Equivalence of term

May 23, 2021

CwF and SplTC Definition

Category with familly

Definition

A Category with familly :

1. $\mathcal{C}$, a category
2. $\mathsf{Ty}, \mathsf{Tm} : \mathcal{C}^{\mathrm{op}} \to \mathsf{Set}$
3. $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$
4. for each $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$
   4.1 $\Gamma.A : \mathcal{C}$ and $\pi_A : \Gamma.A \to \Gamma$,
   4.2 an element te $A : \mathsf{Tm}(\Gamma.A)$, such that $\mathsf{p}(\text{te } A) = Ty\ \pi_A\ A : \mathsf{Ty}(\Gamma.A)$,
   4.3 and the following pullback

$$\begin{tikzcd} \mathsf{y}(\Gamma.A) \arrow[r, "yy(\mathsf{te}_A)"] \arrow[d, "\mathsf{y}(\pi_A)"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathsf{Tm} \arrow[d, "\mathsf{p}"] \\ \mathsf{y}(\Gamma) \arrow[r, "yy(A)"'] & \mathsf{Ty} \end{tikzcd}$$

Split-Type Cateogry

Definition

A **Split-type-category structure** on $\mathcal{C}$ consists of:

1. for each object $\Gamma : \mathcal{C}$, a set $\mathrm{Ty}(\Gamma)$,
2. for each $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, an object $\Gamma.A : \mathcal{C}$ and a morphism $\pi_A : \Gamma.A \to \Gamma$,
3. for each map $f : \Gamma' \to \Gamma$, a function $\_^* \;:\; \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma')$ and axiom for indenty and composition, denoted $A \mapsto f^*A$,
4. for each Γ, $A : \mathrm{Ty}(\Gamma)$, and $f : \Gamma' \to \Gamma$, a morphism $\mathrm{q}(f, A) : \Gamma'.f^*A \to \Gamma.A$ with axiom for indentiy and composition, such that

$$\begin{tikzcd} \Gamma'.f^*A \arrow[r, "\mathrm{q}(f,A)"] \arrow[d, "\pi_{f^*A}"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \Gamma.A \arrow[d, "\pi_A"] \\ \Gamma' \arrow[r, "f"] & \Gamma \end{tikzcd}$$

Term in CwF and SplTC

Term definitions

Display Term type

for each $\Gamma : \mathcal{C}, A : \text{Ty } \Gamma$,

$$tm\ A = \sum_{(a : Tm\ \Gamma)} (p\ a = A)$$

Term type

for each $\Gamma : \mathcal{C}, A : \text{Ty } \Gamma$,

$$tm\ A = \sum_{(s : \mathcal{C}[\![\Gamma, \Gamma.A]\!])} (s \circ \pi_A = identity\ \Gamma)$$

Category with familly

Split-Type Category

Equivalence

Equivalence

$$
\begin{tikzcd}
 & & \sum\limits_{X,Y,Z} \mathsf{compat}_X(Y,Z) \arrow[dl, leftrightarrow, "\simeq"'] \arrow[dr, leftrightarrow, "\simeq"] & & \\
\mathsf{splty}(\mathcal{C}) \arrow[r, leftrightarrow, "\simeq"] & \sum\limits_{X:\mathsf{objext}(\mathcal{C})} \mathsf{qmor}(X) \arrow[dr] & & \sum\limits_{X:\mathsf{objext}(\mathcal{C})} (X) \arrow[r, leftrightarrow, "\simeq"] \arrow[dl] & \mathsf{cwf}(\mathcal{C}) \\
 & & \mathsf{objext}(\mathcal{C}) & &
\end{tikzcd}
$$

Object Extension structure

Definition

A **Object extension structure** on $\mathcal{C}$ consists of:

1. a functor $\mathrm{Ty} : \mathcal{C}^{op} \to \mathrm{Set}$
2. for each $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, an object $\Gamma.A : \mathcal{C}$ and a morphism $\pi_A : \Gamma.A \to \Gamma$,

$\Rightarrow$ Only reordering form both definitions.

qq-morphism structure

Definition

A **qq-morphism structure** on $\mathcal{C}$ and O an object extension structure consists of:
for each Γ, $A : \mathrm{Ty}(\Gamma)$, and $f : \Gamma' \to \Gamma$,
a morphism $\mathrm{q}(f, A) : \Gamma'.f^*A \to \Gamma.A$ with axiom for indentiy and composition, such that

$$\begin{tikzcd} \Gamma'.f^*A \arrow[r, "\mathrm{q}(f,A)"] \arrow[d, "\pi_{f^*A}"] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d, "\pi_A"] \\ \Gamma' \arrow[r, "f"] & \Gamma \end{tikzcd}$$

$$\Rightarrow \mathrm{spltype} C \approx \sum_{O:\mathrm{objext}\ C} \mathrm{qmor}\ O$$

Term structure

Definition

A **term structure** on $\mathcal{C}$ and O an object extension structure consists of:

1. a presheaf $\mathrm{Tm} : \mathcal{C}^{op} \to \mathrm{Set}$, a natural transformation $p : \mathrm{Tm} \to \mathrm{Ty}$
2. for each $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$
   2.1 $\Gamma.A : \mathcal{C}$ and $\pi_A : \Gamma.A \to \Gamma$,
   2.2 an element $\mathrm{te}\ A : \mathrm{Tm}(\Gamma.A)$, such that $\mathrm{p}(\mathrm{te}\ A) = Ty\ \pi_A\ A : \mathrm{Ty}(\Gamma.A)$,
   2.3 and the following pullback
$$\begin{tikzcd} \mathrm{y}(\Gamma.A) \arrow[r, "yy(\mathrm{te}_A)"] \arrow[d, "\mathrm{y}(\pi_A)"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{Tm} \arrow[d, "\mathrm{p}"] \\ \mathrm{y}(\Gamma) \arrow[r, "yy(A)"'] & \mathrm{Ty} \end{tikzcd}$$

$$\Rightarrow \mathrm{cwf}\, C \approx \sum_{O:\mathrm{objext}\, C} \mathrm{termstruc}\ O$$

Equivalence

$$
\begin{tikzcd}
 & & \color{red}{\sum_{X,Y,Z} \mathsf{compat}_X(Y,Z)} \arrow[dr, leftrightarrow, "\simeq"] & & \\
\mathsf{splty}(\mathcal{C}) \arrow[r, leftrightarrow, "\simeq"] & \sum_{X:\mathsf{objext}(\mathcal{C})} \mathsf{qmor}(X) \arrow[ur, leftrightarrow, "\simeq"] \arrow[dr] & & \sum_{X:\mathsf{objext}(\mathcal{C})} (X) \arrow[r, leftrightarrow, "\simeq"] \arrow[dl] & \mathsf{cwf}(\mathcal{C}) \\
 & & \mathsf{objext}(\mathcal{C}) & &
\end{tikzcd}
$$

Compatibility

Compatibility

a qq-morphism structure Q and a term structre T over O are compatible if

$$te(f^* \ A) = q(f, A)^* te(A)$$

term-compatibility

for each qq-morphism structure Q over O , We can define
$T^C := \sum_{(T:term\_structure \ O)}, compatible \ T \ Q.$

qq-compatibility

for each term structure T over O , We can define
$Q^C := \sum_{(Q:qq\_morphism\_structure \ O)}, compatible \ T \ Q.$

Compatibility

weq_term_qq

$qq\_morphism\_structureO \simeq term\_structureO$

suffle

$\sum_{(T:term\_structure\ O)}, Q^C \simeq \sum_{(Q:qq\_morphism\_structure\ O)}, T^C$

forget_compat_qq

$\sum_{(T:term\_structure\ O)}, Q^C \simeq term\_structure\ O$

forget_compat_qq

$\sum_{(Q:qq\_morphism\_structure\ O)}, T^C \simeq qq\_morphism\_structure\ O$

Proof : By showing that T^C , Q^C are contractible

Canonical term_stucture form a qq_structure

- Same object structure so same Ty, _._ ,π_
- For all $\Gamma : \mathcal{C}$, $Tm = \prod_{\Gamma:\mathcal{C}} \sum_{A:\,Ty\ \Gamma} \sum_{s:\mathcal{C}[\![\Gamma,\Gamma.A]\!])} s \circ \pi_A = identity\ \Gamma$
- For all $\Gamma, \Delta : \mathcal{C}$, $f : \mathcal{C}[\![\Gamma, \Delta]\!]$,
  $Tm\ f := (fun\ A => A, , fun(s, , ids) => pb\_of\_section(qq_\pi\_Pb\ Q)\ ids)$
- So with that, $Tm : \mathcal{C}^{op} \to \mathrm{Set}$
- For all $\Gamma : \mathcal{C}$, $a :\ Tm\ \Gamma$, $p\ a = pr1\ a$, natural transformation.
- $te\ A = (\pi_A^*\ A, q(\pi_A, A)\pi_A)$

Term and Type equivalence

With a split type structure as Context

Context

- a Category $\mathcal{C}$
- A Split Type Structure over $\mathcal{C}$, SC
- The asssociate object extension structure O
- the associate qq structure Q
- the associate term structre T
- the associate Category with familly structure CWF

Type Equivalence

reind_type

$$reind\_type\ \{\Gamma\Delta : \mathcal{C}\}\ (f : \mathcal{C}[\![\Delta, \Gamma]\!]) : Ty\ \Gamma \to Ty\ \Delta$$

- Almost everything work
- even the reindextion just by reflexivity
- and just by reflexivity
- Since $\color{red}{Ty} :\equiv \color{blue}{Ty}$

Type Equivalence

What doesn't work (for now)

- Context extension doesn't work
- $\Gamma.A$ can be interpreted as a element of type $pr1\ CWF$ but not clear that $\Gamma.A = \Gamma.A$
- So, Type familly suffer for the same problem
- $\mathsf{Ty}\ \Gamma.A = Ty\ \Gamma.A$ but not $\mathsf{Ty}\ \Gamma.A = Ty\ \Gamma.A$
- So, Dependants Types also doesn't work (same problem)

Term definitions – callback

Display Term type

for each $\Gamma : \mathcal{C}, A : \mathrm{Ty}\ \Gamma$,

$$tm\ A = \sum_{(a : Tm\ \Gamma)} (p\ a = A)$$

Category with familly

Term type

for each $\Gamma : \mathcal{C}, A : \mathrm{Ty}\ \Gamma$,

$$tm\ A = \sum_{(s : \mathcal{C}[\![\Gamma, \Gamma . A]\!])} (s \circ \pi_A = identity\ \Gamma)$$

Split-Type Category

Term form a qq-structure

tm_form_qq

$$Tm = \prod_{\Gamma:\mathcal{C}} \sum_{A:Ty\ \Gamma} \sum_{s:\mathcal{C}[\![\Gamma,\Gamma.A]\!])} s \circ \pi_A = identity\ \Gamma$$

$\Rightarrow$ Create a intermediate display term type with Tm

tm_inter

$$tm\_inter\ \{\Gamma : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma) = \sum_{a:Tm\ \Gamma} (pr_1\ a = A)$$

first part of the equivalence

tm_equiv_inter

$$\textit{tm\_inter}\ \{\Gamma : \mathcal{C}\}\ (A : \mathsf{Ty}\ \Gamma) \simeq \textit{tm}\ \{\Gamma : \mathcal{C}\}\ (A : \mathsf{Ty}\ \Gamma)$$

$\Rightarrow$ Just by path induction

$\Leftarrow$ Just by formation rule of sigma types

second part of the equivalence

tm_equiv_interbis

$$\mathit{tm\_inter}\ \{\Gamma : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma) \simeq \mathit{tm}\ \{\Gamma : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma)$$

$$\text{So, } \sum_{a : \mathit{Tm}\ \Gamma} (pr_1\ a = A) = \sum_{(a : \mathit{Tm}\ \Gamma)} (p\ a = A)$$

Which is just reflexivity, since $\mathit{Tm} :\equiv \mathit{Tm}$ and p on term is defined as pr1 cf

tm_equiv

$$\mathit{tm}\ \{\Gamma : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma) \simeq \mathit{tm}\ \{\Gamma : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma)$$

transport of term

transport_tm

$$transportf\_tm\ \{\Gamma : \mathcal{C}\}\ \{A\ A' : \mathrm{Ty}\ \Gamma\}\ (e : A = A') : tm\ A \simeq tm\ A'$$

Equivalence and Transport

For all $\Gamma : \mathcal{C}$, A , $A' : \mathrm{Ty}\ \Gamma$, $e : A = A'$, $a : tm\ A$

$$transport\_tm\ e\ (tm\_equiv\ a) = (tm\_equiv\ (transportf\_tm\ e\ a))$$

reindexation of term – Proof

- Complete proof
- no intermediate step with a *tm_inter*-like
- Since *tm_inter* is a Sigma Type with a proposition as the second part, Only

$$pr_1 \; reind\_tm\_inter \; e \; (tm\_equiv \; a) = pr_1(tm\_equiv \; (reind\_tm \; e \; a))$$

- and after that just path induction on e

reindexation of term

reind_tm

$$reind\_tm\ \{\Gamma\Delta : \mathcal{C}\}\ (A : \mathrm{Ty}\ \Gamma)\ (f : \mathcal{C}[\![\Delta, \Gamma]\!]) : tm\ A \to tm\ (reind\_type\ f\ \ A)$$

Equivalence and Transport

For all Γ , $\Delta : \mathcal{C}$, $A : \mathrm{Ty}\ \Gamma$, $f : \mathcal{C}[\![\Delta, \Gamma]\!]$, $a : tm\ A$

$$reind\_tm\ e\ (tm\_equiv\ \ a) = (tm\_equiv\ \ (reind\_tm\ e\ \ a))$$

reindexation of term – Proof

- Complete proof
- intermediate step with a *reind_tm_inter* just like the equivalence.
- Since *tm_inter* is a Sigma Type with a proposition as the second part, Only

$$pr_1\ reind\_tm\_inter\ e\ (tm\_equiv\ a) = pr_1(tm\_equiv\ (reind\_tm\ e\ a))$$

- Ugly and take a bit of time but work

Term and Type equivalence With a category with familly structure as Context

Global

- For now only testing
- But seem harder
- no complete automatic Ty equivalence