi_\Theta(y, A\{f\}, B\{\mathrm{q}(f, A, x, y)\})$, provided $y \in \mathrm{Fr}(\Theta)$

(λ-subst) $\lambda_{\Gamma,A,B}(x, b)\{f\} = \lambda_{\Theta,A\{f\},B\{\mathrm{q}(f,A,x,y)\}}(y, b\{\mathrm{q}(f, A, x, y)\})$, provided $y \in \mathrm{Fr}(\Theta)$

pp-subst) $\mathrm{App}_{\Gamma,x,A,B}(c, a)\{f\} = \mathrm{App}_{\Theta,y,A\{f\},B\{\mathrm{q}(f,A,x,y)\}}(c\{f\}, a\{f\})$, provided $y \in \mathrm{Fr}(\Theta)$.

Analyzing the syntax

With the notations

$$\begin{aligned}\Pi(A,(x)B) &= \Pi(x,A,B)\\ \lambda((x)b) &= \lambda(x,b)\end{aligned}$$

we can rewrite the term formers as the rules

- $$\frac{\Gamma : \mathrm{Ctx} \quad x : \mathrm{Fr}(\Gamma) \quad A : \mathrm{Ty}(\Gamma) \quad B : \mathrm{Ty}(\Gamma.(x : A))}{\Pi(A,(x)B) : \mathrm{Ty}(\Gamma)}$$
- $$\frac{\Gamma : \mathrm{Ctx}, x : \mathrm{Fr}(\Gamma) \quad A : \mathrm{Ty}(\Gamma) \quad B : \mathrm{Ty}(\Gamma.(x : A)) \quad b : \mathrm{Tm}(\Gamma.(x : A),}{\lambda((x)b) : \mathrm{Tm}(\Gamma, \Pi(A,(x)B))}$$
- $\Gamma : \mathrm{Ctx}, x : \mathrm{Fr}(\Gamma), A : \mathrm{Ty}(\Gamma), B : \mathrm{Ty}(\Gamma.(x : A)), c :$ $\mathrm{Tm}(\Gamma, \Pi(A,(x)B)), a : \mathrm{Tm}(\Gamma, A)$ / $\mathrm{App}(c, a) :$ $\mathrm{Tm}(\Gamma, B\{\langle 1_\Gamma, x := a\rangle_A\})$

Demonstration of resulting syntax

We drop several of the subscripts and annotations in these examples. We also write

$$\begin{aligned}&\langle x_1 := a_1, x_2 := a_2, \dots, x_n := a_n\rangle \\ &= \langle \dots \langle\langle !, x_1 := a_1\rangle, x_2 := a_2\rangle, \dots, x_n := a_n\rangle.\end{aligned}$$

Note that $\langle\rangle = !$ and $\langle x_1 := a_1\rangle = \langle !, x_1 := a_1\rangle$.
For a context $\Gamma = \top.x_1 : A_1.x_2 : A_2.\dots.x_n : A_n$ we note that its identity is

$$1_\Gamma = \langle x_1 := \mathrm{v}_{x_1}, x_2 := \mathrm{v}_{x_2}, \dots, x_n := \mathrm{v}_{x_n}\rangle.$$

Consider a simple example with a binary relation R between two types and a function f.
Let $A, B \in \mathrm{Ty}(\top)$. Then $\top.x : A \in \mathcal{C}$ and $\mathrm{p} = ! : \top.x : A \longrightarrow \top$.
Thus $B\{\mathrm{p}\} \in \mathrm{Ty}(\top.x : A)$. Form $\top.x : A.y : B\{\mathrm{p}\} :\in \mathcal{C}$.
Let $R \in \mathrm{Ty}(\top.x : A.y : B\{\mathrm{p}\})$ and $f \in \mathrm{Tm}(\top.x : A, B\{\mathrm{p}\})$. Now $1_{\top.x:A} : \top.x : A \longrightarrow \top.x : A$ gives

$$\langle 1_{\top.x:A}, y := f\rangle : \top.x : A \longrightarrow \top.x : A, y : B\{\mathrm{p}\}$$

and hence

$$R\{\langle 1_{\top.x:A}, y := f\rangle\} \in \mathrm{Ty}(\top.x : A).$$

Rewriting the identity and using $f = f\{1_{\top.x:A}\}$ we get

$$R\{\langle x := \mathrm{v}_x, y := f\{\langle x := \mathrm{v}_x\rangle\}\rangle\} \in \mathrm{Ty}(\top.x : A).$$

Thus

$$\Pi(A, (x)R\{\langle x := \mathrm{v}_x, y := f\{\langle x := \mathrm{v}_x\rangle\}\rangle\}) \in \mathrm{Ty}(\top).$$

Assume furthermore that

$$g \in \mathrm{Tm}(\top.x : A, R\{\langle x := \mathrm{v}_x, y := f\{\langle x := \mathrm{v}_x\rangle\}\rangle\}).$$

Rewriting the identity this becomes

$$g\{\langle x := \mathrm{v}_x\rangle\} \in \mathrm{Tm}(\top.x : A, R\{\langle x := \mathrm{v}_x, y := f\{\langle x := \mathrm{v}_x\rangle\}\rangle\}).$$

Thus

$$\lambda((x)g\{\langle x := \mathrm{v}_x\rangle\}) \in \Pi(A, (x)R\{\langle x := \mathrm{v}_x, y := f\{\langle x := \mathrm{v}_x\rangle\}\rangle\}) \in \mathrm{Ty}(\top$$

where $(\lambda x : A)b$ is short for $\lambda_{\Gamma,A,B}(x, b)$. Writing $E[a_1, \ldots, a_n]$ for $E\{\langle x_1 := a_1, \ldots, x_n := a_n\rangle\}$ and E type or term in the context $\top.x_1 : A_1.x_2 : A_2. \ldots .x_n : A_n$, we can express this as

$$\lambda((x)g[\mathrm{v}_x]) \in \Pi(A, (x)R[\mathrm{v}_x, f[\mathrm{v}_x]]) \in \mathrm{Ty}(\top),$$

We have a twist map

$$\langle x := \mathrm{v}_x, y := \mathrm{v}_y\{p\}\rangle : \top.y : B.x : A\{p\} \longrightarrow \top.x : A.y : B\{p\}$$

Thus

$$R\{\langle x := \mathrm{v}_x, y := \mathrm{v}_y\{p\}\rangle\} \in \mathrm{Ty}(\top.y : B.x : A\{p\}).$$

With the abbreviation principle above

$$R[\mathrm{v}_x, \mathrm{v}_y\{p\}] \in \mathrm{Ty}(\top.y : B.x : A\{p\}).$$

Furthermore

$$\Pi(A\{p\}, (x)R[\mathrm{v}_x, \mathrm{v}_y\{p\}]) \in \mathrm{Ty}(\top.y : B). \qquad (3)$$

This suggests that projections p should be forgotten in writing out the formulas in order to produce the standard variableful presentation. The last would then be

$$\Pi(A, (x)R[\mathrm{v}_x, \mathrm{v}_y]) \in \mathrm{Ty}(\top.y : B).$$

Martin-Löf's substitution calculus - polymorphic, free cwfs?

The first-order part of the Substitution Calculus (Martin-Löf 1992) has the judgment forms

$$\Gamma : \text{context}$$

$$\Gamma = \Delta : \text{context}$$

$$\gamma : \Delta \longrightarrow \Gamma$$

$$\gamma = \delta : \Delta \longrightarrow \Gamma$$

$$\Gamma \Longrightarrow A : \text{type}$$

$$\Gamma \Longrightarrow A = B : \text{type}$$

$$\Gamma \Longrightarrow a : A$$

$$\Gamma \Longrightarrow a = b : A$$

One can also add (as in Tasistro 1993) a judgement for *context end extension*

$$\Gamma \preceq \Delta : \text{context}$$

to explicate this notion.

There are two rules for context formation

$$\frac{}{() : \mathrm{context}} \qquad \frac{\Gamma : \mathrm{context} \quad \Gamma \Longrightarrow A : \mathrm{type}}{(\Gamma, x : A) : \mathrm{context}}$$

The thinning rules ("context polymorphism") are:

$$\frac{\gamma : \Delta \longrightarrow \Gamma}{\gamma : \Theta \longrightarrow \Gamma}\ (\Delta \preceq \Theta) \qquad \frac{\gamma : \Delta \longrightarrow \Gamma}{\gamma : \Delta \longrightarrow \Theta}\ (\Theta \preceq \Gamma)$$

$$\frac{\gamma : \Delta \Longrightarrow A : \mathrm{type}}{\gamma : \Theta \Longrightarrow A : \mathrm{type}}\ (\Delta \preceq \Theta) \qquad \frac{\gamma : \Delta \Longrightarrow a : A}{\gamma : \Theta \Longrightarrow a : A}\ (\Delta \preceq \Theta)$$

The composition rules and associativity laws (omitted here).
Unit laws:

$$\frac{\Gamma : \mathrm{context}}{() : \Gamma \longrightarrow \Gamma}$$

$$\frac{\gamma : \Delta \longrightarrow \Gamma}{()\gamma = \gamma : \Delta \longrightarrow \Gamma} \qquad \frac{\gamma : \Delta \longrightarrow \Gamma}{\gamma() = \gamma : \Delta \longrightarrow \Gamma}$$

$$\frac{\Gamma \longrightarrow A : \mathrm{type}}{\Gamma \longrightarrow A() = A : \mathrm{type}} \qquad \frac{\Gamma \longrightarrow a : A}{\Gamma \longrightarrow a() = a : A}$$

Updating laws and β, η, ξ-rules

$$\frac{\gamma : \Delta \longrightarrow \Gamma \quad \Delta \Longrightarrow a : A\gamma}{(\gamma, x = a) : \Delta \longrightarrow \Gamma, x : A}$$

$$\frac{\delta : \Theta \longrightarrow \Delta \quad \gamma : \Delta \longrightarrow \Gamma \quad \Delta \Longrightarrow a : A\gamma}{(\gamma, x = a)\delta = (\gamma\delta, x = a\delta) : \Theta \longrightarrow \Gamma, x : A}$$

$$\frac{\gamma : \Delta \longrightarrow \Gamma \quad \Delta \Longrightarrow a : A\gamma}{(\gamma, x = a) = \gamma : \Delta \longrightarrow \Gamma} \qquad \frac{\gamma : \Delta \longrightarrow \Gamma \quad \Delta \Longrightarrow a : A\gamma}{\Delta \Longrightarrow x(\gamma, x = a) = a : A\gamma}$$

$$\frac{\gamma : \Delta \longrightarrow ()}{\gamma = () : \Delta \longrightarrow ()} \qquad \frac{\gamma : \Delta \longrightarrow \Gamma, x : A}{\gamma = (\gamma, x = x\gamma) : \Delta \longrightarrow \Gamma, x : A}$$

$$\frac{\gamma : \Delta \longrightarrow () \quad \gamma : \Delta \longrightarrow ()}{\gamma = \delta : \Delta \longrightarrow ()}$$

$$\frac{\gamma : \Delta \longrightarrow \Gamma, x : A \quad \delta : \Delta \longrightarrow \Gamma, x : A \quad \gamma = \delta : \Delta \longrightarrow \Gamma \quad \Delta \Longrightarrow x\gamma = x\delta}{\gamma = \delta : \Delta \longrightarrow \Gamma, x : A}$$

Variable law: derive $\Gamma \Longrightarrow x : A$ when $x : A$ is declared in Γ.

Interpretation Substitution Calculus by a cwfn $\mathcal{C}$?

- Contexts: $\Gamma \in \mathrm{Ob}(\mathcal{C})$
- Context maps from Δ to Γ:

$$\Delta \succeq \Delta' \xrightarrow{f} \Gamma' \succeq \Gamma$$

- Types over Γ:

$$\Gamma \succeq \Gamma' \qquad A \in \mathrm{Ty}(\Gamma')$$

- Elements of type A over Γ:

$$\Gamma \succeq \Gamma' \qquad A \in \mathrm{Ty}(\Gamma') \qquad a \in \mathrm{Tm}(\Gamma', A)$$

Interpreting equalities

- $(\Delta \succeq \Delta' \xrightarrow{f} \Gamma' \succeq \Gamma) \sim (\Delta \succeq \Delta'' \xrightarrow{g} \Gamma'' \succeq \Gamma)$ iff
$$\mathrm{p}_{\Gamma',\Gamma} \circ f \circ \mathrm{p}_{\Delta,\Delta'} = \mathrm{p}_{\Gamma'',\Gamma} \circ g \circ \mathrm{p}_{\Delta,\Delta''}$$
- $(\Gamma \succeq \Gamma'; A \in \mathrm{Ty}(\Gamma')) \sim (\Gamma \succeq \Gamma''; B \in \mathrm{Ty}(\Gamma''))$ iff
$$A\{\mathrm{p}_{\Gamma,\Gamma'}\} = B\{\mathrm{p}_{\Gamma,\Gamma''}\}$$
- $(\Gamma \succeq \Gamma'; A \in \mathrm{Ty}(\Gamma'); a \in \mathrm{Tm}(\Gamma', A)) \sim$
$(\Gamma \succeq \Gamma''; B \in \mathrm{Ty}(\Gamma''); b \in \mathrm{Tm}(\Gamma'', B))$ iff
$$A\{\mathrm{p}_{\Gamma,\Gamma'}\} = B\{\mathrm{p}_{\Gamma,\Gamma''}\} \text{ and } a\{\mathrm{p}_{\Gamma,\Gamma'}\} = b\{\mathrm{p}_{\Gamma,\Gamma''}\}.$$

III. Formalizing models of type theory in type theory

An equivalent (to cwfs) standard framework for interpreting dependent type theory is a category with attributes (Cartmell) .

III. Formalizing models of type theory in type theory

An equivalent (to cwfs) standard framework for interpreting dependent type theory is a category with attributes (Cartmell) .

- This modeling usually takes place in set theory.
- An important example is Hofmann's model (Hofmann 1997), which is built from the syntax and judgements of intensional type theory, and is used to translate proofs in extensional type theory into proofs in the intensional theory.

III. Formalizing models of type theory in type theory

An equivalent (to cwfs) standard framework for interpreting dependent type theory is a category with attributes (Cartmell) .

- This modeling usually takes place in set theory.
- An important example is Hofmann's model (Hofmann 1997), which is built from the syntax and judgements of intensional type theory, and is used to translate proofs in extensional type theory into proofs in the intensional theory.
- However, it is of interest to do the modeling in type theory itself, e.g. for the purpose of formal verification, and for foundational reasons. (Dybjer, *Internal Type Theory,* 1995 and onwards).

Categories with attributes is a variant of cwfs where the elements $\mathrm{Tm}(\Gamma, A)$ are defined as sections of projections $\mathrm{p}(A) : \Gamma.A \longrightarrow \Gamma$. It seems slightly easier to formalize.

Definition 1. A category with attributes (cwa) consists of the data

(a) A category $\mathcal{C}$ with a terminal object $\top$.

(b) A functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$.

(c) For each $A \in \mathrm{Ty}(\Gamma)$, an object $\Gamma.A$ in $\mathcal{C}$ and a morphism

$$\mathsf{p}(A) = \mathsf{p}_\Gamma(A) : \Gamma.A \longrightarrow \Gamma \text{ in } \mathcal{C}.$$

(d) The final datum tells how substitutions interact with context extensions: For each $f : \Delta \longrightarrow \Gamma$ and $A \in \mathrm{Ty}(\Gamma)$, there is a morphism $\mathsf{q}(f, A) = \mathsf{q}_\Gamma(f, A) : \Delta.(A\{f\}) \longrightarrow \Gamma.A$ in $\mathcal{C}$ such that

$$\begin{tikzcd}
\Delta.(A\{f\}) \arrow[r, "\mathsf{q}(f{,}A)"] \arrow[d, "\mathsf{p}(A\{f\})"'] & \Gamma.A \arrow[d, "\mathsf{p}(A)"] \\
\Delta \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

is a pullback, and furthermore functoriality conditions

(d.1) $\mathsf{q}(1_\Gamma, A) = 1_{\Gamma.A}$

(d.2) $\mathsf{q}(f \circ g, A) = \mathsf{q}(f, A) \circ \mathsf{q}(g, A\{f\})$ for $\Theta \xrightarrow{g} \Delta \xrightarrow{f} \Gamma$.

Example: The set-theoretic interpretation of the pullback diagram in (d) is then

$$\begin{tikzcd}
\Sigma_{y\in\Delta}.A(f(y)) \arrow[r, "\mathsf{q}"] \arrow[d, "\mathsf{p}"'] & \Sigma_{x\in\Gamma}.A(x) \arrow[d, "\mathsf{p}"] \\
\Delta \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

where $\mathsf{p}(\langle u, v\rangle) = u$ and $\mathsf{q}(\langle y, s\rangle) = \langle f(y), s\rangle$.

By assuming that the universe U is closed under further constructions one can verify axioms for type theoretic constructions like Π, W, I-types etc.

Indirect model in MLTT

In view of the fact that intensional Martin-Löf type theory interprets the universe V of CZF (Aczel 1978, 1986) we get an indirect intepretation of the extensional M-L type theory in the intensional one.

Aczel's interpretation has been formalized in various proof assistants: LEGO (N.P. Mendler, 1990), Agda 1 (M.Takeyama mid 1990s), Coq (P. and Wilander 2011). In the latter interpretation a full faithful functor

$$V \longrightarrow \mathrm{Setoids}$$

is explicitly constructed.

However, we are interested in more direct interpretations.

Setoids

In type theories the notion of set is usually understood in the sense of Bishop as a type together with an equivalence relation, also called a setoid

$$A = (|A|, =_A)$$

where $|A|$ is a type and $=_A$ is an equivalence relation on $|A|$.

Setoids

In type theories the notion of set is usually understood in the sense of Bishop as a type together with an equivalence relation, also called a setoid

$$A = (|A|, =_A)$$

where $|A|$ is a type and $=_A$ is an equivalence relation on $|A|$.
An extensional function $f : A \longrightarrow B$ between setoids is a function $|A| \longrightarrow |B|$ which respects the equivalence relations, i.e.

$$(\forall x, y : |A|)[x =_A y \Longrightarrow f(x) =_B f(y)]$$

Two such functions f and g are extensionally equal ($f =_{\mathrm{ext}} g$) if $(\forall x : |A|)(f(x) =_B g(x))$.

The setoids and extensional functions in a (constructive) type theory form an e-category **Setoids** and its properties can be described in the same abstract way as the category **Sets** in a (constructive) set theory.

The setoids and extensional functions in a (constructive) type theory form an e-category **Setoids** and its properties can be described in the same abstract way as the category **Sets** in a (constructive) set theory.

The properties of the category reflects the possibilities and limitations of constructions in the background theory.

The notion of e-category is a variant of the standard notion of category, but where no equality relation is required on objects.

An e-category $\mathcal{C}$ consists of a type $\mathrm{Ob}\,\mathcal{C}$ of objects, together with a setoid $\mathcal{C}(A, B)$ of morphisms for every pair of objects A and B. The composition is an extensional function

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \longrightarrow \mathcal{C}(A, C)$$

which satisfies the usual monoid laws.

The notion of e-category is a variant of the standard notion of category, but where no equality relation is required on objects.

An e-category $\mathcal{C}$ consists of a type $\mathrm{Ob}\,\mathcal{C}$ of objects, together with a setoid $\mathcal{C}(A, B)$ of morphisms for every pair of objects A and B. The composition is an extensional function

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \longrightarrow \mathcal{C}(A, C)$$

which satisfies the usual monoid laws.

An e-functor is a functor where the object part is just a function between types. There is no equality of objects to respect.

E-cwas - cwas in type theory

Referring back to Definition 1, in order for the equations (d.1) and (d.2) to make sense, we need the *object equalities*

$$\Gamma.(A\{1_\Gamma\}) = \Gamma.A \qquad \mathsf{p}(A\{1_\Gamma\}) = \mathsf{p}(A)$$

and

$$\Theta.(A\{f \circ g\}) = \Theta.(A\{f\}\{g\})$$

and moreover

$$\mathsf{p}(A\{f \circ g\}) = \mathsf{p}(A\{f\}\{g\}).$$

E-cwas - cwas in type theory

Referring back to Definition 1, in order for the equations (d.1) and (d.2) to make sense, we need the *object equalities*

$$\Gamma.(A\{1_\Gamma\}) = \Gamma.A \qquad \mathsf{p}(A\{1_\Gamma\}) = \mathsf{p}(A)$$

and

$$\Theta.(A\{f \circ g\}) = \Theta.(A\{f\}\{g\})$$

and moreover

$$\mathsf{p}(A\{f \circ g\}) = \mathsf{p}(A\{f\}\{g\}).$$

They follow from the functoriality of Ty and by requiring the object equality $\Gamma.A = \Gamma.A'$ and $\mathsf{p}(A) = \mathsf{p}(A')$ whenever $A = A'$.

E-cwas - cwas in type theory

Referring back to Definition 1, in order for the equations (d.1) and (d.2) to make sense, we need the *object equalities*

$$\Gamma.(A\{1_\Gamma\}) = \Gamma.A \qquad \mathsf{p}(A\{1_\Gamma\}) = \mathsf{p}(A)$$

and

$$\Theta.(A\{f \circ g\}) = \Theta.(A\{f\}\{g\})$$

and moreover

$$\mathsf{p}(A\{f \circ g\}) = \mathsf{p}(A\{f\}\{g\}).$$

They follow from the functoriality of Ty and by requiring the object equality $\Gamma.A = \Gamma.A'$ and $\mathsf{p}(A) = \mathsf{p}(A')$ whenever $A = A'$.

This notion of cwa is not appropriate for categories $\mathcal{C}$ that lack object equality, like e-categories.

We modify the structure slightly to the setting of e-categories:

Definition 2. An e-category with attributes (ecwa) consists of the following data (a) - (d):

(a) An e-category $\mathcal{C}$ with a terminal object $\top$.

(b) An e-functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Setoids}$.

(c) There is an e-functor $\Delta_\Gamma : \mathrm{Ty}(\Gamma)^\# \longrightarrow \mathcal{C}/\Gamma$. For $A \in \mathrm{Ob}\ \mathrm{Ty}(\Gamma)$, write $\Delta_\Gamma(A) = (\mathsf{p}(A) : \Gamma.A \longrightarrow \Gamma)$.

We modify the structure slightly to the setting of e-categories:

Definition 2. An e-category with attributes (ecwa) consists of the following data (a) - (d):

(a) An e-category $\mathcal{C}$ with a terminal object $\top$.

(b) An e-functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Setoids}$.

(c) There is an e-functor $\Delta_\Gamma : \mathrm{Ty}(\Gamma)^\# \longrightarrow \mathcal{C}/\Gamma$. For $A \in \mathrm{Ob}\ \mathrm{Ty}(\Gamma)$, write $\Delta_\Gamma(A) = (\mathsf{p}(A) : \Gamma.A \longrightarrow \Gamma)$.

Here $S^\#$ denotes the discrete e-category induced by a setoid S. $\mathcal{C}/\Gamma$ denotes the slice e-category of $\mathcal{C}$ over Γ.

Thus for any proof object t of $A =_{\mathrm{Ty}(\Gamma)} A'$, $\Delta_\Gamma(t) : \Gamma.A \longrightarrow \Gamma.A'$ is an isomorphism such that

$$\begin{tikzcd}
\Gamma.A \arrow[rr, "\Delta_\Gamma(t)"] \arrow[dr, swap, "p(A)"] & & \Gamma.A' \arrow[dl, "p(A')"] \\
& \Gamma &
\end{tikzcd}$$

commutes.

Thus for any proof object t of $A =_{\mathrm{Ty}(\Gamma)} A'$, $\Delta_\Gamma(t) : \Gamma.A \longrightarrow \Gamma.A'$ is an isomorphism such that

$$\begin{tikzcd}
\Gamma.A \arrow[rr, "\Delta_\Gamma(t)"] \arrow[dr, "p(A)"'] & & \Gamma.A' \arrow[dl, "p(A')"] \\
& \Gamma &
\end{tikzcd}$$

commutes. Moreover, $\Delta_\Gamma(t)$ is independent of t and

$$\Delta_\Gamma(t) = 1_{\Gamma.A} \qquad (t \text{ proof of } A =_{\mathrm{Ty}(\Gamma)} A)$$

$$\Delta_\Gamma(s \circ t) = \Delta_\Gamma(s) \circ \Delta_\Gamma(t) \quad (t \text{ pf. of } A =_{\mathrm{Ty}(\Gamma)} A' \text{ and } s \text{ pf. of } A' =_{\mathrm{Ty}(\Gamma)} A'$$

(A particular feature of the slice of an e-category is that equalities of objects over the base turn into isomorphism.)

(d) For each $f : \Psi \longrightarrow \Gamma$ and $A \in \mathrm{Ty}(\Gamma)$, there is a morphism $\mathsf{q}(f, A) : \Psi.A\{f\} \longrightarrow \Gamma.A$ in $\mathcal{C}$ such that

$$\begin{tikzcd}
\Psi.A\{f\} \arrow[r, "\mathsf{q}(f,A)"] \arrow[d, "\mathsf{p}(A\{f\})"'] & \Gamma.A \arrow[d, "\mathsf{p}(A)"] \\
\Psi \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

is a pullback, and moreover these morphisms satisfy

(d.1) $\mathsf{q}(1_\Gamma, A) \circ \Delta_\Gamma(t) = 1_{\Gamma.A}$ where t is any pf. for $\mathrm{Ty}(1_\Gamma)(A) = A$.

(d.2) $\mathsf{q}(f \circ g, A) \circ \Delta_\Theta(t) = \mathsf{q}(f, A) \circ \mathsf{q}(g, A\{f\})$ for $\Theta \xrightarrow{g} \Psi \xrightarrow{f} \Gamma$ and where t is any pf. for $A\{f \circ g\} =_{\mathrm{Ty}(\Theta)} A\{f\}\{g\}$.

Note the type correcting isomorphisms $\Delta(t)$.

Further in condition (d), note that if $f = f' : \Psi \longrightarrow \Gamma$, s is a proof of $A = A' \in \mathrm{Ty}(\Gamma)$ and t is a proof of $A\{f\} = A'\{f'\}$, then by the pullback properly,

$$\mathsf{q}(f, A) \circ \Delta_\Psi(t) = \Delta_\Gamma(s) \circ \mathsf{q}(f', A').$$

Note the type correcting isomorphisms $\Delta(t)$.

Further in condition (d), note that if $f = f' : \Psi \longrightarrow \Gamma$, s is a proof of $A = A' \in \mathrm{Ty}(\Gamma)$ and t is a proof of $A\{f\} = A'\{f'\}$, then by the pullback properly,

$$\mathsf{q}(f, A) \circ \Delta_\Psi(t) = \Delta_\Gamma(s) \circ \mathsf{q}(f', A').$$

Equality on finite contexts. Note that if $f : \Psi \longrightarrow \Gamma$ is an isomorphism, and $A \in \mathrm{Ty}(\Psi)$, $B \in \mathrm{Ty}(\Gamma)$ and t is proof of $A =_{\mathrm{Ty}(\Psi)} B\{f\}$, then get isomorphism:

$$\mathsf{q}(f, B) \circ \Delta(t) : \Psi.A \longrightarrow \Gamma.B$$

Thus we can inductively define when two finite contexts are equal.

Interpretation

The definition of ecwas suggests introducing the following judgements about types

- $\Gamma \vdash A \,\mathrm{type}$ meaning $A \in \mathrm{Ty}(\Gamma)$
- $\Gamma \vdash A = A'$ meaning $A =_{\mathrm{Ty}(\Gamma)} A'$ where $A, A' \in \mathrm{Ty}(\Gamma)$.

Interpretation

The definition of ecwas suggests introducing the following judgements about types

- $\Gamma \vdash A \text{ type}$ meaning $A \in \mathrm{Ty}(\Gamma)$
- $\Gamma \vdash A = A'$ meaning $A =_{\mathrm{Ty}(\Gamma)} A'$ where $A, A' \in \mathrm{Ty}(\Gamma)$.

Define $\mathrm{Tm}(\Gamma, A)$, the elements of A in the context Γ, to be the setoid of sections of $\mathsf{p}(A) : \Gamma.A \longrightarrow \Gamma$.

Note that if r is a proof for $A =_{\mathrm{Ty}(\Gamma)} A'$, then $M \in \mathrm{Tm}(\Gamma, A)$ implies $\Delta_\Gamma(r) \circ M \in \mathrm{Tm}(\Gamma, A')$.

(Cf. canonical isomorphisms of Curien 1993.)

Now assuming a term a always come with an "original" type A, written as a pair (a, A), we introduce the further judgements

Now assuming a term a always come with an "original" type A, written as a pair (a, A), we introduce the further judgements

- $\Gamma \vdash (a, A) : A'$ meaning

  $a \in \mathrm{Tm}(\Gamma, A)$ and $A =_{T(\Gamma)} A'$.

Now assuming a term a always come with an "original" type A, written as a pair (a, A), we introduce the further judgements

- $\Gamma \vdash (a, A) : A'$ meaning

  $a \in \mathrm{Tm}(\Gamma, A)$ and $A =_{T(\Gamma)} A'$.

- $\Gamma \vdash (a, A) = (a', A') : A''$ meaning

  $\Gamma \vdash (a, A) : A''$ and $\Gamma \vdash (a, A') : A''$ and that there is a proof r of $A =_{\mathrm{Ty}(\Gamma)} A'$ such that $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$.

The rules

$$\boxed{\frac{\Gamma \vdash s : A \qquad \Gamma \vdash A = B}{\Gamma \vdash s : B}}$$

$$\boxed{\frac{\Gamma \vdash s = t : A \qquad \Gamma \vdash A = B}{\Gamma \vdash s = t : B}}$$

are immediately justified.

Substitution into terms

For $a \in \mathrm{Tm}(\Gamma, A)$ and $f : \Psi \longrightarrow \Gamma$, define $a\{f\} : \Psi \longrightarrow \Psi.A\{f\}$ as the unique morphism (see diagram below) with $\mathsf{p}(A\{f\})) \circ a\{f\} = 1_\Psi$ and $\mathsf{q}(f, A) \circ a\{f\} = a \circ f$. Thus $a\{f\} \in \mathrm{Tm}(\Psi, A\{f\})$.

$$\begin{tikzcd}
\Psi \arrow[dr, dotted, "a\{f\}"] \arrow[drr, "a \circ f"] \arrow[ddr, "1"'] & & \\
& \Psi.A\{f\} \arrow[r, "\mathsf{q}(f,A)"'] \arrow[d, "\mathsf{p}(A\{f\})"] & \Gamma.A \arrow[d, "\mathsf{p}(A)"] \\
& \Psi \arrow[r, "f"] & \Gamma
\end{tikzcd}$$

Stability under substitution

Thus if

$$\Gamma \vdash (a, A) : A'$$

and $f : \Psi \longrightarrow \Gamma$, we have $a\{f\} \in \mathrm{Tm}(\Psi, A\{f\})$ and $A\{f\} =_{\mathrm{Ty}(\Psi)} A'\{f\}$, so

$$\Psi \vdash (a\{f\}, A\{f\}) : A'\{f\}$$

Thus the rule

$$\boxed{\dfrac{\Gamma \vdash s : A' \qquad f : \Psi \longrightarrow \Gamma}{\Psi \vdash s\{f\} : A'\{f\}}}$$

is justified.

Moreover, if

$$\Gamma \vdash (a, A) = (a', A') : A''$$

we have $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$ for some proof r of $A =_{T(\Gamma)} A'$.
Then

$$\Delta_\Gamma(r) \circ a \circ f = a' \circ f$$

Moreover, if

$$\Gamma \vdash (a, A) = (a', A') : A''$$

we have $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$ for some proof r of $A =_{T(\Gamma)} A'$. Then

$$\Delta_\Gamma(r) \circ a \circ f = a' \circ f$$

and hence

$$\Delta_\Gamma(r) \circ \mathsf{q}(f, A) \circ a\{f\} = a' \circ f.$$

Moreover, if

$$\Gamma \vdash (a, A) = (a', A') : A''$$

we have $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$ for some proof r of $A =_{T(\Gamma)} A'$. Then

$$\Delta_\Gamma(r) \circ a \circ f = a' \circ f$$

and hence

$$\Delta_\Gamma(r) \circ \mathsf{q}(f, A) \circ a\{f\} = a' \circ f.$$

So by remark above there is a proof s of $\mathrm{Ty}(f)(A) = \mathrm{Ty}(f)(A')$ such that $\Delta_\Gamma(r) \circ \mathsf{q}(f, A) = \mathsf{q}(f, A') \circ \Delta_\Psi(s)$ and hence

$$\mathsf{q}(f, A') \circ \Delta_\Psi(s) \circ a\{f\} = a' \circ f.$$

Moreover, if

$$\Gamma \vdash (a, A) = (a', A') : A''$$

we have $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$ for some proof r of $A =_{T(\Gamma)} A'$. Then

$$\Delta_\Gamma(r) \circ a \circ f = a' \circ f$$

and hence

$$\Delta_\Gamma(r) \circ \mathsf{q}(f, A) \circ a\{f\} = a' \circ f.$$

So by remark above there is a proof s of $\mathrm{Ty}(f)(A) = \mathrm{Ty}(f)(A')$ such that $\Delta_\Gamma(r) \circ \mathsf{q}(f, A) = \mathsf{q}(f, A') \circ \Delta_\Psi(s)$ and hence

$$\mathsf{q}(f, A') \circ \Delta_\Psi(s) \circ a\{f\} = a' \circ f.$$

It follows by uniqueness that $\Delta_\Psi(s) \circ a\{f\} = a'\{f\}$, so indeed

$$\Psi \vdash (a\{f\}, A\{f\}) = (a'\{f\}, A'\{f\}) : A''\{f\}.$$

Moreover, if

$$\Gamma \vdash (a, A) = (a', A') : A''$$

we have $\Delta_\Gamma(r) \circ a =_{\mathrm{Tm}(\Gamma, A')} a'$ for some proof r of $A =_{T(\Gamma)} A'$. Then

$$\Delta_\Gamma(r) \circ a \circ f = a' \circ f$$

and hence

$$\Delta_\Gamma(r) \circ \mathsf{q}(f, A) \circ a\{f\} = a' \circ f.$$

So by remark above there is a proof s of $\mathrm{Ty}(f)(A) = \mathrm{Ty}(f)(A')$ such that $\Delta_\Gamma(r) \circ \mathsf{q}(f, A) = \mathsf{q}(f, A') \circ \Delta_\Psi(s)$ and hence

$$\mathsf{q}(f, A') \circ \Delta_\Psi(s) \circ a\{f\} = a' \circ f.$$

It follows by uniqueness that $\Delta_\Psi(s) \circ a\{f\} = a'\{f\}$, so indeed

$$\Psi \vdash (a\{f\}, A\{f\}) = (a'\{f\}, A'\{f\}) : A''\{f\}.$$

This justifies also the rule:

$$\boxed{\dfrac{\Gamma \vdash s = t : A'' \qquad f : \Psi \longrightarrow \Gamma}{\Psi \vdash s\{f\} = t\{f\} : A''\{f\}}}$$

The notion of Π-types is similar for cwas as for cwfs, but note that elements are particular context maps. For ecwas we need to make appropriate changes.

An ecwa supports Π-types if for $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$ there is a type $\Pi(A, B) \in \mathrm{Ty}(\Gamma)$, and moreover for every $b \in \mathrm{Tm}(\Gamma.A, B)$ there is an element $\lambda_{A,B}(b) \in \mathrm{Tm}(\Gamma, \Pi(A, B))$, and furthermore for any $M \in \mathrm{Tm}(\Gamma, \Pi(A, B))$ and any $a \in \mathrm{Tm}(\Gamma, A)$ there is an element $\mathsf{App}_{A,B}(c, a) \in \mathrm{Tm}(\Gamma, B\{a\})$, such that the following equations hold for any $f : \Psi \longrightarrow \Gamma$:

- (β-red) $\mathsf{App}_{A,B}(\lambda_{A,B}(b), a) =_{\mathrm{Tm}(\Gamma, B\{a\})} P\{a\}$, [as before]
- (Π-subst) $\Pi(A, B)\{f\} =_{\mathrm{Ty}(B)} \Pi(A\{f\}, B\{\mathsf{q}(f, A)\})$, [as before]

This part of the definition has type adjustments:

- (λ-subst)

  $$\lambda_{A,B}(b)\{f\} =_{\mathrm{Tm}(B,\Pi(A,B)\{f\})} \Delta_B(t)\circ\lambda_{A\{f\},B\{\mathsf{q}(f,A)\}}(b\{\mathsf{q}(f,A)\}),$$

  for any proof t of $\Pi(A\{f\}, B\{\mathsf{q}(f,A)\}) =_{\mathrm{Ty}(B)} \Pi(A,B)\{f\}$, This part of the definition has type adjustments:

- (λ-subst)

  $$\lambda_{A,B}(b)\{f\} =_{\mathrm{Tm}(B,\Pi(A,B)\{f\})} \Delta_B(t)\circ\lambda_{A\{f\},B\{\mathsf{q}(f,A)\}}(b\{\mathsf{q}(f,A)\}),$$

  for any proof t of $\Pi(A\{f\}, B\{\mathsf{q}(f,A)\}) =_{\mathrm{Ty}(B)} \Pi(A,B)\{f\}$,
- (App-subst)

  $$\mathsf{App}_{A,B}(c,a)\{f\} =_{\mathrm{Tm}(\cdots)} \Delta_\Psi(s)\circ\mathsf{App}_{A\{f\},B\{\mathsf{q}(f,A)\}}(\Delta_\Psi(t)\circ c\{f\}, a\{f$$

  for any proof s of $B\{\mathsf{q}(f,A)\}\{a\{f\}\} =_{\mathrm{Ty}(\Psi)} B\{a\}\{f\}$ and any proof t of $\Pi(A,B)\{f\} =_{\mathrm{Ty}(\Psi)} \Pi(A\{f\}, B\{\mathsf{q}(f,A)\})$.

Furthermore there are the following extensionality conditions on Π, λ and App:

- (Π-cong) if s is a proof of $A =_{\mathrm{Ty}(\Gamma)} A'$ and for $B \in \mathrm{Ty}(\Gamma.A)$, $B' \in \mathrm{Ty}(\Gamma.A')$ with $B =_{\mathrm{Ty}(\Gamma.A)} B'\{\Delta(s)\}$, then

$$\Pi(A, B) =_{\mathrm{Ty}(\Gamma)} \Pi(A', B').$$

Furthermore there are the following extensionality conditions on Π, λ and App:

- (Π-cong) if s is a proof of $A =_{\mathrm{Ty}(\Gamma)} A'$ and for $B \in \mathrm{Ty}(\Gamma.A)$, $B' \in \mathrm{Ty}(\Gamma.A')$ with $B =_{\mathrm{Ty}(\Gamma.A)} B'\{\Delta(s)\}$, then
$$\Pi(A, B) =_{\mathrm{Ty}(\Gamma)} \Pi(A', B').$$
- (λ-cong) if $b =_{\mathrm{Tm}(\Gamma.A,B)} b'$, then
$$\lambda_{A,B}(b) =_{\mathrm{Tm}(\Gamma,\Pi(A,B))} \lambda_{A,B}(b')$$

Furthermore there are the following extensionality conditions on Π, λ and App:

- (Π-cong) if s is a proof of $A =_{\mathrm{Ty}(\Gamma)} A'$ and for $B \in \mathrm{Ty}(\Gamma.A)$, $B' \in \mathrm{Ty}(\Gamma.A')$ with $B =_{\mathrm{Ty}(\Gamma.A)} B'\{\Delta(s)\}$, then
$$\Pi(A, B) =_{\mathrm{Ty}(\Gamma)} \Pi(A', B').$$
- (λ-cong) if $b =_{\mathrm{Tm}(\Gamma.A,B)} b'$, then
$$\lambda_{A,B}(b) =_{\mathrm{Tm}(\Gamma,\Pi(A,B))} \lambda_{A,B}(b')$$
- (App-cong) if $c =_{\mathrm{Tm}(\Gamma,\Pi(A,B))} c'$ and $a =_{\mathrm{Tm}(\Gamma,A)} a'$ then
$$\Delta_\Gamma(s) \circ \mathrm{App}_{A,B}(c, a) =_{\mathrm{Tm}(\Gamma,B\{a'\})} \mathrm{App}_{A,B}(c', a'),$$
where s is any proof of $B\{a\} =_{\mathrm{Ty}(\Gamma)} B\{a'\}$.

Bounded categories with attributes

Here all types are subobjects of a fixed setoid, the bounding setoid, thought of a setoid of raw types. This brings the possibility to eliminate the type correcting Δs in the formulation above.

A category (with equality) on objects can be formulated essentially algebraically in type theory with three setoids $\mathrm{Ob}(\mathcal{C})$, $\mathrm{Arr}(\mathcal{C})$ and $\mathrm{Cmp}(\mathcal{C})$ of object, arrows and composable arrows, respectively. There are extensional functions

$$1 : \mathrm{Ob} \longrightarrow \mathrm{Arr}$$

$$\mathrm{dom}, \mathrm{cod} : \mathrm{Arr} \longrightarrow \mathrm{Ob}$$

$$\mathrm{cmp} : \mathrm{Cmp} \longrightarrow \mathrm{Arr} \qquad \mathrm{fst}, \mathrm{snd} : \mathrm{Cmp} \longrightarrow \mathrm{Arr}$$

satisfying familiar equations, with convention that

$$\mathrm{cod}(\mathrm{fst}(u)) = \mathrm{dom}(\mathrm{snd}(u)).$$

Let B be a setoid. The class of subobjects (or subsetoids) of B (denoted $\mathrm{Sub}(B)$) consists of all pairs $S = (\underline{S}, m_S)$ where $\underline{S}$ is a setoid, and $m_S : \underline{S} \longrightarrow S$ is an extensional injective function. An element $x \in B$ *is a member of* S if there is some $u \in \underline{S}$ with $m_S(u) =_B x$, and this case we write

$$x \mathrel{\dot{\in}} S.$$

From this we get the usual derived relations $\doteq$ and $\dot{\subseteq}$ between subobjects. A binary relation on B is a subobject R of $B \times B$. Composition $R \circ S$ of such binary relations is defined as expected. For subsetoids X, Y of B. We say that $R \in \mathrm{Sub}(B \times B)$ is *relation from X to Y*, if $(x, y) \mathrel{\dot{\in}} R$ implies $x \mathrel{\dot{\in}} X$ and $y \mathrel{\dot{\in}} Y$. The relation is *total on X* if $x \mathrel{\dot{\in}} X$ implies that $(x, y) \mathrel{\dot{\in}} R$ for some $y \in B$. A relation is *functional* if

$$(x, y) \mathrel{\dot{\in}} R, (x, z) \mathrel{\dot{\in}} R \Longrightarrow y =_B z.$$

A total functional relation from A to B is in a certain sense equivalent to a setoid extensional function $\underline{A} \longrightarrow \underline{B}$, but we shall not utilize this below.

Definition. A *bounded category with attributes* (bcwa) consists of the following data

(a) A category $\mathcal{C}$ with a terminal object $\top$.

(b1) A set B thought of a set of raw types.

(b2) An extensional function $\mathrm{Ty} : \mathrm{Ob}(\mathcal{C}) \longrightarrow \mathrm{Sub}(\mathrm{B})$.

(b3) An extensional function $\mathrm{Ty} : \mathrm{Arr}(\mathcal{C}) \longrightarrow \mathrm{Sub}(\mathrm{B} \times \mathrm{B})$ such that

- for each $f \in \mathrm{Arr}(\mathcal{C})$, $\mathrm{Ty}(f)$ is a total functional relation from $\mathrm{Ty}(\mathrm{cod}(f))$ to $\mathrm{T}(\mathrm{dom}(f))$,
- for each $\Gamma \in \mathrm{Ob}(\mathcal{C})$, $\mathrm{T}(1_\Gamma)$ is the identity relation on $\mathrm{Ty}(\Gamma)$
- for each $u \in \mathrm{Cmp}(\mathcal{C})$,

$$\mathrm{Ty}(\mathrm{cmp}(u)) = \mathrm{Ty}(\mathrm{fst}(u)) \circ \mathrm{Ty}(\mathrm{snd}(u))$$

(c) For each $\Gamma \in \mathrm{Ob}(\mathcal{C})$, each $A \in \mathrm{B}$, and each proof $q : (A \mathrel{\dot\in} \mathrm{Ty}(\Gamma))$, an arrow $\mathsf{p}_\Gamma(A, q) \in \mathrm{Arr}(\mathcal{C})$ with $\mathrm{cod}(\mathsf{p}_\Gamma(A, q)) = \Gamma$ satifying the extensionality condition

$$\Gamma = \Gamma', A = A' \Longrightarrow \mathsf{p}_\Gamma(A, q) = \mathsf{p}_{\Gamma'}(A', q').$$

Write $\mathrm{Ext}(\Gamma, A, q)$ or $\Gamma.A$ for $\mathrm{dom}(\mathsf{p}_\Gamma(A, q))$

(d) For each $f : \Delta \longrightarrow \Gamma$ and $A \in \mathrm{B}$ with $r : (A \dot{\in} \mathrm{Ty}(\mathrm{dom}(f)))$, there is a $\mathsf{q}_\Gamma(f, A, r) \in \mathrm{Arr}(\mathcal{C})$ satisfying the extensionality condition

$$\Gamma = \Gamma', A = A', f = f' \Longrightarrow \mathsf{q}_\Gamma(f, A, r) = \mathsf{q}_{\Gamma'}(f', A', r').$$

and $\mathsf{q}(f, A, r) = \mathsf{q}_\Gamma(f, A, r) \in \mathrm{Arr}(\mathcal{C})$ such that, for $B \in \mathrm{B}$ and $r' : ((A, B) \dot{\in} \mathrm{Ty}(f))$,

$$\begin{tikzcd} \cdot \arrow[r, "\mathsf{q}(f{,}A{,}r)"] \arrow[d, "\mathsf{p}(B{,}r')"'] & \cdot \arrow[d, "\mathsf{p}(A{,}r)"] \\ \cdot \arrow[r, "f"'] & \cdot \end{tikzcd}$$

is well composed and is a pullback, and furthermore these functoriality conditions hold:

(d.1) $\mathsf{q}(1_\Gamma, A, r) = 1_{\Gamma.A}$

(d.2) $\mathsf{q}(\mathrm{cmp}(u), A, r) = \mathsf{q}(\mathrm{fst}(u), A, r') \circ \mathsf{q}(\mathrm{snd}(u), B, r'')$ for $u \in \mathrm{Cmp}(\mathcal{C})$ and $((A, B) \dot{\in} T(\mathrm{fst}(u)))$

Conjecture: Every type generated ecwa can be turned in to a bcwa and this can be formalized in Coq.

References

J. Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic,* 32(1986), 209 – 243.

J. Cartmell. *Generalized algebraic theories and contextual categories.* DPhil. Thesis Oxford 1978.

P.-L. Curien. Substitution up to isomorphism. *Fundamenta Informaticae,* 19(1993), 51 – 85

P. Dybjer. Internal Type Theory. In: *Types for Proofs and Programs* (eds. S. Berardi and M. Coppo) Lecture Notes in Computer Science, Vol. 1158, 1996, pp 120 – 134

References (cont.)

M. Hofmann. Syntax and semantics of dependent types. In: *Semantics and Logics of Computation* (eds. A. Pitts and P. Dybjer), Cambridge University Press 1997.

P. Martin-Löf. *The substitution calculus.* Notes from a seminar 1992.

A. Tasistro. *Formulation of Martin-Löf's theory of types with explicit substitutions.* Licentiate Thesis, Chalmers University and Göteborg University 1993.