First Steps in Synthetic Tait Computability

The Objective Metatheory of Cubical Type Theory

Jonathan Sterling

Carnegie Mellon University

September 13, 2021

To my mother, LeeAnn.

Heartfelt thanks to Bob for making this whole experience possible!

Dependent type theory is...

Dependent type theory is...

a language for math

Dependent type theory is...

a language for homotopical math

Dependent type theory is...

a programming language

Dependent type theory is...

a programming language + program logic

Dependent type theory is...

a metalanguage for PL syntax

Dependent type theory is...

a metalanguage for PL semantics

Requirements of type theoretic tools

Requirements of type theoretic tools

Semantic properties

Requirements of type theoretic tools

(all models)

Semantic properties

Requirements of type theoretic tools

(all models)

Semantic properties

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

(just the syntax)

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality

(just the syntax)

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients

(just the syntax)

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)

(just the syntax)

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- consistency

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- consistency
- closed term computation

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- consistency
- closed term computation
- decidable type checking

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- **consistency**
- **closed term computation**
- **decidable type checking**

Swedish philosophy

Requirements of type theoretic tools

(all models)

Semantic properties

- function extensionality
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- consistency
- **closed term computation**
- **decidable type checking**

programming

Requirements of type theoretic tools

(all models)

Semantic properties

- **function extensionality**
- effective quotients
- unique choice ($\forall\exists! \Rightarrow \exists\forall$)
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- **consistency**
- closed term computation
- **decidable type checking**

program logic

Requirements of type theoretic tools

(all models)

Semantic properties

- **function extensionality**
- **effective quotients**
- **unique choice ($\forall\exists! \Rightarrow \exists\forall$)**
- type extensionality (univalence)?

(just the syntax)

Syntactic properties

- **consistency**
- closed term computation
- **decidable type checking**

general mathematics

Requirements of type theoretic tools

(all models)

Semantic properties

- **function extensionality**
- **effective quotients**
- **unique choice ($\forall\exists! \Rightarrow \exists\forall$)**
- **type extensionality (univalence)?**

(just the syntax)

Syntactic properties

- **consistency**
- closed term computation
- **decidable type checking**

homotopical mathematics

How are conventional implementations doing?

Conventional designs cannot satisfy all requirements.

How are conventional implementations doing?

Conventional designs cannot satisfy all requirements.

- **Nuprl:** effective quotients incompatible with PER semantics; equality reflection defeats type checking

How are conventional implementations doing?

Conventional designs cannot satisfy all requirements.

- **Nuprl:** effective quotients incompatible with PER semantics; equality reflection defeats type checking
- **Coq, Lean:** sequestered **Prop** universe $\Rightarrow$ trade-off between computation & unique choice

How are conventional implementations doing?

Conventional designs cannot satisfy all requirements.

- **Nuprl:** effective quotients incompatible with PER semantics; equality reflection defeats type checking
- **Coq, Lean:** sequestered **Prop** universe $\Rightarrow$ trade-off between computation & unique choice
- **Agda, Coq, Lean:** Martin-Löf identity type does not derive funext

How are conventional implementations doing?

Conventional designs cannot satisfy all requirements.

- **Nuprl:** effective quotients incompatible with PER semantics; equality reflection defeats type checking
- **Coq, Lean:** sequestered **Prop** universe $\Rightarrow$ trade-off between computation & unique choice
- **Agda, Coq, Lean:** Martin-Löf identity type does not derive funext
- **Nuprl, Agda, Coq, Lean:** 1-dimensional equality incompatible with univalence

What we've been working on

Our aim has been to achieve all goals at once; HoTT achieves the semantic goals, but it is not a PL. Cubical type theory[1] designed to reconcile all these constraints.

[1] Bezem, Coquand, and Huber (2014), Angiuli, Hou (Favonia), and Harper (2017), Cohen, Coquand, Huber, and Mörtberg (2017), Awodey (2018), and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2021)

What we've been working on

Our aim has been to achieve all goals at once; HoTT achieves the semantic goals, but it is not a PL. Cubical type theory[1] designed to reconcile all these constraints.

Success? Both **`redtt`** [S., Favonia] and Cubical Agda($*$) were conjectured to meet all requirements modulo implementation bugs and features known to be inconsistent.

[1] Bezem, Coquand, and Huber (2014), Angiuli, Hou (Favonia), and Harper (2017), Cohen, Coquand, Huber, and Mörtberg (2017), Awodey (2018), and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2021)

What we've been working on

Our aim has been to achieve all goals at once; HoTT achieves the semantic goals, but it is not a PL. Cubical type theory[1] designed to reconcile all these constraints.

Success? Both **red**`tt` [S., Favonia] and Cubical Agda($*$) were conjectured to meet all requirements modulo implementation bugs and features known to be inconsistent.

This dissertation proves that the type theories underlying both **red**`tt` and Cubical Agda have **decidable type checking**.

[1] Bezem, Coquand, and Huber (2014), Angiuli, Hou (Favonia), and Harper (2017), Cohen, Coquand, Huber, and Mörtberg (2017), Awodey (2018), and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2021)

What we've been working on

Our aim has been to achieve all goals at once; HoTT achieves the semantic goals, but it is not a PL. Cubical type theory[1] designed to reconcile all these constraints.

Success? Both `redtt` [S., Favonia] and Cubical Agda($*$) were conjectured to meet all requirements modulo implementation bugs and features known to be inconsistent.

This dissertation proves that the type theories underlying both `redtt` and Cubical Agda have **decidable type checking**. The main ingredient is a new technique called ***synthetic Tait computability*** (STC) abstracting Artin gluing and logical relations.

[1] Bezem, Coquand, and Huber (2014), Angiuli, Hou (Favonia), and Harper (2017), Cohen, Coquand, Huber, and Mörtberg (2017), Awodey (2018), and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2021)

1. Cubical type theory

What is cubical type theory / □TT?

□TT is an extension of Martin-Löf's Type Theory by an interval:

- a new sort $\Gamma \vdash \mathbb{I}$ and context extension $\Gamma, i : \mathbb{I}$
- with endpoints $\Gamma \vdash 0, 1 : \mathbb{I}$

Why? A new way to think about equality (paths) as *figures* of shape $\mathbb{I}$.

$$(a_0 =_A a_1) :\equiv \{p : \mathbb{I} \to A \mid p(0) \equiv a_0 \wedge p(1) \equiv a_1\}$$

Supports function extensionality, type extensionality (univalence), and effective quotients like Homotopy Type Theory/HoTT,[2] but has stronger syntactic/computational properties.

[2] Univalent Foundations Program (2013)

Computation in □TT: prior art

The state of the art (Huber, 2018; Angiuli, Hou (Favonia), and Harper, 2018):

Theorem (Cubical canonicity)

If $\vec{\imath} : \mathbb{I}^n \vdash M(\vec{\imath}) : \mathsf{bool}$ *is a closed n-cube of booleans, then either* $\vec{\imath} : \mathbb{I}^n \vdash M(\vec{\imath}) \equiv \mathsf{tt} : \mathsf{bool}$ *or* $\vec{\imath} : \mathbb{I}^n \vdash M(\vec{\imath}) \equiv \mathsf{ff} : \mathsf{bool}$.

Hence **□TT** is programming language.

Cubical canonicity is only about computation of closed *n*-cubes.
But **implementation** (type checking, elaboration) requires computation in *arbitrary* contexts Γ, *i.e.* normalization.

Results of this dissertation

I have proved the following suite of results for **$\square$TT** with a countable cumulative hierarchy of universes:[3]

Theorem (Normalization)

*There is a computable function assigning to every type $\Gamma \vdash A$ and every term $\Gamma \vdash a : A$ of **$\square$TT** a unique normal form.*

Corollary (Decidability of equality)

*Judgmental equality $\Gamma \vdash A \equiv B$ and $\Gamma \vdash a \equiv b : A$ in **$\square$TT** is decidable.*

Corollary (Injectivity of type constructors)

If $\Gamma \vdash \Pi(A, B) \equiv \Pi(A', B')$ then $\Gamma \vdash A \equiv A'$ and $\Gamma, x : A \vdash B(x) \equiv B'(x)$.

[3] The preliminary result for **$\square$TT** without universes is j.w.w. Angiuli published in LICS'21 (Sterling and Angiuli, 2021).

2. Synthetic Tait computability

Proving metatheorems using Tait's method

In 1967, Tait introduced his *method of computability*[4]; Tait computability has remained our only scalable tool for proving metatheorems for logics and type theory (canonicity, normalization, parametricity, conservativity, *etc.*).[5]

[4] a.k.a. logical relations/predicates

[5] Gentzen's cut elimination an elegant alternative in some cases, but rarely scales beyond toy examples.

Proving metatheorems using Tait's method

In 1967, Tait introduced his *method of computability*[4]; Tait computability has remained our only scalable tool for proving metatheorems for logics and type theory (canonicity, normalization, parametricity, conservativity, *etc.*).[5]

Idea: an "interpretation" that equips each type A with an predicate $[\![A]\!]$ on elements of A; then show that all *terms* preserve the predicates.

1. First choose the predicate at base type to make soundness of the interpretation imply the desired metatheorem.
2. Then "draw the rest of the owl".

[4] a.k.a. logical relations/predicates

[5] Gentzen's cut elimination an elegant alternative in some cases, but rarely scales beyond toy examples.

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Q1: given a type A, what is the *domain* of $[\![A]\!]$? closed terms, open terms, typed, ??

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Q1: given a type A, what is the *domain* of $[\![A]\!]$? closed terms, open terms, typed, ??
Q2: what properties must $[\![A]\!]$ satisfy? closure under subst., ren., head expansion, ??

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Q1: given a type A, what is the *domain* of $[\![A]\!]$? closed terms, open terms, typed, ??
Q2: what properties must $[\![A]\!]$ satisfy? closure under subst., ren., head expansion, ??
Q3: does our proof actually depend on the chosen transition relation $\mapsto^*$?

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Q1: given a type A, what is the *domain* of $[\![A]\!]$? closed terms, open terms, typed, ??
Q2: what properties must $[\![A]\!]$ satisfy? closure under subst., ren., head expansion, ??
Q3: does our proof actually depend on the chosen transition relation $\mapsto^*$?
Q4: why are the predicates attached to connectives $(\to, \times, \ldots)$ the way they are?

Operational Tait computability

First define operational semantics $\mapsto^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{aligned} [\![\mathsf{bool}]\!](b) &:= (b \mapsto^* \mathsf{tt} \vee b \mapsto^* \mathsf{ff}) \\ [\![A \to B]\!](f) &:= (\forall x : A. [\![A]\!](x) \to [\![B]\!](f(x))) \end{aligned}$$

Q1: given a type A, what is the *domain* of $[\![A]\!]$? closed terms, open terms, typed, ??
Q2: what properties must $[\![A]\!]$ satisfy? closure under subst., ren., head expansion, ??
Q3: does our proof actually depend on the chosen transition relation $\mapsto^*$?
Q4: why are the predicates attached to connectives $(\to, \times, \ldots)$ the way they are?

(None of the above have satisfactory answers in operational Tait computability.)

The outer limits of operational Tait computability

Specifying and verifying the domain and closure conditions of computability predicates for *cubical* ***canonicity*** proved nearly intractable, *pace* Huber (2018) and Angiuli, Hou (Favonia), and Harper (2018).

Motivated S., Angiuli, and Gratzer to pursue an *algebraic*/gluing-based version of Tait computability for **□TT**[6] à la Coquand (2018), as suggested by Awodey.

Idea: work only with *quotiented* typed terms, make computability predicates proof-relevant. **Outcome:** all difficulties disappeared for cubical canonicity, normalization still required fundamentally new ideas (this dissertation).

Synthetic Tait computability = type theoretic abstraction of the algebraic gluing argument à la Orton and Pitts (2016).

[6]Sterling, Angiuli, and Gratzer (2019)

Introducing synthetic Tait computability

What is ***synthetic*** *about synthetic Tait computability?*

Introducing synthetic Tait computability

*What is **synthetic** about synthetic Tait computability?*

Analytic methods explain domain objects in terms of their encoding as something totally different. **Synthetic** methods explain domain objects in terms of their relation to each other.

Introducing synthetic Tait computability

*What is **synthetic** about synthetic Tait computability?*

Analytic methods explain domain objects in terms of their encoding as something totally different. **Synthetic** methods explain domain objects in terms of their relation to each other.

	analytic	**synthetic**
geometry	the Cartesian plane, $\mathbb{R}^n$	Euclid's postulates
metatheory	logical relations, Artin gluing	STC

Introducing synthetic Tait computability

*What is **synthetic** about synthetic Tait computability?*

Analytic methods explain domain objects in terms of their encoding as something totally different. **Synthetic** methods explain domain objects in terms of their relation to each other.

	analytic	**synthetic**
geometry	the Cartesian plane, $\mathbb{R}^n$	Euclid's postulates
metatheory	logical relations, Artin gluing	STC

STC abstracts logical relations by isolating the relationship between syntax and semantics as a pair of modalities.[7]

Expressive enough to recover and simplify existing LR arguments. **More importantly**, STC gave me new geometrical intuitions that I used to solve cubical normalization.

[7](For experts: STC is the internal language of topoi equipped with open/closed partitions.)

Mixing syntax and semantics

What is really going on in Tait computability? We are *immersing* syntax in a more powerful language (the language of computability predicates) that can express the semantic invariants we want.

(Smoother to develop and use if we generalize to **computability *structures***, *i.e.* **proof-relevant** computability predicates.[8])

[8] *cf.* logical relations for universes and strong sums

Mixing syntax and semantics

What is really going on in Tait computability? We are *immersing* syntax in a more powerful language (the language of computability predicates) that can express the semantic invariants we want.

(Smoother to develop and use if we generalize to **computability *structures***, *i.e.* **proof-relevant** computability predicates.[8])

e.g. the computability structure of the booleans:

$$[\![\mathsf{bool}]\!] := \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

[8] *cf.* logical relations for universes and strong sums

Piecing together syntax and semantics

Computability structures built from syntax and semantics.

Piecing together syntax and semantics

Computability structures built from $\boxed{\text{syntax}}$ and $\boxed{\text{semantics}}$. These can be mixed and matched, but the satisfy some laws:

- Both $\boxed{-}$ and $\boxed{-}$ are *lex idempotent monads.*[9]

[9] They are open and closed modalities in the sense of topos theory (Artin, Grothendieck, and Verdier, 1972; Mac Lane and Moerdijk, 1992; Rijke, Shulman, and Spitters, 2020).

Piecing together syntax and semantics

Computability structures built from $\boxed{\text{syntax}}$ and $\boxed{\text{semantics}}$. These can be mixed and matched, but the satisfy some laws:

- Both $\boxed{-}$ and $\boxed{-}$ are *lex idempotent monads.*[9]
- **Complementarity:** semantic things are syntactically trivial, *i.e.* $\boxed{\boxed{A}} \cong$ unit but not the other way around.

[9] They are open and closed modalities in the sense of topos theory (Artin, Grothendieck, and Verdier, 1972; Mac Lane and Moerdijk, 1992; Rijke, Shulman, and Spitters, 2020).

Piecing together syntax and semantics

Computability structures built from syntax and semantics. These can be mixed and matched, but the satisfy some laws:

- Both $\colorbox{blue}{$-$}$ and $\colorbox{magenta}{$-$}$ are *lex idempotent monads.*[9]
- **Complementarity:** semantic things are syntactically trivial, *i.e.* $\colorbox{blue}{$\colorbox{magenta}{A}$} \cong$ unit but not the other way around.
- **Fracture:** any computability structure A can be reconstructed from $\colorbox{blue}{$A$}$, $\colorbox{magenta}{$A$}$, and $\colorbox{magenta}{$\colorbox{blue}{A}$}$.

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \colorbox{blue}{A} \arrow[d] \\
\colorbox{magenta}{A} \arrow[r] & \colorbox{magenta}{$\colorbox{blue}{$A$}$}
\end{tikzcd}$$

[9] They are open and closed modalities in the sense of topos theory (Artin, Grothendieck, and Verdier, 1972; Mac Lane and Moerdijk, 1992; Rijke, Shulman, and Spitters, 2020).

The language of synthetic Tait computability

Definition

STC = type theory + modalities $\boxed{-}$/$\boxed{-}$ that behave as above.

The language of synthetic Tait computability

Definition

STC = type theory + modalities $\boxed{-}$/$\boxed{-}$ that behave as above.

Equivalently, extend type theory by a generic proposition $¶ : \mathbf{Prop}$ and define $\boxed{A} := A^{¶}$ and $\boxed{A} := A \cup_{A \times ¶} ¶$.

Internal language of topoi formed by *Artin gluing* (Artin, Grothendieck, and Verdier, 1972; Wraith, 1974; Rijke, Shulman, and Spitters, 2020).

Example: synthetic computability structure of the booleans

$$[\![\mathsf{bool}]\!] = \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

Example: synthetic computability structure of the booleans

$$[\![\mathsf{bool}]\!] = \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

The syntactic part of $[\![\mathsf{bool}]\!]$ is the syntactic booleans.

$$\boxed{[\![\mathsf{bool}]\!]} \cong \boxed{\left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}}$$

Example: synthetic computability structure of the booleans

$$[\![\mathsf{bool}]\!] = \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

The syntactic part of $[\![\mathsf{bool}]\!]$ is the syntactic booleans.

$$\begin{aligned} \boxed{[\![\mathsf{bool}]\!]} &\cong \boxed{\left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \\ &\cong \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \end{aligned}$$

Example: synthetic computability structure of the booleans

$$[\![\mathsf{bool}]\!] = \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

The syntactic part of $[\![\mathsf{bool}]\!]$ is the syntactic booleans.

$$\begin{aligned}
\boxed{[\![\mathsf{bool}]\!]} &\cong \boxed{\left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \\
&\cong \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \\
&\cong \left(x : \boxed{\mathsf{bool}}\right) \times \mathsf{unit}
\end{aligned}$$

Example: synthetic computability structure of the booleans

$$[\![\mathsf{bool}]\!] = \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}$$

The syntactic part of $[\![\mathsf{bool}]\!]$ is the syntactic booleans.

$$\begin{aligned}
\boxed{[\![\mathsf{bool}]\!]} &\cong \boxed{\left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \\
&\cong \left(x : \boxed{\mathsf{bool}}\right) \times \boxed{\boxed{\boxed{x = \mathsf{tt}} + \boxed{x = \mathsf{ff}}}} \\
&\cong \left(x : \boxed{\mathsf{bool}}\right) \times \mathsf{unit} \\
&\cong \boxed{\mathsf{bool}}
\end{aligned}$$

3. From the general to the particular. . .

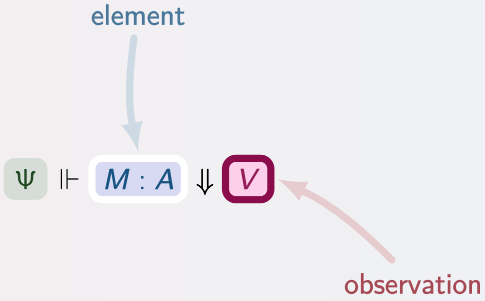
In what contexts do we compute?

$$\Psi \Vdash M : A \Downarrow V$$

In what contexts do we compute?

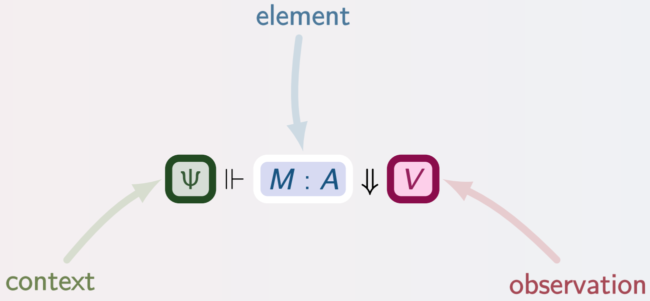
$$\overset{\text{element}}{\Psi \Vdash \boxed{M : A} \Downarrow V}$$

In what contexts do we compute?

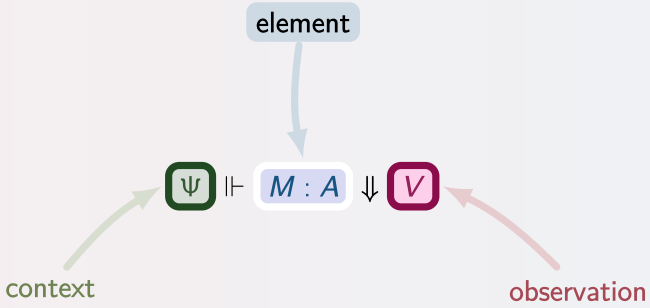


$$\boxed{\Psi} \Vdash \boxed{M : A} \Downarrow \boxed{V}$$

In what contexts do we compute?

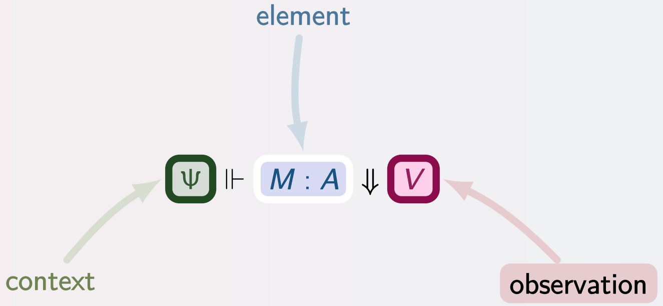


In what contexts do we compute?



canonicity: $A \in \{\mathsf{nat}\}$; **normalization:** $A \in \{\Psi \vdash \mathit{type}\}$

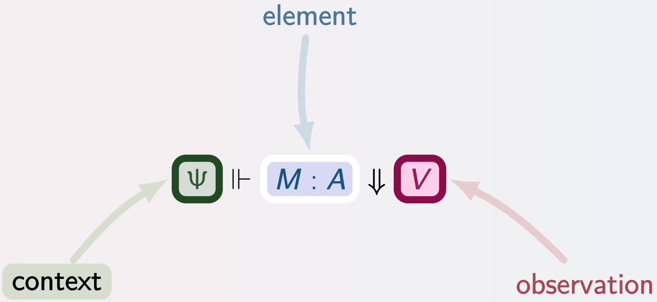
In what contexts do we compute?



canonicity: $A \in \{\mathsf{nat}\}$; **normalization:** $A \in \{\Psi \vdash \mathit{type}\}$

canonicity: $V \in \mathbb{N}$; **normalization:** $V \in \left\{\Psi \Vdash^{\beta\eta}_{\mathbf{nf}} A\right\}$

In what contexts do we compute?



$$\boxed{\Psi} \Vdash \boxed{M : A} \Downarrow \boxed{V}$$

canonicity: $A \in \{\mathsf{nat}\}$; **normalization:** $A \in \{\Psi \vdash \mathit{type}\}$

canonicity: $V \in \mathbb{N}$; **normalization:** $V \in \left\{\Psi \Vdash^{\beta\eta}_{\mathbf{nf}} A\right\}$

canonicity: $\Gamma \in \{\cdot\}$; **cubical canonicity:** $\Gamma \in \{\mathbb{I}^n \mid n \in \mathbb{N}\}$; **normalization:** $\Gamma \in \{\vdash \mathit{ctx}\}$

Stability (or lack thereof) of observation

$$\boxed{x : \mathsf{nat}} \Vdash \boxed{x : \mathsf{nat}} \Downarrow \boxed{\mathbf{var}(x)}$$

Stability (or lack thereof) of observation

$$\boxed{\cdot} \xrightarrow{[\mathsf{fib}(9)/x]} \boxed{x : \mathsf{nat}} \Vdash \boxed{x : \mathsf{nat}} \Downarrow \boxed{\mathbf{var}(x)}$$

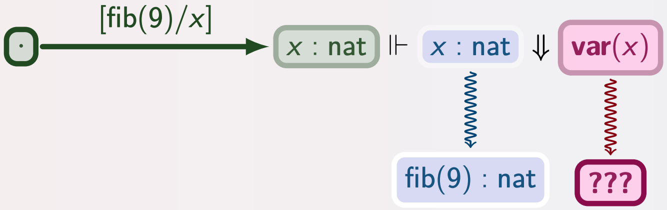
Stability (or lack thereof) of observation

$$\begin{tikzcd}
\boxed{\cdot} \arrow[r, "{[\mathsf{fib}(9)/x]}"] & \boxed{x : \mathsf{nat}} \arrow[r, phantom, "\Vdash"] & \boxed{x : \mathsf{nat}} \arrow[r, phantom, "\Downarrow"] \arrow[d, rightsquigarrow] & \boxed{\mathbf{var}(x)} \\
 & & \boxed{\mathsf{fib}(9) : \mathsf{nat}} &
\end{tikzcd}$$

Stability (or lack thereof) of observation

$$\begin{tikzcd}
\boxed{\cdot} \arrow[r, "{[\mathsf{fib}(9)/x]}"] & \boxed{x : \mathsf{nat}} \arrow[r, phantom, "\Vdash"] & \boxed{x : \mathsf{nat}} \arrow[d, squiggly] \arrow[r, phantom, "\Downarrow"] & \boxed{\mathbf{var}(x)} \arrow[d, squiggly] \\
 & & \boxed{\mathsf{fib}(9) : \mathsf{nat}} & \boxed{???}
\end{tikzcd}$$

Stability (or lack thereof) of observation


$$\begin{tikzcd} \boxed{\cdot} \arrow[r, "{[\mathsf{fib}(9)/x]}"] & \boxed{x : \mathsf{nat}} \arrow[r, phantom, "\Vdash"] & \boxed{x : \mathsf{nat}} \arrow[d, squiggly] \arrow[r, phantom, "\Downarrow"] & \boxed{\mathbf{var}(x)} \arrow[d, squiggly] \\ & & \boxed{\mathsf{fib}(9) : \mathsf{nat}} & \boxed{\mathbf{???}} \end{tikzcd}$$

In plain type theory, neutral observations (elimination forms blocked on variables) are closed under *renaming*, but not full substitution.

Stability (or lack thereof) of observation

$$\begin{tikzcd}
\boxed{\cdot} \arrow[r, "{[\mathsf{fib}(9)/x]}"] & \boxed{x : \mathsf{nat}} \arrow[r, phantom, "\Vdash"] & \boxed{x : \mathsf{nat}} \arrow[d, squiggly] \arrow[r, phantom, "\Downarrow"] & \boxed{\mathbf{var}(x)} \arrow[d, squiggly] \\
 & & \boxed{\mathsf{fib}(9) : \mathsf{nat}} & \boxed{???}
\end{tikzcd}$$

In plain type theory, neutral observations (elimination forms blocked on variables) are closed under *renaming*, but not full substitution.

Therefore normalization takes place over the category $\mathcal{R}$ of contexts and *structural renamings* (weakening, swapping, contraction).

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for **□TT**. The problem lies with the interval:

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for **□TT**. The problem lies with the interval:

$$\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}, i : \mathbb{I}} \Vdash \boxed{(p \mathbin{@} i)\,9 : \mathsf{nat}} \Downarrow \boxed{\mathbf{app}(\mathbf{pathapp}(\mathbf{var}(p), i), \mathbf{su}^9(\mathbf{ze}))}$$

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for □TT. The problem lies with the interval:

$$\begin{tikzcd}
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}, i : \mathbb{I}} \Vdash \boxed{(p \mathbin{@} i)\, 9 : \mathsf{nat}} \Downarrow \boxed{\mathbf{app}(\mathbf{pathapp}(\mathbf{var}(p), i), \mathbf{su}^{9}(\mathbf{ze}))} \\
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}} \arrow[u, "{[0/i]}"]
\end{tikzcd}$$

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for □TT. The problem lies with the interval:

$$\begin{tikzcd}
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}, i : \mathbb{I}} \arrow[r, phantom, "\Vdash"] & \boxed{(p \mathbin{@} i)\, 9 : \mathsf{nat}} \arrow[d, squiggly] \arrow[r, phantom, "\Downarrow"] & \boxed{\mathbf{app}(\mathbf{pathapp}(\mathbf{var}(p), i), \mathbf{su}^{9}(\mathbf{ze}))} \\
 & \boxed{\mathsf{fib}(9) : \mathsf{nat}} & \\
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}} \arrow[uu, "{[0/i]}"] & &
\end{tikzcd}$$

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for □TT. The problem lies with the interval:

$$\begin{tikzcd}
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}, i : \mathbb{I}} & \Vdash & \boxed{(p \mathbin{@} i)\, 9 : \mathsf{nat}} \arrow[d, squiggly] & \Downarrow & \boxed{\mathbf{app}(\mathbf{pathapp}(\mathbf{var}(p), i), \mathbf{su}^9(\mathbf{ze}))} \arrow[d, squiggly] \\
 & & \boxed{\mathsf{fib}(9) : \mathsf{nat}} & & \boxed{???} \\
\boxed{p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}} \arrow[uu, "{[0/i]}"] & & & &
\end{tikzcd}$$

What goes wrong for □TT?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for **□TT**. The problem lies with the interval:

$$\begin{tikzcd}
p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib}, i : \mathbb{I} \arrow[r, phantom, "\Vdash"] & (p \mathbin{@} i)\, 9 : \mathsf{nat} \arrow[d, rightsquigarrow] \arrow[r, phantom, "\Downarrow"] & \mathbf{app}(\mathbf{pathapp}(\mathbf{var}(p), i), \mathbf{su}^{9}(\mathbf{ze})) \arrow[d, rightsquigarrow] \\
 & \mathsf{fib}(9) : \mathsf{nat} & ??? \\
p : \mathsf{fib} =_{\mathsf{nat}\to\mathsf{nat}} \mathsf{fib} \arrow[uu, "{[0/i]}"] & &
\end{tikzcd}$$

We shouldn't remove $[0/i], [1/i]$ from the category of contexts and renamings because we need $\mathbb{I}$ to restrict to something *representable* in $\mathrm{Pr}(\mathcal{R})$, *c.f.* **tininess** criterion (Licata, Orton, Pitts, and Spitters, 2018).

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of **instability***:

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of* ***instability***:

$$\partial(\mathbf{var}(x)) = \bot$$

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of **instability***:

$$\partial(\mathbf{var}(x)) = \bot$$
$$\partial(\mathbf{app}(E, M)) = \partial E$$

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of **instability***:

$$\begin{aligned}\partial(\mathbf{var}(x)) &= \bot\\ \partial(\mathbf{app}(E, M)) &= \partial E\\ \partial(\mathbf{fst}(E)) &= \partial E\end{aligned}$$

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of **instability***:

$$\begin{aligned}
\partial(\mathbf{var}(x)) &= \bot \\
\partial(\mathbf{app}(E, M)) &= \partial E \\
\partial(\mathbf{fst}(E)) &= \partial E \\
\partial(\mathbf{pathapp}(E, r)) &= \partial E \vee (r = 0) \vee (r = 1)
\end{aligned}$$

The power of dialectical thinking: geometrical negation

Thesis: neutrals need to have a cubical substitution action (tininess of $\mathbb{I}$).

Antithesis: positive neutrality is not a cubical notion: under face maps $[0/i], [1/i]$ a neutral observation can cease 'being neutral' and needs to 'compute'.

Synthesis: the conditions away from which a term is neutral *are* cubical. Write ∂E for this *frontier of **instability***:

$$\begin{aligned}
\partial(\mathbf{var}(x)) &= \bot \\
\partial(\mathbf{app}(E, M)) &= \partial E \\
\partial(\mathbf{fst}(E)) &= \partial E \\
\partial(\mathbf{pathapp}(E, r)) &= \partial E \vee (r = 0) \vee (r = 1)
\end{aligned}$$

Therefore we define an inductive family $\mathbf{Ne}_\phi(A)$ with $\boxed{\mathbf{Ne}_\phi(A)} \cong \boxed{A}$ comprised of neutrals e with $\partial e = \phi$. Traditional neutrals $\mathbf{Ne}_\bot(A)$; to model destabilization, $\mathbf{Ne}_\top(A) \cong \boxed{A}$.

Normalization via Tait's yoga

Tait (1967) introduced the famous *saturation yoga* for normalization:

$$\mathbf{Ne}(A) \subseteq [\![A]\!] \subseteq \mathbf{Nf}(A)$$

Normalization via Tait's yoga

Tait (1967) introduced the famous *saturation yoga* for normalization:[10]

$$\begin{tikzcd}
\mathbf{Ne}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & \llbracket A \rrbracket \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
& A &
\end{tikzcd}$$

[10] *cf.* normalization by evaluation in the style of Fiore (2002), Altenkirch, Hofmann, and Streicher (1995), Altenkirch and Kaposi (2016), and Coquand (2019)

Normalization via Tait's yoga

Tait (1967) introduced the famous *saturation yoga* for normalization:[10]

$$\begin{tikzcd}
\mathbf{Ne}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & [\![A]\!] \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
 & A &
\end{tikzcd}$$

"reflection"

[10] *cf.* normalization by evaluation in the style of Fiore (2002), Altenkirch, Hofmann, and Streicher (1995), Altenkirch and Kaposi (2016), and Coquand (2019)

Normalization via Tait's yoga

Tait (1967) introduced the famous *saturation yoga* for normalization:[10]

$$\begin{tikzcd}
\mathbf{Ne}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & {[\![A]\!]} \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl, "\text{“reification”}"] \\
& A &
\end{tikzcd}$$

[10] *cf.* normalization by evaluation in the style of Fiore (2002), Altenkirch, Hofmann, and Streicher (1995), Altenkirch and Kaposi (2016), and Coquand (2019)

Normalization via Tait's yoga

Tait (1967) introduced the famous *saturation yoga* for normalization:[10]

$$\begin{tikzcd}
\mathbf{Ne}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & \llbracket A \rrbracket \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
& A &
\end{tikzcd}$$

"saturation / Tait's yoga"

[10] *cf.* normalization by evaluation in the style of Fiore (2002), Altenkirch, Hofmann, and Streicher (1995), Altenkirch and Kaposi (2016), and Coquand (2019)

Yogic injury: unstable neutrals

$$\begin{tikzcd}
\mathbf{Ne}_{\phi}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & \llbracket A \rrbracket \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
 & A &
\end{tikzcd}$$

Yogic injury: unstable neutrals

$$\begin{tikzcd}
\mathbf{Ne}_\phi(A) \arrow[r, "\uparrow_A"] \arrow[dr] & {[\![A]\!]} \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
 & A &
\end{tikzcd}$$

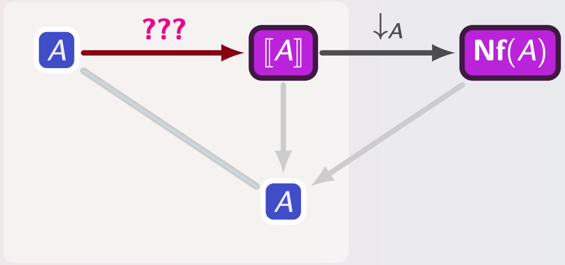
What if $\phi = \top$?

Yogic injury: unstable neutrals

$$\begin{tikzcd}
\mathbf{Ne}_{\top}(A) \arrow[r, "\uparrow_A"] \arrow[dr] & [\![A]\!] \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
& A &
\end{tikzcd}$$

What if $\phi = \top$?

Yogic injury: unstable neutrals



$$\begin{tikzcd}
A \arrow[r, "???"] \arrow[dr, no head] & \llbracket A \rrbracket \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
& A &
\end{tikzcd}$$

What if $\phi = \top$?

Yogic injury: unstable neutrals

$$\begin{tikzcd}
A \arrow[r, "???"] \arrow[dr, no head] & {[\![A]\!]} \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
& A &
\end{tikzcd}$$

What if $\phi = \top$? We must strengthen the "induction hypothesis".

Stabilization of neutrals

To strengthen the Tait reflection hypothesis, we **glue** unstable neutrals together with compatible computability data along their frontiers of instability.

$$\begin{tikzcd}
\mathbf{Ne}_\phi(A) \wr_\phi [\![A]\!] \arrow[r, dotted] \arrow[dd, dotted] \arrow[ddr, phantom, "\lrcorner", very near start] & \mathbf{Ne}_\phi(A) \arrow[d] \\
 & A \arrow[d] \\
\phi \Rightarrow [\![A]\!] \arrow[r] & \phi \Rightarrow A
\end{tikzcd}$$

Stabilization of neutrals

To strengthen the Tait reflection hypothesis, we **glue** unstable neutrals together with compatible computability data along their frontiers of instability.

$$\begin{tikzcd}
\mathbf{Ne}_\phi(A) \wr_\phi \llbracket A \rrbracket \arrow[r, dotted] \arrow[dd, dotted] \arrow[ddr, phantom, "\lrcorner", very near start] & \mathbf{Ne}_\phi(A) \arrow[d] \\
& A \arrow[d] \\
\phi \Rightarrow \llbracket A \rrbracket \arrow[r] & \phi \Rightarrow A
\end{tikzcd}$$

“stabilized neutrals”

A spectrum of computability data

$$\begin{tikzcd}
\mathbf{Ne}_\perp(A) \arrow[r, rightsquigarrow, "\mathbf{Ne}_\phi \wr_\phi \llbracket A \rrbracket"] \arrow[d, mapsto] & \llbracket A \rrbracket \arrow[d, mapsto] \\
\perp \arrow[r, rightsquigarrow, "\phi"] & \top
\end{tikzcd}$$

Stabilization **interpolates** between neutrals and computability data.

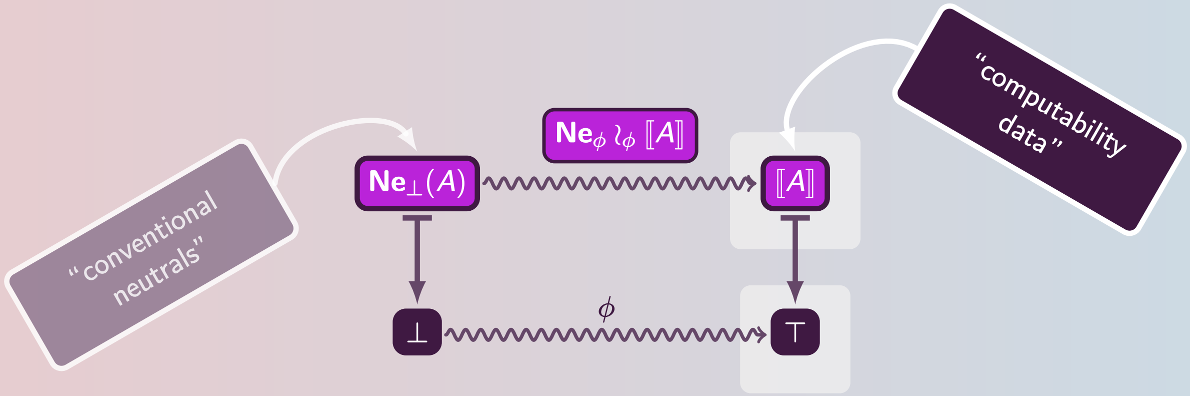
A spectrum of computability data

“conventional neutrals”

$$\begin{tikzcd}
\mathbf{Ne}_\perp(A) \arrow[r, rightsquigarrow, "\mathbf{Ne}_\phi \wr_\phi [\![A]\!]"] \arrow[d, mapsto] & {[\![A]\!]} \arrow[d, mapsto] \\
\perp \arrow[r, rightsquigarrow, "\phi"] & \top
\end{tikzcd}$$

Stabilization **interpolates** between neutrals and computability data.

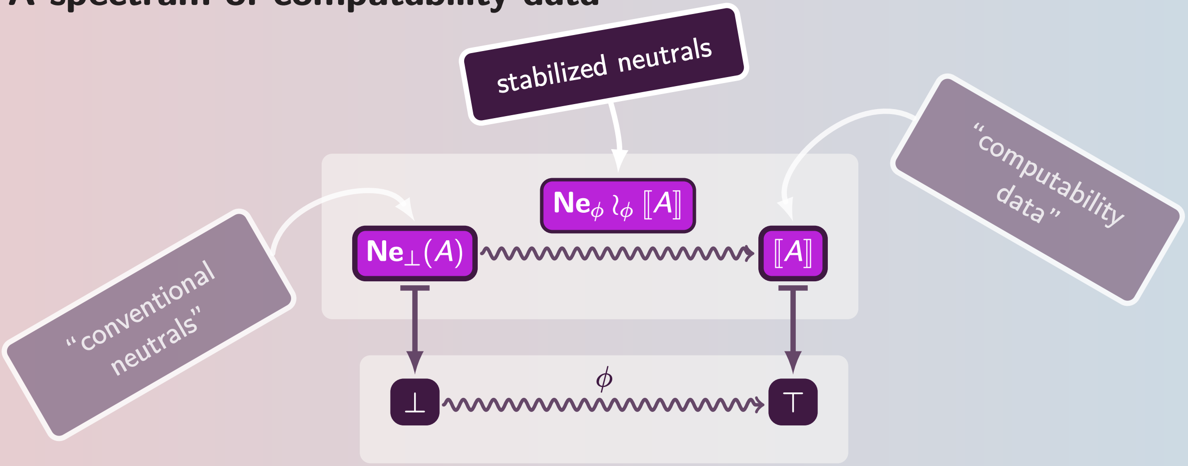
A spectrum of computability data



$$\begin{tikzcd} \mathbf{Ne}_\perp(A) \arrow[r, rightsquigarrow] \arrow[d, mapsto] & \llbracket A \rrbracket \arrow[d, mapsto] \\ \perp \arrow[r, rightsquigarrow, "\phi"] & \top \end{tikzcd}$$

Stabilization **interpolates** between neutrals and computability data.

A spectrum of computability data



$$\begin{tikzcd}
& \mathbf{Ne}_{\phi} \wr_{\phi} [\![A]\!] & \\
\mathbf{Ne}_{\perp}(A) \arrow[rr, squiggly] \arrow[d, |->] & & {[\![A]\!]} \arrow[d, |->] \\
\perp \arrow[rr, squiggly, "\phi"] & & \top
\end{tikzcd}$$

Stabilization **interpolates** between neutrals and computability data.

The stabilized Tait yoga

$$\begin{tikzcd}
 & & \phi \Rightarrow \llbracket A \rrbracket & & \\
\mathbf{Ne}_{\phi}(A) \wr_{\phi} \llbracket A \rrbracket \arrow[urr] \arrow[rr, "\uparrow_{A}^{\phi}"] \arrow[d] \arrow[drr] & & \llbracket A \rrbracket \arrow[u] \arrow[rr, "\downarrow_{A}"] \arrow[d] & & \mathbf{Nf}(A) \arrow[dll] \\
\mathbf{Ne}_{\phi}(A) \arrow[rr] & & A & &
\end{tikzcd}$$

The stabilized Tait yoga

$$\begin{tikzcd}
 & \phi \Rightarrow [\![A]\!] & \\
\mathbf{Ne}_\phi(A) \wr_\phi [\![A]\!] \arrow[ur] \arrow[r, "\uparrow^\phi_A"] \arrow[d] \arrow[dr] & [\![A]\!] \arrow[u] \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
\mathbf{Ne}_\phi(A) \arrow[r] & A &
\end{tikzcd}$$

The stabilized Tait yoga

$$\begin{tikzcd}
 & \phi \Rightarrow [\![A]\!] & \\
\mathbf{Ne}_\phi(A) \wr_\phi [\![A]\!] \arrow[ur] \arrow[r, "\uparrow^\phi_A"] \arrow[d] \arrow[dr] & {[\![A]\!]} \arrow[u] \arrow[r, "\downarrow_A"] \arrow[d] & \mathbf{Nf}(A) \arrow[dl] \\
\mathbf{Ne}_\phi(A) \arrow[r] & A &
\end{tikzcd}$$

Lemma (Saturation)

Every type of **□TT** *is closed under the* ***stabilized*** *Tait yoga.*

Summary of results

Lemma (Saturation)

Every type of **□TT** *is closed under the* ***stabilized*** *Tait yoga.*

The above is employed to obtain our main results:

Theorem (Normalization)

There is a computable function assigning to every type $\Gamma \vdash A$ *and every term* $\Gamma \vdash a : A$ *of* **□TT** *a unique normal form.*

Corollary (Decidability of equality)

Judgmental equality $\Gamma \vdash A \equiv B$ *and* $\Gamma \vdash a \equiv b : A$ *in* **□TT** *is decidable.*

Corollary (Injectivity of type constructors)

If $\Gamma \vdash \Pi(A, B) \equiv \Pi(A', B')$ *then* $\Gamma \vdash A \equiv A'$ *and* $\Gamma, x : A \vdash B(x) \equiv B'(x)$.

4. Taking stock

A computational conspectus on cubes. . .

The community designed **□TT** with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:

A computational conspectus on cubes. . .

The community designed **$\square$TT** with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:

1. **constructive model in cubical sets**
   by Cohen, Coquand, Huber, and Mörtberg (2017) and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2019).

A computational conspectus on cubes...

The community designed **□TT** with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:

1. **constructive model in cubical sets**
   by Cohen, Coquand, Huber, and Mörtberg (2017) and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2019).
2. **computational interpretation of closed *n*-cubes**
   by Angiuli, Hou (Favonia), and Harper (2018) and Huber (2018).

A computational conspectus on cubes...

The community designed **$\square$TT** with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:

1. **constructive model in cubical sets**
   by Cohen, Coquand, Huber, and Mörtberg (2017) and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2019).
2. **computational interpretation of closed *n*-cubes**
   by Angiuli, Hou (Favonia), and Harper (2018) and Huber (2018).
3. **standard model in homotopy types**
   by Awodey, Cavallo, Coquand, Riehl, and Sattler (forthcoming).

A computational conspectus on cubes. . .

The community designed **□TT** with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:

1. **constructive model in cubical sets**
   by Cohen, Coquand, Huber, and Mörtberg (2017) and Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata (2019).
2. **computational interpretation of closed *n*-cubes**
   by Angiuli, Hou (Favonia), and Harper (2018) and Huber (2018).
3. **standard model in homotopy types**
   by Awodey, Cavallo, Coquand, Riehl, and Sattler (forthcoming).
4. **computational interpretation of open terms**
   by Sterling and Angiuli (2021), this dissertation.

What's next for cubical type theory?

We have done more than enough cubical type *theory*. Time for applications!

- **applications to programming and verification**
  Cavallo and Harper (2020), Angiuli, Cavallo, Mörtberg, and Zeuner (2021), and Kidney and Wu (2021)
- **applications to denotational semantics**
  Møgelberg and Veltri (2019), Veltri and Vezzosi (2020), and Møgelberg and Vezzosi (2021)
- **applications to ordinary mathematics**
  Forsberg, Xu, and Ghani (2020)
- **applications to synthetic homotopy theory**
  Mörtberg and Pujet (2020), Cavallo (2021), and Brunerie, Ljungström, and Mörtberg (2021)

The era of synthetic Tait computability

- [LICS'21] **Normalization for cubical type theory** (Sterling and Angiuli, 2021)
- [JACM] **Logical Relations As Types: Proof-Relevant Parametricity for Program Modules** (Sterling and Harper, 2021)
- **Normalization for multi-modal type theory** (Gratzer, 2021).
- **A cost-aware logical framework** (Niu, Sterling, Grodin, and Harper, 2021)

Let a hundred phase distinctions bloom!

STC also leads to new perspectives on classic PL problems, *cf.* S. and Harper's analysis of the static/dynamic **phase distinction** and sealing in terms of STC.

Let a hundred phase distinctions bloom!

STC also leads to new perspectives on classic PL problems, *cf.* S. and Harper's analysis of the static/dynamic **phase distinction** and sealing in terms of STC.

program modules	static	dynamic
logical relations	syntax	semantics
type refinements	computation	specification
resource analysis	behavior	complexity
security / IFC	public	classified

Sterling, Jonathan and Robert Harper (2021). "Logical Relations As Types: Proof-Relevant Parametricity for Program Modules". In: *Journal of the ACM*. To appear. arXiv: `2010.08599 [cs.PL]`.

Let a hundred phase distinctions bloom!

STC also leads to new perspectives on classic PL problems, *cf.* S. and Harper's analysis of the static/dynamic **phase distinction** and sealing in terms of STC.

program modules	static	dynamic
logical relations	syntax	semantics
type refinements	computation	specification
resource analysis	behavior	complexity
security / IFC	public	classified

Gratzer, Daniel (2021). *Normalization for Multimodal Type Theory*. arXiv: 2106.01414 [cs.LO].

Sterling, Jonathan and Carlo Angiuli (July 2021). "Normalization for Cubical Type Theory". In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1–15. DOI: 10.1109/LICS52264.2021.9470719. arXiv: 2101.11479 [cs.LO].

Sterling, Jonathan and Robert Harper (2021). "Logical Relations As Types: Proof-Relevant Parametricity for Program Modules". In: *Journal of the ACM*. To appear. arXiv: 2010.08599 [cs.PL].

Let a hundred phase distinctions bloom!

STC also leads to new perspectives on classic PL problems, *cf.* S. and Harper's analysis of the static/dynamic **phase distinction** and sealing in terms of STC.

program modules	static	dynamic
logical relations	syntax	semantics
type refinements	computation	specification
resource analysis	behavior	complexity
security / IFC	public	classified

Niu, Yue, Jonathan Sterling, Harrison Grodin, and Robert Harper (2021). *A cost-aware logical framework*. arXiv: `2107.04663 [cs.PL]`.

Let a hundred phase distinctions bloom!

STC also leads to new perspectives on classic PL problems, *cf.* S. and Harper's analysis of the static/dynamic **phase distinction** and sealing in terms of STC.

program modules	static	dynamic
logical relations	syntax	semantics
type refinements	computation	specification
resource analysis	behavior	complexity
security / IFC	public	classified

Sterling, Jonathan, Stephanie Balzer, and Robert Harper (2021). "Abstract phase distinctions and noninterference". Work in progress.

Thanks!

- Part I — syntax of dependent type theory *c.* 2021
- Part II — mathematical background (topos theory, universes)
- Part III — synthetic Tait computability, synthetic normalization for MLTT
- Part IV — cubical type theory, all main theorems

References I

Tait, W. W. (1967). “Intensional Interpretations of Functionals of Finite Type I”. In: *The Journal of Symbolic Logic* 32.2, pp. 198–212. ISSN: 00224812. URL: http://www.jstor.org/stable/2271658.

Artin, Michael, Alexander Grothendieck, and Jean-Louis Verdier (1972). *Théorie des topos et cohomologie étale des schémas*. Séminaire de Géométrie Algébrique du Bois-Marie 1963–1964 (SGA 4), Dirigé par M. Artin, A. Grothendieck, et J.-L. Verdier. Avec la collaboration de N. Bourbaki, P. Deligne et B. Saint-Donat, Lecture Notes in Mathematics, Vol. 269, 270, 305. Berlin: Springer-Verlag.

Wraith, Gavin (1974). “Artin glueing”. In: *Journal of Pure and Applied Algebra* 4.3, pp. 345–348. ISSN: 0022-4049. DOI: 10.1016/0022-4049(74)90014-0.

Mac Lane, Saunders and Ieke Moerdijk (1992). *Sheaves in geometry and logic: a first introduction to topos theory*. Universitext. New York: Springer. ISBN: 0-387-97710-4.

Altenkirch, Thorsten, Martin Hofmann, and Thomas Streicher (1995). “Categorical reconstruction of a reduction free normalization proof”. In: *Category Theory and Computer Science*. Ed. by David Pitt, David E. Rydeheard, and Peter Johnstone. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 182–199. ISBN: 978-3-540-44661-3.

Fiore, Marcelo (2002). “Semantic Analysis of Normalisation by Evaluation for Typed Lambda Calculus”. In: *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. PPDP ’02. Pittsburgh, PA, USA: Association for Computing Machinery, pp. 26–37. ISBN: 1-58113-528-9. DOI: 10.1145/571157.571161.

References II

Univalent Foundations Program, The (2013). *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book.

Bezem, Marc, Thierry Coquand, and Simon Huber (2014). “A Model of Type Theory in Cubical Sets”. In: *19th International Conference on Types for Proofs and Programs (TYPES 2013)*. Ed. by Ralph Matthes and Aleksy Schubert. Vol. 26. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 107–128. ISBN: 978-3-939897-72-9. DOI: 10.4230/LIPIcs.TYPES.2013.107. URL: http://drops.dagstuhl.de/opus/volltexte/2014/4628.

Altenkirch, Thorsten and Ambrus Kaposi (2016). “Normalisation by Evaluation for Dependent Types”. In: *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*. Ed. by Delia Kesner and Brigitte Pientka. Vol. 52. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 6:1–6:16. ISBN: 978-3-95977-010-1. DOI: 10.4230/LIPIcs.FSCD.2016.6. URL: http://drops.dagstuhl.de/opus/volltexte/2016/5972.

Orton, Ian and Andrew M. Pitts (2016). “Axioms for Modelling Cubical Type Theory in a Topos”. In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Ed. by Jean-Marc Talbot and Laurent Regnier. Vol. 62. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 24:1–24:19. ISBN: 978-3-95977-022-4. DOI: 10.4230/LIPIcs.CSL.2016.24.

References III

Angiuli, Carlo, Kuen-Bang Hou (Favonia), and Robert Harper (2017). *Computational Higher Type Theory III: Univalent Universes and Exact Equality*. arXiv: 1712.01800 [cs.LO]. URL: https://arxiv.org/abs/1712.01800.

Cohen, Cyril, Thierry Coquand, Simon Huber, and Anders Mörtberg (Nov. 2017). "Cubical Type Theory: a constructive interpretation of the univalence axiom". In: *IfCoLog Journal of Logics and their Applications* 4.10, pp. 3127–3169. URL: http://www.collegepublications.co.uk/journals/ifcolog/?00019.

Angiuli, Carlo, Kuen-Bang Hou (Favonia), and Robert Harper (2018). "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities". In: *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*. Ed. by Dan Ghica and Achim Jung. Vol. 119. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 6:1–6:17. ISBN: 978-3-95977-088-0. DOI: 10.4230/LIPIcs.CSL.2018.6. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9673.

Awodey, Steve (2018). "A cubical model of homotopy type theory". In: *Annals of Pure and Applied Logic* 169.12. Logic Colloquium 2015, pp. 1270–1294. ISSN: 0168-0072. DOI: 10.1016/j.apal.2018.08.002.

Coquand, Thierry (Oct. 2018). *Canonicity and normalisation for Dependent Type Theory*. arXiv: 1810.09367 [cs.PL]. URL: https://arxiv.org/abs/1810.09367.

Huber, Simon (June 13, 2018). "Canonicity for Cubical Type Theory". In: *Journal of Automated Reasoning*. ISSN: 1573-0670. DOI: 10.1007/s10817-018-9469-1.

References IV

Licata, Daniel R., Ian Orton, Andrew M. Pitts, and Bas Spitters (2018). “Internal Universes in Models of Homotopy Type Theory”. In: *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, 22:1–22:17. DOI: 10.4230/LIPIcs.FSCD.2018.22.

Angiuli, Carlo, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata (Feb. 2019). *Syntax and Models of Cartesian Cubical Type Theory*. Preprint. URL: https://github.com/dlicata335/cart-cube.

Coquand, Thierry (2019). “Canonicity and normalization for dependent type theory”. In: *Theoretical Computer Science* 777. In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I, pp. 184–191. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2019.01.015. arXiv: 1810.09367 [cs.PL].

Møgelberg, Rasmus Ejlers and Niccolò Veltri (Jan. 2019). “Bisimulation as Path Type for Guarded Recursive Types”. In: *Proceedings of the ACM on Programming Languages* 3.POPL. DOI: 10.1145/3290317.

Sterling, Jonathan, Carlo Angiuli, and Daniel Gratzer (2019). “Cubical Syntax for Reflection-Free Extensional Equality”. In: *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*. Ed. by Herman Geuvers. Vol. 131. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 31:1–31:25. ISBN: 978-3-95977-107-8. DOI: 10.4230/LIPIcs.FSCD.2019.31. arXiv: 1904.08562 [cs.LO]. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10538.

References V

Cavallo, Evan and Robert Harper (2020). “Internal Parametricity for Cubical Type Theory”. In: *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. Ed. by Maribel Fernández and Anca Muscholl. Vol. 152. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 13:1–13:17. ISBN: 978-3-95977-132-0. DOI: 10.4230/LIPIcs.CSL.2020.13. URL: https://drops.dagstuhl.de/opus/volltexte/2020/11656.

Forsberg, Fredrik Nordvall, Chuangjie Xu, and Neil Ghani (2020). “Three Equivalent Ordinal Notation Systems in Cubical Agda”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. New Orleans, LA, USA: Association for Computing Machinery, pp. 172–185. ISBN: 978-1-4503-7097-4. DOI: 10.1145/3372885.3373835.

Mörtberg, Anders and Loïc Pujet (2020). “Cubical Synthetic Homotopy Theory”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. New Orleans, LA, USA: Association for Computing Machinery, pp. 158–171. ISBN: 978-1-4503-7097-4. DOI: 10.1145/3372885.3373825.

Rijke, Egbert, Michael Shulman, and Bas Spitters (Jan. 2020). “Modalities in homotopy type theory”. In: *Logical Methods in Computer Science* Volume 16, Issue 1. DOI: 10.23638/LMCS-16(1:2)2020. arXiv: 1706.07526 [math.CT]. URL: https://lmcs.episciences.org/6015.

References VI

Veltri, Niccolò and Andrea Vezzosi (2020). “Formalizing π-Calculus in Guarded Cubical Agda”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. New Orleans, LA, USA: Association for Computing Machinery, pp. 270–283. ISBN: 978-1-4503-7097-4. DOI: 10.1145/3372885.3373814.

Angiuli, Carlo, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata (Mar. 2021). *Syntax and Models of Cartesian Cubical Type Theory*. Preprint. URL: https://www.cs.cmu.edu/~cangiuli/papers/abcfhl.pdf.

Angiuli, Carlo, Evan Cavallo, Anders Mörtberg, and Max Zeuner (Jan. 2021). “Internalizing Representation Independence with Univalence”. In: *Proceedings of the ACM on Programming Languages* 5.POPL, pp. 1–30. DOI: 10.1145/3434293.

Brunerie, Guillaume, Axel Ljungström, and Anders Mörtberg (2021). “Synthetic Cohomology Theory in Cubical Agda”. Preprint.

Cavallo, Evan (2021). “Higher Inductive Types and Internal Parametricity for Cubical Type Theory”. PhD thesis. Carnegie Mellon University.

Gratzer, Daniel (2021). *Normalization for Multimodal Type Theory*. arXiv: 2106.01414 [cs.LO].

Kidney, Donnacha Oisín and Nicolas Wu (Aug. 2021). “Algebras for Weighted Search”. In: *Proceedings of the ACM on Programming Languages* 5.ICFP. DOI: 10.1145/3473577.

Møgelberg, Rasmus Ejlers and Andrea Vezzosi (2021). “Two Guarded Recursive Powerdomains for Applicative Simulation”. In: *MFPS37: 37th Conference on Mathematical Foundations of Programming Semantics*.

References VII

Niu, Yue, Jonathan Sterling, Harrison Grodin, and Robert Harper (2021). *A cost-aware logical framework*. arXiv: 2107.04663 [cs.PL].

Sterling, Jonathan and Carlo Angiuli (July 2021). "Normalization for Cubical Type Theory". In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1–15. DOI: 10.1109/LICS52264.2021.9470719. arXiv: 2101.11479 [cs.LO].

Sterling, Jonathan and Robert Harper (2021). "Logical Relations As Types: Proof-Relevant Parametricity for Program Modules". In: *Journal of the ACM*. To appear. arXiv: 2010.08599 [cs.PL].