Why is equality interesting?

Ambrus Kaposi

Dept. of Programming Languages and Compilers,
Eötvös Loránd University,
Budapest

5th World Logic Day
Rényi Institute, Budapest
13 January 2023

Plan

- First order logic as a second order generalised algebraic theory (SOGAT)
- Type theory as a SOGAT
- Type theory and set theory
- Equality type in type theory

FOL as SOGAT

$$
\begin{aligned}
\mathsf{Tm} &: \mathsf{Set}^{+} \\
\mathsf{For} &: \mathsf{Set} \\
- \supset - &: \mathsf{For} \to \mathsf{For} \to \mathsf{For} \\
\forall &: (\mathsf{Tm} \to \mathsf{For}) \to \mathsf{For} \\
\mathsf{Eq} &: \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{For}
\end{aligned}
$$

FOL as SOGAT

$$
\begin{array}{ll}
\mathsf{Tm} & : \mathsf{Set}^{+} \\
\mathsf{For} & : \mathsf{Set} \\
- \supset - & : \mathsf{For} \to \mathsf{For} \to \mathsf{For} \\
\forall & : (\mathsf{Tm} \to \mathsf{For}) \to \mathsf{For} \\
\mathsf{Eq} & : \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{For} \\
\hline
\mathsf{Pf} & : \mathsf{For} \to \mathsf{Set}^{+} \\
 & : (p, q : \mathsf{Pf}\, A) \to p = q \\
 & : (\mathsf{Pf}\, A \to \mathsf{Pf}\, B) \leftrightarrow \mathsf{Pf}\,(A \supset B) \\
 & : ((t : \mathsf{Tm}) \to \mathsf{Pf}\,(A\, t)) \leftrightarrow \mathsf{Pf}\,(\forall A) \\
 & : (t = t') \leftrightarrow \mathsf{Pf}\,(\mathsf{Eq}\, t\, t')
\end{array}
$$

Graphs

As a two-sorted algebraic theory:

$$\begin{aligned} \mathsf{V} &: \mathsf{Set} \\ \mathsf{E} &: \mathsf{Set} \\ \mathsf{start} &: \mathsf{E} \to \mathsf{V} \\ \mathsf{end} &: \mathsf{E} \to \mathsf{V} \end{aligned}$$

As a generalised algebraic theory:

$$\begin{aligned} \mathsf{V} &: \mathsf{Set} \\ \mathsf{E} &: \mathsf{V} \to \mathsf{V} \to \mathsf{Set} \end{aligned}$$

Transitive graphs

As an essentially algebraic theory:

$$
\begin{aligned}
\mathsf{V} &: \mathsf{Set} \\
\mathsf{E} &: \mathsf{Set} \\
\mathsf{start} &: \mathsf{E} \to \mathsf{V} \\
\mathsf{end} &: \mathsf{E} \to \mathsf{V} \\
\mathsf{tran} &: (f, g : \mathsf{E}) \to \mathsf{end}\, f = \mathsf{start}\, g \to \mathsf{E} \\
&: \mathsf{start}\,(\mathsf{tran}\, f\, g\, e) = \mathsf{start}\, f \\
&: \mathsf{end}\,(\mathsf{tran}\, f\, g\, e) = \mathsf{end}\, g
\end{aligned}
$$

As a generalised algebraic theory:

$$
\begin{aligned}
\mathsf{V} &: \mathsf{Set} \\
\mathsf{E} &: \mathsf{V} \to \mathsf{V} \to \mathsf{Set} \\
\mathsf{tran} &: \mathsf{E}\, x\, y \to \mathsf{E}\, y\, z \to \mathsf{E}\, x\, z
\end{aligned}
$$

FOL as SOGAT

$$
\begin{array}{ll}
\mathsf{Tm} & : \mathsf{Set}^{+} \\
\mathsf{For} & : \mathsf{Set} \\
- \supset - & : \mathsf{For} \to \mathsf{For} \to \mathsf{For} \\
\forall & : (\mathsf{Tm} \to \mathsf{For}) \to \mathsf{For} \\
\mathsf{Eq} & : \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{For} \\
\hline
\mathsf{Pf} & : \mathsf{For} \to \mathsf{Set}^{+} \\
 & : (p, q : \mathsf{Pf}\, A) \to p = q \\
 & : (\mathsf{Pf}\, A \to \mathsf{Pf}\, B) \leftrightarrow \mathsf{Pf}\,(A \supset B) \\
 & : ((t : \mathsf{Tm}) \to \mathsf{Pf}\,(A\, t)) \leftrightarrow \mathsf{Pf}\,(\forall A) \\
 & : (t = t') \leftrightarrow \mathsf{Pf}\,(\mathsf{Eq}\, t\, t')
\end{array}
$$

- algebraic theory, signature with equations
- generalised (Cartmell)
- second order
- every SOGAT has a first order GAT presentation: a category $\mathcal{C}$ together with the signature interpreted in presheaves over $\mathcal{C}$
- category of algebras
- syntax = initial algebra

Untyped lambda calculus as SOGAT

$$
\begin{aligned}
\mathsf{Tm} &: \mathsf{Set}^{+} \\
\mathsf{lam} &: (\mathsf{Tm} \to \mathsf{Tm}) \to \mathsf{Tm} \\
- \$ - &: \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{Tm} \\
\beta &: (\mathsf{lam}\, t) \,\$\, t' = t\, t'
\end{aligned}
$$

Simply typed lambda calculus as SOGAT

$$
\begin{aligned}
\mathsf{Ty} &: \mathsf{Set} \\
\mathsf{Tm} &: \mathsf{Ty} \to \mathsf{Set}^{+} \\
- \Rightarrow - &: \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\
\mathsf{lam} &: (\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \cong \mathsf{Tm}\, (A \Rightarrow B) : - \$ -
\end{aligned}
$$

Type theory as SOGAT (i)

$$
\begin{aligned}
&\mathsf{Ty} && : \mathsf{Set} \\
&\mathsf{Tm} && : \mathsf{Ty} \to \mathsf{Set}^{+} \\
&\Pi && : (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty} \\
&\mathsf{lam} && : \big((t : \mathsf{Tm}\,A) \to \mathsf{Tm}\,(B\,t)\big) \cong \mathsf{Tm}\,(\Pi\,A\,B) : - \,\$\, -
\end{aligned}
$$

Type theory as SOGAT (ii)

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^{+} \\
\hline
\Pi & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{lam} & : ((t : \mathsf{Tm}\,A) \to \mathsf{Tm}\,(B\,t)) \cong \mathsf{Tm}\,(\Pi\,A\,B) : -\,\$\,- \\
\hline
\mathsf{U} & : \mathsf{Ty} \\
\mathsf{El} & : \mathsf{Tm}\,\mathsf{U} \cong \mathsf{Ty} \\
\hline
\mathbb{N} & : \mathsf{Ty} \\
\mathsf{zero} & : \mathsf{Tm}\,\mathbb{N} \\
\mathsf{suc} & : \mathsf{Tm}\,\mathbb{N} \to \mathsf{Tm}\,\mathbb{N} \\
\mathsf{ind}\mathbb{N} & : (A : \mathsf{Tm}\,\mathbb{N} \to \mathsf{Ty}) \to \mathsf{Tm}\,(A\,\mathsf{zero}) \to \\
 & \quad \big((t : \mathsf{Tm}\,\mathbb{N}) \to \mathsf{Tm}\,(C\,t) \to \mathsf{Tm}\,(C\,(\mathsf{suc}\,t))\big) \to \\
 & \quad (t : \mathsf{Tm}\,\mathbb{N}) \to \mathsf{Tm}\,(A\,t) \\
\mathbb{N}\beta_1 & : \mathsf{ind}\mathbb{N}\,A\,z\,s\,\mathsf{zero} = z \\
\mathbb{N}\beta_2 & : \mathsf{ind}\mathbb{N}\,A\,z\,s\,(\mathsf{suc}\,t) = s\,t\,(\mathsf{ind}\mathbb{N}\,A\,z\,s\,t) \\
\hline
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty}
\end{array}
$$

Type theory vs. Set theory

$\mathsf{suc}\,\mathsf{zero} : \mathsf{Tm}\,\mathbb{N}$

$(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N}) : \mathsf{For}$

$\cdots : \mathsf{Pf}\,(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N})$

Type theory vs. Set theory

$\mathsf{suc}\,\mathsf{zero} : \mathsf{Tm}\,\mathbb{N}$

$(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N}) : \mathsf{For}$

$\cdots : \mathsf{Pf}\,(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N})$

$\mathsf{reflexivity} : \mathsf{Tm}\,(\mathsf{Eq}_{\mathbb{N}}\,(2+3)\,5)$

$\cdots : \mathsf{Pf}\,(\mathsf{Eq}\,(2+3)\,5)$

Type theory vs. Set theory

Type theory	Set theory
$\mathsf{suc}\,\mathsf{zero} : \mathsf{Tm}\,\mathbb{N}$	$(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N}) : \mathsf{For}$
	$\cdots : \mathsf{Pf}\,(\mathsf{suc}\,\mathsf{zero} \in \mathbb{N})$
$\mathsf{reflexivity} : \mathsf{Tm}\,(\mathsf{Eq}_{\mathbb{N}}\,(2+3)\,5)$	$\cdots : \mathsf{Pf}\,(\mathsf{Eq}\,(2+3)\,5)$
not expressible	$(\mathsf{zero} \in \mathsf{suc}\,\mathsf{zero}) : \mathsf{For}$

Type theory vs. Set theory

$\mathsf{suc\,zero} : \mathsf{Tm}\,\mathbb{N}$	$(\mathsf{suc\,zero} \in \mathbb{N}) : \mathsf{For}$
	$\cdots : \mathsf{Pf}\,(\mathsf{suc\,zero} \in \mathbb{N})$
$\mathsf{reflexivity} : \mathsf{Tm}\,(\mathsf{Eq}_{\mathbb{N}}\,(2+3)\,5)$	$\cdots : \mathsf{Pf}\,(\mathsf{Eq}\,(2+3)\,5)$
not expressible	$(\mathsf{zero} \in \mathsf{suc\,zero}) : \mathsf{For}$
everything respects isomorphisms (Voevodsky's univalence axiom)	$0 \in \{0,1\}$ and $0 \notin \{1,2\}$

Type checking

$\text{parse} : \text{String} \rightarrow \text{Maybe}\,\text{Pretm}$

Type checking

$$
\begin{aligned}
\mathsf{parse} &: \mathsf{String} \to \mathsf{Maybe}\,\mathsf{Pretm} \\
\mathsf{infer} &: \mathsf{Pretm} \to \mathsf{Maybe}\,\big((A : \mathsf{Ty}) \times \mathsf{Tm}\,A\big)
\end{aligned}
$$

Type checking

$$
\begin{aligned}
&\mathsf{parse} : \mathsf{String} \to \mathsf{Maybe}\,\textcolor{red}{\mathsf{Pretm}}\\
&\mathsf{infer}\ \ : \textcolor{red}{\mathsf{Pretm}} \to \mathsf{Maybe}\big((A : \mathsf{Ty}) \times \mathsf{Tm}\,A\big)\\
&\mathsf{infer}\,(\textcolor{red}{t\,\$\,t'}) := \mathsf{if}\ \mathsf{infer}\,\textcolor{red}{t} = (A \Rightarrow B, t)\ \mathsf{and}\\
&\qquad\qquad\qquad\quad\ \mathsf{infer}\,\textcolor{red}{t'} = (A', t')\ \mathsf{and}\ A = A'\ \mathsf{then}\ (B, t\,\$\,t')
\end{aligned}
$$

Type checking

$$\begin{aligned}
&\mathsf{parse} : \mathsf{String} \to \mathsf{Maybe}\,\mathsf{Pretm} \\
&\mathsf{infer}\ \ : \mathsf{Pretm} \to \mathsf{Maybe}\,\big((A : \mathsf{Ty}) \times \mathsf{Tm}\,A\big) \\
&\mathsf{infer}\,(t \,\$\, t') := \text{if } \mathsf{infer}\ t = (A \Rightarrow B, t) \text{ and} \\
&\qquad\qquad\qquad\quad \mathsf{infer}\ t' = (A', t') \text{ and } A = A' \text{ then } (B, t \,\$\, t')
\end{aligned}$$

- So equality of Ty (and Tm) should be decidable.

Type checking

$$
\begin{aligned}
&\text{parse} : \text{String} \to \text{Maybe}\,\text{Pretm} \\
&\text{infer} \;\, : \text{Pretm} \to \text{Maybe}\big((A : \text{Ty}) \times \text{Tm}\,A\big) \\
&\text{infer}\,(t \,\$\, t') := \text{if infer}\,t = (A \Rightarrow B, t) \text{ and} \\
&\qquad\qquad\qquad\quad \text{infer}\,t' = (A', t') \text{ and } A = A' \text{ then } (B, t \,\$\, t')
\end{aligned}
$$

- So equality of Ty (and Tm) should be decidable.
- Implemented through normalisation.

Type checking

$$\text{parse} : \text{String} \to \text{Maybe}\,\text{Pretm}$$
$$\text{infer} \;\; : \text{Pretm} \to \text{Maybe}\,\big((A : \text{Ty}) \times \text{Tm}\,A\big)$$
$$\begin{aligned}\text{infer}\,(t\,\$\,t') := \text{if infer}\;t = (A \Rightarrow B, t)\text{ and}\\ \text{infer}\;t' = (A', t')\text{ and } A = A' \text{ then } (B, t\,\$\,t')\end{aligned}$$

- So equality of Ty (and Tm) should be decidable.
- Implemented through normalisation.
- Implementations: Agda, Coq, Idris, Lean

Variants of the equality type

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^{+} \\
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\, A \to \mathsf{Tm}\, A \to \mathsf{Ty} \\
\hline
\end{array}
$$

Variants of the equality type

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^{+} \\
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty} \\
\hline
 & : (t = t') \cong \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \\
\hline
\end{array}
$$

Variants of the equality type

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^+ \\
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty} \\
\hline
 & : (t = t') \cong \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \\
\hline
\mathsf{refl} & : \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t) \\
\mathsf{transport} & : (P : \mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \to \mathsf{Tm}\,(P\,t) \to \mathsf{Tm}\,(P\,t') \\
\beta & : \mathsf{transport}\,P\,\mathsf{refl}\,u = u \\
\hline
\end{array}
$$

Variants of the equality type

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^{+} \\
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty} \\
\hline
 & : (t = t') \cong \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \\
\hline
\mathsf{refl} & : \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t) \\
\mathsf{transport} & : (P : \mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \to \mathsf{Tm}\,(P\,t) \to \mathsf{Tm}\,(P\,t') \\
\beta & : \mathsf{transport}\,P\,\mathsf{refl}\,u = u \\
\hline
 & : \mathsf{Eq}_A\,t\,t' = (f : \mathbb{I} \to A) \text{ such that } f\,\$\,0 = t \text{ and } f\,\$\,1 = t' \\
\hline
\end{array}
$$

Variants of the equality type

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set}^{+} \\
\mathsf{Eq} & : (A : \mathsf{Ty}) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty} \\
\hline
 & : (t = t') \cong \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \\
\hline
\mathsf{refl} & : \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t) \\
\mathsf{transport} & : (P : \mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Tm}\,(\mathsf{Eq}_A\,t\,t') \to \mathsf{Tm}\,(P\,t) \to \mathsf{Tm}\,(P\,t') \\
\beta & : \mathsf{transport}\,P\,\mathsf{refl}\,u = u \\
\hline
 & : \mathsf{Eq}_A\,t\,t' = (f : \mathbb{I} \to A) \text{ such that } f \mathbin{\$} 0 = t \text{ and } f \mathbin{\$} 1 = t' \\
\hline
 & : \mathsf{Eq}_{\mathbb{N}}\,\mathsf{zero}\,\mathsf{zero} = \top \\
 & : \mathsf{Eq}_{\mathbb{N}}\,(\mathsf{suc}\,t)\,(\mathsf{suc}\,t') = \mathsf{Eq}_{\mathbb{N}}\,t\,t' \\
 & : \mathsf{Eq}_{A \Rightarrow B}\,t\,t' = \Pi(x : A).\mathsf{Eq}_B\,(t \mathbin{\$} x)\,(t' \mathbin{\$} x) \\
 & : \mathsf{Eq}_{\mathsf{U}}\,A\,A' = \mathsf{Iso}\,A\,A'
\end{array}
$$

Difficulty: what are the rules for refl?

Summary

- A language with binders is a SOGAT
- Dependent types
- Lots of static information in type theory which ensures respecting isos
- Computer implementations
- What is a good type theory with univalence?