Homotopy Type Theory Class

LMFI 2019-2020

Categories with families

Hugo Herbelin

Models of type theory

- Categories with families (Dybjer) whose internal language is type theory
- Contextual categories (Cartmell)
- Realisability models (e.g. to prove termination and consistency)
- Syntactic models (using type theory as the target rather than the meta-language)
- ...

See e.g. Taichi Uemura, *A General Framework for the Semantics of Type Theory*, 2019.

Categories with families

We shall model:

- Well-formed contexts (Γ **ok** up to conversion)
- Types well-formed in some well-formed context ($\Gamma \vdash A$ **type** up to conversion)
- Terms well-typed of some well-formed type in some well-formed context ($\Gamma \vdash t : A$ up to conversion)

Categories with families: bibliography

There is a very large bibliography on categories with families. Here are a few references:

- Peter Dybjer, *Internal type theory.*, 1995
- Thierry Coquand, *Models of Type Theory* (TYPES summer school), 2007
- Daniel Fridlender, Miguel Pagano, *Pure type systems with explicit substitutions*, 2015
- Ambrus Kaposi, András Kovács, Thorsten Altenkirch, *Constructing quotient Inductive-Inductive Types*, 2019

Categories with families

- The model of well-formed contexts: A category $\mathcal{C}$ of contexts Γ with substitutions σ as morphisms.
- The model of well-formed types and terms: A functor from $\mathcal{C}$ to the category $\mathcal{F}am$ of families of sets standing for the sets of terms t such that $\Gamma \vdash t : A$ with these sets indexed over types in the set of types A such that $\Gamma \vdash A$ **type**
- Together with a terminal object in $\mathcal{C}$ standing for the empty context $\square$ and a *context comprehension* operation standing for the context formation $\Gamma, a : A$

Note also that conversion will be interpreted as equality of the metalanguage

The category of contexts (set-theoretic presentation)

A category of contexts $\mathcal{C}$ is a category in the standard sense. That is, it is made of:

- A set of objects $\mathsf{Ob}_{\mathcal{C}}$, written Ctx, standing for contexts
- For each two contexts Γ and Γ', a set of morphisms $\mathsf{Hom}_{\mathcal{C}}(\Gamma, \Gamma')$, written $\Gamma \vdash \Gamma'$, standing for the set of substitutions from Γ to Γ'
- For each context Γ, an identity morphism id_Γ in $\Gamma \vdash \Gamma$ (the identity substitution)
- For each substitutions σ in $\Gamma \vdash \Gamma'$ and σ' in $\Gamma' \vdash \Gamma''$, their composition $\sigma' \circ \sigma$ in $\Gamma \vdash \Gamma''$
- Neutrality of identity on the left: $\mathsf{id}_{\Gamma'} \circ \sigma = \sigma$ seen as a cut-elimination rule
- Neutrality of identity on the right: $\sigma \circ \mathsf{id}_\Gamma = \sigma$, also seen as cut-elimination rule
- Associativity of composition: $(\sigma'' \circ \sigma') \circ \sigma = \sigma'' \circ (\sigma' \circ \sigma)$, seen as a commutation of cuts

The category of contexts (type-theoretic presentation)

- Objects/Contexts

$$\mathsf{Ctx} \ : \ \mathsf{U}_0$$

- Homomorphisms/Substitutions

$$\Gamma \vdash \Gamma' \ : \ \mathsf{U}_0 \quad \text{for } \Gamma, \Gamma' : \mathsf{Ctx}$$

- Identity/Axiom-rule

$$\mathsf{id}_\Gamma \ : \ \Gamma \vdash \Gamma \quad \text{for } \Gamma : \mathsf{Ctx}$$

- Composition/Cut-rule

$$\sigma' \circ \sigma \ : \ \Gamma \vdash \Gamma'' \quad \text{for } \sigma : (\Gamma \vdash \Gamma'), \sigma' : (\Gamma' \vdash \Gamma'')$$

- Left-Neutrality/Cut-Axiom-elimination

$$\mathsf{idleft} \ : \ \mathsf{id}_{\Gamma'} \circ \sigma = \sigma \quad \text{for } \sigma : (\Gamma \vdash \Gamma')$$

- Right-Neutrality/Axiom-Cut-elimination

$$\mathsf{idright} \ : \ \sigma \circ \mathsf{id}_\Gamma = \sigma \quad \text{for } \sigma : (\Gamma \vdash \Gamma')$$

- Associativity/Cut-Cut-commutation

$$\mathsf{compassoc} \ : \ (\sigma'' \circ \sigma') \circ \sigma = \sigma'' \circ (\sigma' \circ \sigma) \quad \text{for } \sigma : (\Gamma \vdash \Gamma'), \sigma' : (\Gamma' \vdash \Gamma''), \sigma'' : (\Gamma'' \vdash \Gamma'$$

The categories of contexts (Coq syntax)

Let's use Coq as the meta-language. A category (seen as a type system of contexts and substitutions) is represented as a record:

```
Record ContextCalculus := {
  Ctx : Type;
  Subst (Γ Γ':Ctx) : Type where "Γ ⊢ Γ'" := (Subst Γ Γ');
  Axiom (Γ:Ctx) : Γ ⊢ Γ where "id" := (Axiom _);
  Cut (Γ Γ' Γ'':Ctx) (σ:Γ ⊢ Γ') (σ':Γ' ⊢ Γ'') : Γ ⊢ Γ''
    where "σ' o σ" := (Cut _ _ _ σ σ');
  Comp_cut_axiom (Γ Γ':Ctx) (σ:Γ ⊢ Γ') : id o σ = σ;
  Comp_axiom_cut (Γ Γ':Ctx) (σ:Γ ⊢ Γ') : σ o id = σ;
  Comp_cut_cut (Γ Γ' Γ'':Ctx) (σ:Γ ⊢ Γ') (σ:Γ' ⊢ Γ'') (σ'':Γ'' ⊢ Γ''')
    : (σ'' o σ') o σ = σ'' o (σ' o σ);
}
```

Internal language of the category of contexts (presentation as a HIT)

$$
\begin{array}{lll}
\textit{Contexts} & \Gamma & ::= \ \ldots \\
\textit{Substitutions} & \sigma & ::= \ \mathsf{id} \mid \sigma' \circ \sigma \\
\textit{Substitution equality} & \theta & ::= \ \mathsf{idleft} \mid \mathsf{idright} \mid \mathsf{compassoc}
\end{array}
$$

$$
\frac{}{\mathsf{id} : (\Gamma \vdash \Gamma)} \qquad \frac{\sigma : (\Gamma \vdash \Gamma') \qquad \sigma' : (\Gamma' \vdash \Gamma'')}{\sigma' \circ \sigma : (\Gamma \vdash \Gamma'')}
$$

$$
\frac{\sigma : (\Gamma \vdash \Gamma')}{\mathsf{idleft}_\sigma : \mathsf{id} \circ \sigma =_{\Gamma \vdash \Gamma'} \sigma} \qquad \frac{\sigma : (\Gamma \vdash \Gamma')}{\mathsf{idright}_\sigma : \sigma \circ \mathsf{id} =_{\Gamma \vdash \Gamma'} \sigma}
$$

$$
\frac{\sigma : (\Gamma \vdash \Gamma') \qquad \sigma' : (\Gamma' \vdash \Gamma'') \qquad \sigma'' : (\Gamma'' \vdash \Gamma''')}{\mathsf{compassoc}_{\sigma,\sigma',\sigma''} : (\sigma'' \circ \sigma') \circ \sigma =_{\Gamma \vdash \Gamma'''} \sigma'' \circ (\sigma' \circ \sigma)}
$$

A functor from $\mathcal{C}$ to the category $\mathcal{F}am$ of families of sets (set-theoretic presentation)

A functor F from $\mathcal{C}$ to the category $\mathcal{F}am$ of families of sets is given by:

- To each object Γ in **Ctx** a family of sets, that is:
  - an index set $F(\Gamma)$, written $\Gamma \vdash \textbf{type}$, of types over Γ
  - for each index A in $\Gamma \vdash \textbf{type}$, a set $F(\Gamma, A)$, written $\Gamma \vdash A$, of terms of type A in Γ
- To each morphism σ in $\Gamma \vdash \Gamma'$, a map of family of sets, that is:
  - a map $F(\sigma)$ from A in $\Gamma \vdash \textbf{type}$ to $\Gamma' \vdash \textbf{type}$ written $A[\sigma]$ and standing for applying the substitution σ to A
  - for each index A, a map $F(\sigma, A)$ from t in $\Gamma \vdash A$ to $\Gamma \vdash A[\sigma]$, written $t[\sigma]$, and standing for applying the substitution σ to t
- The functorial properties:
  - $A[\textbf{id}] = A$ and $t[\textbf{id}] = t$
  - $A[\sigma' \circ \sigma] = A[\sigma'][\sigma]$ and $t[\sigma' \circ \sigma] = t[\sigma'][\sigma]$

Note: The functor laws for term substitution will eventually be derivable when all the structure will be in place

Internal language of the functor from the category of contexts to families of sets (presentation as a HIT in cubical type theory)

$$\begin{array}{llll} \textit{Types} & A ::= A[\sigma] & \textit{Type equality} & E ::= \mathsf{typesubstid}_A \mid \mathsf{typesubstcomp}_{A,\sigma,\sigma'} \\ \textit{Terms} & t ::= t[\sigma] & \textit{Term equality} & e ::= \mathsf{termsubstid}_t \mid \mathsf{termsubstcomp}_{A,t,\sigma,\sigma'} \end{array}$$

$$\frac{A : (\Gamma \vdash \mathsf{type}) \qquad \sigma : (\Gamma \vdash \Gamma')}{A[\sigma] : (\Gamma' \vdash \mathsf{type})} \qquad \frac{t : (\Gamma \vdash A) \qquad \sigma : (\Gamma \vdash \Gamma')}{t[\sigma] : (\Gamma' \vdash A[\sigma])}$$

$$\frac{A : (\Gamma \vdash \mathsf{type})}{\mathsf{typesubstid}_A : A[\mathsf{id}] =_{\Gamma \vdash \mathsf{type}} A} \qquad \frac{t : (\Gamma \vdash A)}{\mathsf{termsubstid}_t : t[\mathsf{id}] =_{\lambda i.(\Gamma \vdash \mathsf{typesubstid}_A\, i)} t}$$

$$\frac{A : (\Gamma \vdash \mathsf{type}) \qquad \sigma : (\Gamma \vdash \Gamma') \qquad \sigma' : (\Gamma' \vdash \Gamma'')}{\mathsf{typesubstcomp}_{A,\sigma,\sigma'} : A[\sigma' \circ \sigma] =_{\Gamma'' \vdash \mathsf{type}} A[\sigma'][\sigma]}$$

$$\frac{t : (\Gamma \vdash A) \qquad \sigma : (\Gamma \vdash \Gamma') \qquad \sigma' : (\Gamma' \vdash \Gamma'')}{\mathsf{termsubstcomp}_{A,t,\sigma,\sigma'} : t[\sigma' \circ \sigma] =_{\lambda i.\Gamma'' \vdash (\mathsf{typesubstcomp}_{A,\sigma,\sigma'}\, i)} t[\sigma'][\sigma]}$$

Note: We use Cubical Type Theory's notation $=_{\lambda i.T(i)}$ for an equality comparing two objets in equal types $T(0)$ and $T(1)$ and notation $=_T$ for homogeneous equality.

A context comprehension structure

The last ingredients of a category of family over $\mathcal{C}$ and F are:

- A terminal object in the category of context, that is:
  - A terminal object $\square$ seen as the empty context
  - A terminal morphism $\langle\rangle_\Gamma$ from each object Γ, seen as an empty substitution
  - The universal property of the terminal morphism $\langle\rangle_{\Gamma'} \circ \sigma = \langle\rangle_\Gamma$ for any σ in $\Gamma \vdash \Gamma'$
- A context comprehension operator, that is:
  - to each object Γ and type A in $\Gamma \vdash \mathbf{type}$, an object Γ, A in $\mathcal{C}$
  - together with first and second projections
    - $\pi_1(\sigma)$ a substitution in $\Gamma \vdash \Gamma'$ for σ in $\Gamma \vdash \Gamma', A$
    - $\pi_2(\sigma)$ a term in $\Gamma \vdash A[\pi_1(\sigma)]$ for σ in $\Gamma \vdash \Gamma', A$
  - for each σ in $\Gamma \vdash \Gamma'$ and t in $\Gamma \vdash A[\sigma]$, a unique pair $\langle\sigma, t\rangle$ in $\Gamma \vdash \Gamma', A[\sigma]$ such that $\pi_1\langle\sigma, t\rangle = \sigma$ and $\pi_2\langle\sigma, t\rangle = t$

Internal language of the comprehension structure (presentation as a HIT in cubical type theory)

$$
\begin{array}{lrcl}
\textit{Contexts} & \Gamma & ::= & \square \mid \Gamma, A \\
\textit{Substitutions} & \sigma & ::= & \ldots \mid \langle\rangle \mid \pi_1(\sigma) \mid \langle \sigma, t \rangle \\
\textit{Terms} & t & ::= & \ldots \mid \pi_2(\sigma) \\
\textit{Substitution equality} & E & ::= & \mathsf{terminal} \mid \mathsf{terminaleta} \mid \mathsf{substproj1} \mid \mathsf{compreheta} \\
\textit{Term equality} & e & ::= & \mathsf{substproj2} \mid \mathsf{comprehcomp}
\end{array}
$$

$$\frac{}{\square\ \mathsf{ok}} \qquad \frac{\Gamma\ \mathsf{ok} \qquad A : (\Gamma \vdash \mathsf{type})}{\Gamma, A\ \mathsf{ok}}$$

$$\frac{\sigma : (\Gamma' \vdash \Gamma) \qquad A : (\Gamma \vdash \mathsf{type}) \qquad t : (\Gamma' \vdash A[\sigma])}{\langle \sigma, t \rangle : (\Gamma' \vdash \Gamma, A[\sigma])}$$

$$\frac{\sigma : (\Gamma' \vdash \Gamma, A)}{\pi_1(\sigma) : (\Gamma' \vdash \Gamma)} \qquad \frac{\sigma : (\Gamma' \vdash \Gamma, A)}{\pi_2(\sigma) : (\Gamma' \vdash A[\pi_1(\sigma)])}$$

Internal language of the comprehension structure, continued (presentation as a HIT in cubical type theory)

$$\frac{\sigma : (\Gamma \vdash \Gamma')}{\mathsf{terminal} : \langle\rangle \circ \sigma =_{\Gamma \vdash \square} \langle\rangle} \qquad \frac{\sigma : (\Gamma \vdash \square)}{\mathsf{terminaleta} : \sigma =_{\Gamma \vdash \square} \langle\rangle}$$

$$\frac{\sigma : (\Gamma \vdash \Gamma') \qquad t : (\Gamma \vdash A[\sigma])}{\mathsf{substproj1} : \pi_1(\langle \sigma, t \rangle) =_{\Gamma \vdash \Gamma'} \sigma} \qquad \frac{\sigma : (\Gamma \vdash \Gamma') \qquad t : (\Gamma \vdash A[\sigma])}{\mathsf{substproj2} : \pi_2(\langle \sigma, t \rangle) =_{\lambda i.(\Gamma \vdash A[\mathsf{substproj1}\, i])} t}$$

$$\frac{\sigma : (\Gamma \vdash \Gamma') \qquad \sigma' : (\Gamma' \vdash \Gamma'') \qquad t : (\Gamma'' \vdash t : A[\sigma'])}{\mathsf{comprehcomp} : \langle \sigma', t \rangle \circ \sigma =_{\Gamma \vdash \Gamma'', A} \langle \sigma' \circ \sigma, (\mathsf{subst\ typesubstcomp\ in}\ t[\sigma]) \rangle} \quad A['][]\Gamma''\Gamma\Gamma''t\sigma : s$$

Conclusions

- Syntactic model vs internal syntax

  A syntax defines a model and a model defines a syntax. Categories with families is the class of models generated by a presentation of type theory with explicit substitutions.

  Conversely, the initial model of categories with families is a presentation of type theory with explicit substitutions.

- Intrinsic vs extrinsic syntax

  The syntax of type theory can be considered in two flavours:

  – Extrinsic syntax is the syntax with proof terms and derivations characterizing well-typed proof terms

  – Intrinsic syntax is the syntax without proof terms, where the proof information is only in the derivation