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: $\mathsf{PSh}_i \subseteq \mathsf{U}$. 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:

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

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 "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 "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 "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.)

Yoneda: conversion between internal & external views

GLF Axiom: 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:

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

and Y preserves all other structure strictly.

Notation: we write Λ for inverses of Y.

LC examples, sugar

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

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

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:

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

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 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

Sugar for contexts:

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

Embedding dependent type theories

Sugar for contexts:

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

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 dependent type theories

Sugar for contexts:

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

This suggests the notation:

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

With implicit Y:

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

Sugar for $\mathbf{Tm}_M$. 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 dependent type theories

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

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

but is reasonable in sugary GLF notation:

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

It's a fair amount of sugar, but we can always rigorously desugar when it doubt!

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

Sketch of 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, the categorical part (Base, In) is purely for bookkeeping, we can't do synthetic category theory. We can only do interesting things with presheaves.

GLF contexts are *trees of categories* where tree morphisms only have interesting action on "discrete" parts of the tree.

Sketch of 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. A *category telescope* is either the terminal category, or it is (inductively) of the form $C \triangleright \mathrm{Disc}\, A \triangleright B$ where C is a category telescope. We write $C : \mathrm{CatTel}$ for a category telescope.

Definition. A tree of categories is inductively defined as:

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

Sketch of 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 Tree 1. Some examples for semantic contexts. We have $\mathbb{N}_i : \mathsf{PSh}_i$. We use $- \triangleright -$ for "context extension" in presheaves as well.

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

Sketch of 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 $\mathsf{Tree}\,1$. Some examples for semantic contexts. We have $\mathbb{N}_i : \mathsf{PSh}_i$. We use $- \triangleright -$ for "context extension" in presheaves as well.

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

- A Base points to a node of the tree.
- An In 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 $C : \mathsf{Cat}_j$ adds a new subtree at node j.