A Generalized Logical Framework

András Kovács[1], Christian Sattler[1]

[1]University of Gothenburg & Chalmers University of Technology

18 Apr 2025, EuroProofNet WG6 meeting, Genoa

Overview

1. Two-level type theories (2LTT):
   - metaprogramming over a **single model** of a **single type theory**.
   - the chosen model is defined **outside the system**.
   - **only a second-order ("internal")** view on the model.

Overview

1. Two-level type theories (2LTT):
   - metaprogramming over a **single model** of a **single type theory**.
   - the chosen model is defined **outside the system**.
   - **only a second-order ("internal")** view on the model.
2. Generalized logical framework (GLF):
   - metaprogramming over **any number of models** of **any number of type theories**.
   - models are defined **inside the system**.
   - both a **first-order/external** and a **second-order/internal** view on each model.

Overview

1. Two-level type theories (2LTT):
   - metaprogramming over a **single model** of a **single type theory**.
   - the chosen model is defined **outside the system**.
   - **only a second-order ("internal")** view on the model.
2. Generalized logical framework (GLF):
   - metaprogramming over **any number of models** of **any number of type theories**.
   - models are defined **inside the system**.
   - both a **first-order/external** and a **second-order/internal** view on each model.
   - *No substructural modalities*.

Overview

1. Two-level type theories (2LTT):
   - metaprogramming over a **single model** of a **single type theory**.
   - the chosen model is defined **outside the system**.
   - **only a second-order ("internal")** view on the model.
2. Generalized logical framework (GLF):
   - metaprogramming over **any number of models** of **any number of type theories**.
   - models are defined **inside the system**.
   - both a **first-order/external** and a **second-order/internal** view on each model.
   - *No substructural modalities*.

In this talk:

1. A syntax of GLF + examples + increasing amount of syntactic sugar.
2. A short overview of semantics.

GLF basic universes & type formers

$\mathbf{U}$	$: \mathbf{U}$	A universe of that supports ETT.
$\mathbf{Base}$	$: \mathbf{U}$	Type of "base categories".
$\mathbf{1}$	$: \mathbf{Base}$	The terminal category as a base category.
$\mathbf{PSh}$	$: \mathbf{Base} \to \mathbf{U}$	Universes of presheaves. Cumulativity: $\mathrm{PSh}_i \subseteq \mathrm{U}$. Supports ETT.
		We can only eliminate from PSh_i to PSh_i.
$\mathbf{Cat}_i$	$: \mathbf{PSh}_i$	$:=$ *type of categories in* PSh_i
$\mathbf{In}$	$: \mathbf{Cat}_i \to \mathbf{U}$	"Permission token" for working in presheaves over some $\mathbb{C} : \mathrm{Cat}_i$.
$\mathbf{base}$	$: \mathbf{In}\,\mathbb{C} \to \mathbf{Base}$	"Using the permission".

We use type-in-type everywhere for simplicity, i.e. $\mathrm{U} : \mathrm{U}$ and $\mathrm{PSh}_i : \mathrm{PSh}_i$.

Basic things we can do

$$\mathsf{U} : \mathsf{U} \qquad \mathsf{Base} : \mathsf{U} \qquad \mathbf{1} : \mathsf{Base} \qquad \mathsf{PSh} : \mathsf{Base} \to \mathsf{U}$$

$$\mathsf{Cat}_i : \mathsf{PSh}_i := \textit{type of cats in } \mathsf{PSh}_i \qquad \mathsf{In} : \mathsf{Cat}_i \to \mathsf{U} \qquad \mathsf{base} : \mathsf{In}\,\mathbb{C} \to \mathsf{Base}$$

PSh_1 is a universe supporting ETT (semantically: universe of sets).

Basic things we can do

$$\mathsf{U} : \mathsf{U} \qquad \mathsf{Base} : \mathsf{U} \qquad \mathbf{1} : \mathsf{Base} \qquad \mathsf{PSh} : \mathsf{Base} \to \mathsf{U}$$

$$\mathsf{Cat}_i : \mathsf{PSh}_i := \textit{type of cats in } \mathsf{PSh}_i \qquad \mathsf{In} : \mathsf{Cat}_i \to \mathsf{U} \qquad \mathsf{base} : \mathsf{In}\,\mathbb{C} \to \mathsf{Base}$$

PSh_1 is a universe supporting ETT (semantically: universe of sets).

We can define some $\mathbb{C} : \mathsf{Cat}_1$, where $\mathsf{Obj}(\mathbb{C}) : \mathsf{PSh}_1$.

Basic things we can do

$$\mathsf{U} : \mathsf{U} \qquad \mathsf{Base} : \mathsf{U} \qquad \mathbf{1} : \mathsf{Base} \qquad \mathsf{PSh} : \mathsf{Base} \to \mathsf{U}$$

$$\mathsf{Cat}_i : \mathsf{PSh}_i := \textit{type of cats in } \mathsf{PSh}_i \qquad \mathsf{In} : \mathsf{Cat}_i \to \mathsf{U} \qquad \mathsf{base} : \mathsf{In}\,\mathbb{C} \to \mathsf{Base}$$

PSh_1 is a universe supporting ETT (semantically: universe of sets).

We can define some $\mathbb{C} : \mathsf{Cat}_1$, where $\mathsf{Obj}(\mathbb{C}) : \mathsf{PSh}_1$.

Now, under the assumption of $i : \mathsf{In}\,\mathbb{C}$, we can form the universe $\mathsf{PSh}_{(\mathsf{base}\,i)}$, which is semantically the universe of presheaves over $\mathbb{C}$.

Basic things we can do

$$\mathsf{U} : \mathsf{U} \qquad \mathsf{Base} : \mathsf{U} \qquad \mathbf{1} : \mathsf{Base} \qquad \mathsf{PSh} : \mathsf{Base} \to \mathsf{U}$$
$$\mathsf{Cat}_i : \mathsf{PSh}_i := \textit{type of cats in } \mathsf{PSh}_i \qquad \mathsf{In} : \mathsf{Cat}_i \to \mathsf{U} \qquad \mathsf{base} : \mathsf{In}\,\mathbb{C} \to \mathsf{Base}$$

PSh_1 is a universe supporting ETT (semantically: universe of sets).

We can define some $\mathbb{C} : \mathsf{Cat}_1$, where $\mathsf{Obj}(\mathbb{C}) : \mathsf{PSh}_1$.

Now, under the assumption of $i : \mathsf{In}\,\mathbb{C}$, we can form the universe $\mathsf{PSh}_{(\mathsf{base}\,i)}$, which is semantically the universe of presheaves over $\mathbb{C}$.

At this point, we have no interesting interaction between PSh_1 and PSh_i.

Syntactic sugar: we'll omit "base" in the following.

Example: embedding pure lambda calculus

A **second-order model of pure LC** in PSh_i consists of:

$$
\begin{aligned}
&\mathsf{Tm} : \mathsf{PSh}_i \\
&\mathsf{lam} : (\mathsf{Tm} \to \mathsf{Tm}) \to \mathsf{Tm} \\
&\text{–\$–} : \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{Tm} \\
&\beta \quad : \mathsf{lam}\, f \,\$\, t = f\ t \\
&\eta \quad : \mathsf{lam}\,(\lambda x.\, t \,\$\, x) = t
\end{aligned}
$$

We define $\mathsf{SMod}_i : \mathsf{PSh}_i$ as the above Σ-type.

Example: embedding pure lambda calculus

A **first-order model of pure LC** consists of:

- A category of contexts and substitutions with $\mathsf{Con} : \mathsf{PSh}_i$, $\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{PSh}_i$ and terminal object $\bullet$.
- $\mathsf{Tm} : \mathsf{Con} \to \mathsf{PSh}_i$, plus a term substitution operation.
- A context extension operation $- \triangleright : \mathsf{Con} \to \mathsf{Con}$ such that $\mathsf{Sub}\,\Gamma\,(\Delta \triangleright) \simeq \mathsf{Sub}\,\Gamma\,\Delta \times \mathsf{Tm}\,\Gamma$.
- A natural isomorphism $\mathsf{Tm}\,(\Gamma \triangleright) \simeq \mathsf{Tm}\,\Gamma$ whose components are λ and application.

We define $\mathsf{FMod}_i : \mathsf{PSh}_i$ as the above Σ-type.

FMod is mechanically derivable from SMod.[1]

[1] Ambrus Kaposi & Szumi Xie: *Second-Order Generalised Algebraic Theories*.

Example: embedding pure lambda calculus

GLF Axiom 1

Assuming $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.

(In "$\mathsf{In}\, M$" we implicitly convert M to its underlying category.)

Now we have a 2LTT inside PSh_j:

- ETT type formers in PSh_j comprise the outer level.
- S_j comprises the inner level.

Example: embedding pure lambda calculus

GLF Axiom 1

Assuming $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.
(In "$\mathsf{In}\, M$" we implicitly convert M to its underlying category.)

Now we have a 2LTT inside PSh_j:

- ETT type formers in PSh_j comprise the outer level.
- S_j comprises the inner level.

Y-combinator as example:

$$\mathsf{YC} : \mathsf{Tm}_{\mathsf{S}_j}$$

$$\mathsf{YC} := \mathsf{lam}_{\mathsf{S}_j}(\lambda f.\, (\mathsf{lam}_{\mathsf{S}_j}(\lambda x.\, x\, \$_{\mathsf{S}_j}\, x))\, \$_{\mathsf{S}_j}\, (\mathsf{lam}_{\mathsf{S}_j}(\lambda f.\, \mathsf{lam}_{\mathsf{S}_j}(\lambda x.\, f\, \$_{\mathsf{S}_j}\, (x\, \$_{\mathsf{S}_j}\, x)))))$$

Example: embedding pure lambda calculus

GLF Axiom 1

Assuming $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.
(In "$\mathsf{In}\, M$" we implicitly convert M to its underlying category.)

Now we have a 2LTT inside PSh_j:

- ETT type formers in PSh_j comprise the outer level.
- S_j comprises the inner level.

Y-combinator as example:

$$\mathsf{YC} : \mathsf{Tm}_{\mathsf{S}_j}$$
$$\mathsf{YC} := \mathsf{lam}_{\mathsf{S}_j}(\lambda f.\, (\mathsf{lam}_{\mathsf{S}_j}(\lambda x.\, x\, \$_{\mathsf{S}_j}\, x))\, \$_{\mathsf{S}_j}\, (\mathsf{lam}_{\mathsf{S}_j}(\lambda f.\, \mathsf{lam}_{\mathsf{S}_j}(\lambda x.\, f\, \$_{\mathsf{S}_j}\, (x\, \$_{\mathsf{S}_j}\, x)))))$$

With a reasonable amount of sugar:

$$\mathsf{YC} : \mathsf{Tm}_{\mathsf{S}_j}$$
$$\mathsf{YC} := \mathsf{lam}\, f.\, (\mathsf{lam}\, x.\, x\, x)\, (\mathsf{lam}\, f.\, \mathsf{lam}\, x.\, f\, (x\, x))$$

- More generally, we have the previous axiom for every second-order generalized algebraic theory.
- Hence: all 2LTTs are syntactic fragments of GLF.
- For each 2LTT, the semantics of GLF restricts to the standard presheaf semantics of the 2LTT.

Moving between internal & external views

GLF Axiom: Yoneda embedding for pure LC

Assuming $M : \mathsf{FMod}_i$, we have

$$\begin{aligned}
\mathsf{Y} : \mathsf{Con}_M \quad &\to ((j : \mathsf{In}_M) \to \mathsf{PSh}_j) \\
\mathsf{Y} : \mathsf{Sub}_M\, \Gamma\, \Delta &\simeq ((j : \mathsf{In}_M) \to \mathsf{Y}\, \Gamma\, j \to \mathsf{Y}\, \Delta\, j) \\
\mathsf{Y} : \mathsf{Tm}_M\, \Gamma \quad &\simeq ((j : \mathsf{In}_M) \to \mathsf{Y}\, \Gamma\, j \to \mathsf{Tm}_{\mathsf{S}_j})
\end{aligned}$$

such that Y preserves empty context and context extension up to iso:

$$\begin{aligned}
\mathsf{Y}\, \bullet\, j \quad &\simeq \top \\
\mathsf{Y}\, (\Gamma \triangleright)\, j &\simeq \mathsf{Y}\, \Gamma\, j \times \mathsf{Tm}_{\mathsf{S}_j}
\end{aligned}$$

and Y preserves all other structure strictly.

Notation: we write Λ for inverse Y.

LC examples, sugar

Y and Λ allow ad-hoc switching between first-order and second-order notation. Let's redefine some operations using second-order notation:

$$\mathsf{id} : \mathsf{Sub}_M\,\Gamma\,\Gamma \qquad \mathsf{comp} : \mathsf{Sub}_M\,\Delta\,\Theta \to \mathsf{Sub}_M\,\Gamma\,\Delta \to \mathsf{Sub}_M\,\Gamma\,\Theta$$

$$\mathsf{id} := \Lambda\,(\lambda\, j\, \gamma.\, \gamma) \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,(\lambda\, j\, \gamma.\, \mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma\, j)\, j)$$

LC examples, sugar

Y and Λ allow ad-hoc switching between first-order and second-order notation. Let's redefine some operations using second-order notation:

$$\mathsf{id} : \mathsf{Sub}_M\,\Gamma\,\Gamma \qquad \mathsf{comp} : \mathsf{Sub}_M\,\Delta\,\Theta \to \mathsf{Sub}_M\,\Gamma\,\Delta \to \mathsf{Sub}_M\,\Gamma\,\Theta$$
$$\mathsf{id} := \Lambda\,(\lambda\, j\, \gamma.\, \gamma) \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,(\lambda\, j\, \gamma.\, \mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma\, j)\, j)$$

With reasonable amount of sugar:

$$\mathsf{id} := \Lambda\,\gamma.\,\gamma \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,\gamma.\,\mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma)$$

LC examples, sugar

Y and Λ allow ad-hoc switching between first-order and second-order notation. Let's redefine some operations using second-order notation:

$$\mathsf{id} : \mathsf{Sub}_M\,\Gamma\,\Gamma \qquad \mathsf{comp} : \mathsf{Sub}_M\,\Delta\,\Theta \to \mathsf{Sub}_M\,\Gamma\,\Delta \to \mathsf{Sub}_M\,\Gamma\,\Theta$$
$$\mathsf{id} := \Lambda\,(\lambda\, j\, \gamma.\, \gamma) \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,(\lambda\, j\, \gamma.\, \mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma\, j)\, j)$$

With reasonable amount of sugar:

$$\mathsf{id} := \Lambda\,\gamma.\,\gamma \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,\gamma.\,\mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma)$$

Or even:

$$\mathsf{comp}\,\sigma\,\delta := \Lambda\,\gamma.\,\sigma\,(\delta\,\gamma)$$

LC examples, sugar

Y and Λ allow ad-hoc switching between first-order and second-order notation. Let's redefine some operations using second-order notation:

$$\mathsf{id} : \mathsf{Sub}_M\,\Gamma\,\Gamma \qquad \mathsf{comp} : \mathsf{Sub}_M\,\Delta\,\Theta \to \mathsf{Sub}_M\,\Gamma\,\Delta \to \mathsf{Sub}_M\,\Gamma\,\Theta$$
$$\mathsf{id} := \Lambda\,(\lambda\, j\, \gamma.\, \gamma) \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,(\lambda\, j\, \gamma.\, \mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma\, j)\, j)$$

With reasonable amount of sugar:

$$\mathsf{id} := \Lambda\,\gamma.\,\gamma \qquad \mathsf{comp}\,\sigma\,\delta := \Lambda\,\gamma.\,\mathsf{Y}\,\sigma\,(\mathsf{Y}\,\delta\,\gamma)$$

Or even:

$$\mathsf{comp}\,\sigma\,\delta := \Lambda\,\gamma.\,\sigma\,(\delta\,\gamma)$$

Example for "pattern matching" notation (Y preserves extended contexts):

$$\mathsf{p} : \mathsf{Sub}_M\,(\Gamma\,\triangleright)\,\Gamma$$
$$\mathsf{p} := \Lambda\,(\gamma,\, \alpha).\,\gamma$$

Second-order named notation

- When working with CwF-s, De Bruijn indices and substitutions can be hard to read.
- Handwaved "named" binders in CwFs have been used a couple of times.
- GLF provides a rigorous implementation of such notation.

Embedding a type theory

In a first order model, we have:

$$\begin{aligned}
&\mathsf{Con} : \mathsf{PSh}_i \\
&\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Ty} \ : \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Tm} \ : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

In a second order model, we have

$$\begin{aligned}
&\mathsf{Ty} \ : \mathsf{PSh}_i \\
&\mathsf{Tm} : \mathsf{Ty} \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

Embedding a type theory

In a first order model, we have:

$$\begin{aligned}
&\mathsf{Con} : \mathsf{PSh}_i \\
&\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Ty} \ : \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Tm} \ : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

In a second order model, we have

$$\begin{aligned}
&\mathsf{Ty} \ : \mathsf{PSh}_i \\
&\mathsf{Tm} : \mathsf{Ty} \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

Sugar for contexts & sorts.

$$(\Gamma \triangleright A \triangleright B) : \mathsf{Con}_M \quad \text{is equal to} \quad \Gamma \triangleright (\Lambda\,\gamma.\mathsf{Y}A\,\gamma) \triangleright (\Lambda\,(\gamma,\,\alpha).\mathsf{Y}B\,(\gamma,\,\alpha))$$

Embedding a type theory

In a first order model, we have:

$$\begin{aligned}
&\mathsf{Con} : \mathsf{PSh}_i \\
&\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Ty} \ : \mathsf{Con} \to \mathsf{PSh}_i \\
&\mathsf{Tm} \ : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

In a second order model, we have

$$\begin{aligned}
&\mathsf{Ty} \ : \mathsf{PSh}_i \\
&\mathsf{Tm} : \mathsf{Ty} \to \mathsf{PSh}_i \\
&\ldots
\end{aligned}$$

Sugar for contexts & sorts.

$$(\Gamma \triangleright A \triangleright B) : \mathsf{Con}_M \quad \text{is equal to} \quad \Gamma \triangleright (\Lambda\,\gamma.\mathsf{Y}A\,\gamma) \triangleright (\Lambda\,(\gamma,\,\alpha).\mathsf{Y}B\,(\gamma,\,\alpha))$$

This suggests the notation:

$$(\gamma : \Gamma,\, \alpha : \mathsf{Y}A\,\gamma,\, \beta : \mathsf{Y}B\,(\gamma,\,\alpha)) : \mathsf{Con}_M$$

With implicit Y:

$$(\gamma : \Gamma,\, \alpha : A\,\gamma,\, \beta : B\,(\gamma,\,\alpha)) : \mathsf{Con}_M$$

Embedding a type theory

More "contextual" Sugar for Tm_M. We have

$$\mathsf{Tm}_M\,(\Gamma \triangleright A \triangleright B)\,C \;=\; \mathsf{Tm}_M\,(\Gamma \triangleright A \triangleright B)\,(\Lambda\,(\gamma,\,\alpha,\,\beta).\,B\,(\gamma,\,\alpha,\,\beta))$$

which suggests the notation

$$\mathsf{Tm}_M\,(\gamma : \Gamma,\,\alpha : A\,\gamma,\,\beta : B\,(\gamma,\,\alpha))\,(B\,(\gamma,\,\alpha,\,\beta))$$

Embedding a type theory

Example: a construction which looks kind of awful in explicit CwF notation.[2]

$$
\begin{aligned}
&\mathsf{Con}^\circ\,\Gamma && := \mathsf{Ty}\,(F\,\Gamma)\\
&\mathsf{Ty}^\circ\,\Gamma^\circ\,A && := \mathsf{Ty}\,(F\,\Gamma \triangleright \Gamma^\circ \triangleright F\,A[\mathsf{p}])\\
&\mathsf{Tm}^\circ\,\Gamma^\circ\,A^\circ\,t && := \mathsf{Tm}\,(F\,\Gamma \triangleright \Gamma^\circ)\,(A^\circ[\mathsf{id},\, F\,t[\mathsf{p}]])\\
&\Gamma^\circ \triangleright^\circ A^\circ && := \Sigma(\Gamma^\circ[\mathsf{p} \circ F_{\triangleright.1}])(A^\circ[\mathsf{p} \circ F_{\triangleright.1} \circ \mathsf{p},\, 0,\, \mathsf{q}[F_{\triangleright.1} \circ \mathsf{p}]])\\
&\ldots
\end{aligned}
$$

But is reasonable in sugary GLF notation:

$$
\begin{aligned}
&\mathsf{Con}^\circ\,\Gamma && := \mathsf{Ty}\,(\gamma : F\,\Gamma)\\
&\mathsf{Ty}^\circ\,\Gamma^\circ\,A && := \mathsf{Ty}\,(\gamma : F\,\Gamma,\, \gamma^\circ : \Gamma^\circ\,\gamma,\, \alpha : F\,A\,\gamma)\\
&\Gamma^\circ \triangleright^\circ A^\circ && := \Lambda\,(F_{\triangleright.1}(\gamma,\, \alpha)).\,\Sigma(\gamma^\circ : \Gamma^\circ\,\gamma).\,(A^\circ\,(\gamma,\, \gamma^\circ,\, \alpha))\\
&\ldots
\end{aligned}
$$

[2] Kaposi, Huber, Sattler: *Gluing for Type Theory*