Models of dependent types and dependently typed logic and their formalization in type theory

Erik Palmgren
Stockholm University
Department of Mathematics

Constructive Mathematics and Models of Type Theory
I.H.P. Paris, June 2 – 6, 2014

Overview of this talk

(I) General goal and some obstacles
(II) Type theories from iterated presheaves
(III) Formalizing models of type theory in type theory
(IV) First-order logic with dependent types [if time allows]

I. General goal and some obstacles

A longstanding goal: A constructive model theory of depedent type theory, more specially 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 specially 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): Interpretation 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 and a (first-class) object of B.

II. Models of MLTT from iterated presheaves

It is well-known that the presheaves over a category

$$\mathrm{PSh}(\mathbb{C}) = \mathrm{Sets}^{\mathbb{C}^{\mathrm{op}}}$$

gives rise to a model of MLTT, by construction of a category with attributes/families (Hofmann 1997) closed under the standard type formers.

This model may be constructed even in constructive set theory.

II. Models of MLTT from iterated presheaves

It is well-known that the presheaves over a category

$$\mathrm{PSh}(\mathbb{C}) = \mathrm{Sets}^{\mathbb{C}^{\mathrm{op}}}$$

gives rise to a model of MLTT, by construction of a category with attributes/families (Hofmann 1997) closed under the standard type formers.

This model may be constructed even in constructive set theory.

For $\mathbb{C} = \Delta$, this is the simplicial sets model. The Kan simplicial set model of MLTT+Univalence (Voevodsky) is a subcategory of $\mathrm{PSh}(\Delta)$, for which it turns out that there obstacles to a constructive model (Bezem, Coquand). A constructive model of Univalence is given by Bezem, Coquand and Huber (2013) using cubical sets which live in $\mathrm{PSh}(\square)$.

Iterated presheaves

Let $\mathbb{C}$ be a small category. Let P be a presheaf on $\mathbb{C}$. The category of elements of P, denoted

$$\Sigma(\mathbb{C}, P),$$

consists of objects (a, x) where a is an object in $\mathbb{C}$ and x is an element of $P(a)$. A morphism $\alpha : (a, x) \longrightarrow (b, y)$ is a $\mathbb{C}$-morphism $\alpha : a \longrightarrow b$, such that $P(\alpha)(y) = x$.

Iterated presheaves

Let $\mathbb{C}$ be a small category. Let P be a presheaf on $\mathbb{C}$. The category of elements of P, denoted

$$\Sigma(\mathbb{C}, P),$$

consists of objects (a, x) where a is an object in $\mathbb{C}$ and x is an element of $P(a)$. A morphism $\alpha : (a, x) \longrightarrow (b, y)$ is a $\mathbb{C}$-morphism $\alpha : a \longrightarrow b$, such that $P(\alpha)(y) = x$. $\Sigma(\mathbb{C}, P)$ is the well-known *Grothendieck construction* and is usually denoted

$$\int_{\mathbb{C}} P.$$

This category is again small. There is a projection functor $\pi_P = \pi : \Sigma(\mathbb{C}, P) \longrightarrow \mathbb{C}$ defined by $\pi(a, x) = a$ and $\pi(\alpha) = \alpha$, for $\alpha : (a, x) \longrightarrow (b, y)$.

It is well-known that the Grothendieck construction gives the following equivalence

$$\mathrm{PSh}(\mathbb{C})/P \simeq \mathrm{PSh}(\Sigma(\mathbb{C}, P)).$$

It is well-known that the Grothendieck construction gives the following equivalence

$$\mathrm{PSh}(\mathbb{C})/P \simeq \mathrm{PSh}(\Sigma(\mathbb{C}, P)).$$

This suggests a relation to semantics of dependent types. One can iterate the Grothendieck construction as follows. Write

$$\Sigma(\mathbb{C}) = \mathbb{C}$$

$$\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_n) = \Sigma(\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_{n-1}), P_n).$$

Here $P_{k+1} \in \mathrm{PSh}(\Sigma(\mathbb{C}, P_1, \ldots, P_k))$ for each $k = 0, \ldots, n-1$.

It is well-known that the Grothendieck construction gives the following equivalence

$$\mathrm{PSh}(\mathbb{C})/P \simeq \mathrm{PSh}(\Sigma(\mathbb{C}, P)).$$

This suggests a relation to semantics of dependent types. One can iterate the Grothendieck construction as follows. Write

$$\Sigma(\mathbb{C}) = \mathbb{C}$$

$$\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_n) = \Sigma(\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_{n-1}), P_n).$$

Here $P_{k+1} \in \mathrm{PSh}(\Sigma(\mathbb{C}, P_1, \ldots, P_k))$ for each $k = 0, \ldots, n-1$. Note that

$$\pi_{P_{k+1}} : \Sigma(\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_k), P_{k+1}) \longrightarrow \Sigma(\mathbb{C}, P_1, P_2, \ldots, P_k).$$

Using these projections, define the iterated first projection functor

$$\pi^*_{P_1, P_2, \ldots, P_n} =_{\mathrm{def}} \pi_{P_1} \circ \pi_{P_2} \circ \cdots \circ \pi_{P_n} : \Sigma(\mathbb{C}, P_1, P_2, \ldots, P_n) \longrightarrow \mathbb{C}.$$

We explicate these constructions to see the connection to contexts of type theory. The objects of the category $\Sigma(\mathbb{C}, P_1, P_2, \dots, P_n)$ have the form $(\cdots((a, x_1), x_2), \dots, x_n)$ but we shall write them as $(a, x_1, \dots, x_n)$. Thus

$$a \in \mathbb{C}, x_1 \in P_1(a), x_2 \in P_2(a, x_1), \dots, x_n \in P_n(a, x_1, \dots, x_{n-1}).$$

We explicate these constructions to see the connection to contexts of type theory. The objects of the category $\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_n)$ have the form $(\cdots((a, x_1), x_2), \ldots, x_n)$ but we shall write them as $(a, x_1, \ldots, x_n)$. Thus

$$a \in \mathbb{C}, x_1 \in P_1(a), x_2 \in P_2(a, x_1), \ldots, x_n \in P_n(a, x_1, \ldots, x_{n-1}).$$

Proposition A morphism

$$\alpha : (a, x_1, \ldots, x_n) \longrightarrow (b, y_1, \ldots, y_n)$$

in $\Sigma(\mathbb{C}, P_1, P_2, \ldots, P_n)$ is given by a morphism $\alpha : a \longrightarrow b$ in $\mathbb{C}$ such that

$$P_1(\alpha)(y_1) = x_1, P_2(\alpha)(y_2) = x_2, \ldots, P_n(\alpha)(y_n) = x_n.$$

A category with attributes from iterated presheaves

Let $\mathbb{C}$ be a fixed small category.

Define a category $\mathrm{MPSh}(\mathbb{C}) = M$ to have as objects finite sequences $\overline{P} = [P_1, \ldots, P_n]$, $n \geq 0$, such that $P_{k+1} \in \mathrm{PSh}(\Sigma(\mathbb{C}, P_1, \ldots, P_k))$ for each $k = 0, \ldots, n-1$.

A category with attributes from iterated presheaves

Let $\mathbb{C}$ be a fixed small category.

Define a category $\mathrm{MPSh}(\mathbb{C}) = M$ to have as objects finite sequences $\overline{P} = [P_1, \dots, P_n]$, $n \geq 0$, such that $P_{k+1} \in \mathrm{PSh}(\Sigma(\mathbb{C}, P_1, \dots, P_k))$ for each $k = 0, \dots, n-1$.

Define the set of morphisms $\mathrm{Hom}_M(\overline{P}, \overline{Q})$ as a subset of the functors from $\Sigma(\overline{P})$ to $\Sigma(\overline{Q})$, as follows:

$$\mathrm{Hom}_M(\overline{P}, \overline{Q}) = \{f \in \Sigma(\overline{Q})^{\Sigma(\overline{P})} : \pi^*_{\overline{Q}} \circ f = \pi^*_{\overline{P}}\}. \qquad (1)$$

Notice that since $\pi^*_{\overline{Q}}(\beta) = \beta$ and $\pi^*_{\overline{P}}(\alpha) = \alpha$ for all arrows β in $\Sigma(\overline{Q})$, and α in $\Sigma(\overline{P})$, it holds for $f \in \mathrm{Hom}_M(\overline{P}, \overline{Q})$,

$$f(\alpha) = \alpha.$$

Theorem $\mathrm{MPSh}(\mathbb{C})$ is a category with terminal object $[]$.

The restriction in the hom-sets of $\mathrm{MPSh}(\mathbb{C})$ yields the following characterization.

Lemma For $P, Q \in \mathrm{PSh}(\mathbb{C})$ there is a bijection

$$\mathrm{Hom}_{\mathrm{MPSh}(\mathbb{C})}([P],[Q]) \cong \mathrm{Hom}_{\mathrm{PSh}(\mathbb{C})}(P, Q).$$

The restriction in the hom-sets of $\mathrm{MPSh}(\mathbb{C})$ yields the following characterization.

Lemma For $P, Q \in \mathrm{PSh}(\mathbb{C})$ there is a bijection

$$\mathrm{Hom}_{\mathrm{MPSh}(\mathbb{C})}([P],[Q]) \cong \mathrm{Hom}_{\mathrm{PSh}(\mathbb{C})}(P,Q).$$

Theorem. $[\cdot] : \mathrm{PSh}(\mathbb{C}) \longrightarrow \mathrm{MPSh}(\mathbb{C})$ is a full and faithful functor.

Composing the Yoneda embedding with the above embedding we get:

Corollary. $[\cdot] \circ \mathbf{y} : \mathbb{C} \longrightarrow \mathrm{MPSh}(\mathbb{C})$ is a full and faithful functor.

Theorem Let $\overline{P} = [P_1, \ldots, P_n]$ and $\overline{Q} = [Q_1, \ldots, Q_m]$ be objects of $\mathrm{MPSh}(\mathbb{C})$. An $\mathrm{MPSh}(\mathbb{C})$-morphism $f : \overline{P} \longrightarrow \overline{Q}$ is given by m components $f_1, \ldots, f_m$, which are such that for objects $(a, \overline{x})$ of $\Sigma(\overline{P})$:

$$f(a, \overline{x}) = (a, f_1(a, \overline{x}), \ldots, f_m(a, \overline{x})) \tag{2}$$

and

$$f_1(a, \overline{x}) \in Q_1(a), f_2(a, \overline{x}) \in Q_2(a, f_1(a, \overline{x})), \ldots,$$
$$f_m(a, \overline{x}) \in Q_m(a, f_1(a, \overline{x}), \ldots, f_{m-1}(a, \overline{x}))$$

Theorem Let $\overline{P} = [P_1, \dots, P_n]$ and $\overline{Q} = [Q_1, \dots, Q_m]$ be objects of $\mathrm{MPSh}(\mathbb{C})$. An $\mathrm{MPSh}(\mathbb{C})$-morphism $f : \overline{P} \longrightarrow \overline{Q}$ is given by m components $f_1, \dots, f_m$, which are such that for objects $(a, \overline{x})$ of $\Sigma(\overline{P})$:

$$f(a, \overline{x}) = (a, f_1(a, \overline{x}), \dots, f_m(a, \overline{x})) \tag{2}$$

and

$$\begin{gathered} f_1(a, \overline{x}) \in Q_1(a), f_2(a, \overline{x}) \in Q_2(a, f_1(a, \overline{x})), \dots, \\ f_m(a, \overline{x}) \in Q_m(a, f_1(a, \overline{x}), \dots, f_{m-1}(a, \overline{x})) \end{gathered}$$

Moreover for each morphism $\alpha : (a, \overline{x}) \longrightarrow (b, \overline{y})$ in $\Sigma(\overline{P})$, the following naturality equations hold

$$\begin{aligned} Q_1(\alpha)(f_1(b, \overline{y})) &= f_1(a, P_1(\alpha)(y_1), \dots, P_n(\alpha)(y_n)) \\ &\vdots \\ Q_m(\alpha)(f_m(b, \overline{y})) &= f_m(a, P_1(\alpha)(y_1), \dots, P_n(\alpha)(y_n)) \end{aligned}$$

Example. For $R \in \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P}))$, the projection functor π_R is morphism $[\overline{P}, R] \longrightarrow \overline{P}$ in $\mathrm{MPSh}(\mathbb{C})$.

Example. For $R \in \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P}))$, the projection functor π_R is morphism $[\overline{P}, R] \longrightarrow \overline{P}$ in $\mathrm{MPSh}(\mathbb{C})$.

Example. Sections. Let $Q \in \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P}))$, where $\overline{P} = [P_1, \ldots, P_n]$. Consider a $\mathrm{MPSh}(\mathbb{C})$-morphism $s : \overline{P} \longrightarrow [\overline{P}, Q]$ which is a section of π_Q, that is, it satisfies $\pi_Q \circ s = \mathrm{id}_{\overline{P}}$. By Theorem above it follows that s is specified by s' such that

$$s(a, \overline{x}) = (a, \overline{x}, s'(a, \overline{x})),$$

where $s'(a, \overline{x}) \in Q(a, \overline{x})$ and $(a, \overline{x}) \in \Sigma(\mathbb{C}, \overline{P})$, and for $\alpha : (a, \overline{x}) \longrightarrow (b, \overline{y})$,

$$Q(\alpha)(s'(b, \overline{y})) = s'(a, P_1(\alpha)(y_1), \ldots, P_n(\alpha)(y_n)).$$

For $n = 0$, this is

$$Q(\alpha)(s'(b)) = s'(a).$$

An equivalence

Theorem. The category $\mathrm{MPSh}(\mathbb{C})$ is equivalent to $\mathrm{PSh}(\mathbb{C})$.

An equivalence

Theorem. The category $\mathrm{MPSh}(\mathbb{C})$ is equivalent to $\mathrm{PSh}(\mathbb{C})$.

Note: for $\mathbb{C} = 1$ this gives an equivalent to the category of sets.

An equivalence

Theorem. The category $\mathrm{MPSh}(\mathbb{C})$ is equivalent to $\mathrm{PSh}(\mathbb{C})$.

Note: for $\mathbb{C} = 1$ this gives an equivalent to the category of sets.

Proof. The following functor $\Sigma^* : \mathrm{MPSh}(\mathbb{C}) \longrightarrow \mathrm{PSh}(\mathbb{C})$ gives the equivalence

For any object $\overline{P}$ of $\mathrm{MPSh}(\mathbb{C})$ define the presheaf $\Sigma^*(\overline{P})$ on $\mathbb{C}$ by letting

$\Sigma^*(\overline{P})(a) = \{(x_1, \ldots, x_n) : x_1 \in P_1(a), \ldots, x_n \in P_n(a, x_1, \ldots, x_n)\}$,

and for $\alpha : b \longrightarrow a$, assigning

$$\Sigma^*(\overline{P})(\alpha)((x_1, \ldots, x_n)) = (P_1(\alpha)(x_1), \ldots, P_n(\alpha)(x_n)).$$

An equivalence

Theorem. The category $\mathrm{MPSh}(\mathbb{C})$ is equivalent to $\mathrm{PSh}(\mathbb{C})$.

Note: for $\mathbb{C} = 1$ this gives an equivalent to the category of sets.

Proof. The following functor $\Sigma^* : \mathrm{MPSh}(\mathbb{C}) \longrightarrow \mathrm{PSh}(\mathbb{C})$ gives the equivalence

For any object $\overline{P}$ of $\mathrm{MPSh}(\mathbb{C})$ define the presheaf $\Sigma^*(\overline{P})$ on $\mathbb{C}$ by letting

$$\Sigma^*(\overline{P})(a) = \{(x_1, \ldots, x_n) : x_1 \in P_1(a), \ldots, x_n \in P_n(a, x_1, \ldots, x_n)\},$$

and for $\alpha : b \longrightarrow a$, assigning

$$\Sigma^*(\overline{P})(\alpha)((x_1, \ldots, x_n)) = (P_1(\alpha)(x_1), \ldots, P_n(\alpha)(x_n)).$$

For $f : \overline{P} \longrightarrow \overline{Q}$ in $\mathrm{MPSh}(\mathbb{C})$ define a natural transformation

$$\Sigma^*(f) : \Sigma^*(\overline{P}) \longrightarrow \Sigma^*(\overline{Q})$$

by letting

$$\Sigma^*(f)_a((x_1, \ldots, x_n)) = (f_1(a, x_1, \ldots, x_n), \ldots, f_m(a, x_1, \ldots, x_n)).$$

Here $f_1, \ldots, f_m$ are as in equation (2). $\square$

The following will give the types in a context $\overline{P}$. Define for each $\overline{P} \in \mathrm{MPSh}(\mathbb{C})$,

$$\mathrm{T}(\overline{P}) = \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P})).$$

The following will give the types in a context $\overline{P}$. Define for each $\overline{P} \in \mathrm{MPSh}(\mathbb{C})$,

$$\mathrm{T}(\overline{P}) = \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P})).$$

For a morphism $f : \overline{Q} \longrightarrow \overline{P}$, and $S \in \mathrm{T}(\overline{P})$, let

$$\mathrm{T}(f)(S) = S \circ f \in \mathrm{T}(\overline{Q}).$$

Thus T is a contravariant functor. We write $S\{f\}$ for $S \circ f$.

The following will give the types in a context $\overline{P}$. Define for each $\overline{P} \in \mathrm{MPSh}(\mathbb{C})$,

$$\mathrm{T}(\overline{P}) = \mathrm{PSh}(\Sigma(\mathbb{C}, \overline{P})).$$

For a morphism $f : \overline{Q} \longrightarrow \overline{P}$, and $S \in \mathrm{T}(\overline{P})$, let

$$\mathrm{T}(f)(S) = S \circ f \in \mathrm{T}(\overline{Q}).$$

Thus T is a contravariant functor. We write $S\{f\}$ for $S \circ f$.
For $Q \in \mathrm{T}$ define its set of elements as the sections of π_Q

$$\mathrm{E}(\overline{P}, Q) = \{s : \overline{P} \longrightarrow [\overline{P}, Q] : \pi_Q \circ s = \mathrm{id}_{\overline{P}}\}$$

These data give rise to a natural category with attributes.

Theorem. Let $M = \mathrm{MPSh}(\mathbb{C})$ for a small category $\mathbb{C}$. For $S \in \mathrm{T}(\overline{P})$ and $f : \overline{Q} \longrightarrow \overline{P}$, the functor $q_{S,f} = q : [\overline{Q}, S \circ f] \longrightarrow [\overline{P}, S]$ defined by

$$q(a, \overline{x}, u) = (f(a, \overline{x}), u) \quad \text{and} \quad q(\alpha) = f(\alpha) \quad (\alpha : (a, \overline{x}, u) \longrightarrow (b, \overline{y}, v))$$

makes the following into a pullback square in M:

$$\begin{tikzcd}
\overline{Q}, S \circ f \arrow[r, "q_{S,f}"] \arrow[d, "\pi_{S\circ f}"'] & \overline{P}, S \arrow[d, "\pi_S"] \\
\overline{Q} \arrow[r, "f"'] & \overline{P}
\end{tikzcd} \tag{3}$$

with appropriate functoriality in f.

The Π-construction

Let $\overline{R} = [R_1, \ldots, R_n] \in \mathrm{MPSh}(\mathbb{C})$. Let $P \in \mathrm{PSh}(\Sigma(\overline{R}))$ and $Q \in \mathrm{PSh}(\Sigma(\overline{R}, P))$. Define a presheaf $\Pi(P, Q)$ over $\Sigma(\overline{R})$ as follows (cf. Kripke semantics):

For $(a, \overline{x}) \in \Sigma(\overline{R})$, let

$$\begin{aligned}
\Pi(P,Q)(a,\overline{x}) \quad = \quad \Big\{ & h \in (\Pi b \in \mathbb{C})(\Pi f : b \to a) \\
& (\Pi v \in P(b, \overline{R}(f)(\overline{x}))Q(b, \overline{R}(f)(\overline{x}), v) \mid \\
& \forall b \in \mathbb{C}, \forall f : b \to a, \forall v \in P(b, \overline{R}(f)(\overline{x})), \\
& \forall c \in \mathbb{C}, \forall \beta : c \to b, \\
& \quad Q(\beta)(h(b, f, v)) = h(c, f \circ \beta, P(\beta)(v)) \Big\}
\end{aligned}$$

The Π-construction

Let $\overline{R} = [R_1, \ldots, R_n] \in \mathrm{MPSh}(\mathbb{C})$. Let $P \in \mathrm{PSh}(\Sigma(\overline{R}))$ and $Q \in \mathrm{PSh}(\Sigma(\overline{R}, P))$. Define a presheaf $\Pi(P, Q)$ over $\Sigma(\overline{R})$ as follows (cf. Kripke semantics):

For $(a, \overline{x}) \in \Sigma(\overline{R})$, let

$$\begin{aligned}\Pi(P,Q)(a,\overline{x}) \quad = \quad \Big\{ h \in (\Pi b \in \mathbb{C})(\Pi f : b \to a) \\ (\Pi v \in P(b, \overline{R}(f)(\overline{x}))Q(b, \overline{R}(f)(\overline{x}), v) \mid \\ \forall b \in \mathbb{C}, \forall f : b \to a, \forall v \in P(b, \overline{R}(f)(\overline{x})), \\ \forall c \in \mathbb{C}, \forall \beta : c \to b, \\ Q(\beta)(h(b, f, v)) = h(c, f \circ \beta, P(\beta)(v)) \Big\}\end{aligned}$$

Here $\overline{R}(f)(\overline{x})$ is $R_1(f)(x_1), \ldots, R_n(f)(x_n)$.

For $\alpha : (a', \overline{x}') \to (a, \overline{x})$ and $h \in \Pi(P, Q)(a, \overline{x})$ define $\Pi(P, Q)(\alpha)(h) = h'$ by

$$h'(b, f, v) = h(b, \alpha \circ f, v), \tag{4}$$

for $b \in \mathbb{C}, f : b \to a', v \in P(b, \overline{R}(f)(\overline{x}'))$.

Σ-construction

Let $\overline{R} = [R_1, \ldots, R_n] \in \mathrm{MPSh}(\mathbb{C})$. Let $P \in \mathrm{PSh}(\Sigma(\overline{R}))$ and $Q \in \mathrm{PSh}(\Sigma(\overline{R}, P))$. We define a presheaf $\dot{\Sigma}(P, Q)$ over $\Sigma(\overline{R})$ as follows. For $(a, \overline{x}) \in \Sigma(\overline{R})$, let

$$\dot{\Sigma}(P, Q)(a, \overline{x}) = \{(u, v) : u \in P(a, \overline{x}), v \in Q(a, \overline{x}, u)\}.$$

For $\alpha : (a', \overline{x}') \to (a, \overline{x})$ and $h \in \dot{\Sigma}(P, Q)(a, \overline{x})$ define

$$\dot{\Sigma}(P, Q)(\alpha)(u, v) = (P(\alpha)(u), Q(\alpha)(v)).$$

Explication of the Π-case over a simple category

Suppose that $\mathbb{C}$ is the category $0 \to 2 \leftarrow 1$.

Let $\overline{R} = [R_1, \ldots, R_n] \in \mathrm{MPSh}(\mathbb{C})$. Let $P \in \mathrm{PSh}(\Sigma(\overline{R}))$ and $Q \in \mathrm{PSh}(\Sigma(\overline{R}, P))$. Now the definition of $\Pi(P, Q)$ simplifies for $a = 0, 1$, since there are only identity arrows into a, the naturality condition becomes void, so we have:

$$\begin{aligned} \Pi(P, Q)(a, \overline{x}) &= (\Pi b \in \mathbb{C})(\Pi f : b \to a)(\Pi v \in P(b, \overline{R}(f)(\overline{x})))Q(b, \overline{R}(f \\ &\cong (\Pi v \in P(b, \overline{x})Q(b, \overline{x}, v) \end{aligned}$$

Explication of the Π-case over a simple category

Suppose that $\mathbb{C}$ is the category $0 \to 2 \leftarrow 1$.

Let $\overline{R} = [R_1, \ldots, R_n] \in \mathrm{MPSh}(\mathbb{C})$. Let $P \in \mathrm{PSh}(\Sigma(\overline{R}))$ and $Q \in \mathrm{PSh}(\Sigma(\overline{R}, P))$. Now the definition of $\Pi(P, Q)$ simplifies for $a = 0, 1$, since there are only identity arrows into a, the naturality condition becomes void, so we have:

$$\begin{array}{rcl} \Pi(P,Q)(a,\overline{x}) & = & (\Pi b \in \mathbb{C})(\Pi f : b \to a)(\Pi v \in P(b, \overline{R}(f)(\overline{x})))Q(b, \overline{R}(f \\ & \cong & (\Pi v \in P(b, \overline{x})Q(b, \overline{x}, v) \end{array}$$

For $a = 2$, the naturality condition has a few nontrivial cases, which can be simplified to

$$\begin{array}{rcl} \Pi(P,Q)(2,\overline{x}) & = & \Big\{ h \in (\Pi b \in \mathbb{C})(\Pi f : b \to 2) \\ & & \quad (\Pi v \in P(b, \overline{R}(f)(\overline{x}))Q(b, \overline{R}(f)(\overline{x}), v) \mid \\ & & \quad \forall v \in P(2, \overline{x}), \\ & & \qquad Q(f_{02})(h(2, \mathrm{id}_2, v)) = h(0, f_{02}, P(f_{02})(v)), \\ & & \qquad Q(f_{12})(h(2, \mathrm{id}_2, v)) = h(1, f_{12}, P(f_{12})(v)) \Big\} \end{array}$$

The constructions are still not standard intensional type theory since there are equations involved in some type formers. Can these be eliminated?

The constructions are still not standard intensional type theory since there are equations involved in some type formers. Can these be eliminated?

Basic pairs as a model of type theory.

Martin-Löf 2012 (seminar), Tonelli 2013 (MSc Thesis, Padova/Stockholm):

The judgement $A \ \mathrm{type}$ is interpreted as the following three standard judgements

$A^0 \ \mathrm{type}$
$A^1 \ \mathrm{type}$
$x : A^0, y : A^1 \Longrightarrow A^2(x, y) \ \mathrm{type}$

The constructions are still not standard intensional type theory since there are equations involved in some type formers. Can these be eliminated?

Basic pairs as a model of type theory.

Martin-Löf 2012 (seminar), Tonelli 2013 (MSc Thesis, Padova/Stockholm):

The judgement $A\ \mathrm{type}$ is interpreted as the following three standard judgements

$A^0\ \mathrm{type}$
$A^1\ \mathrm{type}$
$x : A^0, y : A^1 \Longrightarrow A^2(x, y)\ \mathrm{type}$

Now $(x) : (x : A^0, y : A^1, z : A^2(x, y)) \to (x : A^0)$ and $(y) : (x : A^0, y : A^1, z : A^2(x, y)) \to (y : A^1)$ so the triple (A^0, A^1, A^2) can be considered as a presheaf over $0 \to 2 \leftarrow 1$.

General contexts in the basic pair model are triples of contexts:

$$x_1 : A^0_1, \dots, x_n : A^0_n(x_1, \dots, x_{n-1}) \text{ context}$$

$$y_1 : A^1_1, \dots, y_n : A^1_n(y_1, \dots, y_{n-1}) \text{ context}$$

$$\begin{aligned} x_1 : A^0_1, y_1 : A^1_1, z_1 : A^2_1(x_1, y_1), \dots, \\ x_n : A^0_n(x_1, \dots, x_{n-1}), y_n : A^1_n(y_1, \dots, y_{n-1}), \\ z_n : A^2_n(x_1, y_1, z_1, \dots, x_{n-1}, y_{n-1}, z_{n-1}, x_n, y_n) \text{ context.} \end{aligned}$$

Martin-Löf's Π-relation type in Coq

```
Inductive Pi (A: Set)(B: A -> Set) :Set :=
  lamPi: (forall b:(forall x:A, B x), Pi A B).

Inductive PiRel (A: Set)(A': Set)(P:A -> A'-> Set)
            (B: A -> Set)(B': A' -> Set)
            (Q: (forall x:A, forall x':A', forall p:P x x',
                 B x -> B' x' -> Set))
            : (Pi A B) -> (Pi A' B') -> Set :=
   lamPiRel:
           (forall b: (forall x:A, B x),
            forall b': (forall x:A', B' x),
            forall q: (forall x:A, forall x':A',
                               forall p: P x x',
                               Q x x' p (b x) (b' x')),
      PiRel A A' P B B' Q  (lamPi A B b)(lamPi A' B' b')).
```

Makkai's presentation of dependent type signatures

Makkai (1995) uses a categorical way of presenting dependent types in his seminal version of dependently sorted logic, FOLDS.

Makkai's presentation of dependent type signatures

Makkai (1995) uses a categorical way of presenting dependent types in his seminal version of dependently sorted logic, FOLDS.

A FOLDS signature is a category K, which is

- skeletal (isomorphic objects are identical)
- one-way (no endomorphisms except identities)
- finitely branching (only finitely many arrows are going out from each object)

Makkai's presentation of dependent type signatures

Makkai (1995) uses a categorical way of presenting dependent types in his seminal version of dependently sorted logic, FOLDS.

A FOLDS signature is a category K, which is

- skeletal (isomorphic objects are identical)
- one-way (no endomorphisms except identities)
- finitely branching (only finitely many arrows are going out from each object)

Any functor $K \to \mathrm{Sets}$ (i.e.presheaf on K^{op}) gives a model of dependent sets.

Examples:

$K = 0 \leftarrow 2 \rightarrow 1$ is the example above.

$K = \Delta_{semi}^{op}$ gives semisimplicial sets

A method for translating from Makkai's presentation to standard type-theoretic presentation:

Define a binary relation $\leq$ on the objects of K by

$$A \leq B \Longleftrightarrow_{\mathrm{def}} \text{there is a morphism from } B \text{ to } A.$$

By the properties of K this is a well-founded partial order. Let $\leq^*$ be a linear order that extends $\leq$.

A method for translating from Makkai's presentation to standard type-theoretic presentation:

Define a binary relation $\leq$ on the objects of K by

$$A \leq B \Longleftrightarrow_{\mathrm{def}} \text{there is a morphism from } B \text{ to } A.$$

By the properties of K this is a well-founded partial order. Let $\leq^*$ be a linear order that extends $\leq$. Construct for every object A a non-repeating enumeration

$$x_1^A, \ldots, x_{n(A)}^A$$

of all the non-identity morphisms with domain A, so that

$$\mathrm{cod}(x_i^A) \leq^* \mathrm{cod}(x_j^A) \text{ whenever } i < j.$$

We shall below define a signature where the objects of K are the type symbols and where the morphisms of K serve as variables in the declarations.
For each object A of K we will have a type declaration$\mathcal{D}_A$:

$$\Gamma_A \Longrightarrow A(\mathrm{Var}(\Gamma_A))\ \mathrm{type}$$

by induction on $\leq$.

If $\Gamma_A = y_1 : B_1, \ldots, y_m : B_m$, the declaration says more explicitly

$$y_1 : B_1, \ldots, y_m : B_m \Longrightarrow A(y_1, \ldots, y_m)\ \mathrm{type}$$

where A is the type symbol declared.

If $\sigma = u_1, \ldots, u_n$ is a sequence of morphisms in K whose all of whose domains are equal to the codomain of $v \in \mathrm{Arr}\ K$, we write σ^v for

$$u_1 v, \ldots, u_n v.$$

If $\sigma = u_1, \ldots, u_n$ is a sequence of morphisms in K whose all of whose domains are equal to the codomain of $v \in \mathrm{Arr}\ K$, we write σ^v for

$$u_1 v, \ldots, u_n v.$$

Let

$$\Gamma_A = x_1 : C_1(\mathrm{Var}(\Gamma_{C_1})^{x_1}), \ldots, x_n : C_n(\mathrm{Var}(\Gamma_{C_n})^{x_n})$$

where $x_i = x_i^A$, $C_i = \mathrm{cod}(x_i)$ and $n = n(A)$.

If $\sigma = u_1, \ldots, u_n$ is a sequence of morphisms in K whose all of whose domains are equal to the codomain of $v \in \mathrm{Arr}\, K$, we write σ^v for

$$u_1 v, \ldots, u_n v.$$

Let

$$\Gamma_A = x_1 : C_1(\mathrm{Var}(\Gamma_{C_1})^{x_1}), \ldots, x_n : C_n(\mathrm{Var}(\Gamma_{C_n})^{x_n})$$

where $x_i = x_i^A$, $C_i = \mathrm{cod}(x_i)$ and $n = n(A)$.

If

$$B_1 <^* B_2 <^* \cdots <^* B_N <^* \cdots$$

are all the objects of K, then the pre-signature corresponding to the vocabulary K is

$$\Sigma_K = [\mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_N}, \ldots].$$

Examples

For $K = 0 \leftarrow 2 \rightarrow 1$ with the order $0 <^* 1 <^* 2$ we get

$$\begin{aligned} &\implies 0 \text{ type} \\ &\implies 1 \text{ type} \\ f_{20} : 0, f_{21} : 1 &\implies 2(f_{20}, f_{21}) \text{ type} \end{aligned}$$

Note: (awkwardly) f_{20}, f_{21} are variable symbols, and $0, 1, 2$ are type symbols.

The product category $K_n = K \times [n]$ gives the general forms of the types in contexts in the basic pair model. Here $[n]$ is the category corresponding to the ordinal n.

Examples

$$l, r : \mathrm{Eq} \to \mathrm{Hom} \qquad d, c : \mathrm{Hom} \to \mathrm{Ob}.$$

with $dl = dr$ and $cl = cr$.
Order consistently

$$\mathrm{Ob} <^* \mathrm{Hom} <^* \mathrm{Eq}$$

Examples

$$l, r : \mathrm{Eq} \to \mathrm{Hom} \qquad d, c : \mathrm{Hom} \to \mathrm{Ob}.$$

with $dl = dr$ and $cl = cr$.
Order consistently

$$\mathrm{Ob} <^* \mathrm{Hom} <^* \mathrm{Eq}$$

Then we get

$$\begin{aligned} &\Longrightarrow \quad \mathrm{Ob\ type} \\ d, c : \mathrm{Ob} &\Longrightarrow \quad \mathrm{Hom}(d, c)\ \mathrm{type} \\ dl, cl : \mathrm{Ob}, l : \mathrm{Hom}(d, c)^l, r : \mathrm{Hom}(d, c)^r &\Longrightarrow \quad E(l, k)\ \mathrm{type} \end{aligned}$$

Mulitplying, using the equalities of the category, and the renaming $x = dl$, $y = cl$ gives as the last line

$$x, y : \mathrm{Ob}, l : \mathrm{Hom}(x, y), r : \mathrm{Hom}(x, y) \Longrightarrow E(l, k)\ \mathrm{type}$$

Next we can use the notion of homomorphism between signatures of dependent types of Cartmell 1986, to obtain a counterpart to natural transformations. Thus we get a type theory corresponding to $\mathrm{PSh}(K^{\mathrm{op}})$.

Next we can use the notion of homomorphism between signatures of dependent types of Cartmell 1986, to obtain a counterpart to natural transformations. Thus we get a type theory corresponding to $\mathrm{PSh}(K^{\mathrm{op}})$.

A possible direction would be to extend Martin-Löf's and Tonelli's model to general FOLDS signatures K using the method above. This ought to be a model of kind α, interpreting definitional equalities as definitional equalities in some MLTT extended with certain generalized inductive types.

III. Formalizing models of type theory in type theory

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

III. Formalizing models of type theory in type theory

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.
- 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

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.
- 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).

Definition

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

$$\sigma\{f\} \text{ for } T(f)(\sigma).$$

Definition

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

$$\sigma\{f\} \text{ for } T(f)(\sigma).$$

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

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

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) The final datum tells how substitutions interact with context extensions:

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

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

is a pullback, and furthermore functoriality conditions

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

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

is a pullback, and furthermore functoriality conditions

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

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

Example:

In set theory (ZF or CZF) we may construct a cwa from any set U which contains a singleton set and is Σ-*closed* in the sense that if $A \in U$ and $F : A \longrightarrow U$ is any function then the following Σ-set belongs to U

$$\Sigma_{x \in A} F(x) = \{\langle x, y \rangle : x \in A, y \in F(x)\}.$$

Then we can take $\mathcal{C}$ to be the full subcategory of sets with objects in U. It is small. Moreover

$$T =_{\mathrm{def}} \mathrm{Set}(\cdot, U) : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}.$$

For $\Gamma \in \mathcal{C}$ and $\sigma \in T(\Gamma)$

$$\Gamma.\sigma =_{\mathrm{def}} \Sigma_{x \in \Gamma} \sigma(x).$$

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

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

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 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.

ecwas - 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*

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

and

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

and moreover

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

ecwas - 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*

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

and

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

and moreover

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

They follow from the functoriality of T and by requiring the object equality $\Gamma.\sigma = \Gamma.\sigma'$ and $\mathsf{p}(\sigma) = \mathsf{p}(\sigma')$ whenever $\sigma = \sigma'$.

ecwas - 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*

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

and

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

and moreover

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

They follow from the functoriality of T and by requiring the object equality $\Gamma.\sigma = \Gamma.\sigma'$ and $\mathsf{p}(\sigma) = \mathsf{p}(\sigma')$ whenever $\sigma = \sigma'$.

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 1.

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

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

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 1.

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

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

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 $\sigma =_{T(\Gamma)} \sigma'$, $\Delta_\Gamma(t) : \Gamma.\sigma \longrightarrow \Gamma.\sigma'$ is an isomorphism such that

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

commutes.

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

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

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

$$\Delta_\Gamma(t) = 1_{\Gamma.\sigma} \qquad (t \text{ proof of } \sigma =_{T(\Gamma)} \sigma)$$

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

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

(d) For each $f : B \longrightarrow \Gamma$ and $\sigma \in T(\Gamma)$, there is a morphism $\mathrm{q}(f, \sigma) : B.\sigma\{f\} \longrightarrow \Gamma.\sigma$ in $\mathcal{C}$ such that

$$\begin{tikzcd}
B.\sigma\{f\} \arrow[r, "\mathrm{q}(f,\sigma)"] \arrow[d, "\mathrm{p}(\sigma\{f\})"'] & \Gamma.\sigma \arrow[d, "\mathrm{p}(\sigma)"] \\
B \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

is a pullback, and moreover these morphisms satisfy

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

(d.2) $\mathrm{q}(f \circ g, \sigma) \circ \Delta_A(t) = \mathrm{q}(f, \sigma) \circ \mathrm{q}(g, \sigma\{f\})$ for $A \xrightarrow{g} B \xrightarrow{f} \Gamma$ and where t is any pf. for $\sigma\{f \circ g\} =_{TA} \sigma\{f\}\{g\}$.

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

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

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

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

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

$$\mathrm{q}(f, \sigma) \circ \Delta_B(t) = \Delta_\Gamma(s) \circ \mathrm{q}(f', \sigma').$$

Equality on finite contexts. Note that if $f : B \longrightarrow \Gamma$ is an isomotphism, and $\sigma \in T(B)$, $\tau \in T(\Gamma)$ and t is proof of $\sigma =_{T(B)} \tau\{f\}$, then

$$\mathrm{q}(f, \tau) \circ \Delta(t) : B.\sigma \longrightarrow \Gamma.\tau$$

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

Interpretation

The definition of ecwas suggests introducing the following judgements about types

- $\Gamma \vdash \sigma \text{ type}$ meaning $\sigma \in T(\Gamma)$
- $\Gamma \vdash \sigma = \sigma'$ meaning $\sigma =_{T(\Gamma)} \sigma'$ where $\sigma, \sigma' \in T(\Gamma)$.

Interpretation

The definition of ecwas suggests introducing the following judgements about types

- $\Gamma \vdash \sigma \ \mathrm{type}$ meaning $\sigma \in T(\Gamma)$
- $\Gamma \vdash \sigma = \sigma'$ meaning $\sigma =_{T(\Gamma)} \sigma'$ where $\sigma, \sigma' \in T(\Gamma)$.

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

Note that if r is a proof for $\sigma =_{T(\Gamma)} \sigma'$, then $M \in E(\Gamma, \sigma)$ implies $\Delta_\Gamma(r) \circ M \in E(\Gamma, \sigma')$.

(Cf. canonical isomorphisms of Curien 1993.)

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

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

- $\Gamma \vdash (M, \sigma) : \sigma'$ meaning

  $M \in E(\Gamma, \sigma)$ and $\sigma =_{T(\Gamma)} \sigma'$.

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

- $\Gamma \vdash (M, \sigma) : \sigma'$ meaning

  $M \in E(\Gamma, \sigma)$ and $\sigma =_{T(\Gamma)} \sigma'$.

- $\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$ meaning

  $\Gamma \vdash (M, \sigma) : \sigma''$ and $\Gamma \vdash (M, \sigma') : \sigma''$ and that there is a proof r of $\sigma =_{T(\Gamma)} \sigma'$ such that $\Delta_\Gamma(r) \circ M =_{E(\Gamma, \sigma')} M'$.

The rules

$$\boxed{\frac{\Gamma \vdash s : \sigma \qquad \Gamma \vdash \sigma = \tau}{\Gamma \vdash s : \tau}}$$

$$\boxed{\frac{\Gamma \vdash s = t : \sigma \qquad \Gamma \vdash \sigma = \tau}{\Gamma \vdash s = t : \tau}}$$

are immediately justified.

Substitution into terms

For $M \in E(\Gamma, \sigma)$ and $f : B \longrightarrow \Gamma$, define $M\{f\} : B \longrightarrow B.\sigma\{f\}$ as the unique morphism (see diagram below) with $\mathsf{p}(\sigma\{f\})) \circ M\{f\} = 1_B$ and $\mathsf{q}(f, \sigma) \circ M\{f\} = M \circ f$. Thus $M\{f\} \in E(B, \sigma\{f\})$.

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

Stability under substitution

Thus if

$$\Gamma \vdash (M, \sigma) : \sigma'$$

and $f : B \longrightarrow \Gamma$, we have $M\{f\} \in E(B, \sigma\{f\})$ and $\sigma\{f\} =_{T(B)} \sigma'\{f\}$, so

$$B \vdash (M\{f\}, \sigma\{f\}) : \sigma'\{f\}$$

Thus the rule

$$\boxed{\frac{\Gamma \vdash s : \sigma' \qquad f : B \longrightarrow \Gamma}{B \vdash s\{f\} : \sigma'\{f\}}}$$

is justified.

Moreover, if

$$\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$$

we have $\Delta_\Gamma(r) \circ M =_{E(\Gamma,\sigma')} M'$ for some proof r of $\sigma =_{T(\Gamma)} \sigma'$.
Then

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

Moreover, if

$$\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$$

we have $\Delta_\Gamma(r) \circ M =_{E(\Gamma,\sigma')} M'$ for some proof r of $\sigma =_{T(\Gamma)} \sigma'$. Then

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

and hence

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

Moreover, if

$$\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$$

we have $\Delta_\Gamma(r) \circ M =_{E(\Gamma,\sigma')} M'$ for some proof r of $\sigma =_{T(\Gamma)} \sigma'$. Then

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

and hence

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

So by remark above there is a proof s of $T(f)(\sigma) = T(f)(\sigma')$ such that $\Delta_\Gamma(r) \circ \mathsf{q}(f, \sigma) = \mathsf{q}(f, \sigma') \circ \Delta_B(s)$ and hence

$$\mathsf{q}(f, \sigma') \circ \Delta_B(s) \circ M\{f\} = M' \circ f.$$

Moreover, if

$$\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$$

we have $\Delta_\Gamma(r) \circ M =_{E(\Gamma,\sigma')} M'$ for some proof r of $\sigma =_{T(\Gamma)} \sigma'$. Then

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

and hence

$$\Delta_\Gamma(r) \circ \mathrm{q}(f, \sigma) \circ M\{f\} = M' \circ f.$$

So by remark above there is a proof s of $T(f)(\sigma) = T(f)(\sigma')$ such that $\Delta_\Gamma(r) \circ \mathrm{q}(f, \sigma) = \mathrm{q}(f, \sigma') \circ \Delta_B(s)$ and hence

$$\mathrm{q}(f, \sigma') \circ \Delta_B(s) \circ M\{f\} = M' \circ f.$$

It follows by uniqueness that $\Delta_B(s) \circ M\{f\} = M'\{f\}$, so indeed

$$B \vdash (M\{f\}, \sigma\{f\}) = (M'\{f\}, \sigma'\{f\}) : \sigma''\{f\}.$$

Moreover, if

$$\Gamma \vdash (M, \sigma) = (M', \sigma') : \sigma''$$

we have $\Delta_\Gamma(r) \circ M =_{E(\Gamma,\sigma')} M'$ for some proof r of $\sigma =_{T(\Gamma)} \sigma'$. Then

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

and hence

$$\Delta_\Gamma(r) \circ \mathrm{q}(f, \sigma) \circ M\{f\} = M' \circ f.$$

So by remark above there is a proof s of $T(f)(\sigma) = T(f)(\sigma')$ such that $\Delta_\Gamma(r) \circ \mathrm{q}(f, \sigma) = \mathrm{q}(f, \sigma') \circ \Delta_B(s)$ and hence

$$\mathrm{q}(f, \sigma') \circ \Delta_B(s) \circ M\{f\} = M' \circ f.$$

It follows by uniqueness that $\Delta_B(s) \circ M\{f\} = M'\{f\}$, so indeed

$$B \vdash (M\{f\}, \sigma\{f\}) = (M'\{f\}, \sigma'\{f\}) : \sigma''\{f\}.$$

This justifies also the rule:

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

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore for any $M \in E(\Gamma, \Pi(\sigma, \tau))$ and any $N \in E(\Gamma, \sigma)$ there is an element

$$\mathsf{App}_{\sigma,\tau}(M, N) \in E(\Gamma, \tau\{N\}),$$

such that

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore for any $M \in E(\Gamma, \Pi(\sigma, \tau))$ and any $N \in E(\Gamma, \sigma)$ there is an element

$$\mathsf{App}_{\sigma,\tau}(M, N) \in E(\Gamma, \tau\{N\}),$$

such that the following equations hold for any subst. $f : B \longrightarrow \Gamma$:

- (β-red) $\mathsf{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P), N) = P\{N\}$,

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore for any $M \in E(\Gamma, \Pi(\sigma, \tau))$ and any $N \in E(\Gamma, \sigma)$ there is an element

$$\mathrm{App}_{\sigma,\tau}(M, N) \in E(\Gamma, \tau\{N\}),$$

such that the following equations hold for any subst. $f : B \longrightarrow \Gamma$:

- (β-red) $\mathrm{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P), N) = P\{N\}$,
- (Π-subst) $\Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathrm{q}(f, \sigma)\})$,

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore for any $M \in E(\Gamma, \Pi(\sigma, \tau))$ and any $N \in E(\Gamma, \sigma)$ there is an element

$$\mathrm{App}_{\sigma,\tau}(M, N) \in E(\Gamma, \tau\{N\}),$$

such that the following equations hold for any subst. $f : B \longrightarrow \Gamma$:

- (β-red) $\mathrm{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P), N) = P\{N\}$,
- (Π-subst) $\Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathsf{q}(f, \sigma)\})$,
- (λ-subst) $\lambda_{\sigma,\tau}(P)\{f\} = \lambda_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(P\{\mathsf{q}(f, \sigma)\})$,

From (Hofmann 1994) we take the following definition, but adapt it in the obvious way to cwas.

A cwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type

$$\Pi(\sigma, \tau) \in T(\Gamma),$$

and moreover for every $P \in E(\Gamma.\sigma, \tau)$ there is an element

$$\lambda_{\sigma,\tau}(P) \in E(\Gamma, \Pi(\sigma, \tau)),$$

and furthermore for any $M \in E(\Gamma, \Pi(\sigma, \tau))$ and any $N \in E(\Gamma, \sigma)$ there is an element

$$\mathsf{App}_{\sigma,\tau}(M, N) \in E(\Gamma, \tau\{N\}),$$

such that the following equations hold for any subst. $f : B \longrightarrow \Gamma$:

- (β-red) $\mathsf{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P), N) = P\{N\}$,
- (Π-subst) $\Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathsf{q}(f, \sigma)\})$,
- (λ-subst) $\lambda_{\sigma,\tau}(P)\{f\} = \lambda_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(P\{\mathsf{q}(f, \sigma)\})$,
- (App-subst) $\mathsf{App}_{\sigma,\tau}(M, N)\{f\} = \mathsf{App}_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(M\{f\}, N\{f\})$.

Adapting this to ecwas ... the first part is similar

An ecwa supports Π-types if for $\sigma \in T(\Gamma)$ and $\tau \in T(\Gamma.\sigma)$ there is a type $\Pi(\sigma,\tau) \in T(\Gamma)$, and moreover for every $P \in E(\Gamma.\sigma,\tau)$ there is an element $\lambda_{\sigma,\tau}(P) \in E(\Gamma,\Pi(\sigma,\tau))$, and furthermore for any $M \in E(\Gamma,\Pi(\sigma,\tau))$ and any $N \in E(\Gamma,\sigma)$ there is an element $\mathsf{App}_{\sigma,\tau}(M,N) \in E(\Gamma,\tau\{N\})$, such that the following equations hold for any $f : B \longrightarrow \Gamma$:

- (β-red) $\mathsf{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P),N) =_{E(\Gamma,\tau\{N\})} P\{N\}$, [as before]
- (Π-subst) $\Pi(\sigma,\tau)\{f\} =_{T(B)} \Pi(\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\})$, [as before]

This part of the definition has type adjustments:

- (λ-subst)

  $$\lambda_{\sigma,\tau}(P)\{f\} =_{E(B,\Pi(\sigma,\tau)\{f\})} \Delta_B(t)\circ\lambda_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(P\{\mathsf{q}(f,\sigma)\}),$$

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

- (λ-subst)

  $$\lambda_{\sigma,\tau}(P)\{f\} =_{E(B,\Pi(\sigma,\tau)\{f\})} \Delta_B(t)\circ\lambda_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(P\{\mathsf{q}(f,\sigma)\}),$$

  for any proof t of $\Pi(\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}) =_{T(B)} \Pi(\sigma,\tau)\{f\}$,
- (App-subst)

  $$\mathsf{App}_{\sigma,\tau}(M,N)\{f\} =_{E(\cdots)} \Delta_B(s)\circ\mathsf{App}_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(\Delta_B(t)\circ M\{f\}, N\{$$

  for any proof s of $\tau\{\mathsf{q}(f,\sigma)\}\{N\{f\}\} =_{T(B)} \tau\{N\}\{f\}$ and any proof t of $\Pi(\sigma,\tau)\{f\} =_{T(B)} \Pi(\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\})$.

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

- (Π-cong) if s is a proof of $\sigma =_{T(\Gamma)} \sigma'$ and for $\tau \in T(\Gamma.\sigma)$, $\tau' \in T(\Gamma.\sigma')$ with $\tau =_{T(\Gamma.\sigma)} \tau'\{\Delta(s)\}$, then

$$\Pi(\sigma, \tau) =_{T(\Gamma)} \Pi(\sigma', \tau').$$

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

- (Π-cong) if s is a proof of $\sigma =_{T(\Gamma)} \sigma'$ and for $\tau \in T(\Gamma.\sigma)$, $\tau' \in T(\Gamma.\sigma')$ with $\tau =_{T(\Gamma.\sigma)} \tau'\{\Delta(s)\}$, then
$$\Pi(\sigma, \tau) =_{T(\Gamma)} \Pi(\sigma', \tau').$$
- (λ-cong) if $P =_{E(\Gamma.\sigma,\tau)} P'$, then
$$\lambda_{\sigma,\tau}(P) =_{E(\Gamma,\Pi(\sigma,\tau))} \lambda_{\sigma,\tau}(P')$$

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

- (Π-cong) if s is a proof of $\sigma =_{T(\Gamma)} \sigma'$ and for $\tau \in T(\Gamma.\sigma)$, $\tau' \in T(\Gamma.\sigma')$ with $\tau =_{T(\Gamma.\sigma)} \tau'\{\Delta(s)\}$, then

$$\Pi(\sigma, \tau) =_{T(\Gamma)} \Pi(\sigma', \tau').$$

- (λ-cong) if $P =_{E(\Gamma.\sigma,\tau)} P'$, then

$$\lambda_{\sigma,\tau}(P) =_{E(\Gamma,\Pi(\sigma,\tau))} \lambda_{\sigma,\tau}(P')$$

- (App-cong) if $M =_{E(\Gamma,\Pi(\sigma,\tau))} M'$ and $N =_{E(\Gamma,\sigma)} N'$ then

$$\Delta_\Gamma(s) \circ \mathsf{App}_{\sigma,\tau}(M, N) =_{E(\Gamma,\tau\{N'\})} \mathsf{App}_{\sigma,\tau}(M', N'),$$

  where s is any proof of $\tau\{N\} =_{T(\Gamma)} \tau\{N'\}$.

IV. First-order logic with dependent types

A dependently typed (or sorted) system of first-order logic, FOLDS, was introduced and studied by Makkai (1995, 1998, 2013).

The purpose of FOLDS is to provide a natural logical system for formalizing (higher) category theory.

FOLDS can be considered as a so-called logic enriched type theory (Maietti-Sambin 2005, Gambino-Aczel 2006) where the underlying type theory is very rudimentary.

We present a system like FOLDS which is essentially due to Belo (2008) and based on Cartmell's (1986) generalized algebraic theories but without equality of types. Unlike Makkai we allow function symbols.

Single-sorted vs multi-sorted first-order logic

The standard classical (or intutionistic) first-order logic assumes one non-empty (inhabited) domain of quantification. The latter restriction is due to the formulation of the existence introduction rule

$$\frac{\vdash \top[t/x]}{\vdash \exists x.\top}\ (\exists I) \tag{5}$$

(we may let $t = x$)
In many-sorted logics, especially those used in the categorical logic, one specifies the possible free variables of involved formulas

$$\frac{x_1 : A_1, \ldots, x_n : A_n; \Delta \vdash \psi[t/x]}{x_1 : A_1, \ldots, x_n : A_n; \Delta \vdash (\exists x : A)\psi}\ (\exists I)$$

if $FV(t) \subseteq \{x_1, \ldots, x_n\}$ and t has sort A. The derivation (5) is not possible for $n = 0$.

Many-sorted logic can be formulated (sequent-style) using judgements of the form

$$\Gamma; \Delta \vdash \phi$$

where

$$\Gamma = x_1 : A_1, \ldots, x_n : A_n$$

is a sequence of variables x_i with associated sort A_i, and where

$$\Delta = \phi_1, \ldots, \phi_m$$

is a sequence of formulas assumed to be true, and with their free variables listed in Γ, the formula proved true ϕ has also its free variables in Γ.

In standard many-sorted logic the sorts A_i belong to a given set of sorts and the order of the variables x_i does not matter

$$x_1 : A_1, \ldots, x_n : A_n; \Delta \vdash \phi$$

In standard many-sorted logic the sorts A_i belong to a given set of sorts and the order of the variables x_i does not matter

$$x_1 : A_1, \ldots, x_n : A_n; \Delta \vdash \phi$$

We may generalize this situation by letting the sort or types be given by a type theory. This is the idea of *logic enriched type theory* (Gambino and Aczel 2006).

We introduce three new judgement forms

$A \text{ type}$	— A is a type
$a : A$	— a is an element (term) of type A
$\phi \text{ formula}$	

... and their hypothetical versions

$$x_1 : A_1, \ldots, x_n : A_n \Longrightarrow A \text{ type}$$

$$x_1 : A_1, \ldots, x_n : A_n \Longrightarrow a : A$$

$$x_1 : A_1, \ldots, x_n : A_n \Longrightarrow \phi \text{ formula}$$

We will consider a type theory in which it is possible to introduce a finite number of dependent types and dependent function.

A representative example is the language of a category (cf. Cartmell 1986, Makkai 1995).

1. $\langle\rangle \Longrightarrow \mathsf{Ob}\ \mathrm{type}$
2. $x : \mathsf{Ob}, y : \mathsf{Ob} \Longrightarrow \mathsf{Hom}(x, y)\ \mathrm{type}$
3. $x : \mathsf{Ob} \Longrightarrow 1_x : \mathsf{Hom}(x, x)$
4. $x : \mathsf{Ob}, y : \mathsf{Ob}, z : \mathsf{Ob}, g : \mathsf{Hom}(y, z), f : \mathsf{Hom}(x, y) \Longrightarrow$ $g \circ_{x,y,z} f : \mathsf{Hom}(x, z)$

This is will be considered as a dependent signature declaring the constants Ob, Hom, 1 and $\circ$.
A complication is that we have to know that constants are declared in legal contexts, e.g. in (4) above we have to know its context is legal relative to (1) – (3).

Example 1: E-category

Signature as above but add an equality predicate only on Hom-types

$$x, y : \mathsf{Ob}, f, g : \mathsf{Hom}(x, y) \Longrightarrow f =_{x,y} g \text{ formula.}$$

Axioms in DTFOL/type theory

$$\begin{aligned}
&x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y); \vdash f =_{x,y} f.\\
&x, y : \mathsf{Ob}, f, g : \mathsf{Hom}(x, y); f =_{x,y} g \vdash g =_{x,y} .\\
&x, y, z : \mathsf{Ob}, f, g, h : \mathsf{Hom}(x, y); f =_{x,y} g, g =_{x,y} h \vdash f =_{x,y} h.\\
&x, y, z : \mathsf{Ob}, f, h : \mathsf{Hom}(x, y), g, k : \mathsf{Hom}(y, z);\\
&\qquad\qquad f =_{x,y} h, g =_{y,z} k \vdash g \circ_{x,y,z} f =_{x,z} k \circ_{x,y,z} h
\end{aligned}$$

$$\begin{aligned}
&x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y); \vdash 1_y \circ_{x,y,y} f =_{x,y} f.\\
&x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y); \vdash f \circ_{x,x,y} 1_x =_{x,y} f.\\
&x, y, z, w : \mathsf{Ob}, f : \mathsf{Hom}(x, y), g : \mathsf{Hom}(y, z), h : \mathsf{Hom}(z, w);\\
&\qquad\qquad \vdash h \circ_{x,z,w} (g \circ_{x,y,z} f) =_{x,w} (h \circ_{y,z,w} g) \circ_{x,y,w} f.
\end{aligned}$$

Example 2: E-bicategory

Signature for a bicategory (objects, arrows, transformations):

1. $\langle\rangle \Longrightarrow \mathsf{O}$ type
2. $x, y : \mathsf{O} \Longrightarrow \mathsf{A}(x, y)$ type
3. $x : \mathsf{O} \Longrightarrow 1_x : \mathsf{A}(x, x)$
4. $x, y, z : \mathsf{O}, g : \mathsf{A}(y, z), f : \mathsf{A}(x, y) \Longrightarrow g \circ_{x,y,z} f : \mathsf{A}(x, z).$
5. $x, y : \mathsf{O}, f, g : \mathsf{A}(x, y) \Longrightarrow \mathsf{T}(x, y, f, g)$ type
6. $x, y : \mathsf{O}, f : \mathsf{A}(x, y) \Longrightarrow i_f : \mathsf{T}(x, y, f, f)$ type
7. $x, y : \mathsf{O}, f, g, h : \mathsf{A}(x, y), \beta : \mathsf{T}(x, y, g, h), \alpha : \mathsf{T}(x, y, f, g) \Longrightarrow$
   $\beta \cdot_{x,y,f,g,h} \alpha : \mathsf{T}(x, y, f, h)$
8. $x, y, z : \mathsf{O}, f, h : \mathsf{A}(x, y), g, k : \mathsf{A}(y, z), \beta : \mathsf{T}(g, k), \alpha :$
   $\mathsf{T}(f, h) \Longrightarrow \beta *_{x,y,z,f,h,g,k} \alpha : \mathsf{T}(x, z, g \circ_{x,y,z} f, k \circ_{x,y,z} h)$
9. $x, y, z, w : \mathsf{O}, f : \mathsf{A}(x, y), g : \mathsf{A}(y, z), h : \mathsf{A}(z, w) \Longrightarrow$
   $\mathsf{a}_{x,y,z,w,f,g,h} : \mathsf{T}(x, w, (h \circ_{y,z,w} g) \circ_{x,y,w} f, h \circ_{x,z,w} (g \circ_{x,y,z} f))$
10. $x, y : \mathsf{O}, f : \mathsf{A}(x, y) \Longrightarrow \mathsf{l}_f : \mathsf{T}(x, y, 1_y \circ_{x,y,y} f, f)$
11. $x, y : \mathsf{O}, f : \mathsf{A}(x, y) \Longrightarrow \mathsf{r}_f : \mathsf{T}(x, y, f \circ_{x,x,y} 1_x, f)$

Further we add only congruence relations on the transformations:

$$x, y : \mathsf{O}, f, g : \mathsf{A}(x, y), \alpha, \beta : \mathsf{T}(x, y, f, g) \Longrightarrow \alpha =_{x,y,f,g} \beta \text{ formula.}$$

Depently typed first-order logic: Pre-syntax

We assume that F and T are two disjoint sets of symbols equipped with arities $0, 1, 2, \dots$.

Pre-terms: formed using F as function symbols and variables in the usual way

Pre-types: $S(t_1, \dots, t_n)$ where $S \in T$ is a type symbol, and $t_1, \dots, t_n$ are pre-terms, $n = \mathrm{ar}(S)$

Pre-contexts: $\langle x_1 : A_1, \dots, x_n : A_n \rangle$, where $x_1, \dots, x_n$ are distinct variables, $A_1, \dots, A_n$ are pre-types and $FV(A_i) \subseteq \{x_1, \dots, x_{i-1}\}$.

Pre-syntax (cont.)

Pre-declarations: Either
of types:

$$x_1 : A_1, \ldots, x_n : A_n \Longrightarrow S(x_1, \ldots, x_n) \text{ type}$$

encoded as a pair (Γ, S) where $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ pre-context,
$S \in T$ and $\mathsf{ar}(S) = |\Gamma|$,
or of functions:

$$x_1 : A_1, \ldots, x_n : A_n \Longrightarrow f(x_1, \ldots, x_n) : U$$

endcoded as a triple (Γ, f, U) where $\Gamma = x_1 : A_1, \ldots, x_n : A_n$
pre-context, U pre-type, $FV(U) \subseteq FV(\Gamma)$, $\mathsf{ar}(f) = |\Gamma|$.

Pre-signature: a finite sequence $[D_1, \ldots, D_n]$ of pre-declarations.

Legal syntax — syntactic judgements

Contexts, types, elements and signatures are generated by simultaneous induction given by rules (R1) – (R9)

$$\frac{\Sigma \text{ signature}}{\langle\rangle \text{ context } [\Sigma]} \text{ (R1)}$$

$$\frac{\Gamma \text{ context } [\Sigma] \quad \Gamma \Longrightarrow A \text{ type } [\Sigma]}{\Gamma, x : A \text{ context } [\Sigma]} \text{ (R2)}, x \in \mathit{Var} \setminus \mathrm{Var}(\Gamma)$$

$$\frac{x_1 : A_1 \ldots, x_n : A_n \text{ cont. } [\Sigma]}{x_1 : A_1 \ldots, x_n : A_n \Longrightarrow x_i : A_i \ [\Sigma]} \text{ (R4)}$$

To formulate the next two rules we introduce the notion of context map. Let Δ and $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ be two contexts relative to Σ. A sequence $(a_1, \ldots, a_n)$ of preterms is called a *context map* $\Delta \longrightarrow \Gamma$ if

$$\begin{array}{l}
\Delta \text{ context} \\
\Gamma \text{ context} \\
\Delta \Longrightarrow a_1 : A_1 \ [\Sigma] \\
\Delta \Longrightarrow a_2 : A_2[a_1/x_1] \ [\Sigma] \\
\quad \vdots \\
\Delta \Longrightarrow a_n : A_n[a_1 \ldots, a_{n-1}/x_1, \ldots, x_{n-1}] \ [\Sigma]
\end{array}$$

We write $(a_1, \ldots, a_n) : \Delta \longrightarrow \Gamma \ [\Sigma]$ for the conjunction of these judgements.

The sequence $(a_1, \ldots, a_n)$ can be plugged into correct function and type declarations according to the next rules.

$$\frac{\begin{array}{l}(a_1, \ldots, a_n) : \Delta \longrightarrow \langle x_1 : A_1, \ldots, x_n : A_n \rangle \ [\Sigma] \\ (\langle x_1 : A_1 \ldots, x_n : A_n \rangle, S) \text{ decl. in } \Sigma\end{array}}{\Delta \Longrightarrow S(a_1, \ldots, a_n) \text{ type } [\Sigma]} \ \text{(R5)}$$

$$\frac{\begin{array}{l}(a_1, \ldots, a_n) : \Delta \longrightarrow \langle x_1 : A_1, \ldots, x_n : A_n \rangle \ [\Sigma] \\ (\langle x_1 : A_1 \ldots, x_n : A_n \rangle, f, U) \text{ decl. in } \Sigma \\ \Delta \Longrightarrow U[a_1, \ldots, a_n / x_1, \ldots, x_n] \text{ type } [\Sigma]\end{array}}{\Delta \Longrightarrow f(a_1, \ldots, a_n) : U[a_1, \ldots, a_n / x_1, \ldots, x_n] \ [\Sigma]} \ \text{(R6)}$$

The final three rules are concerned with the formation of correct signatures, i.e. sequences of function and type declarations.

$$\frac{}{[]\ \text{signature}}\ (\text{R7})$$

$$\frac{\Sigma\ \text{sig.} \quad \Gamma\ \text{con.}\ [\Sigma] \quad S \in T \quad S\ \textsf{not decl. in}\ \Sigma \quad |\Gamma| = \text{ar}(S)}{[\Sigma, (\Gamma, S)]\ \text{sig.}}\ (\text{R8})$$

$$\frac{\Sigma\ \text{sig.} \quad \Gamma\ \text{con.}\ [\Sigma] \quad \Gamma \Longrightarrow U\ \text{type}\ [\Sigma] \quad f \in F\ \textsf{not decl. in}\ \Sigma \quad |\Gamma| = \text{ar}(f}{[\Sigma, (\Gamma, f, U)]\ \text{sig.}}$$

The necessary substitution and weakening lemmas can be proved by induction on the derivations:

Substitution lemma Let $(s_1, \ldots, s_n) : \Theta \longrightarrow \Gamma$ be a context map, where $\Gamma = x_1 : A_1, \ldots, x_n : A_n$.

(a) If $\Gamma \Longrightarrow B \ \mathrm{type}$, then $\Theta \Longrightarrow B[s_1, \ldots, s_n/x_1, \ldots, x_n] \ \mathrm{type}$.

(b) If $\Gamma \Longrightarrow b : B$, then
$\Theta \Longrightarrow b[s_1, \ldots, s_n/x_1, \ldots, x_n] : B[s_1, \ldots, s_n/x_1, \ldots, x_n] \ \mathrm{type}$

Weakening lemma Suppose that $\Gamma \Longrightarrow B \ \mathrm{type}$. Let y be a variable not in $\mathrm{FV}(\Gamma, \Theta)$.

(a) If $\Gamma, \Theta \ \mathrm{context}$, then $\Gamma, y : B, \Theta \ \mathrm{context}$

(b) If $\Gamma, \Theta \Longrightarrow A \ \mathrm{type}$, then $\Gamma, y : B, \Theta \Longrightarrow A \ \mathrm{type}$

(c) If $\Gamma, \Theta \Longrightarrow a : A$, then $\Gamma, y : B, \Theta \Longrightarrow a : A$

First-order formulas with dependent sorts

Similar to ordinary many-sorted first-order logic.

Predicate symbols are given by a set P of symbols with an arity.

Given a term signature Σ, we may to any

$$x_1 : A_1, \ldots, x_n : A_n \text{ context } [\Sigma]$$

assign a new n-ary predicate symbol $R \in P$. This gives a predicate declaration

$$(\langle x_1 : A_1, \ldots, x_n : A_n \rangle, R).$$

Thus a signature for first-order logic with dependent sorts consists of a term signature Σ and a sequence of predicate declarations $\Pi = [E_1, \ldots, E_m]$, where all predicates declared are distinct. Given this we can form the set of formulas in each variable context Γ over $[\Sigma; \Pi]$:

- For each predicate declaration (Δ, R) in Π, we assume the rule

$$\frac{\Gamma \text{ context } [\Sigma] \quad (a_1, \ldots, a_n) : \Gamma \longrightarrow \Delta}{\Gamma \Rightarrow R(a_1, \ldots, a_n) \text{ formula } [\Sigma; \Pi]}$$

- $$\frac{\Gamma \text{ context } [\Sigma]}{\Gamma \Rightarrow \bot \text{ formula } [\Sigma; \Pi]} \qquad \frac{\Gamma \text{ context } [\Sigma]}{\Gamma \Rightarrow \top \text{ formula } [\Sigma; \Pi]}$$

- $$\frac{\Gamma \Rightarrow \phi \text{ formula} \quad \Gamma \Rightarrow \psi \text{ formula } [\Sigma; \Pi]}{\Gamma \Rightarrow (\phi \wedge \psi) \text{ formula } [\Sigma; \Pi]}$$
$$\frac{\Gamma \Rightarrow \phi \text{ formula} \quad \Gamma \Rightarrow \psi \text{ formula } [\Sigma; \Pi]}{\Gamma \Rightarrow (\phi \vee \psi) \text{ formula } [\Sigma; \Pi]}$$
$$\frac{\Gamma \Rightarrow \phi \text{ formula} \quad \Gamma \Rightarrow \psi \text{ formula } [\Sigma; \Pi]}{\Gamma \Rightarrow (\phi \to \psi) \text{ formula } [\Sigma; \Pi]}$$

- $$\frac{\Gamma, x : A \Rightarrow \phi \text{ formula } [\Sigma; \Pi]}{\Gamma \Rightarrow (\forall x : A)\phi \text{ formula } [\Sigma; \Pi]} \qquad \frac{\Gamma, x : A \Rightarrow \phi \text{ formula } [\Sigma; \Pi]}{\Gamma \Rightarrow (\exists x : A)\phi \text{ formula } [\Sigma; \Pi]}$$

Logic

Assumption rule:

$$\frac{\Gamma \Rightarrow \phi_1, \ldots, \phi_n \text{ formulas}}{\Gamma; \phi_1, \ldots, \phi_m \vdash \phi_i}$$

Propositional rules:

$$\frac{\Gamma \Rightarrow \phi \text{ formula} \quad \Gamma; \Delta \vdash \bot}{\Gamma; \Delta \vdash \phi}\ (\bot E) \qquad \frac{\Gamma \Rightarrow \Delta \text{ formulas}}{\Gamma; \Delta \vdash \top}\ (\top I)$$

$$\frac{\Gamma; \Delta \vdash \phi \quad \Gamma; \Delta \vdash \psi}{\Gamma; \Delta \vdash \phi \wedge \psi}\ (\wedge I) \qquad \frac{\Gamma; \Delta \vdash \phi \wedge \psi}{\Gamma; \Delta \vdash \phi}\ (\wedge E_1) \quad \frac{\Gamma; \Delta \vdash \phi \wedge \psi}{\Gamma; \Delta \vdash \psi}\ (\wedge E_2)$$

$$\frac{\Gamma; \Delta, \phi \vdash \psi}{\Gamma; \Delta \vdash \phi \rightarrow \psi}\ (\rightarrow I) \qquad \frac{\Gamma; \Delta \vdash \phi \rightarrow \psi \quad \Gamma; \Delta \vdash \phi}{\Gamma; \Delta \vdash \psi}\ (\rightarrow E)$$

$$\frac{\Gamma;\Delta, \vdash \phi \quad \Gamma \Longrightarrow \psi \text{ formula}}{\Gamma;\Delta \vdash \phi \vee \psi}\ (\vee I_1) \qquad \frac{\Gamma;\Delta, \vdash \psi \quad \Gamma \Longrightarrow \phi \text{ formula}}{\Gamma;\Delta \vdash \phi \vee \psi}\ (\vee I_2)$$

$$\frac{\Gamma;\Delta \vdash \phi \vee \psi \quad \Gamma;\Delta,\phi \vdash \theta \quad \Gamma;\Delta,\psi \vdash \theta}{\Gamma;\Delta \vdash \theta}\ (\vee E)$$

Quantifier rules: (usual variable conditions in blue)

$$\frac{\Gamma, x : A; \Delta \vdash \psi \quad \Gamma \Rightarrow \Delta \text{ formulas}}{\Gamma; \Delta \vdash (\forall x : A)\psi} \ (\forall I)$$

$$\frac{\Gamma; \Delta \vdash (\forall x : A)\psi \quad \Gamma \Rightarrow t : A}{\Gamma; \Delta \vdash \psi[t/x]} \ (\forall E)$$

$$\frac{\Gamma \Rightarrow t : A \quad \Gamma, x : A \Rightarrow \psi \text{ formula} \quad \Gamma; \Delta \vdash \psi[t/x]}{\Gamma; \Delta \vdash (\exists x : A)\psi} \ (\exists I)$$

$$\frac{\Gamma; \Delta \vdash (\exists x : A)\psi \quad \Gamma, x : A; \Delta, \psi \vdash \phi \quad \Gamma \Rightarrow \Delta, \phi \text{ formulas}}{\Gamma; \Delta \vdash \phi} \ (\exists E)$$

We assume capture free substitution and α-equivalence, so that substitutions into quantified formulas are always possible.

Partial functions - in classical logic

In classical (non-dependent, many-sorted,) logic a functional relation, can be extended to a function.
Suppose

$$x =_A u, y =_B z, \phi(x, y) \vdash \phi(u, z)$$

and

$$\phi(x, y), \phi(x, z) \vdash y =_B z$$

Suppose $*$ is some constant in B. We can introduce a total function symbol $f : A \to B$ with defining axiom

$$f(x) =_B y \leftrightarrow \phi(x, y) \vee y =_B * \wedge \neg(\exists y : B)\, \phi(x, y).$$

In intuitionistic logic this is not possible, unless the domain of definition of the relation ϕ is decidable.

Partial functions - in dependently typed FOL

Suppose again

$$x =_A u, y =_B z, \phi(x, y) \vdash \phi(u, z)$$

and

$$\phi(x, y), \phi(x, z) \vdash y =_B z$$

Introduce a type D for the domain of definition of ϕ

$$x : A \Rightarrow D(x) \ \mathrm{type}$$

and an axiom

$$x : A, y : B; \phi(x, y) \vdash (\exists p : D(x))\top$$

and a dependent function symbol f

$$x : A, p : D(x) \Rightarrow f(x, p) : B$$

with axiom

$$x : A, p : D(x); \vdash \phi(x, f(x, p))$$

This works out as it should ...

Local propositions-as-types

Consider a fixed signature. Suppose $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is a context and that ϕ is a formula in that context.

Add a new dependent type to the signature

$$\Gamma \Longrightarrow F(x_1, \ldots, x_n) \text{ type}$$

Then add two axioms over the extended signature

$$\Gamma, p : F(x_1, \ldots, x_n); \vdash \phi$$

$$\Gamma; \phi \vdash (\exists p : F(x_1, \ldots, x_n))\top$$

Truth of $\phi (= \phi(x_1, \ldots, x_n))$ has thus been encoded as inhabitedness of $F(x_1, \ldots, x_n)$.

Standard semantics in Martin-Löf type theory

A signature Σ, Π is interpreted by a sequence of constant declarations. These will be postulates of type theory.
The standard semantics of a judgement in dependently typed first-order logic

$$x_1 : A_1, \ldots, x_n : A_n; \phi_1, \ldots, \phi_m \vdash \psi$$

will be the type-theoretic judgement

$$x_1 : A_1, \ldots, x_n : A_n, \phi_1 \text{ true}, \ldots, \phi_m \text{ true} \Longrightarrow \psi \text{ true}$$

which can be paraphrased as the existence of a term q such that

$$x_1 : A_1, \ldots, x_n : A_n, p_1 : \phi_1, \ldots, p_m : \phi_m \Longrightarrow q : \psi.$$

This is interpretation is straightforward in Coq and Agda and should be easily supported by these proof assistants.

As we consider the standard semantics to be M-L type theory (MLTT) the notion of function will be the (intensional) functions or operations of that theory. The axiom of choice is valid in MLTT with this notion of function, since it does not require the functions to respect prescribed equivalence relations. In E-categories we may for example have functions that chose pullbacks and projections from the arrow data.

$$\begin{tikzcd}
\mathsf{P}(X, Y, Z, f, g) \arrow[r, "\mathsf{p}_1(x,y,z,f,g)"] \arrow[d, "\mathsf{p}_2(x,y,z,f,g)"'] & X \arrow[d, "f"] \\
Y \arrow[r, "g"'] & Z
\end{tikzcd}$$

But it is not necessary that these functions respect equalities. Functions that respect equalities will be called extensional functions.

CETCS - constructive version of Lawvere's ETCS

The Elementary Theory of the Category of Sets (ETCS) is a classical first-order axiomatization of the category of sets and function (Lawvere 1963). The set-theoretic constructions possible are basically those of Zermelo's set theory Z. In modern categorical terms ETCS is a well-pointed boolean topos with the axiom of choice. S. MacLane has argued (not quite successfully) such theories are enough for mathematical practice.

Unlike set theory its theorems are invariant under set-isomorphism

$$\phi(A) \text{ and } A \cong B \text{ implies } \phi(B) \qquad (A, B \text{ are sets})$$

CETCS is a constructive version of ETCS suitable for formalizing elementary parts of Bishop's constructive mathematics (P. 2012).

CETCS

Signature: to Σ_{Cat} we add declarations for terminal object and pullbacks

1. $\langle\rangle \Longrightarrow 1 : \mathsf{Ob}$
2. $x, y, z : \mathsf{Ob}, f : \mathsf{Hom}(x, z), g : \mathsf{Hom}(y, z) \Longrightarrow \mathsf{P}(x, y, z, f, g) : \mathsf{Ob}$
3. $-" - \Longrightarrow \mathsf{p}_1(x, y, z, f, g) : \mathrm{Hom}(\mathsf{P}(x, y, z, f, g), x)$
4. $-" - \Longrightarrow \mathsf{p}_2(x, y, z, f, g) : \mathrm{Hom}(\mathsf{P}(x, y, z, f, g), y)$

and similarly for initial object and pushouts. Further we can add the Π-construction

5. $x, y, z : \mathsf{Ob}, f : \mathsf{Hom}(x, y), g : \mathsf{Hom}(z, x) \Longrightarrow \Pi(x, y, z, f, g) : \mathsf{Ob}$
6. $-" - \Longrightarrow \pi(x, y, z, f, g) : \mathsf{Hom}(\Pi(x, y, z, f, g), y)$
7. $-" - \Longrightarrow \mathrm{ev}(x, y, z, f, g) : \mathsf{Hom}(\mathsf{P}(x, \Pi(x, y, z, f, g), y, f, \pi(x, y, z, f, g)), z)$

That the object 1 is terminal is formalized as

$$\begin{array}{l} X : \mathrm{Ob}, f, g : \mathrm{Hom}(X, 1); \vdash f =_{X,1} g \\ X : \mathrm{Ob}; \vdash (\exists f : \mathrm{Hom}(X, 1))\top \end{array} \tag{6}$$

That pullbacks exists are given by (dropping some subscripts and derivable arguments)

$$\begin{array}{l} X, Y, Z : \mathrm{Ob}, f : \mathrm{Hom}(X, Z), g : \mathrm{Hom}(Y, Z); \vdash \\ \qquad f \circ \mathrm{p}_1(f, g) = g \circ \mathrm{p}_2(f, g) \\ X, Y, Z, W : \mathrm{Ob}, f : \mathrm{Hom}(X, Z), g : \mathrm{Hom}(Y, Z), h : \mathrm{Hom}(W, X), k : \mathrm{Hor} \\ \qquad f \circ h = g \circ k \vdash \\ \qquad (\exists t : \mathrm{Hom}(W, \mathrm{P}(f, g))) \\ \qquad\qquad \mathrm{p}_1(f, g) \circ t =_{w,x} h \wedge \mathrm{p}_2(f, g) \circ t = k. \\ X, Y, Z, W : \mathrm{Ob}, f : \mathrm{Hom}(X, Z), g : \mathrm{Hom}(Y, Z), t, s : \mathrm{Hom}(w, \mathrm{P}(f, g))); \\ \qquad \mathrm{p}_1(f, g) \circ t = \mathrm{p}_1(f, g) \circ s, \\ \qquad \mathrm{p}_2(f, g) \circ t = \mathrm{p}_2(f, g) \circ s \vdash t = s \end{array} \tag{7}$$

A morphism $x : 1 \longrightarrow X$ is called an *element* of X. We write this as $x \in X$. For $x, x' \in X$ we write $x =_{1,X} x'$ as $x =_X x'$. A morphism $f : X \longrightarrow Y$ is called *surjective* if for every $y \in Y$, there is $x \in X$ such that $f \circ x =_Y y$. The following axiom states that 1 is a strong generator of the category:

(G) Every surjective mono $X \longrightarrow Y$ is an isomorphism.

We consider as in (P. 2012) a sequence of maps $\alpha_1 : P \longrightarrow X_1, \dots, \alpha_1 : P \longrightarrow X_n$ that are jointly monic as a relation between $X_1, \dots, X_n$ and for elements $x_1 \in X_1, \dots, x_n \in X_n$ we write

$$(x_1, \dots, x_n) \,\epsilon\, (\alpha_1, \dots, \alpha_n)$$

if there is $p \in P$ with $\alpha_1 \circ p =_{X_1} x_1, \dots, \alpha_n \circ p =_{X_n} x_n$. A relation $\alpha_1 : P \longrightarrow X_1, \alpha_2 : P \longrightarrow X_2$ is called a *partial function from X_1 to X_2* if α is mono.

Existence of dependent products Π is formulated as follows

For any mappings $Y \xrightarrow{g} X \xrightarrow{f} I$ we have a commutative diagram

$$\begin{tikzcd}
Y \arrow[rrd, "g"'] & & \mathsf{P}(f, \pi(f,g)) \arrow[ll, "\mathrm{ev}(f,g)"'] \arrow[r, "\mathsf{p}_1"] \arrow[d, "\mathsf{p}_2"'] & \Pi(f,g) \arrow[d, "\pi(f,g)"] \\
& & X \arrow[r, "f"'] & I
\end{tikzcd} \tag{8}$$

where for any for any element $i \in I$ and any partial function $\psi =_{\mathrm{def}} (\xi : R \longrightarrow X, \upsilon : R \longrightarrow Y)$ satisfying (a) and (b):

(a) for all $x \in X$, $y \in Y$, $(x,y) \epsilon \psi$ implies $gy =_X x$ and $fx =_I i$,

(b) if $fx =_I i$, then there is $y \in Y$ with $(x,y) \epsilon \psi$

there is a unique $s \in \Pi(f,g)$ s.t. $\pi(f,g) \circ s =_I i$ and for all $x \in X$, $y \in Y$,

$$(s,x,y) \epsilon \alpha \Longleftrightarrow (x,y) \epsilon \psi. \tag{9}$$

Here $\alpha =_{\mathrm{def}} (\mathsf{p}_1, \mathsf{p}_2, \mathrm{ev}(f,g))$.

Further axioms include: Natural numbers object (for recursion and induction), non-triviality of binary sums.
and optionally a presentation axiom with new constants

8. $x : \mathrm{Ob} \Longrightarrow \mathrm{Pre}(x) : \mathrm{Ob}$
9. $x : \mathrm{Ob} \Longrightarrow \mathrm{pre}_x : \mathrm{Hom}(\mathrm{Pre}(x), x)$

(PA) For every object x, $\mathrm{Pre}(x)$ is projective and $\mathrm{pre}_x : \mathrm{Pre}(x) \twoheadrightarrow x$ is surjective.

This says that every object is a quotient of an object on which the axiom of choice is valid.

Axiomatizing the category of small categories

An early proposal:

F.W. Lawvere: The category of categories as a foundation for mathematics, In: *La Jolla conference on categorical algebra,* Springer-Verlag, 1966, pp. 1-20.

Is there a constructive version of this theory ?

A start:

O.Wilander: An E-bicategory of E-categories exemplifying a type-theoretic approach to bicategories. In: O.Wilander, *On constructive sets and partial structures.* PhD Thesis, Uppsala University 2011.

Project: carry out such a axiomatization using dependently typed FOL, with E-categories in type theory rather than categories in sets as intended model.

General semantics of dependently typed FOL

The type system we introduced may be interpreted by a cwa. This is a category $\mathcal{E}$ (contexts and context maps) with a contravariant functor $\mathsf{Ty} : \mathcal{E}^{\mathrm{op}} \longrightarrow \mathrm{Set}$ (the possible types in a context and the action on them by substitution). There is moreover an operation . that extends contexts $\Gamma \in \mathrm{Ob}(\mathcal{E})$ by a type $B \in \mathrm{Ty}(\Gamma)$: $\Gamma.B \in \mathrm{Ob}(\mathcal{E})$. The extended context has a projection down to the orginal context

$$\mathsf{p}_{\Gamma,B} : \Gamma.B \longrightarrow \Gamma.$$

The logic is then interpreted using another contravariant functor $\mathcal{E}^{\mathrm{op}} \longrightarrow \mathrm{Heyting}$ similarly to a hyper doctrine or tripos.

The logical part :

- A functor $P : \mathcal{E}^{op} \longrightarrow \mathrm{Heyting}$ into the category of Heyting-algebras with maps preserving all propositional logic operations ($\wedge$, $\vee$, $\rightarrow$,$\top$,$\bot$)
- For any $\Gamma \in \mathrm{Ob}(\mathcal{E})$ and $S \in \mathrm{Ty}(\Gamma)$ monotone operations

  $$\forall_{\Gamma,S}, \exists_{\Gamma,S} : P(\Gamma.S) \longrightarrow P(\Gamma)$$

  such that for $Q \in P(\Gamma)$, $R \in P(\Gamma.S)$,

  - $Q \leq \forall_{\Gamma,S}(R) \Longleftrightarrow P(\mathrm{p}_{\Gamma,S})(Q) \leq R$
  - $\exists_{\Gamma,S}(R) \leq Q \Longleftrightarrow R \leq P(\mathrm{p}_{\Gamma,S})(Q)$.

- (Beck-Chevalley) For the pullback square

$$\begin{tikzcd}
\Delta.S\{\sigma\} \arrow[r, "\mathsf{q}_{\Delta,\Gamma,S}(\sigma)"] \arrow[d, "\mathsf{p}_{\Delta,S\{\sigma\}}"'] & \Gamma.S \arrow[d, "\mathsf{p}_{\Gamma,S}"] \\
\Delta \arrow[r, "\sigma"'] & \Gamma
\end{tikzcd} \tag{10}$$

  we have for $R \in P(\Gamma.S)$,

  - $P(\sigma)(\forall_{\Gamma,S}(R)) = \forall_{\Delta,S\{\sigma\}}(P(q_{\Delta,\Gamma,S})(\sigma)(R))$,
  - $P(\sigma)(\exists_{\Gamma,S}(R)) = \exists_{\Delta,S\{\sigma\}}(P(q_{\Delta,\Gamma,S})(\sigma)(R))$.

The Lindenbaum-Tarski model

We indicate how to construct a universal Heyting-algebra model.

Theorem. The contexts and context maps for a fixed signature form a category with attributes.

Proof. Let Σ be a fixed signature. Let $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ be a context with respect to the signature. By rules (R3) and (R3) we have for all $i = 1, \ldots, n$

$$\Gamma \Longrightarrow A_i \text{ type}$$

and

$$\Gamma \Longrightarrow x_i : A_i$$

Now trivially, $A_i = A_i[x_1, \ldots, x_{i-1}/x_1, \ldots, x_{i-1}]$ so

$$1_\Gamma =_{\text{def}} (x_1, \ldots, x_n) : \Gamma \longrightarrow \Gamma$$

is a context map. This will be the identity.

Suppose that $\Delta = y_1 : B_1, \ldots, y_m : B_m$ and $\Theta = z_1 : C_1, \ldots, z_k : C_k$ are contexts and that

$$\sigma = (s_1, \ldots, s_m) : \Gamma \longrightarrow \Delta \text{ and } \tau = (t_1, \ldots, t_k) : \Delta \longrightarrow \Theta$$

are context maps. The vector of terms

$$\tau \circ \sigma =_{\mathrm{def}} \big(t_1[s_1, \ldots, s_m/y_1, \ldots, y_m], \ldots, t_k[s_1, \ldots, s_m/y_1, \ldots, y_m]\big)$$

is a context map $\Gamma \longrightarrow \Theta$, the composition of τ and σ.

It is straightforward to show that this gives a category where the empty context $\langle\rangle$ is terminal.

The following square is a pullback diagram for any variable $v \notin FV(\Delta)$, and $u \notin FV(\Gamma)$:

$$\begin{tikzcd}
\Delta, v : S[s_1, \ldots, s_n/x_1, \ldots, x_n] \arrow[r, "(s_1, \ldots, s_n, v)"] \arrow[d, "(y_1, \ldots, y_m)"'] & \Gamma, u : S \arrow[d, "(x_1, \ldots, x_n)"] \\
\Delta \arrow[r, "(s_1, \ldots, s_n)"'] & \Gamma
\end{tikzcd}$$

Assume that the variables form a set V with decidable equality, and that there is a function fresh such that for any list of variables $x_1, \ldots, x_n$,

$$\mathrm{fresh}(x_1, \ldots, x_n) \in V \setminus \{x_1, \ldots, x_n\}.$$

For a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ write

$$\mathrm{fresh}(\Gamma) = \mathrm{fresh}(x_1, \ldots, x_n).$$

Then this becomes the required pullback diagram of the CwA:

$$\begin{tikzcd}
\Delta, \mathrm{fresh}(\Delta) : S[s_1, \ldots, s_n / x_1, \ldots, x_n] \arrow[r, "{(s_1, \ldots, s_n, \mathrm{fresh}(\Gamma))}"] \arrow[d, "{(y_1, \ldots, y_m)}"'] & \Gamma, \mathrm{fresh}(\Delta) : S \arrow[d, "{(x_1, \ldots, x_n)}"] \\
\Delta \arrow[r, "{(s_1, \ldots, s_n)}"'] & \Gamma
\end{tikzcd}$$

Let $\mathcal{C}_\Sigma$ be the category of contexts and context maps relative to a signature Σ. We define a functor $\mathsf{Ty} : \mathcal{C}_\Sigma^{\mathrm{op}} \longrightarrow \mathrm{Set}$ which assigns types to contexts

$$\mathsf{Ty}(\Gamma) = \{S \in \mathrm{pretype} : \Gamma \Longrightarrow S \ \mathrm{type}\}$$

and for a context map $\sigma : \Delta \longrightarrow \Gamma$, let

$$\mathsf{Ty}(\sigma)(S) = S[\sigma/\Gamma].$$

Here

$$S[\sigma/\Gamma] = S[s_1, \ldots, s_n/x_1, \ldots, x_n]$$

Let $\mathcal{S} = [\Sigma, \Pi]$ be a fixed signature.

Theorem. Let Γ be a context in the signature. Then the set of formulas in the context

$$P(\Gamma) = \{\phi : \Gamma \Longrightarrow \phi\}$$

is a Heyting pre-algebra ordered $(\leq)$ by derivability

$$\phi \leq \psi \Longleftrightarrow_{\mathrm{def}} \Gamma; \phi \vdash \psi.$$

Substitution is an operation that preserves order and the propositional connectives.
For a context map $f : \Delta \longrightarrow \Gamma$ define $P(f) : P(\Gamma) \longrightarrow P(\Delta)$ by

$$P(f)(\phi) = \phi[f/\Gamma]$$

Theorem. For a context map $\sigma : \Delta \longrightarrow \Gamma$, $P(\sigma)$ is a morphism of pre-Heyting algebras, i.e. it preserves the order and the operations $\wedge$, $\vee$, $\rightarrow$, $\top$ and $\bot$. Moreover this assignment is pseudo-functorial in the sense that for another context map $\tau : \Theta \longrightarrow \Delta$

$$P(\tau \circ \sigma)(\phi) \equiv P(\tau)(P(\sigma)(\phi))$$

and

$$P(1_\Gamma)(\phi) \equiv \phi.$$

Here $\phi \equiv \psi$ means $\phi \leq \psi$ and $\psi \leq \phi$.

Suppose Γ context and $\Gamma \Rightarrow S : \mathrm{type}$. We have the projection context map $\mathsf{p}_{\Gamma,S}; \Gamma.S \longrightarrow \Gamma$. Define $\forall_{\Gamma,S}, \exists_{\Gamma,S} : P(\Gamma.A) \longrightarrow P(\Gamma)$ by

$$\forall_{\Gamma,S}(\phi) = (\forall\, \mathsf{fr}(\Gamma) : S)(\phi)$$

and

$$\exists_{\Gamma,S}(\phi) = (\exists\, \mathsf{fr}(\Gamma) : S)(\phi).$$

Theorem. Suppose Γ context and $\Gamma \Rightarrow S : \mathrm{type}$. Then

(a) $\forall_{\Gamma,S}$ and $\exists_{\Gamma,S}$ are monotone

(b) For $Q \in P(\Gamma)$, $R \in P(\Gamma.S)$,

$$Q \leq \forall_{\Gamma,S}(R) \iff P(\mathsf{p}_{\Gamma,S})(Q) \leq R$$

(c) For $Q \in P(\Gamma)$, $R \in P(\Gamma.S)$,

$$\exists_{\Gamma,S}(R) \leq Q \iff R \leq P(\mathsf{p}_{\Gamma,S})(Q).$$

Theorem. (Beck-Chevalley condition) Suppose that Γ is a context and S is a type in context Γ. Let $\sigma : \Delta \longrightarrow \Gamma$ be a context map. Then for $\phi \in P(\Gamma.S)$

(a) $P(\sigma)(\forall_{\Gamma,S}(\phi)) \equiv \forall_{\Delta,\mathrm{Ty}(\sigma)(S)}(P(q_{\Delta,\Gamma,S}(\sigma)(\phi)))$

(b) $P(\sigma)(\exists_{\Gamma,S}(\phi)) \equiv \exists_{\Delta,\mathrm{Ty}(\sigma)(S)}(P(q_{\Delta,\Gamma,S}(\sigma)(\phi)))$

References

J.F. Belo. Dependently Sorted Logic. In: M. Miculan, I. Scagnetto, and F. Honsell (Eds.): *TYPES 2007*, LNCS 4941, pp. 33 – 50, Springer 2008.

J.F. Belo. *Foundations of Dependently Sorted Logic.* PhD Thesis, Manchester 2008.

N. Gambino and P. Aczel. The generalized type-theoretic interpretation of constructive set theory. *Journal of Symbolic Logic* 71(2006), 67 – 103.

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

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

T. Coquand, P. Dybjer, E. Palmgren and A. Setzer. *Type-theoretic foundation of constructive mathematics.* Notes distributed at TYPES Summer School, Göteborg, August 2005.

References (cont.)

P.T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium.* Vol 1 and 2. Oxford University Press 2002.

M. Hofmann. Syntax and semantics of dependent types. In: *Semantics and Logics of Computation.* Cambridge University Press 1997.

M.E. Maietti and G. Sambin. Towards a minimalist foundation for constructive mathematics. In: *From Sets and Types to Topology and Analysis* (eds. L. Crosilla and P. Schuster) Oxford University Press 2005, 91 – 114.

M.E. Maietti. Modular correspondence between dependent type theories and categories including pretopoi and topoi. *Mathematical Structures in Computer Science* 15(2005), 1089 –1149.

M. Makkai. First-order logic with dependent sorts, with applications to category theory. Preprint 1995. Available from the author's webpages.

References (cont.)

M. Makkai. Towards a categorical foundation of mathematics. In: *Logic Colloquium '95* (eds. J.A. Makowsky and E.V. Ravve) Lecture Notes in Logic, vol. 11, Association for Symbolic Logic 1998, 153 – 190.

M. Makkai. The theory of abstract sets based on first-order logic with depend types. Preprint 2013. Available from the author's webpages.

P. Martin-Löf (1984). *Intuitionistic Type Theory.* Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980. Bibliopolis.

E. Palmgren and S. Vickers. Partial Horn logic and cartesian categories. *Annals of Pure and Applied Logic,* 145(2007), 314 – 355.

E. Palmgren. Constructivist and structuralist foundations: Bishop's and Lawvere's theories of sets. *Annals of Pure and Applied Logic,* 163(2012), 1384 – 1399.

References (cont.)

S. Tonelli. *Investigations into a model of type theory based on the concept of basic pairs.* MSc Thesis Padova/Stockholm 2013
URL:
www2.math.su.se/gemensamt/grund/exjobb/matte/2013/rep8/report.pdf

O. Wilander. *On Constructive Sets and Partial Structures.* PhD dissertation in Mathematics. Uppsala University 2011.