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 + \textit{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

Semantic properties

(function extensionality)

(effective quotients)

(unique choice (∀∃! ⇒ ∃∀))

(type extensionality (univalence)?)

.consistency

.closed term computation

(decidable type checking)

(all models)
Requirements of type theoretic tools

Semantic properties

(all models)

Syntactic properties
Requirements of type theoretic tools

Semantic properties

(all models)

Syntactic properties

(Just the syntax)
Requirements of type theoretic tools

Semantic properties
- function extensionality

Syntactic properties

(all models)

(just the syntax)
Requirements of type theoretic tools

Semantic properties
- function extensionality
- effective quotients

Syntactic properties
- consistency
- closed term computation
- decidable type checking

(all models)

(just the syntax)
Requirements of type theoretic tools

Semantic properties
- function extensionality
- effective quotients
- unique choice (\(\forall \exists \Rightarrow \exists \forall\))

Syntactic properties

(all models)

(just the syntax)
Requirements of type theoretic tools

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

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

*(all models)* *(just the syntax)*
Requirements of type theoretic tools

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

**Syntactic properties**
- consistency

*(all models)*

*(just the syntax)*
Requirements of type theoretic tools

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

Syntactic properties
- consistency
- closed term computation

(all models) (just the syntax)
Requirements of type theoretic tools

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

(all models)

Syntactic properties
- consistency
- closed term computation
- decidable type checking

(just the syntax)
Requirements of type theoretic tools

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

Syntactic properties
- consistency
- closed term computation
- decidable type checking

Swedish philosophy
Requirements of type theoretic tools

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

Syntactic properties
- consistency
- closed term computation
- decidable type checking

programming
Requirements of type theoretic tools

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

Syntactic properties
- consistency
- closed term computation
- decidable type checking

program logic
Requirements of type theoretic tools

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

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

**general mathematics**

*(all models)*

*(just the syntax)*
Requirements of type theoretic tools

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

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 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.

---

\(^1\)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)
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 / \( \boxdot \text{TT} \) ?

\( \boxdot \text{TT} \) 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 \( \mathbb{I} \).

\[
(a_0 =_A a_1) := \{ p : \mathbb{I} \to A \mid p(0) \equiv a_0 \land p(1) \equiv a_1 \}
\]

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

---

\(^2\) Univalent Foundations Program (2013)
Computation in $\textit{TT}$: prior art

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

**Theorem (Cubical canonicity)**

If $\vec{i} : I^n \vdash M(\vec{i}) : \text{bool}$ is a closed $n$-cube of booleans, then either

$\vec{i} : I^n \vdash M(\vec{i}) \equiv \text{tt} : \text{bool}$ or $\vec{i} : I^n \vdash M(\vec{i}) \equiv \text{ff} : \text{bool}$.

Hence $\textit{TT}$ is programming language.

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

I have proved the following suite of results for $\text{TT}$ 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 $\text{TT}$ a unique normal form.

**Corollary (Decidability of equality)**

Judgmental equality $\Gamma \vdash A \equiv B$ and $\Gamma \vdash a \equiv b : A$ in $\text{TT}$ 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 $\text{TT}$ without universes is j.w.w. Angiuli published in LICS'21 (Sterling and Angiuli, 2021).
2. Synthetic Tait computability
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.
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 $\leftrightarrow^*$ on raw closed terms.

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$\begin{align*}
[\text{bool}](b) &:= (b \leftrightarrow^* \text{tt} \lor b \leftrightarrow^* \text{ff}) \\
[A \rightarrow B](f) &:= (\forall x : A. [A](x) \rightarrow [B](f(x)))
\end{align*}$$
First define operational semantics $\mapsto^*$ on raw closed terms.

**Example (Canonicity)**

To prove canonicity, we choose the following predicates:

\[
\begin{align*}
\text{[bool]}(b) & := (b \mapsto^* \text{tt} \lor b \mapsto^* \text{ff}) \\
\text{[A \to B]}(f) & := (\forall x : A. \text{[A]}(x) \to \text{[B]}(f(x)))
\end{align*}
\]

**Q1:** given a type $A$, what is the *domain* of $\text{[A]}$? closed terms, open terms, typed, ??
Operational Tait computability

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

Example (Canonicity)

To prove canonicity, we choose the following predicates:

$$[\text{bool}](b) := (b \rightarrow^* \text{tt} \lor b \rightarrow^* \text{ff})$$
$$[A \rightarrow B](f) := (\forall x : A.[A](x) \rightarrow [B](f(x)))$$

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

**Example (Canonicity)**

To prove canonicity, we choose the following predicates:

- $\text{[bool]}(b) := (b \mapsto^* \texttt{tt} \lor b \mapsto^* \texttt{ff})$
- $\text{[A \to B]}(f) := (\forall x : A. \text{[A]}(x) \to \text{[B]}(f(x)))$

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

**Q2:** what properties must $\text{[A]}$ satisfy? closure under subst., ren., head expansion, ??

**Q3:** does our proof actually depend on the chosen transition relation $\mapsto^*$?
First define operational semantics \( \rightsquigarrow^* \) on raw closed terms.

**Example (Canonicity)**

To prove canonicity, we choose the following predicates:

\[
\begin{align*}
[\text{bool}](b) &:= (b \rightsquigarrow^* \text{tt} \lor b \rightsquigarrow^* \text{ff}) \\
[A \rightarrow B](f) &:= (\forall x : A. [A](x) \rightarrow [B](f(x)))
\end{align*}
\]

**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 \( \rightsquigarrow^*? 

**Q4:** why are the predicates attached to connectives \((\rightarrow, \times, \ldots)\) 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:

$$[\text{bool}](b) := (b \mapsto^* \text{tt} \lor b \mapsto^* \text{ff})$$
$$[A \to B](f) := (\forall x : A.[A](x) \to [B](f(x)))$$

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 ($\to$, $\times$, ... ) 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 \(\mathbb{C}^{\mathbb{C}}\) 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?*
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.

<table>
<thead>
<tr>
<th>analytic</th>
<th>synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>geometry</td>
<td>the Cartesian plane, $\mathbb{R}^n$</td>
</tr>
<tr>
<td>metatheory</td>
<td>logical relations, Artin gluing</td>
</tr>
<tr>
<td></td>
<td>Euclid’s postulates</td>
</tr>
<tr>
<td></td>
<td>STC</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>analytic</th>
<th>synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>geometry</td>
<td>the Cartesian plane, $\mathbb{R}^n$</td>
</tr>
<tr>
<td>metatheory</td>
<td>logical relations, Artin gluing</td>
</tr>
</tbody>
</table>

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.

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

---

⁸cf. 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.

(Smooth to develop and use if we generalize to computability structures, i.e. proof-relevant computability predicates.)

e.g. the computability structure of the booleans:

\[
[\text{bool}] := (x : \text{bool}) \times \begin{cases} x = \text{tt} \\ x = \text{ff} \end{cases}
\]

\footnote{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 syntax and semantics. These can be mixed and matched, but the satisfy some laws:

- Both $\bot$ and $\top$ 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 syntax and semantics. These can be mixed and matched, but the satisfy some laws:

- Both $\mathcal{C}$ and $\mathcal{D}$ are lex idempotent monads.\footnote{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).}
- Complementarity: semantic things are syntactically trivial, i.e. $\mathcal{A} \approx \text{unit but not the other way around.}$
Piecing together syntax and semantics

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

- Both − and − are lex idempotent monads.\(^9\)
- Complementarity: semantic things are syntactically trivial, i.e. \(A \simeq \text{unit but not the other way around.}\)
- Fracture: any computability structure \(A\) can be reconstructed from \(A\), \(\overline{A}\), and \(A\).

\[^9\text{They are open and closed modalities in the sense of topos theory (Artin, Grothendieck, and Verdier, 1972; Mac Lane and Moerdijk, 1992; Rijke, Shulman, and Spitters, 2020).}\]
The language of synthetic Tait computability

Definition

**STC** = type theory + modalities $\square / \neg \neg$ that behave as above.
The language of synthetic Tait computability

Definition

\[ \text{STC} = \text{type theory} + \text{modalities} \quad \square / \quad \bar{\square} \quad \text{that behave as above.} \]

Equivalently, extend type theory by a generic proposition \( \square : \text{Prop} \) and define

\[ A := A\square \quad \text{and} \quad A := A \cup_{A \times \bar{\square}} \bar{\square}. \]

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

\[
\mathbb{[bool]} = (x : \text{bool}) \times (x = \text{tt} + x = \text{ff})
\]
Example: synthetic computability structure of the booleans

\[
\mathbb{[bool]} = (x : \text{bool}) \times (x = tt + x = ff)
\]

The syntactic part of \(\mathbb{[bool]}\) is the syntactic booleans.
Example: synthetic computability structure of the booleans

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

The syntactic part of \[\text{[bool]}\] is the syntactic booleans.
Example: synthetic computability structure of the booleans

\[[\text{bool}] = (x : \text{bool}) \times (x = \text{tt} + x = \text{ff})\]

The syntactic part of \[[\text{bool}]\] is the syntactic booleans.
Example: synthetic computability structure of the booleans

$$[[\text{bool}]] = (x : \text{bool}) \times (x = \text{tt} + x = \text{ff})$$

The syntactic part of $[[\text{bool}]]$ is the syntactic booleans.
3. From the general to the particular...
In what contexts do we compute?

\[\Psi \models M : A \Downarrow V\]
In what contexts do we compute?

\[ \Psi \vdash M : A \Downarrow V \]

- **Canonicity**:
  - \( \psi \in \{ \text{nat} \} \)
  - \( \Gamma \in \{ \text{\cdot} \} \)
  - \( \Gamma \in \{ \text{In} \mid n \in \mathbb{N} \} \)

- **Normalization**:
  - \( \psi \in \{ \Psi \vdash \beta \eta \text{nf} \ A \} \)
  - \( \Gamma \in \{ \vdash \text{ctx} \} \)
In what contexts do we compute?

- **Element:**
  - $\psi \models M : A \Downarrow V$

- **Observation:**
  - $\psi \models M : A \Downarrow V$

**Canonicity:**
- $A \in \{\text{nat}\}$
- $V \in \mathbb{N}$
- $\Gamma \in \{\cdot\}$
- Cubical canonicity: $\Gamma \in \{I | n \in \mathbb{N}\}$

**Normalization:**
- $A \in \{\psi \vdash \text{type}\}$
- $V \in \{\psi \vdash \beta\eta\text{nf}\ A\}$
- $\Gamma \in \{\vdash \text{ctx}\}$
In what contexts do we compute?

canonicity: $A \in \{\text{nat}\}$; normalization: $A \in \{\Psi \vdash \text{type}\}$

canonicity: $V \in \mathbb{N}$; normalization: $V \in \{\Psi \vdash \beta\eta\text{nf}A\}$

canonicity: $\Gamma \in \{\cdot\}$; cubical canonicity: $\Gamma \in \{\text{In} | n \in \mathbb{N}\}$

context

element

observation
In what contexts do we compute?

**canonicity:** $A \in \{\text{nat}\}$; **normalization:** $A \in \{\Psi \vdash \text{type}\}$
In what contexts do we compute?

- **canonicity:** $A \in \{\text{nat}\}$; **normalization:** $A \in \{\Psi \vdash \text{type}\}$

- **canonicity:** $V \in \mathbb{N}$; **normalization:** $V \in \{\Psi \vdash^{\beta \eta} \text{nf} \ A\}$
In what contexts do we compute?

- **canonicity**: $A \in \{\text{nat}\}$; **normalization**: $A \in \{\Psi \vdash \text{type}\}$

- **canonicity**: $V \in \mathbb{N}$; **normalization**: $V \in \{\Psi \vdash^{\beta\eta\text{nf}} A\}$

- **canonicity**: $\Gamma \in \{\cdot\}$; **cubical canonicity**: $\Gamma \in \{I^n \mid n \in \mathbb{N}\}$; **normalization**: $\Gamma \in \{\vdash \text{ctx}\}$
Stability (or lack thereof) of observation

\[
\begin{array}{c}
\vdash \ x : \text{nat} \\
\downarrow \\
\text{var}(x)
\end{array}
\]

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).
Stability (or lack thereof) of observation

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).
Stability (or lack thereof) of observation

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 $R$ of contexts and structural renamings (weakening, swapping, contraction).
Stability (or lack thereof) of observation

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).
Stability (or lack thereof) of observation

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

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 $\text{TT}$?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for $\text{TT}$. The problem lies with the interval:

$\text{fib} : \text{nat} \to \text{nat}$

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

\[
p : \text{fib} =_{\text{nat} \rightarrow \text{nat}} \text{fib}, \ i : \mathbb{I} \vdash (p \circ i) \, 9 : \text{nat} \quad \Downarrow \quad \text{app(pathapp(var(p), i), su}^9(\text{ze}))
\]
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:

\[
p : \text{fib} \to_{\text{nat}} \text{fib}, \ i : \text{II} \vdash (p @ i) \ 9 : \text{nat} \Downarrow \text{app(pathapp(var(p), i), su^9(ze))}
\]
What goes wrong for $\Box\mathbb{T}\mathbf{T}$?

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for $\Box\mathbb{T}\mathbf{T}$. The problem lies with the interval:

\[
p : \text{fib} =_{\text{nat} \rightarrow \text{nat}} \text{fib}, \ i : \ \emptyset \vdash (p \circ i) \ 9 : \text{nat} \Downarrow \text{app(pathapp(var(p), i), su}^9(\text{ze}))
\]

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

\[ p : \text{fib} =_{\text{nat} \rightarrow \text{nat}} \text{fib}, i : I \vdash (p @ i) \, 9 : \text{nat} \Downarrow \text{app}(\text{pathapp}(\text{var}(p), i), \text{su}^9(ze)) \]

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

Unfortunately, just removing the substitutions for which neutral observations are unstable is not practicable for $\mathbb{TT}$. The problem lies with the interval:

$\vdash (p \circ i) \cdot 9 : \text{nat} \Downarrow \text{app(pathapp(var}(p), i), \text{su}^9(\text{ze}))$

We shouldn't remove $[0/i], [1/i]$ from the category of contexts and renamings because we need $\text{I}$ to restrict to something \textit{representable} in $\text{Pr}(\mathcal{R})$, \textit{c.f.} \textit{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 $\partial E$ 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 $\partial E$ for this frontier of instability:

$$\partial(\text{var}(x)) = \bot$$
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 $\partial E$ for this frontier of instability:

$$
\partial(\text{var}(x)) = \perp \\
\partial(\text{app}(E, M)) = \partial E
$$
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 $\partial E$ for this frontier of instability:

\[
\begin{align*}
\partial(\text{var}(x)) &= \perp \\
\partial(\text{app}(E, M)) &= \partial E \\
\partial(\text{fst}(E)) &= \partial E
\end{align*}
\]
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 \( \partial E \) for this *frontier of instability*:

\[
\begin{align*}
\partial(\text{var}(x)) &= \bot \\
\partial(\text{app}(E, M)) &= \partial E \\
\partial(\text{fst}(E)) &= \partial E \\
\partial(\text{pathapp}(E, r)) &= \partial E \lor (r = 0) \lor (r = 1)
\end{align*}
\]
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 $\partial E$ for this frontier of instability:

\[
\begin{align*}
\partial(\text{var}(x)) &= \bot \\
\partial(\text{app}(E, M)) &= \partial E \\
\partial(\text{fst}(E)) &= \partial E \\
\partial(\text{pathapp}(E, r)) &= \partial E \lor (r = 0) \lor (r = 1)
\end{align*}
\]

Therefore we define an inductive family $\text{Ne}_\phi(A)$ with $\text{Ne}_\phi(A) \cong A$ comprised of neutrals $e$ with $\partial e = \phi$. Traditional neutrals $\text{Ne}_\bot(A)$; to model destabilization, $\text{Ne}_\top(A) \cong A$. 
Normalization via Tait’s yoga

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

\[ \text{Ne}(A) \subseteq [A] \subseteq \text{Nf}(A) \]
Normalization via Tait’s yoga

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

Normalization via Tait’s yoga

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

\[
\text{Ne}(A) \xrightarrow{\uparrow A} [A] \xleftarrow{\downarrow A} \text{Nf}(A)
\]

“reflection”

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

\[ \text{Ne}(A) \xrightarrow{\uparrow A} [A] \xrightarrow{\downarrow A} \text{Nf}(A) \]

"reification"

---

Tait (1967) introduced the famous *saturation yoga* for normalization:\(^\text{10}\)

\[ \text{Ne}(A) \xrightarrow{A} [A] \xleftarrow{A} \text{Nf}(A) \]

---

Yogic injury: unstable neutrals

What if $\phi = \top$?

We must strengthen the "induction hypothesis".
Yogic injury: unstable neutrals

What if $\phi = T$?

- $\text{Ne}_\phi(A)$
- $\text{[A]}$
- $\text{Nf}(A)$

We must strengthen the "induction hypothesis".
Yogic injury: unstable neutrals

What if $\phi = T$?
Yogic injury: unstable neutrals

What if $\phi = T$?
Yogic injury: unstable neutrals

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.

\[
\text{Ne}_\phi(A) \bowtie \phi \lbrack [A] \rbrack \quad \text{Ne}_\phi(A)
\]

\[
\phi \Rightarrow [A] \quad \phi \Rightarrow A
\]
Stabilization of neutrals

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

Stabilization **interpolates** between neutrals and computability data.
A spectrum of computability data

Stabilization interpolates between neutrals and computability data.
A spectrum of computability data

Stabilization interpolates between neutrals and computability data.
A spectrum of computability data

Stabilization interpolates between neutrals and computability data.
The stabilized Tait yoga

\[ \text{Lemma (Saturation)} \]

Every type of \( TT \) is closed under the stabilized Tait yoga.
The stabilized Tait yoga

Lemma (Saturation)

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

Lemma (Saturation)

Every type of $\square \text{TT}$ is closed under the stabilized Tait yoga.
**Summary of results**

<table>
<thead>
<tr>
<th>Lemma (Saturation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every type of $\boxdot \text{TT}$ is closed under the stabilized Tait yoga.</td>
</tr>
</tbody>
</table>

The above is employed to obtain our main results:

<table>
<thead>
<tr>
<th>Theorem (Normalization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is a computable function assigning to every type $\Gamma \vdash A$ and every term $\Gamma \vdash a : A$ of $\boxdot \text{TT}$ a unique normal form.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corollary (Decidability of equality)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Judgmental equality $\Gamma \vdash A \equiv B$ and $\Gamma \vdash a \equiv b : A$ in $\boxdot \text{TT}$ is decidable.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corollary (Injectivity of type constructors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $\Gamma \vdash \prod(A, B) \equiv \prod(A', B')$ then $\Gamma \vdash A \equiv A'$ and $\Gamma, x : A \vdash B(x) \equiv B'(x)$.</td>
</tr>
</tbody>
</table>
4. Taking stock
The community designed \( \text{COTT} \) with the explicit aim of finding a computational version of homotopy type theory. We consider the first chapter finally closed:
The community designed $\mathbb{TT}$ 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 $\mathbb{TT}$ 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).
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).
The community designed \( \boxdot \text{TT} \) 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.

<table>
<thead>
<tr>
<th>program modules</th>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical relations</td>
<td>syntax</td>
<td>semantics</td>
</tr>
<tr>
<td>type refinements</td>
<td>computation</td>
<td>specification</td>
</tr>
<tr>
<td>resource analysis</td>
<td>behavior</td>
<td>complexity</td>
</tr>
<tr>
<td>security / IFC</td>
<td>public</td>
<td>classified</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>program modules</th>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical relations</td>
<td>syntax</td>
<td>semantics</td>
</tr>
<tr>
<td>type refinements</td>
<td>computation</td>
<td>specification</td>
</tr>
<tr>
<td>resource analysis</td>
<td>behavior</td>
<td>complexity</td>
</tr>
<tr>
<td>security / IFC</td>
<td>public</td>
<td>classified</td>
</tr>
</tbody>
</table>


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.

<table>
<thead>
<tr>
<th>program modules</th>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical relations</td>
<td>syntax</td>
<td>semantics</td>
</tr>
<tr>
<td>type refinements</td>
<td>computation</td>
<td>specification</td>
</tr>
<tr>
<td>resource analysis</td>
<td>behavior</td>
<td>complexity</td>
</tr>
<tr>
<td>security / IFC</td>
<td>public</td>
<td>classified</td>
</tr>
</tbody>
</table>

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.

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


References II


References IV


References V


References VI


References VII

