Type Theory in Type Theory using a Strictified Syntax

Ambrus Kaposi

Eötvös Loránd University, Budapest

j.w.w. Loïc Pujet

ICFP 2025
Singapore
14 October 2025

What is the syntax of type theory?

What is the syntax of type theory?

`"(λx.x+x) 3"`

`"(λx.x+x) 3a"`

`"((λx.x+x) 3)"`

`"(λx.x + x) 3"`

`"(λx.x+x 3"`

`"6"`

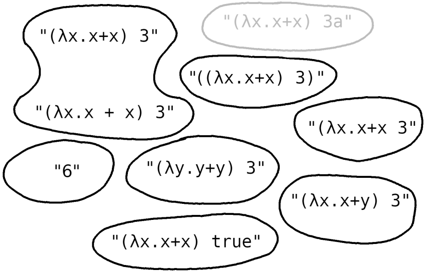
`"(λy.y+y) 3"`

`"(λx.x+y) 3"`

`"(λx.x+x) true"`

string

What is the syntax of type theory?



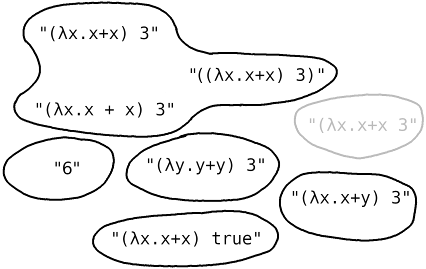
sequence of lexical elements

removal of spaces lexical analysis add spaces

invalid lexical elements

string

What is the syntax of type theory?



abstract syntax tree

removal of extra brackets — parsing — add brackets

bad number of parameters

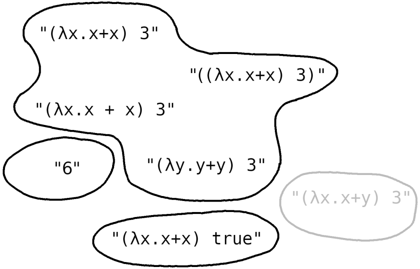
sequence of lexical elements

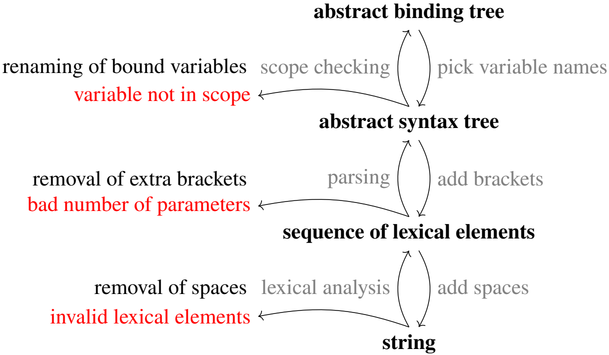
removal of spaces — lexical analysis — add spaces

invalid lexical elements

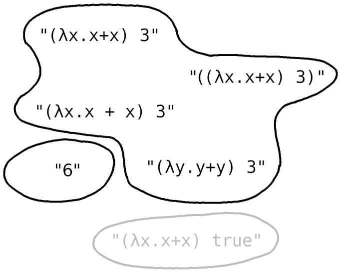
string

What is the syntax of type theory?





What is the syntax of type theory?



well-typed syntax

type checking

non-matching types

abstract binding tree

renaming of bound variables — scope checking — pick variable names

variable not in scope

abstract syntax tree

removal of extra brackets — parsing — add brackets

bad number of parameters

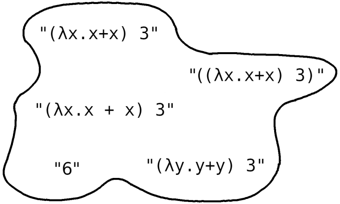
sequence of lexical elements

removal of spaces — lexical analysis — add spaces

invalid lexical elements

string

What is the syntax of type theory?



well-typed syntax quotiented by conversion

operational equivalence — normalise

well-typed syntax

type checking

non-matching types

abstract binding tree

renaming of bound variables — scope checking — pick variable names

variable not in scope

abstract syntax tree

removal of extra brackets — parsing — add brackets

bad number of parameters

sequence of lexical elements

removal of spaces — lexical analysis — add spaces

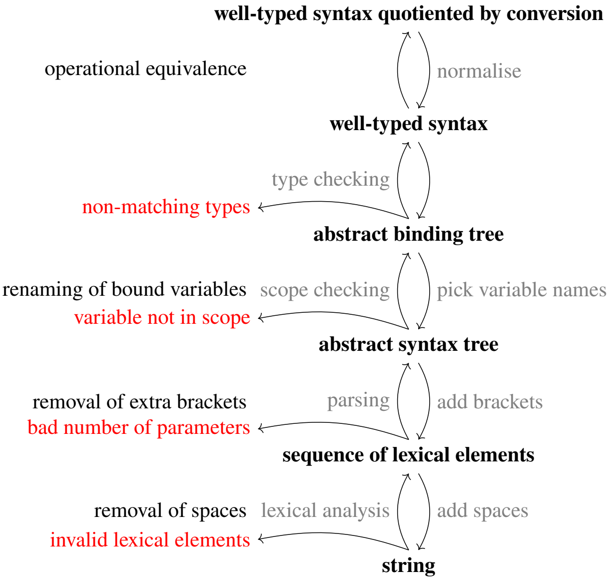
invalid lexical elements

string

What is the syntax of type theory?

Going abstract:

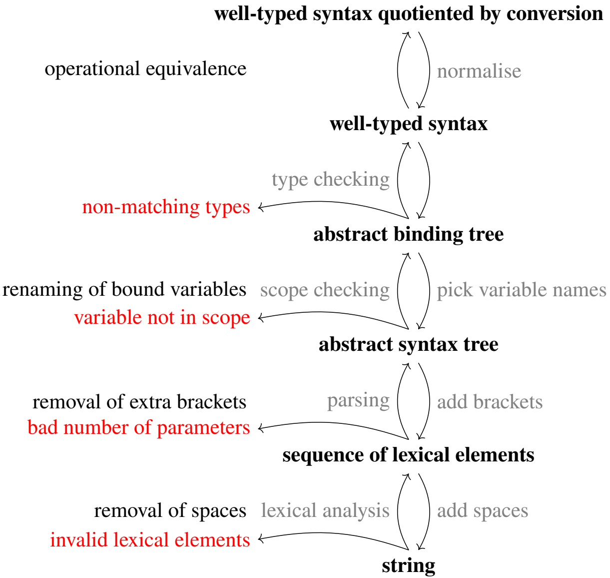
- forces choices:
  - De Bruijn
  - paranoid, not economic
  - explicit substitution



What is the syntax of type theory?

Going abstract:

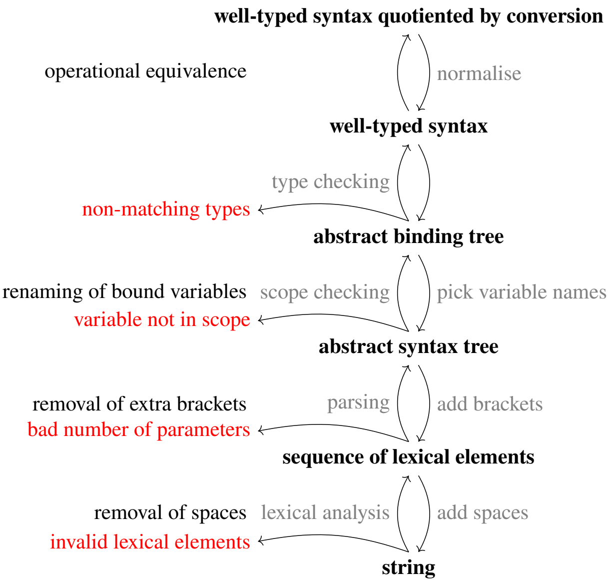
- forces choices:
  - De Bruijn
  - paranoid, not economic
  - explicit substitution
- we can't:
  - count the number of brackets
  - print the term
  - normalise?



What is the syntax of type theory?

Going abstract:

- forces choices:
  - De Bruijn
  - paranoid, not economic
  - explicit substitution
- we can't:
  - count the number of brackets
  - print the term
- we can:
  - normalise
    (gluing a.k.a. proof relevant
    logical predicates)



Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation.

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\, A : \mathbb{N} \to \mathsf{Set}$

$$\mathsf{rev} : \mathsf{Vec}\, A\, n \to \mathsf{Vec}\, A\, n$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\, A : \mathbb{N} \to \mathsf{Set}$

$$\mathsf{rev} : \mathsf{Vec}\, A\, n \to \mathsf{Vec}\, A\, n$$

$$\mathsf{rev}\, [] :\equiv []$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$

$$\begin{aligned}
&\mathsf{rev} : \mathsf{Vec}\,A\,n \to \mathsf{Vec}\,A\,n \\
&\mathsf{rev}\,[\,] :\equiv [\,] \\
&\mathsf{rev}\,\underbrace{(x :: xs)}_{:\mathsf{Vec}\,A\,(1+n)} :\equiv \underbrace{\mathsf{rev}\,xs \mathbin{+\!\!+} (x :: [\,])}_{:\mathsf{Vec}\,A\,(n+1)}
\end{aligned}$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$

$$
\begin{aligned}
&\mathsf{rev} : \mathsf{Vec}\,A\,n \to \mathsf{Vec}\,A\,n \\
&\mathsf{rev}\,[\,] :\equiv [\,] \\
&\mathsf{rev}\,\underbrace{(x :: xs)}_{:\mathsf{Vec}\,A\,(1+n)} :\equiv (\mathsf{comm}_+\,n\,1)_*\,\underbrace{(\mathsf{rev}\,xs +\!\!+ (x :: [\,]))}_{:\mathsf{Vec}\,A\,(n+1)}
\end{aligned}
$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$
   $$\begin{aligned} &\mathsf{rev} : \mathsf{Vec}\,A\,n \to \mathsf{Vec}\,A\,n \\ &\mathsf{rev}\,[] :\equiv [] \\ &\mathsf{rev}\,(x :: xs) :\equiv (\mathsf{comm}_+\,n\,1)_*\,(\mathsf{rev}\,xs \mathbin{+\!+} (x :: [])) \end{aligned}$$
3. substitution laws are propositional, not definitional
   $$(\mathsf{app}\,t\,u)[\sigma] = \mathsf{app}\,(t[\sigma])\,(u[\sigma])$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients ← Cubical TT, OTT solves this
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$
$$
\begin{aligned}
&\mathsf{rev} : \mathsf{Vec}\,A\,n \to \mathsf{Vec}\,A\,n\\
&\mathsf{rev}\,[] :\equiv []\\
&\mathsf{rev}\,(x :: xs) :\equiv (\mathsf{comm}_+\,n\,1)_*\,(\mathsf{rev}\,xs +\!\!+\,(x :: []))
\end{aligned}
$$
3. substitution laws are propositional, not definitional
$$(\mathsf{app}\,t\,u)[\sigma] = \mathsf{app}\,(t[\sigma])\,(u[\sigma])$$

Computer formalisation

In practice, formalisation happens at the ABT level:

- Abel–Öhman–Vezzosi 2018, MetaRocq (2014–2025), Martin-Löf à la Coq (Adjedj–Lennon-Bertrand–Maillard–Pédrot–Pujet 2024), Lean4Lean (Carneiro 2024)

Only very small CwF-style formalisations, and they are difficult to use. No formalisation of gluing-style normalisation. Reasons:

1. the syntax needs quotients ← Cubical TT, OTT solves this
2. $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{Set}$ induces transport hell ← this talk
   $\mathsf{Vec}\,A : \mathbb{N} \to \mathsf{Set}$
   $$\begin{aligned} \mathsf{rev} &: \mathsf{Vec}\,A\,n \to \mathsf{Vec}\,A\,n \\ \mathsf{rev}\,[] &:\equiv [] \\ \mathsf{rev}\,(x :: xs) &:\equiv (\mathsf{comm}_+\,n\,1)_*\,(\mathsf{rev}\,xs \mathbin{+\!\!+} (x :: [])) \end{aligned}$$
3. substitution laws are propositional, not definitional ← this talk
   $$(\mathsf{app}\,t\,u)[\sigma] = \mathsf{app}\,(t[\sigma])\,(u[\sigma])$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

  Examples: $$(\Pi\, A\, B)[\sigma] \equiv \Pi\, (A[\sigma])\, (B[\sigma^{\uparrow}])$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:
$$(\Pi\, A\, B)[\sigma] \equiv \Pi\, (A[\sigma])\, (B[\sigma^{\uparrow}])$$
$$(\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\, (t[\sigma])\, (u[\sigma])$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$(\Pi\, A\, B)[\sigma] \equiv \Pi\, (A[\sigma])\, (B[\sigma^{\uparrow}])$$
$$(\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\, (t[\sigma])\, (u[\sigma])$$
$$A[\mathsf{wk}][\mathsf{id}, t] \equiv A$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$(\Pi\, A\, B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}])$$
$$(\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma])$$
$$A[\mathsf{wk}][\mathsf{id}, t] \equiv A$$
$$(\mathsf{lam}\,\mathsf{v0})[\mathsf{id}, t] \equiv t$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$
\begin{aligned}
(\Pi\, A\, B)[\sigma] &\equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
(\mathsf{app}\, t\, u)[\sigma] &\equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
A[\mathsf{wk}][\mathsf{id}, t] &\equiv A \\
(\mathsf{lam}\, \mathsf{v0})[\mathsf{id}, t] &\equiv t \\
t[\sigma \circ \delta] &\equiv t[\sigma][\delta]
\end{aligned}
$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$
\begin{aligned}
&(\Pi\, A\, B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
&(\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
&A[\mathsf{wk}][\mathsf{id}, t] \equiv A \\
&(\mathsf{lam}\,\mathsf{v0})[\mathsf{id}, t] \equiv t \\
&t[\sigma \circ \delta] \equiv t[\sigma][\delta] \\
&(\sigma, t) \circ \delta \equiv (\sigma \circ \delta, t[\delta])
\end{aligned}
$$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$(\Pi\, A\, B)[\sigma] \equiv \Pi\, (A[\sigma])\, (B[\sigma^{\uparrow}])$$
$$(\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\, (t[\sigma])\, (u[\sigma])$$
$$A[\mathsf{wk}][\mathsf{id}, t] \equiv A$$
$$(\mathsf{lam}\, \mathsf{v0})[\mathsf{id}, t] \equiv t$$
$$t[\sigma \circ \delta] \equiv t[\sigma][\delta]$$
$$(\sigma, t) \circ \delta \equiv (\sigma \circ \delta, t[\delta])$$

Non-example: $\mathsf{id} = (\mathsf{wk}, \mathsf{v0})$

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

$$
\begin{array}{ll}
\text{Examples:} & (\Pi\, A\, B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
& (\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
& A[\mathsf{wk}][\mathsf{id}, t] \equiv A \\
& (\mathsf{lam}\, \mathsf{v0})[\mathsf{id}, t] \equiv t \\
& t[\sigma \circ \delta] \equiv t[\sigma][\delta] \\
& (\sigma, t) \circ \delta \equiv (\sigma \circ \delta, t[\delta]) \\
\text{Non-example:} & \mathsf{id} = (\mathsf{wk}, \mathsf{v0})
\end{array}
$$

- All CwF equations except the above are definitional (strict).

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

Examples:

$$
\begin{aligned}
(\Pi\, A\, B)[\sigma] &\equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
(\mathsf{app}\, t\, u)[\sigma] &\equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
A[\mathsf{wk}][\mathsf{id}, t] &\equiv A \\
(\mathsf{lam}\, \mathsf{v0})[\mathsf{id}, t] &\equiv t \\
t[\sigma \circ \delta] &\equiv t[\sigma][\delta] \\
(\sigma, t) \circ \delta &\equiv (\sigma \circ \delta, t[\delta])
\end{aligned}
$$

Non-example: $\mathsf{id} = (\mathsf{wk}, \mathsf{v0})$

- All CwF equations except the above are definitional (strict).
- Like making comm_{+} definitional.

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

$$\begin{aligned}
\text{Examples:} \quad & (\Pi\, A\, B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
& (\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
& A[\mathsf{wk}][\mathsf{id}, t] \equiv A \\
& (\mathsf{lam}\,\mathsf{v0})[\mathsf{id}, t] \equiv t \\
& t[\sigma \circ \delta] \equiv t[\sigma][\delta] \\
& (\sigma, t) \circ \delta \equiv (\sigma \circ \delta, t[\delta]) \\
\text{Non-example:} \quad & \mathsf{id} = (\mathsf{wk}, \mathsf{v0})
\end{aligned}$$

- All CwF equations except the above are definitional (strict).
- Like making comm_{+} definitional.
- The new model can replace the old one, e.g. a substitution-strict syntax.

Solution

- Model of type theory $\longmapsto$ new isomorphic model with definitional substitution laws.

$$
\begin{array}{ll}
\text{Examples:} & (\Pi\, A\, B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
& (\mathsf{app}\, t\, u)[\sigma] \equiv \mathsf{app}\,(t[\sigma])\,(u[\sigma]) \\
& A[\mathsf{wk}][\mathsf{id}, t] \equiv A \\
& (\mathsf{lam}\,\mathsf{v0})[\mathsf{id}, t] \equiv t \\
& t[\sigma \circ \delta] \equiv t[\sigma][\delta] \\
& (\sigma, t) \circ \delta \equiv (\sigma \circ \delta, t[\delta]) \\
\text{Non-example:} & \mathsf{id} = (\mathsf{wk}, \mathsf{v0})
\end{array}
$$

- All CwF equations except the above are definitional (strict).
- Like making comm_{+} definitional.
- The new model can replace the old one, e.g. a substitution-strict syntax.
- We formalised gluing style canoncity for a small type theory, the Agda proof is as beautiful as the paper one.

Analogs of our strictification technique

- Difference lists in the Haskell Prelude:
  replace $[a]$ by $[a] \to [a]$
  xs represented by $\lambda ys.\, xs +\!\!+\, ys$

Analogs of our strictification technique

- Difference lists in the Haskell Prelude:
  replace $[a]$ by $[a] \to [a]$
  xs represented by $\lambda ys.\, xs +\!\!+\, ys$

- Taking a category Ob, Hom and replacing $Hom(J, I)$ by

$$(\alpha : \forall K. Hom(K, J) \to Hom(K, I)) \qquad \text{such that} \qquad (\alpha\, K\, f) \circ g = \alpha\, L\, (f \circ g).$$

Analogs of our strictification technique

- Difference lists in the Haskell Prelude:
  replace $[a]$ by $[a] \to [a]$
  xs represented by $\lambda ys.\, xs +\!\!+\, ys$

- Taking a category Ob, Hom and replacing $Hom(J, I)$ by

$$\mathsf{y}\,J \xrightarrow{\bullet} \mathsf{y}\,I.$$

  The Yoneda lemma implies $Hom(J, I) \;\cong\; \mathsf{y}\,J \xrightarrow{\bullet} \mathsf{y}\,I.$

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.
  - Pédrot's strict presheaves (LICS 2020)

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.
  - Pédrot's strict presheaves (LICS 2020)

$$F : \mathsf{Disc}(\mathcal{C}) \to \mathcal{C}$$

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.
  - Pédrot's strict presheaves (LICS 2020)

$$F : \mathsf{Disc}(\mathcal{C}) \to \mathcal{C}$$

$$\begin{tikzcd} \mathsf{Psh}(\mathcal{C}) \arrow[rr, bend left, "F^*"] \arrow[rr, phantom, "\perp"] & & \mathsf{Psh}(\mathsf{Disc}(\mathcal{C})) \arrow[ll, bend left, "F_*"] \end{tikzcd}$$

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.
  - Pédrot's strict presheaves (LICS 2020)

$$F : \mathsf{Disc}(\mathcal{C}) \to \mathcal{C}$$

$$\begin{tikzcd} \mathsf{Psh}(\mathcal{C}) \arrow[r, bend left, "F^*"] \arrow[r, phantom, "\perp"] & \mathsf{Psh}(\mathsf{Disc}(\mathcal{C})) \arrow[l, bend left, "F_*"] \end{tikzcd}$$

$$\begin{aligned} |\mathsf{StrictPSh}(\mathcal{C})| :\equiv\ & (\Gamma : \mathcal{C} \to \mathsf{Set}) \\ & \times \big((I : \mathcal{C}) \to ((J : \mathcal{C}) \to \mathcal{C}(J, I) \to \Gamma\, J) \to \mathsf{Prop}\big) \end{aligned}$$

Inspiration for our strictification technique

- Presheaves as in HOAS, LF, 2lTT, SOGATs
- We need a strict CwF_{Π} of presheaves.
  - Pédrot's strict presheaves (LICS 2020)

$$F : \mathsf{Disc}(\mathcal{C}) \to \mathcal{C}$$

$$\begin{tikzcd} \mathsf{Psh}(\mathcal{C}) \arrow[rr, bend left, "F^*"] \arrow[rr, phantom, "\perp"] & & \mathsf{Psh}(\mathsf{Disc}(\mathcal{C})) \arrow[ll, bend left, "F_*"] \end{tikzcd}$$

$$\begin{aligned} |\mathsf{StrictPSh}(\mathcal{C})| :\equiv\ & (\Gamma : \mathcal{C} \to \mathsf{Set}) \\ & \times \big((I : \mathcal{C}) \to ((J : \mathcal{C}) \to \mathcal{C}(J, I) \to \Gamma\, J) \to \mathsf{Prop}\big) \end{aligned}$$

- Also called strictification ($\cong\ \to\ =$, while our method is $=\ \to\ \equiv$):
  - right adjoint splitting (Hofmann 1994)
  - left adjoint splitting, local universes (Lumsdaine–Warren 2015)

The strictification technique

- A universe closed under Σ:

$$
\begin{aligned}
\mathsf{U} &: \mathsf{Set} & \Sigma \quad &: (A : \mathsf{U}) \to (\mathsf{El}\, A \to \mathsf{U}) \to \mathsf{U} \\
\mathsf{El} &: \mathsf{U} \to \mathsf{Set} & -,- &: (a : \mathsf{El}\, A) \times \mathsf{El}\, B \cong \mathsf{El}\, (\Sigma\, A\, B) : \mathsf{fst}, \mathsf{snd}
\end{aligned}
$$

The strictification technique

- A universe closed under Σ:

$$\begin{aligned} &\mathsf{U} : \mathsf{Set} && \Sigma \quad : (A : \mathsf{U}) \to (\mathsf{El}\,A \to \mathsf{U}) \to \mathsf{U} \\ &\mathsf{El} : \mathsf{U} \to \mathsf{Set} && -,- : (a : \mathsf{El}\,A) \times \mathsf{El}\,B \cong \mathsf{El}\,(\Sigma\,A\,B) : \mathsf{fst}, \mathsf{snd} \end{aligned}$$

- Given a universe closed under certain type formers, we build a CwF closed under the same type formers inheriting the substitution calculus from the metatheory:

$$\begin{aligned} &\mathsf{Con} := \mathsf{Set} && \mathsf{Tm}\,\Gamma\,A := (\gamma : \Gamma) \to \mathsf{El}\,(A\,\gamma) \\ &\mathsf{Sub}\,\Delta\,\Gamma := \Delta \to \Gamma && A[\sigma] := A \circ \sigma \\ &\mathsf{Ty}\,\Gamma := \Gamma \to \mathsf{U} && \Sigma\,A\,B := \lambda\gamma.\,\Sigma\,(A\,\gamma)\,(\lambda a.\,B\,(\gamma, a)) \end{aligned}$$

$$(\Sigma\,A\,B)[\tau] = \lambda\delta.\,(A\,(\tau\,\delta))\,(\lambda a.\,B(\tau\,\delta, a)) = \Sigma\,(A[\tau])\,(B[\tau^{\uparrow}])$$

The strictification technique

- A universe closed under Σ:

$$\begin{aligned} &\mathsf{U} : \mathsf{Set} && \Sigma \quad : (A : \mathsf{U}) \to (\mathsf{El}\,A \to \mathsf{U}) \to \mathsf{U} \\ &\mathsf{El} : \mathsf{U} \to \mathsf{Set} && -,- : (a : \mathsf{El}\,A) \times \mathsf{El}\,B \cong \mathsf{El}\,(\Sigma\,A\,B) : \mathsf{fst}, \mathsf{snd} \end{aligned}$$

- Given a universe closed under certain type formers, we build a CwF closed under the same type formers inheriting the substitution calculus from the metatheory:

$$\begin{aligned} &\mathsf{Con} && := \mathsf{Set} && \mathsf{Tm}\,\Gamma\,A := (\gamma : \Gamma) \to \mathsf{El}\,(A\,\gamma) \\ &\mathsf{Sub}\,\Delta\,\Gamma && := \Delta \to \Gamma && A[\sigma] \quad := A \circ \sigma \\ &\mathsf{Ty}\,\Gamma && := \Gamma \to \mathsf{U} && \Sigma\,A\,B \quad := \lambda\gamma.\,\Sigma\,(A\,\gamma)\,(\lambda a.\,B\,(\gamma, a)) \end{aligned}$$

$$(\Sigma\,A\,B)[\tau] = \lambda\delta.\,(A\,(\tau\,\delta))\,(\lambda a.\,B(\tau\,\delta, a)) = \Sigma\,(A[\tau])\,(B[\tau^{\uparrow}])$$

- This is a substitution-strict model (called *contextualisation*, standard/set/type-model).

The strictification technique

- A universe closed under Σ:

$$\begin{aligned} &\mathsf{U} : \mathsf{Set} && \Sigma \quad : (A : \mathsf{U}) \to (\mathsf{El}\, A \to \mathsf{U}) \to \mathsf{U} \\ &\mathsf{El} : \mathsf{U} \to \mathsf{Set} && -,- : (a : \mathsf{El}\, A) \times \mathsf{El}\, B \cong \mathsf{El}\,(\Sigma\, A\, B) : \mathsf{fst}, \mathsf{snd} \end{aligned}$$

- Given a universe closed under certain type formers, we build a CwF closed under the same type formers inheriting the substitution calculus from the metatheory:

$$\begin{aligned} &\mathsf{Con} := \mathsf{Set} && \mathsf{Tm}\,\Gamma\, A := (\gamma : \Gamma) \to \mathsf{El}\,(A\,\gamma) \\ &\mathsf{Sub}\,\Delta\,\Gamma := \Delta \to \Gamma && A[\sigma] := A \circ \sigma \\ &\mathsf{Ty}\,\Gamma := \Gamma \to \mathsf{U} && \Sigma\, A\, B := \lambda\gamma.\, \Sigma\,(A\,\gamma)\,(\lambda a.\, B\,(\gamma, a)) \end{aligned}$$

$$(\Sigma\, A\, B)[\tau] = \lambda\delta.\,(A\,(\tau\,\delta))\,(\lambda a.\, B(\tau\,\delta, a)) = \Sigma\,(A[\tau])\,(B[\tau^{\uparrow}])$$

- This is a substitution-strict model (called *contextualisation*, standard/set/type-model).

- Internally to presheaves over a model supporting some type formers, we have a universe closed under the same type formers.

The strictification technique

- A universe closed under Σ:

$$\begin{aligned} &\mathsf{U} : \mathsf{Set} && \Sigma \quad : (A : \mathsf{U}) \to (\mathsf{El}\, A \to \mathsf{U}) \to \mathsf{U} \\ &\mathsf{El} : \mathsf{U} \to \mathsf{Set} && -,- : (a : \mathsf{El}\, A) \times \mathsf{El}\, B \cong \mathsf{El}\, (\Sigma\, A\, B) : \mathsf{fst}, \mathsf{snd} \end{aligned}$$

- Given a universe closed under certain type formers, we build a CwF closed under the same type formers inheriting the substitution calculus from the metatheory:

$$\begin{aligned} &\mathsf{Con} && := \mathsf{Set} && \mathsf{Tm}\, \Gamma\, A := (\gamma : \Gamma) \to \mathsf{El}\, (A\, \gamma) \\ &\mathsf{Sub}\, \Delta\, \Gamma && := \Delta \to \Gamma && A[\sigma] \quad := A \circ \sigma \\ &\mathsf{Ty}\, \Gamma && := \Gamma \to \mathsf{U} && \Sigma\, A\, B \quad := \lambda\gamma.\, \Sigma\, (A\, \gamma)\, (\lambda a.\, B\, (\gamma, a)) \end{aligned}$$

$$(\Sigma\, A\, B)[\tau] = \lambda\delta.\, (A\, (\tau\, \delta))\, (\lambda a.\, B(\tau\, \delta, a)) = \Sigma\, (A[\tau])\, (B[\tau^{\uparrow}])$$

- This is a substitution-strict model (called *contextualisation*, standard/set/type-model).
- Internally to presheaves over a model supporting some type formers, we have a universe closed under the same type formers.
- We take its contextualisation, then externalise (we need a strict CwF_{Π} of presheaves).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).
- Shallow embedding (K.–Kovács–Kraus MPC 2019).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).
- Shallow embedding (K.–Kovács–Kraus MPC 2019).
- Via conservativity of equality reflection (Winterhalter et al. CPP 2019, Winterhalter ICFP 2024).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).
- Shallow embedding (K.–Kovács–Kraus MPC 2019).
- Via conservativity of equality reflection (Winterhalter et al. CPP 2019, Winterhalter ICFP 2024).
- Rewrite rules (Cockx TYPES 2019, Leray et al. ITP 2024).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).
- Shallow embedding (K.–Kovács–Kraus MPC 2019).
- Via conservativity of equality reflection (Winterhalter et al. CPP 2019, Winterhalter ICFP 2024).
- Rewrite rules (Cockx TYPES 2019, Leray et al. ITP 2024).
- Local universe $\cong \longrightarrow =$ strictification also provides some $= \longrightarrow \equiv$ (Lumsdaine–Warren 2015).

Other ways to strictify

- Just work internally to an LF (Sterling PhD 2022, Bocquet–K.–Sattler FSCD 2023).
- Shallow embedding (K.–Kovács–Kraus MPC 2019).
- Via conservativity of equality reflection (Winterhalter et al. CPP 2019, Winterhalter ICFP 2024).
- Rewrite rules (Cockx TYPES 2019, Leray et al. ITP 2024).
- Local universe $\cong \longrightarrow =$ strictification also provides some $= \longrightarrow \equiv$ (Lumsdaine–Warren 2015).
- Redefining substitution recursively (K. TYPES 2023).

Summary

- A technique for strictifying ($= \longrightarrow \equiv$) the substitution calculus of a model of TT.
  - Intrinsic quotiented syntax is now even nicer than extrinsic syntax.
  - The first computer formalisation of gluing-style canonicity for type theory.

Summary

- A technique for strictifying ($= \to \equiv$) the substitution calculus of a model of TT.
  - Intrinsic quotiented syntax is now even nicer than extrinsic syntax.
  - The first computer formalisation of gluing-style canonicity for type theory.
- Problems:
  - Agda and Coq hang when trying to compute with the strictified syntax.
  - We lose some definitional computation rules for the eliminator of the syntax.

Summary

- A technique for strictifying ($= \to \equiv$) the substitution calculus of a model of TT.
  - Intrinsic quotiented syntax is now even nicer than extrinsic syntax.
  - The first computer formalisation of gluing-style canonicity for type theory.
- Problems:
  - Agda and Coq hang when trying to compute with the strictified syntax.
  - We lose some definitional computation rules for the eliminator of the syntax.
- Future work:
  - Strictify the substitution calculus for a model of any SOGAT.
  - Reusable library.
  - Direct efficient proof assistant support.

Weak CwF

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Set} \\
\mathsf{Sub} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
- \circ - &: \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma \\
\mathsf{ass} &: (\gamma \circ \delta) \circ \theta = \gamma \circ (\delta \circ \theta) \\
\mathsf{id} &: \mathsf{Sub}\,\Gamma\,\Gamma \\
\mathsf{idl} &: \mathsf{id} \circ \gamma = \gamma \\
\mathsf{idr} &: \gamma \circ \mathsf{id} = \gamma \\
\diamond &: \mathsf{Con} \\
\epsilon &: \mathsf{Sub}\,\Gamma\,\diamond \\
\diamond\eta &: (\sigma : \mathsf{Sub}\,\Gamma\,\diamond) \to \sigma = \epsilon \\
-[-] &: \mathsf{Ty}\,\Gamma \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Ty}\,\Delta \\
[\circ] &: A[\gamma \circ \delta] = A[\gamma][\delta] \\
[\mathsf{id}] &: A[\mathsf{id}] = A
\end{aligned}
$$

$$
\begin{aligned}
-[-] &: \mathsf{Tm}\,\Gamma\,A \to (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \\
&\quad\ \mathsf{Tm}\,\Delta\,(A[\gamma]) \\
[\circ] &: [\circ]_*\,(a[\gamma \circ \delta]) = a[\gamma][\delta] \\
[\mathsf{id}] &: [\mathsf{id}]_*\,(a[\mathsf{id}]) = a \\
- \triangleright - &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
-,- &: (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \mathsf{Tm}\,\Delta\,(A[\gamma]) \to \\
&\quad\ \mathsf{Sub}\,\Delta\,(\Gamma \triangleright A) \\
,\circ &: (\gamma, a) \circ \delta = (\gamma \circ \delta, [\circ]_*\,(a[\delta])) \\
\mathsf{p} &: \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma \\
\mathsf{q} &: \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}]) \\
\triangleright\beta_1 &: \mathsf{p} \circ (\gamma, a) = \gamma \\
\triangleright\beta_2 &: ([\circ] \cdot \triangleright\beta_1)_*\,(\mathsf{q}[\gamma, a]) = a \\
\triangleright\eta &: \mathsf{id} = (\mathsf{p}, \mathsf{q})
\end{aligned}
$$

Strict CwF

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Set} \\
\mathsf{Sub} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
- \circ - &: \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma \\
\mathsf{ass} &: (\gamma \circ \delta) \circ \theta \equiv \gamma \circ (\delta \circ \theta) \\
\mathsf{id} &: \mathsf{Sub}\,\Gamma\,\Gamma \\
\mathsf{idl} &: \mathsf{id} \circ \gamma \equiv \gamma \\
\mathsf{idr} &: \gamma \circ \mathsf{id} \equiv \gamma \\
\diamond &: \mathsf{Con} \\
\epsilon &: \mathsf{Sub}\,\Gamma\,\diamond \\
\diamond\eta &: (\sigma : \mathsf{Sub}\,\Gamma\,\diamond) \to \sigma \equiv \epsilon \\
-[-] &: \mathsf{Ty}\,\Gamma \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Ty}\,\Delta \\
[\circ] &: A[\gamma \circ \delta] \equiv A[\gamma][\delta] \\
[\mathsf{id}] &: A[\mathsf{id}] \equiv A
\end{aligned}
$$

$$
\begin{aligned}
-[-] &: \mathsf{Tm}\,\Gamma\,A \to (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \\
&\quad\ \mathsf{Tm}\,\Delta\,(A[\gamma]) \\
[\circ] &: a[\gamma \circ \delta] \equiv a[\gamma][\delta] \\
[\mathsf{id}] &: a[\mathsf{id}] \equiv a \\
- \triangleright - &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
-,- &: (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \mathsf{Tm}\,\Delta\,(A[\gamma]) \to \\
&\quad\ \mathsf{Sub}\,\Delta\,(\Gamma \triangleright A) \\
,\circ &: (\gamma, a) \circ \delta \equiv (\gamma \circ \delta, a[\delta]) \\
\mathsf{p} &: \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma \\
\mathsf{q} &: \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}]) \\
\triangleright\beta_1 &: \mathsf{p} \circ (\gamma, a) \equiv \gamma \\
\triangleright\beta_2 &: \mathsf{q}[\gamma, a] \equiv a \\
\triangleright\eta &: \mathsf{id} = (\mathsf{p}, \mathsf{q})
\end{aligned}
$$

Booleans in a weak CwF (i)

$$
\begin{aligned}
&\mathsf{Bool} &&: \mathsf{Ty}\,\Gamma\\
&\mathsf{Bool}[] &&: \mathsf{Bool}[\gamma] = \mathsf{Bool}\\
&\mathsf{true} &&: \mathsf{Tm}\,\Gamma\,\mathsf{Bool}\\
&\mathsf{true}[] &&: \mathsf{Bool}[]_*\,(\mathsf{true}[\gamma]) = \mathsf{true}\\
&\mathsf{false} &&: \mathsf{Tm}\,\Gamma\,\mathsf{Bool}\\
&\mathsf{false}[] &&: \mathsf{Bool}[]_*\,(\mathsf{false}[\gamma]) = \mathsf{false}\\
&\mathsf{ind} &&: (P : \mathsf{Ty}\,(\Gamma \triangleright \mathsf{Bool})) \to \mathsf{Tm}\,\Gamma\,(P[\langle \mathsf{true}\rangle]) \to \mathsf{Tm}\,\Gamma\,(P[\langle \mathsf{false}\rangle]) \to\\
& &&\quad (b : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) \to \mathsf{Tm}\,\Gamma\,(P[\langle b\rangle])\\
&\mathsf{ind}[] &&: (\alpha\,b)_*\,((\mathsf{ind}\,P\,p\,p'\,b)[\gamma]) =\\
& &&\quad \mathsf{ind}\,(\mathsf{Bool}[]_*\,(P[\gamma^{\uparrow}]))\,(\mathsf{true}[]_*\,((\alpha\,\mathsf{true})_*\,(p[\gamma])))\,(\mathsf{false}[]_*\,((\alpha\,\mathsf{false})_*\,(p'[\gamma])))\\
& &&\qquad (\mathsf{Bool}[]_*\,(b[\gamma]))\\
&\mathsf{Bool}\beta_1 &&: \mathsf{ind}\,P\,p\,p'\,\mathsf{true} = p\\
&\mathsf{Bool}\beta_2 &&: \mathsf{ind}\,P\,p\,p'\,\mathsf{false} = p'
\end{aligned}
$$

where

$$
\alpha : (u : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) \to P[\langle u\rangle][\gamma] = P[\mathsf{Bool}[]_*\,(\gamma^{\uparrow})][\langle \mathsf{Bool}[]_*\,(u[\gamma])\rangle]
$$

Booleans in a weak CwF (ii)

$$
\begin{aligned}
\alpha\, u : & P[\langle u\rangle][\gamma] && =([\circ]) \\
& P[\langle u\rangle \circ \gamma] && \equiv \\
& P[(\mathsf{id}, [\mathsf{id}]_* \, u) \circ \gamma] && =(,\circ) \\
& P[\mathsf{id} \circ \gamma, [\circ]_* \, (([\mathsf{id}]_* \, u)[\gamma])] && =(-[-] \text{ and transport}) \\
& P[\mathsf{id} \circ \gamma, [\circ]_* \, ([\mathsf{id}]_* \, (u[\gamma]))] && =(\mathsf{idl}) \\
& P[\gamma, \mathsf{idl}_* \, ([\circ]_* \, ([\mathsf{id}]_* \, (u[\gamma])))] && =(\cdot_*) \\
& P[\gamma, ([\mathsf{id}] \cdot [\circ] \cdot \mathsf{idl})_* \, (u[\gamma])] && \equiv \\
& P[\gamma, u[\gamma]] && \equiv \\
& P[\gamma, ([\mathsf{id}] \cdot [\mathsf{id}])_* \, (u[\gamma])] && =(\cdot_*) \\
& P[\gamma, [\mathsf{id}]_* \, ([\mathsf{id}]_* \, (u[\gamma]))] && =(\triangleright\beta_2) \\
& P[\gamma, [\mathsf{id}]_* \, (([\circ] \cdot \triangleright\beta_1)_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] && \equiv \\
& P[\gamma, [\mathsf{id}]_* \, (([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr} \cdot [\mathsf{id}])_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] && =(\cdot_*) \\
& P[\gamma, ([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr} \cdot [\mathsf{id}] \cdot [\mathsf{id}])_* \, (\mathsf{q}[\langle u[\gamma]\rangle])] && \equiv \\
& P[\gamma, ([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr})_* \, (\mathsf{q}[\langle u[\gamma]\rangle])] && =(\cdot_*) \\
& P[\gamma, \mathsf{idr}_* \, (\triangleright\beta_{1*} \, (\mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle])))))] && =(\mathsf{idr}) \\
& P[\gamma \circ \mathsf{id}, \triangleright\beta_{1*} \, (\mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))))] && =(\triangleright\beta_1) \\
& P[\gamma \circ (\mathsf{p} \circ \langle u[\gamma]\rangle), \mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle])))] && =(\mathsf{ass}) \\
& P[(\gamma \circ \mathsf{p}) \circ \langle u[\gamma]\rangle, [\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] && =(-[-] \text{ and transport}) \\
& P[(\gamma \circ \mathsf{p}) \circ \langle u[\gamma]\rangle, [\circ]_* \, (([\circ]_* \, \mathsf{q})[\langle u[\gamma]\rangle])] && =(,\circ) \\
& P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle u[\gamma]\rangle] && \equiv \\
& P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle (\mathsf{Bool}[] \cdot \mathsf{Bool}[])_* \, (u[\gamma])\rangle] && =(\cdot_*) \\
& P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle \mathsf{Bool}[]_* \, (\mathsf{Bool}[]_* \, (u[\gamma]))\rangle] && =(\langle - \rangle \text{ and transport}) \\
& P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \mathsf{Bool}[]_* \, \langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] && =(- \circ - \text{ and transport}) \\
& P[(\mathsf{Bool}[]_* \, (\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q})) \circ \langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] && \equiv
\end{aligned}
$$

Substitution-strict booleans in a strict CwF

$$
\begin{aligned}
&\mathsf{Bool} && : \mathsf{Ty}\,\Gamma \\
&\mathsf{Bool}[] && : \mathsf{Bool}[\gamma] \equiv \mathsf{Bool} \\
&\mathsf{true} && : \mathsf{Tm}\,\Gamma\,\mathsf{Bool} \\
&\mathsf{true}[] && : \mathsf{true}[\gamma] \equiv \mathsf{true} \\
&\mathsf{false} && : \mathsf{Tm}\,\Gamma\,\mathsf{Bool} \\
&\mathsf{false}[] && : \mathsf{false}[\gamma] \equiv \mathsf{false} \\
&\mathsf{ind} && : (P : \mathsf{Ty}\,(\Gamma \triangleright \mathsf{Bool})) \to \mathsf{Tm}\,\Gamma\,(P[\langle \mathsf{true} \rangle]) \to \mathsf{Tm}\,\Gamma\,(P[\langle \mathsf{false} \rangle]) \to \\
& && \ (b : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) \to \mathsf{Tm}\,\Gamma\,(P[\langle b \rangle]) \\
&\mathsf{ind}[] && : (\mathsf{ind}\,P\,p\,p'\,b)[\gamma] \equiv \mathsf{ind}\,(P[\gamma^{\uparrow}])\,(p[\gamma])\,(p'[\gamma])\,(b[\gamma]) \\
&\mathsf{Bool}\beta_1 && : \mathsf{ind}\,P\,p\,p'\,\mathsf{true} = p \\
&\mathsf{Bool}\beta_2 && : \mathsf{ind}\,P\,p\,p'\,\mathsf{false} = p'
\end{aligned}
$$