Mechanizing Synthetic Tait Computability in Istari

PLunch
Nov. 14, 2025

Runming Li
j.w.w. Yue Yao, Robert Harper

Mechanizing *Synthetic Tait Computability* in *Istari*

Mechanizing *Synthetic Tait Computability* in *Istari*

Theorem (Canonicity)

Every closed term of type bool is equal to either true or false.

Idea: equip every type A with a computability predicate.

$$\mathsf{true} \quad \vdots \quad \mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}) \qquad\qquad e : \mathsf{bool} \quad \vdots \quad (e = \mathsf{true}) \vee (e = \mathsf{false})$$

Idea: equip every type A with a computability predicate.

$$\begin{array}{c} \mathsf{true} \\ \mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}) \end{array} \quad \vdots \quad \begin{array}{c} e : \mathsf{bool} \\ (e = \mathsf{true}) \vee (e = \mathsf{false}) \end{array}$$

Idea: equip every type A with a computability predicate.

$$\begin{array}{c} \mathsf{true} \\ \mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}) \end{array} \quad \vdots \quad \begin{array}{c} e : \mathsf{bool} \\ (e = \mathsf{true}) \vee (e = \mathsf{false}) \end{array}$$

Idea: equip every type A with a computability predicate.

$$\begin{array}{c} \mathsf{true} \\ \mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}) \end{array} \quad \vdots \quad \begin{array}{c} e : \mathsf{bool} \\ (e = \mathsf{true}) \vee (e = \mathsf{false}) \end{array}$$

TRUE : BOOL

$$\underset{\mathsf{inl}(\checkmark\,:\,\mathsf{true}\,=\,\mathsf{true})}{\mathsf{true}} \;:\; \underset{(e\,=\,\mathsf{true})\,\vee\,(e\,=\,\mathsf{false})}{e\,:\,\mathsf{bool}}$$

$$\mathsf{IF} : \mathsf{BOOL} \to A \to A \to A$$

$$\begin{array}{c} \mathsf{if}\ b\ t\ f \\ \mathsf{case}\ b\ \{\mathsf{inl}\_ \hookrightarrow t \mid \mathsf{inr}\_ \hookrightarrow f\} \end{array} : A$$

The blue terms work *exactly* like the regular terms.

$$\text{IF TRUE}\ t\ f = t$$

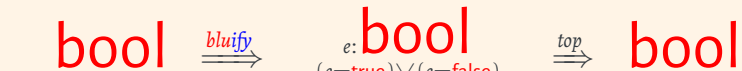
$$\text{if true}\ t\ f \quad \text{case}\ (\text{inl}(\checkmark))\ \{\text{inl}\_ \hookrightarrow t \mid \text{inr}\_ \hookrightarrow f\} = t$$

The blue terms work *exactly* like the regular terms.

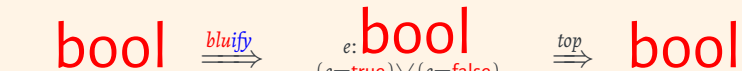
$$\mathsf{IF\ TRUE}\ t\ f = t$$

$$\mathsf{if\ true}\ t\ f \quad \mathsf{case}\ (\mathsf{inl}(\checkmark))\ \{\mathsf{inl\_} \hookrightarrow t \mid \mathsf{inr\_} \hookrightarrow f\} = t$$

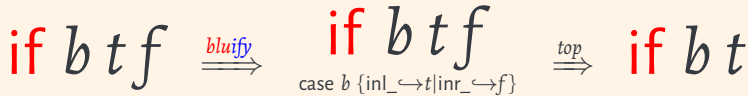
Goal: construct blue counterpart of each red term such that $\cdots$



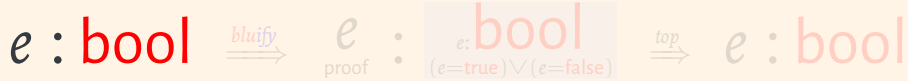
Goal: construct blue counterpart of each red term such that $\cdots$



Goal: construct blue counterpart of each red term such that $\cdots$



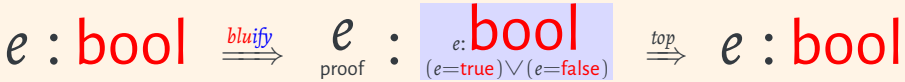
Suppose I have $e : \mathsf{bool}$.

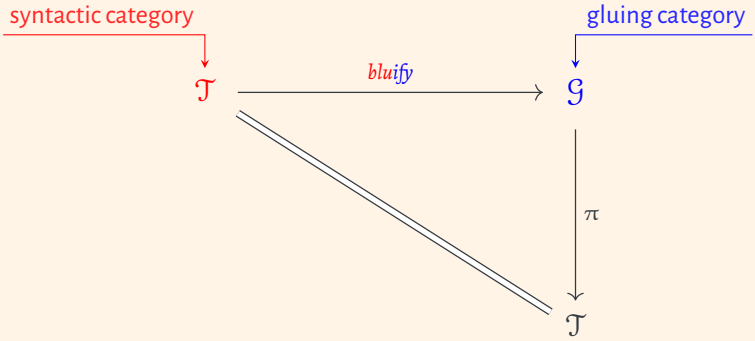


Suppose I have e : bool.

$$e : \mathsf{bool} \xrightarrow{\mathit{bluify}} \underset{\text{proof}}{e} : \underset{(e=\mathsf{true})\vee(e=\mathsf{false})}{{}^{e:}\mathsf{bool}} \xRightarrow{\mathit{top}} e : \mathsf{bool}$$

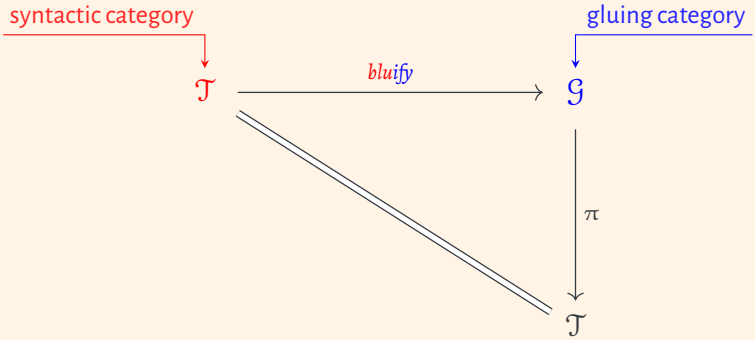
Suppose I have e : bool.





$$\mathit{bluify} \circ \pi = \mathrm{id}_{\mathcal{T}}$$

"fundemental theorem of logical relations"

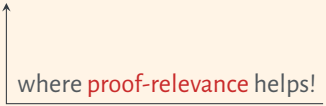


$$\textit{bluify} \circ \pi = \mathrm{id}_{\mathcal{T}}$$

"fundemental theorem of logical relations"

Artin Gluing (Proof-Relevant Logical Relations)

- Scale to dependent types naturally
- Handle universes beautifully



Artin Gluing

need to mechanize them!

- It's a complicated proof with *a lot of* proof obligations
- Some proof obligations are *trivial* but *tedious*

e.g. naturality conditions:
all operations are stable under substitutions

Artin Gluing

need to mechanize them!

- It's a complicated proof with *a lot of* proof obligations
- Some proof obligations are *trivial* but *tedious*

e.g. naturality conditions:
all operations are stable under substitutions

"We leave it to the reader to check that these two operations [$\mathbf{true}_{\mathcal{G}}$ and $\mathbf{false}_{\mathcal{G}}$] are natural in Γ."

"Strictly speaking, one must verify that $\mathbf{Unit}_{\mathcal{G}}$ is stable under substitution but we leave this calculation to the reader."

"We leave the routine calculations that this [$\mathbf{\Pi}_{\mathcal{G}}$] is natural to the reader..."

"We once more leave the naturality conditions for both $\mathbf{U}_{\mathcal{G}}$ and $\mathbf{El}_{\mathcal{G}}$ to the reader."

— *Angiuli & Gratzer, Principles of Dependent Type Theory*

Sterling's observation: modalities can make things synthetic!

Axiom

There is a proposition $\mathrm{syn} : \mathrm{Prop}$ *that controls syntactic v.s. semantic aspects.*

- ○ open modality isolates syntax
- ● closed modality isolates semantics
- ○● $A \cong \mathbf{1}$

Sterling's observation: modalities can make things synthetic!

Axiom

There is a proposition $\mathrm{syn} : \mathrm{Prop}$ *that controls syntactic v.s. semantic aspects.*

- open modality isolates syntax
- closed modality isolates semantics
- $A \cong 1$

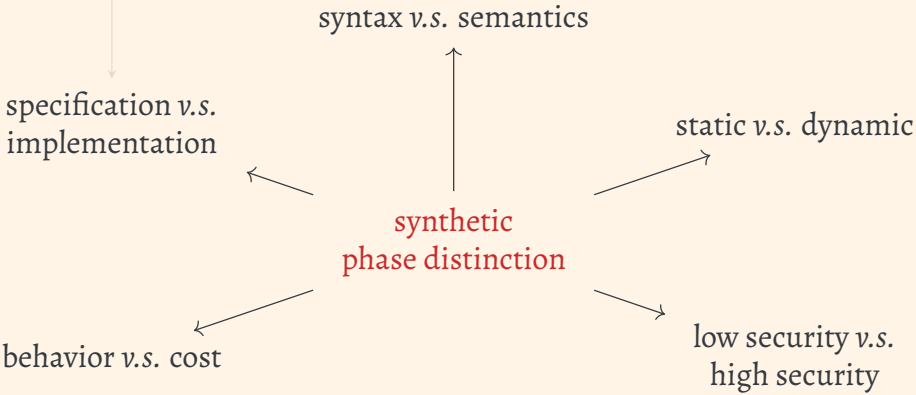
Sterling's observation: modalities can make things synthetic!

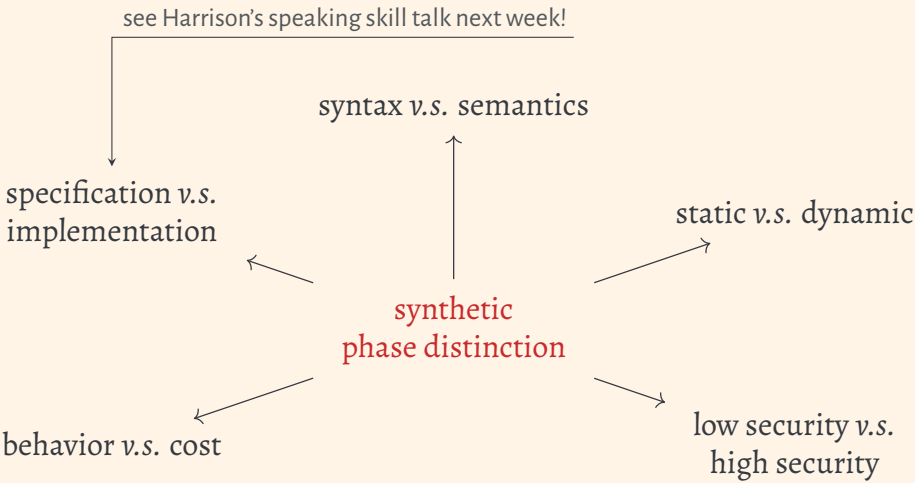
Axiom

There is a proposition $\mathrm{syn} : \mathrm{Prop}$ *that controls syntactic v.s. semantic aspects.*

- ○ open modality isolates syntax
- ● closed modality isolates semantics
- ○●$A \cong \mathbf{1}$

see Harrison's speaking skill talk next week!




see Harrison's speaking skill talk next week!
syntax *v.s.* semantics
specification *v.s.* implementation
static *v.s.* dynamic
synthetic phase distinction
behavior *v.s.* cost
low security *v.s.* high security

Synthetic Tait Computability

Parametricity for an ML module calculus	Sterling & Harper
Normalization for Cartesian Cubical Type Theory	Sterling & Angiuli
Normalization for a multimodal type theory	Gratzer
⋮	
Canonicity for Cost-Aware Logical Framework	Li & Harper

If $e : \mathsf{F}(\mathsf{nat})$, then $e = \mathsf{step}^c(\mathsf{ret}(\mathsf{suc}^n(\mathsf{zero})))$.

A modal dependent type theory with a phase distinction between cost and behavior based on call-by-push-value.

Synthetic Tait Computability

Parametricity for an ML module calculus — Sterling & Harper

Normalization for Cartesian Cubical Type Theory — Sterling & Angiuli

Normalization for a multimodal type theory — Gratzer

⋮

If $e : \mathsf{F}(\mathsf{nat})$, then $e = \mathsf{step}^c(\mathsf{ret}(\mathsf{suc}^n(\mathsf{zero})))$.

Canonicity for Cost-Aware Logical Framework — Li & Harper

A modal dependent type theory with a phase distinction between cost and behavior based on call-by-push-value.

TRUE : BOOL

$$\mathsf{syn} \hookrightarrow \mathsf{true} \;:\; \bigcirc\left(e : \mathsf{bool}\right)$$

$$\eta_{\bullet}(\mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true})) \qquad \bullet\,((e = \mathsf{true}) \vee (e = \mathsf{false}))$$

Observation: $\bigcirc(\mathsf{BOOL} = \mathsf{bool})$ and $\bigcirc(\mathsf{TRUE} = \mathsf{true})$!

TRUE : BOOL

$$\mathsf{syn} \hookrightarrow \mathsf{true} \;:\; \bigcirc\left(e : \mathsf{bool}\right)$$
$$\eta\bullet(\mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true})) \qquad \bullet((e = \mathsf{true}) \vee (e = \mathsf{false}))$$

Observation: $\bigcirc(\mathsf{BOOL} = \mathsf{bool})$ and $\bigcirc(\mathsf{TRUE} = \mathsf{true})$!

TRUE : BOOL

$$\mathsf{syn} \hookrightarrow \underset{\eta_{\bullet}(\mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}))}{\mathsf{true}} : \underset{\bullet((e = \mathsf{true}) \vee (e = \mathsf{false}))}{\bigcirc\left(e : \mathsf{bool}\right)}$$

Observation: $\bigcirc(\mathsf{BOOL} = \mathsf{bool})$ and $\bigcirc(\mathsf{TRUE} = \mathsf{true})$!

TRUE : BOOL

$$\underset{\eta_{\bullet}(\mathsf{inl}(\checkmark : \mathsf{true} = \mathsf{true}))}{\mathsf{syn} \hookrightarrow \mathsf{true}} : \underset{\bullet((e = \mathsf{true}) \vee (e = \mathsf{false}))}{\bigcirc\left(e : \mathsf{bool}\right)}$$

Observation: $\bigcirc(\mathsf{BOOL} = \mathsf{bool})$ and $\bigcirc(\mathsf{TRUE} = \mathsf{true})$!

TRUE : BOOL

$$\underset{\eta_{\bullet}(\mathsf{inl}(\checkmark\,:\,\mathsf{true}=\mathsf{true}))}{\mathsf{syn} \hookrightarrow \mathsf{true}} \;:\; \underset{\bullet\,((e=\mathsf{true})\vee(e=\mathsf{false}))}{\bigcirc\left(e:\mathsf{bool}\right)}$$

Observation: $\bigcirc(\mathsf{BOOL}=\mathsf{bool})$ and $\bigcirc(\mathsf{TRUE}=\mathsf{true})$!

Write $a : \{A \mid \mathrm{syn} \hookrightarrow a_\circ\}$ to mean "$a : A$ with $\bigcirc(a = a_\circ)$".

To prove FTLR, it suffices to construct

$$\begin{aligned}
&\mathsf{BOOL} : \{\mathsf{TP} \mid \mathsf{syn} \hookrightarrow \mathsf{bool}\}\\
&\mathsf{TRUE} : \{\mathsf{BOOL} \mid \mathsf{syn} \hookrightarrow \mathsf{true}\}\\
&\mathsf{FALSE} : \{\mathsf{BOOL} \mid \mathsf{syn} \hookrightarrow \mathsf{false}\}\\
&\mathsf{IF} : \{\mathsf{BOOL} \to A \to A \to A \mid \mathsf{syn} \hookrightarrow \mathsf{if}\}\\
&\vdots
\end{aligned}$$

$\mathsf{TP} : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tp}\}$

$\mathsf{TP} = (A : \mathsf{tp}) \ltimes \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tm}\, A\}$

$\mathsf{TM} : \{\mathsf{TP} \to \mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tm}\}$

$\mathsf{TM}\, A = \pi_2 A$

$\mathsf{BOOL} : \{\mathsf{TP} \mid \mathsf{syn} \hookrightarrow \mathsf{bool}\}$

$\mathsf{BOOL} = [\mathsf{syn} \hookrightarrow \mathsf{bool}, (b : \mathsf{tm}(\mathsf{bool})) \ltimes \bullet((b = \mathsf{true}) + (b = \mathsf{false}))]$

$\mathsf{TRUE} : \{\mathsf{TM}(\mathsf{BOOL}) \mid \mathsf{syn} \hookrightarrow \mathsf{true}\}$

$\mathsf{TRUE} = [\mathsf{syn} \hookrightarrow \mathsf{true}, \eta_{\bullet}(\mathsf{inl}(\checkmark))]$

$$\mathsf{TP} : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tp}\}$$
$$\mathsf{TP} = (A : \mathsf{tp}) \ltimes \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tm}\, A\}$$

$$\mathsf{TM} : \{\mathsf{TP} \to \mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{tm}\}$$
$$\mathsf{TM}\, A = \pi_2 A$$

$$\mathsf{BOOL} : \{\mathsf{TP} \mid \mathsf{syn} \hookrightarrow \mathsf{bool}\}$$
$$\mathsf{BOOL} = [\mathsf{syn} \hookrightarrow \mathsf{bool}, (b : \mathsf{tm}(\mathsf{bool})) \ltimes \bullet((b = \mathsf{true}) + (b = \mathsf{false}))]$$

$$\mathsf{TRUE} : \{\mathsf{TM}(\mathsf{BOOL}) \mid \mathsf{syn} \hookrightarrow \mathsf{true}\}$$
$$\mathsf{TRUE} = [\mathsf{syn} \hookrightarrow \mathsf{true}, \eta_\bullet(\mathsf{inl}(\checkmark))]$$

$$\mathsf{PI} : \{(A : \mathsf{TP}) \to (\mathsf{TM}\, A \to \mathsf{TP}) \to \mathsf{TP} \mid \mathsf{syn} \hookrightarrow \mathsf{pi}\}$$
$$\mathsf{PI}\, A\, B = [\mathsf{syn} \hookrightarrow \mathsf{pi}\, A\, B, (e : \mathsf{tm}(\mathsf{pi}\, A\, B)) \ltimes \{(a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\, e\}]$$

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

$$\mathsf{APP} : \{\mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\}$$
$$\mathsf{APP}\, e\, a = (\pi_2\, e)\, a$$

$$\mathsf{PI}_\beta : \{\mathsf{APP}\, (\mathsf{LAM}\, f)\, a = f\, a \mid \mathsf{syn} \hookrightarrow \mathsf{pi}_\beta\}$$
$$\mathsf{PI}_\beta = \checkmark$$

$$\mathsf{PI} : \{(A : \mathsf{TP}) \to (\mathsf{TM}\, A \to \mathsf{TP}) \to \mathsf{TP} \mid \mathsf{syn} \hookrightarrow \mathsf{pi}\}$$
$$\mathsf{PI}\, A\, B = [\mathsf{syn} \hookrightarrow \mathsf{pi}\, A\, B, (e : \mathsf{tm}(\mathsf{pi}\, A\, B)) \ltimes \{(a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\, e\}]$$

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

$$\mathsf{APP} : \{\mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\}$$
$$\mathsf{APP}\, e\, a = (\pi_2\, e)\, a$$

$$\mathsf{PI}_\beta : \{\mathsf{APP}\, (\mathsf{LAM}\, f)\, a = f\, a \mid \mathsf{syn} \hookrightarrow \mathsf{pi}_\beta\}$$
$$\mathsf{PI}_\beta = \checkmark$$

Proof is short, but can be deceiving!

The proof is short because . . .

Hidden transport and equality reasoning

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

need to cite $\mathsf{pi}_\beta : \mathsf{app}\,(\mathsf{lam}\, f)\, x = f\, x$ to make it type-check

need to "transport" along
$\bigcirc((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) = (a : \mathsf{tm}\, A) \to \mathsf{tm}(B\, a))$
which itself needs to "transport" along $\bigcirc((A : \mathsf{TP}) = (A : \mathsf{tp}))$

The proof is short because . . .

Hidden transport and equality reasoning

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

need to cite $\mathsf{pi}_\beta : \mathsf{app}\,(\mathsf{lam}\, f)\, x = f\, x$ to make it type-check

need to "transport" along
$\bigcirc((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) = (a : \mathsf{tm}\, A) \to \mathsf{tm}(B\, a))$
which itself needs to "transport" along $\bigcirc((A : \mathsf{TP}) = (A : \mathsf{tp}))$

coe : $\{A, B : \mathcal{U}\} \to A = B \to A \to B$

transport : $\{A : \mathcal{U}\}\{P : A \to \mathcal{U}\} \to a = b \to P(a) \to P(b)$

```
lam· = λ {Γ Γ· A} A· {B B· t} t· γ γ· a a· →
       let
         A[γ] = _[_]T {Γ} A {◇} γ
         A[γ]· = _[_]T· {Γ}{Γ·}{A} A· {◇}{◇·}{γ} (λ _ _ → γ·)
         γ↑ = Ct.↑ {Γ}{A}{◇} γ
         γ↑· = dCt.↑· {Γ}{Γ·}{A}{A·}{◇}{◇·}{γ} (λ _ _ → γ·)
         B[γ↑] = _[_]T {Γ ▹ A} B {◇ ▹ A[γ]} γ↑
         B[γ↑]· = _[_]T· {Γ ▹ A}{_▹·_ {Γ} Γ· {A} A·}{B} B· {◇ ▹ A[γ]}{_▹·_ {◇} ◇· {A[γ]} A[γ]·}{γ↑} γ↑·
         sg-a = Ct.sg {◇}{A[γ]} a
         sg-a· = dCt.sg· {◇}{◇·}{A[γ]}{A[γ]·}{a} (λ _ _ → a·)
         γ↑∘sg-a = _∘_ {Γ ▹ A}{◇ ▹ A[γ]} γ↑ {◇} sg-a
         γ↑∘sg-a· = _∘·_ {Γ ▹ A}{_▹·_ {Γ} Γ· {A} A·}{◇ ▹ A[γ]}{_▹·_ {◇} ◇· {A[γ]} A[γ]·}{γ↑} γ↑· {◇}{◇·}{sg-
         t[γ↑] = _[_]t {Γ ▹ A}{B} t {◇ ▹ A[γ]} γ↑
       in coe (cong (B· γ↑∘sg-a γ↑∘sg-a·) (Πβ {◇}{A[γ]}{B[γ↑]}{t[γ↑]}{a} ⁻¹)) (t· γ↑∘sg-a γ↑∘sg-a·)
```

— Kaposi & Pujet, Type Theory in Type Theory using a Strictified Syntax

“Transport Hell”

```
110   id^ : ∀{Γ A} →
111     id {Γ} ^ ≡ transp (λ t → Sub t (Γ ▹ A)) (cong (_▹_ Γ) (sym [id])) id {Γ ▹ A}
112   id^ {Γ} {A} =
113     cong
114       (λ x → π₁ x , π₂ x)
115       (idl p ,= transptransp
116         (Tm (Γ ▹ (A [ id ])))
117         (sym ([∘] A p id)) {cong (A [_]) (idl p)}) ▪
118     transp, [id] ▪
119     cong (transp (λ z → Sub (Γ ▹ (A [ id ])) (Γ ▹ z)) [id]) consid ▪
120     coe2l
121       (sym (cong (λ t → Sub t (Γ ▹ A)) (cong (_▹_ Γ) (sym [id]))))
122       (coecoe
123         (cong (λ z → Sub (Γ ▹ (A [ id ])) (Γ ▹ z)) [id])
124         (sym (cong (λ t → Sub t (Γ ▹ A)) (cong (_▹_ Γ) (sym [id])))) ▪
125     trid [id])
```

Listing 5: `cwf.agda`: weakening of substitution.

— Yee-Jian Tan, Formalizing Dependent Types in Agda

The proof is short because . . .

Implicit coercions

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{APP} : \{\mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\}$$

$$\mathsf{PI}_\beta : \mathsf{APP}\, (\mathsf{LAM}\, f)\, a = f\, a$$

not immediately well-typed!
Unless $\mathsf{LAM} : ((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B)$
and $\mathsf{APP} : \mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)$

The proof is short because . . .

Implicit coercions

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{APP} : \{\mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\}$$

$$\mathsf{PI}_\beta : \mathsf{APP}\ (\mathsf{LAM}\, f)\ a = f\ a$$

not immediately well-typed!
Unless $\mathsf{LAM} : ((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B)$
and $\mathsf{APP} : \mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)$

The proof is short because . . .

Implicit coercions

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{APP} : \{\mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a) \mid \mathsf{syn} \hookrightarrow \mathsf{app}\}$$

$$\mathsf{PI}_\beta : \mathsf{APP}\ (\mathsf{LAM}\, f)\ a = f\ a$$

not immediately well-typed!
Unless $\mathsf{LAM} : ((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B)$
and $\mathsf{APP} : \mathsf{TM}(\mathsf{PI}\, A\, B) \to (a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)$

The proof is short because . . .

- Invisiable function extensionality
- Culmulative universes
- . . .

None of your favorite proof assistants support all of these features!
Except for one . . .

Istari

https://istarilogic.org

A NuPRL-style proof assistant based on computational semantics developed by Karl Crary.

PER model over operational semantics, à la logical relations.

Allows: ill-typed terms, terms with multiple types, untyped reduction, etc[1].

[1] A great tutorial on computational semantics is Angiuli's thesis.

Istari

- Typing judgment is internal to the language.

  $M : A$ is a proposition to be proved by users.

- If $M : A(N)$ and $N = N'$, then $M : A(N')$.

  Propositional equality! Equality reflection in action.

- Type-checking is undecidable!

  If typechecker is stuck, user can manually take over!

Istari

- Typing judgment is internal to the language.

  $M : A$ is a proposition to be proved by users.

- If $M : A(N)$ and $N = N'$, then $M : A(N')$.

  Propositional equality! Equality reflection in action.

- Type-checking is undecidable!

  If typechecker is stuck, user can manually take over!

Istari

- Typing judgment is internal to the language.

  $M : A$ is a proposition to be proved by users.

- If $M : A(N)$ and $N = N'$, then $M : A(N')$.

  Propositional equality! Equality reflection in action.

- Type-checking is undecidable!

  If typechecker is stuck, user can manually take over!

Istari

```
define/id_vec {m n}/
/
  fn v . v
//
  forall(m n : nat). vec(m + n) → vec(n + m)
/;
  unfold /id_vec/.
  introOf /m n v/.
  typecheck.
  apply /Nat.plus_commute/.
qed();
```

a proof that $m + n = n + m$!

Istari

define/*id_vec* {*m n*}/
/
  fn *v* . *v*
//
  forall(*m n* : nat). vec($m + n$) → vec($n + m$)
/;
  unfold /*id_vec*/.
  introOf /*m n v*/.
  typecheck.
  apply /Nat.*plus_commute*/.
qed();

a proof that $m + n = n + m$!

Istari

define/*id_vec* {m n}/
/
  fn v . v
//
  forall(m n : nat). vec($m + n$) $\to$ vec($n + m$)
/;
  unfold /*id_vec*/.
  introOf /m n v/.
  typecheck.
  apply /Nat.*plus_commute*/.
qed();

a proof that $m + n = n + m$!

A library of phase distinction in Istari

- Phase
- Modalities
- Strict glue types
- Extension types

Thanks to Istari's subset type $\{x : A \mid P(x)\}$,
which handles the coercion implicitly: if $a : \{x : A \mid P(x)\}$, then $a : A$.

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue (fn z .lam f) f
//
Ext ((forall(a : TM A).TM(B a)) → TM(PI A B)) (fn z .lam)
/;
  ...
qed();
```

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$
$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue (fn z .lam f) f
//
Ext ((forall(a : TM A).TM(B a)) → TM(PI A B)) (fn z .lam)
/;
  ...
qed();
```

$$\mathsf{LAM} : \{\ ((a : \mathsf{TM}\,A) \to \mathsf{TM}(B\,a)) \to \mathsf{TM}(\mathsf{PI}\,A\,B)\ \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{LAM}\,f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\,f, f]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue (fn z .lam f) f
//
Ext ( (forall(a : TM A).TM(B a)) → TM(PI A B) ) (fn z .lam)
/;
  ...
qed();
```

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue (fn z .lam f) f
//
Ext ((forall(a : TM A).TM(B a)) → TM(PI A B)) ( fn z .lam )
/;
  ...
qed();
```

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{LAM}\, f = [\, \mathsf{syn} \hookrightarrow \mathsf{lam}\, f\, , f]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue ( fn z .lam f ) f
//
Ext ((forall(a : TM A).TM(B a)) → TM(PI A B)) (fn z .lam)
/;
  ...
qed();
```

$$\mathsf{LAM} : \{((a : \mathsf{TM}\, A) \to \mathsf{TM}(B\, a)) \to \mathsf{TM}(\mathsf{PI}\, A\, B) \mid \mathsf{syn} \hookrightarrow \mathsf{lam}\}$$

$$\mathsf{LAM}\, f = [\mathsf{syn} \hookrightarrow \mathsf{lam}\, f, f\,]$$

Istari

```
define/LAM {A B}/
/
  fn f .glue (fn z .lam f) f
//
Ext ((forall(a : TM A).TM(B a)) → TM(PI A B)) (fn z .lam)
/;
  ...
qed();
```

Our mechanized proof is almost verbatim from the on-paper one!

Mechanizing *Synthetic Tait Computability* in *Istari*

synthetic *v.s.* analytic mathematics

A "synthetic" approach to mathematical theories is one which prioritizes the axiomatic description of structures, as opposed to more traditional "analytic" approaches in which these structures are built out of primitive mathematical objects.

— Corinthia

Analytic Cartesian Geometry	**Synthetic Euclidean Geometry**
Points are pairs of real numbers	A line can be drawn between any two points
Lines are sets of points satisfying linear equations	A line segment can be extended to a straight line
Circles are sets of points satisfying certain quadratic equations	Given a point and a distance, a circle can be drawn
⋮	⋮

Analytic Cartesian Geometry	Synthetic Euclidean Geometry
Points are pairs of real numbers	A line can be drawn between any two points
Lines are sets of points satisfying linear equations	A line segment can be extended to a straight line
Circles are sets of points satisfying certain quadratic equations	Given a point and a distance, a circle can be drawn
⋮	⋮

Homotopy Type Theory (HoTT) is synthetic homotopy theory!

Formalize results from homotopy theory in type theory!
(*e.g.* in redtt or Cubical Agda)

Homotopy Type Theory (HoTT) is synthetic homotopy theory!

Formalize results from homotopy theory in type theory!
(*e.g.* in redtt or Cubical Agda)

What is so synthetic about Synthetic Tait Computability?

Axiom

There is a proposition $\mathsf{syn} : \mathsf{Prop}$ *that controls syntactic v.s. semantic aspects.*

This axiom isolates the FTLR condition!

What is so synthetic about Synthetic Tait Computability?

Axiom

There is a proposition $\mathsf{syn} : \mathsf{Prop}$ *that controls syntactic v.s. semantic aspects.*

This axiom isolates the FTLR condition!

What is so synthetic about Synthetic Tait Computability?

Axiom

There is a proposition $\mathsf{syn} : \mathsf{Prop}$ *that controls syntactic v.s. semantic aspects.*

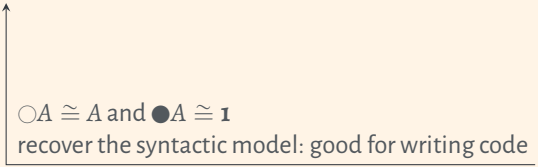
This axiom isolates the FTLR condition!

Analytically there are three ways to justify this axiom:

syn is interpreted as $\top$

Analytically there are three ways to justify this axiom:

syn is interpreted as $\top$



Analytically there are three ways to justify this axiom:

syn is interpreted as $\bot$

$\bigcirc A \cong \mathbf{1}$ and $\bullet A \cong A$
recover the standard model in sets

$\mathsf{TP} = \mathbf{Type}$
$\mathsf{TM}(A) = A$

$\mathsf{BOOL} = \mathbf{1} + \mathbf{1}$
$\mathsf{TRUE} = \mathsf{inl}\ \star$
$\mathsf{FALSE} = \mathsf{inr}\ \star$

$\mathsf{PI}\ A\ B = (a : A) \to B(a)$
$\mathsf{LAM}\ f = f$
$\mathsf{APP}\ f\ x = f\ x$

Analytically there are three ways to justify this axiom:

syn is interpreted as $\bot$

$\bigcirc A \cong \mathbf{1}$ and $\bullet A \cong A$
recover the standard model in sets

$\mathsf{TP} = \mathbf{Type}$
$\mathsf{TM}(A) = A$

$\mathsf{BOOL} = \mathbf{1} + \mathbf{1}$
$\mathsf{TRUE} = \mathsf{inl}\ \star$
$\mathsf{FALSE} = \mathsf{inr}\ \star$

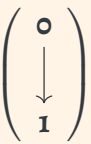
$\mathsf{PI}\ A\ B = (a : A) \to B(a)$
$\mathsf{LAM}\ f = f$
$\mathsf{APP}\ f\ x = f\ x$

Analytically there are three ways to justify this axiom:

syn is interpreted as

recover the logical relations model: good for proving meta-theoretic properties

Analytically there are three ways to justify this axiom:

syn is interpreted as 

recover the logical relations model: good for proving meta-theoretic properties

What is so synthetic about Synthetic Tait Computability?

Axiom

Syntactically, there exists syntax tp, tm, bool, true, false, if, pi, lam, app, · · ·

Second-order syntax remove the burden of substitution and naturality!

in fact, out mechanization does not have any substitution proofs at all!

What is so synthetic about Synthetic Tait Computability?

Axiom

Syntactically, there exists syntax tp, tm, bool, true, false, if, pi, lam, app, · · ·

Second-order syntax remove the burden of substitution and naturality!

in fact, out mechanization does not have any substitution proofs at all!

What is so synthetic about Synthetic Tait Computability?

Axiom

Syntactically, there exists syntax tp, tm, bool, true, false, if, pi, lam, app, · · ·

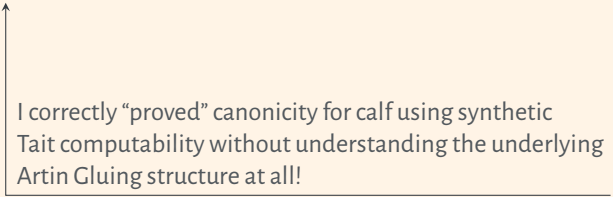
Second-order syntax remove the burden of substitution and naturality!

in fact, out mechanization does not have any substitution proofs at all!

Synthetic methods isolate the core ideas that need attentions.

I correctly "proved" canonicity for calf using synthetic Tait computability without understanding the underlying Artin Gluing structure at all!

Synthetic methods isolate the core ideas that need attentions.



[W]hereas in the past the synthetic and analytic viewpoints were opposed to each other, today they must be viewed as two parts of a whole whose interplay leads to new and useful developments in mathematics.

— Jon Sterling, Naïve logical relations in synthetic Tait computability