A Generalized Logical Framework

András Kovács[1], Christian Sattler[1]

[1]University of Gothenburg & Chalmers University of Technology

12 Jun 2025, TYPES 2025, Glasgow

Overview

1. Two-level type theories (2LTT) & logical frameworks (LF):
   - 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) & logical frameworks (LF):
   - 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) & logical frameworks (LF):
   - 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) & logical frameworks (LF):
   - 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 rules

$\mathbf{U}$	$: \mathbf{U}$	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: $\mathsf{PSh}_i \subseteq \mathsf{U}$. 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 $\mathbb{C} : \mathsf{Cat}_i$.
$\mathbf{base}$	$: \mathbf{In}\,\mathbb{C} \to \mathbf{Base}$	"Using the permission".

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

Basic things we can do

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

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

PSh_1 is a universe supporting ETT. Semantically, PSh_1 is a universe of sets.

Basic things we can do

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

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

PSh_1 is a universe supporting ETT. Semantically, PSh_1 is a 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

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

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

PSh_1 is a universe supporting ETT. Semantically, PSh_1 is a 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

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

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

PSh_1 is a universe supporting ETT. Semantically, PSh_1 is a 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:

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

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** is a unityped category with families with λ-abstraction and application.

For every object theory, the definition of FMod is mechanically derivable from SMod.[1]

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

Example: embedding pure lambda calculus

GLF rule

Internally to presheaves over a first-order model, we have a second-order model.
Formally: given $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.

Example: embedding pure lambda calculus

GLF rule

Internally to presheaves over a first-order model, we have a second-order model.
Formally: given $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.

We have 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 rule

Internally to presheaves over a first-order model, we have a second-order model.
Formally: given $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.

We have 2LTT inside PSh_j:

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

Y-combinator as example:

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

Example: embedding pure lambda calculus

GLF rule

Internally to presheaves over a first-order model, we have a second-order model.
Formally: given $M : \mathsf{FMod}_i$ and $j : \mathsf{In}\, M$, we have $\mathsf{S}_j : \mathsf{SMod}_j$.

We have 2LTT inside PSh_j:

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

Y-combinator as example:

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

With a reasonable amount of sugar:

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

- More generally, we have the previous GLF rule for every second-order generalized algebraic theory.

- More generally, we have the previous GLF rule for every second-order generalized algebraic theory.
- Hence: all 2LTTs are syntactic fragments of GLF.

Yoneda: conversion between internal & external views

GLF rule: Yoneda embedding for pure LC

Assuming $M : \mathsf{FMod}_i$ and writing $\simeq$ for definitional isomorphism, we have

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

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

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

and Y preserves all other structure strictly.

Notation: we write Λ for inverses of 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:

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

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:

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

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:

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

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:

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

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:

$$\begin{array}{ll} \mathsf{p} : \mathsf{Sub}_M\,(\Gamma\,\triangleright)\,\Gamma & \\ \mathsf{p} := \Lambda\,(\gamma,\,\alpha).\,\gamma & \textit{Note: } \mathsf{Y}\,(\Gamma\,\triangleright) \simeq \mathsf{Y}\,\Gamma \times \mathsf{Tms}_j \end{array}$$

Embedding dependent type theories

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 dependent type theories

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

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

Yoneda embedding:

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

Embedding dependent type theories

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

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

[2] Kaposi, Huber, Sattler: *Gluing for Type Theory*, Section 5

Embedding dependent type theories

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

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

but is reasonable in sugary GLF notation:

$$
\begin{aligned}
&\mathsf{Con}^{\circ}\,\Gamma && := \mathsf{Ty}\,(\gamma : F\,\Gamma)\\
&\mathsf{Ty}^{\circ}\,\Gamma^{\circ}\,A && := \mathsf{Ty}\,(\gamma : F\,\Gamma,\,\gamma^{\circ} : \Gamma^{\circ}\,\gamma,\,\alpha : F\,A\,\gamma)\\
&\mathsf{Tm}^{\circ}\,\Gamma^{\circ}\,A^{\circ}\,t && := \mathsf{Tm}\,(\gamma : F\,\Gamma,\,\gamma^{\circ} : \Gamma^{\circ}\,\gamma)\,(A^{\circ}\,(\gamma,\,\gamma^{\circ},\,F\,t\,\gamma))\\
&\Gamma^{\circ} \triangleright^{\circ} A^{\circ} && := \Lambda\,(F_{\triangleright.2}(\gamma,\,\alpha)).\,\Sigma(\gamma^{\circ} : \Gamma^{\circ}\,\gamma) \times A^{\circ}\,(\gamma,\,\gamma^{\circ},\,\alpha)
\end{aligned}
$$

[2] Kaposi, Huber, Sattler: *Gluing for Type Theory*, Section 5

Embedding dependent type theories

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

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

but is reasonable in sugary GLF notation:

$$
\begin{aligned}
&\mathsf{Con}^\circ\,\Gamma && := \mathsf{Ty}\,(\gamma : F\,\Gamma) \\
&\mathsf{Ty}^\circ\,\Gamma^\circ\,A && := \mathsf{Ty}\,(\gamma : F\,\Gamma,\ \gamma^\circ : \Gamma^\circ\,\gamma,\ \alpha : F\,A\,\gamma) \\
&\mathsf{Tm}^\circ\,\Gamma^\circ\,A^\circ\,t && := \mathsf{Tm}\,(\gamma : F\,\Gamma,\ \gamma^\circ : \Gamma^\circ\,\gamma)\,(A^\circ\,(\gamma,\ \gamma^\circ,\ F\,t\,\gamma)) \\
&\Gamma^\circ \triangleright^\circ A^\circ && := \Lambda\,(F_{\triangleright.2}(\gamma,\ \alpha)).\,\Sigma(\gamma^\circ : \Gamma^\circ\,\gamma) \times A^\circ\,(\gamma,\ \gamma^\circ,\ \alpha)
\end{aligned}
$$

It's a lot of sugar, but we can always rigorously desugar when in doubt!

[2] Kaposi, Huber, Sattler: *Gluing for Type Theory*, Section 5

Semantics

Each PSh_i should be an universe of internal presheaves over an internal category.

Semantics

Each PSh_i should be an universe of internal presheaves over an internal category.

We should work with **Cat** somehow, but there are issues with that:

- There's no general Π.
- Π-types of presheaves and universes of presheaves are not stable under reindexing by arbitrary functors.

Semantics

Each PSh_i should be an universe of internal presheaves over an internal category.

We should work with **Cat** somehow, but there are issues with that:

- There's no general Π.
- Π-types of presheaves and universes of presheaves are not stable under reindexing by arbitrary functors.

In GLF, we can't do any interesting categorical reasoning! Base and In are purely for managing internal/external languages.

Semantics

Each PSh_i should be an universe of internal presheaves over an internal category.

We should work with **Cat** somehow, but there are issues with that:

- There's no general Π.
- Π-types of presheaves and universes of presheaves are not stable under reindexing by arbitrary functors.

In GLF, we can't do any interesting categorical reasoning! Base and In are purely for managing internal/external languages.

GLF contexts are modeled as certain *trees of categories*:

- Each node represents a presheaf universe, each edge represents an internal/external switch.
- Tree morphisms only have non-trivial action on discrete data in trees.

Semantics

Notation:

- For a category C and a split fibration A over it, we write $C \triangleright A$ for the total category.
- For a presheaf A, we write $\mathrm{Disc}\, A$ for the derived discrete fibration.

Definition: trees of categories.

$$
\begin{aligned}
&\mathbf{data}\ \mathrm{Tree}\,(B : \mathrm{Cat}) : \mathrm{Set}\ \mathbf{where}\\
&\quad \mathrm{node} : (\Gamma : \mathrm{PSh}\, B)\\
&\qquad\quad \to (n : \mathbb{N})\\
&\qquad\quad \to (C : \mathrm{Fin}\, n \to \mathrm{Fib}\,(B \triangleright \mathrm{Disc}\,\Gamma))\\
&\qquad\quad \to ((i : \mathrm{Fin}\, n) \to \mathrm{Tree}\,(B \triangleright \mathrm{Disc}\,\Gamma \triangleright C\, i))\\
&\qquad\quad \to \mathrm{Tree}\, B
\end{aligned}
$$

Semantics

$$\mathsf{node} : (\Gamma : \mathsf{PSh}\, B)(n : \mathbb{N})(C : \mathsf{Fin}\, n \to \mathsf{Fib}\,(B \triangleright \mathsf{Disc}\,\Gamma)) \to ((i : \mathsf{Fin}\, n) \to \mathsf{Tree}\,(B \triangleright \mathsf{Disc}\,\Gamma \triangleright C\, i))$$
$$\to \mathsf{Tree}\, B$$

A GLF context is an element of $\mathsf{Tree}\,1$. We give some examples for semantic contexts. We have $\mathbb{N}_i : \mathsf{PSh}_i$. We use $- \triangleright -$ for "context extension" in presheaves as well.

$$\begin{aligned}
\bullet \quad & := \mathsf{node}\, 1\, 0\, []\, [] \\
(\bullet \triangleright \mathbb{N}_1) \quad & := \mathsf{node}\,(1 \triangleright \mathbb{N})\, 0\, []\, [] \\
(\bullet \triangleright \mathbb{N}_1 \triangleright \mathsf{In}\, C) \quad & := \mathsf{node}\,(1 \triangleright \mathbb{N})\, 1\, [C]\, [\mathsf{node}\, 1\, 0\, []\, []] \\
(\bullet \triangleright \mathbb{N}_1 \triangleright i : \mathsf{In}\, C \triangleright \mathbb{N}_{(\mathsf{base}\, i)}) & := \mathsf{node}\,(1 \triangleright \mathbb{N})\, 1\, [C]\, [\mathsf{node}\,(1 \triangleright \mathbb{N})\, 0\, []\, []]
\end{aligned}$$

Semantics

$$\mathsf{node} : (\Gamma : \mathsf{PSh}\, B)(n : \mathbb{N})(C : \mathsf{Fin}\, n \to \mathsf{Fib}\,(B \triangleright \mathsf{Disc}\,\Gamma)) \to ((i : \mathsf{Fin}\, n) \to \mathsf{Tree}\,(B \triangleright \mathsf{Disc}\,\Gamma \triangleright C\, i)) \\ \to \mathsf{Tree}\, B$$

A GLF context is an element of $\mathsf{Tree}\, 1$. We give some examples for semantic contexts. We have $\mathbb{N}_i : \mathsf{PSh}_i$. We use $- \triangleright -$ for "context extension" in presheaves as well.

$$\begin{aligned}
\bullet \quad &:= \mathsf{node}\, 1\, 0\, []\, [] \\
(\bullet \triangleright \mathbb{N}_1) \quad &:= \mathsf{node}\,(1 \triangleright \mathbb{N})\, 0\, []\, [] \\
(\bullet \triangleright \mathbb{N}_1 \triangleright \mathsf{In}\, C) \quad &:= \mathsf{node}\,(1 \triangleright \mathbb{N})\, 1\, [C]\, [\mathsf{node}\, 1\, 0\, []\, []] \\
(\bullet \triangleright \mathbb{N}_1 \triangleright i : \mathsf{In}\, C \triangleright \mathbb{N}_{(\mathsf{base}\, i)}) &:= \mathsf{node}\,(1 \triangleright \mathbb{N})\, 1\, [C]\, [\mathsf{node}\,(1 \triangleright \mathbb{N})\, 0\, []\, []]
\end{aligned}$$

- A Base in context Γ points to a node in Γ.
- An $\mathsf{In}\, C$ in context Γ points to a subtree of a node.
- Extending a context with $A : \mathsf{PSh}_i$ extends the presheaf in node i.
- Extending a context with $j : \mathsf{In}\, C$ for $C : \mathsf{Cat}_i$ adds a new subtree at node i.

Further work

- Decide on the exact rules of GLF.
- Compute the specification of Yoneda embedding from SOGAT signatures, define semantics in this generality.
- Investigate syntactic metatheory.
  - For computer implementation, we need to wean ourselves off extensional TT!
    - (but informal extensional GLF is already useful)
  - Definitional isos for Y are unusual in syntax.
  - Simpler syntactic fragments of GLF could be useful & easier to implement.

Further work

- Decide on the exact rules of GLF.
- Compute the specification of Yoneda embedding from SOGAT signatures, define semantics in this generality.
- Investigate syntactic metatheory.
  - For computer implementation, we need to wean ourselves off extensional TT!
    - (but informal extensional GLF is already useful)
  - Definitional isos for Y are unusual in syntax.
  - Simpler syntactic fragments of GLF could be useful & easier to implement.

Advertisement: please consider submitting to the TyDe 2025 workshop! Deadline 22nd June. It's colocated with ICFP in Singapore but remote presentation is possible!

Further work

- Decide on the exact rules of GLF.
- Compute the specification of Yoneda embedding from SOGAT signatures, define semantics in this generality.
- Investigate syntactic metatheory.
  - For computer implementation, we need to wean ourselves off extensional TT!
    - (but informal extensional GLF is already useful)
  - Definitional isos for Y are unusual in syntax.
  - Simpler syntactic fragments of GLF could be useful & easier to implement.

Advertisement: please consider submitting to the TyDe 2025 workshop! Deadline 22nd June. It's colocated with ICFP in Singapore but remote presentation is possible!

Thank you!