Objective Metatheory of (Cubical) Type Theory

Jonathan Sterling

August 31, 2020

The implementation and semantics of dependent type theories can be studied in a syntax-independent way. *Using the semantic techniques of the* ***objective metatheory****, type theorists can obtain succinct and conceptual proofs of formerly intractable results.*

Who is a type theorist?

Who is a type theorist?

Someone who studies the *non-type-theoretic aspects* of type theory.

Who is a type theorist?

Someone who studies the *non-type-theoretic aspects* of type theory.

???

Definition. A statement is called *type theoretic* when it is preserved by morphisms of models of type theory: these hold in all models iff they hold in the syntax of type theory.

non-ty
of type the

Definition. A statement is called *type theoretic* when it is preserved by morphisms of models of type theory: these hold in all models iff they hold in the syntax of type theory.

Examples. Anything expressible as a judgment of type theory; for instance, “There is a function that computes the gcd.”

non-ty
of type the

?

Definition. A statement is called *type theoretic* when it is preserved by morphisms of models of type theory: these hold in all models iff they hold in the syntax of type theory.

Examples. Anything expressible as a judgment of type theory; for instance, “There is a function that computes the gcd.”

Non-examples. “Every closed term of type $\mathbb{N}$ is equal to a numeral”;

Definition. A statement is called *type theoretic* when it is preserved by morphisms of models of type theory: these hold in all models iff they hold in the syntax of type theory.

Examples. Anything expressible as a judgment of type theory; for instance, “There is a function that computes the gcd.”

Non-examples. “Every closed term of type $\mathbb{N}$ is equal to a numeral”; “These two queue implementations are observationally equivalent”;

Definition. A statement is called *type theoretic* when it is preserved by morphisms of models of type theory: these hold in all models iff they hold in the syntax of type theory.

Examples. Anything expressible as a judgment of type theory; for instance, “There is a function that computes the gcd.”

Non-examples. “Every closed term of type $\mathbb{N}$ is equal to a numeral”; “These two queue implementations are observationally equivalent”; “It is decidable whether two types are judgmentally equal.”

DERIVABILITY

About the syntax of type theory. *But, by soundness, also about any model of type theory.*

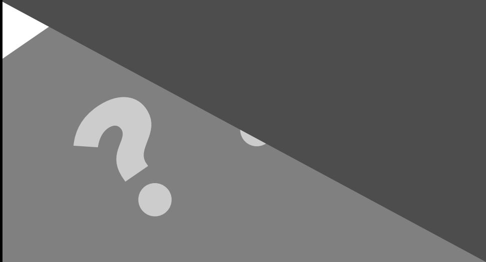
DEDUCTION

$$\frac{\Gamma \vdash A \equiv A' \ \ \mathit{type} \qquad \Gamma \vdash B \equiv B' \ \ \mathit{type}}{\Gamma \vdash (A \to B) \equiv (A' \to B') \ \ \mathit{type}}$$

ype theoretic when it is pre-
of type theory: these hold in all
of type theory.

s a judgment of type theory; for
computes the gcd."

osed term of type $\mathbb{N}$ is equal to
wo queue implementations
ally equivalent"; "It is decidable
o types are judgmentally equal."



DERIVABILITY

About the syntax of type theory.
But, by soundness, also about any model of type theory.

DEDUCTION

$$\frac{\Gamma \vdash A \equiv A' \ \mathit{type} \qquad \Gamma \vdash B \equiv B' \ \mathit{type}}{\Gamma \vdash (A \to B) \equiv (A' \to B') \ \mathit{type}}$$

ADMISSIBILITY

About the syntax of type theory.
(Only the syntax of type theory!)

INVERSION

$$\frac{\Gamma \vdash (A \to B) \equiv (A' \to B') \ \mathit{type}}{\Gamma \vdash A \equiv A' \ \mathit{type} \qquad \Gamma \vdash B \equiv B' \ \mathit{type}}$$

DERIVABILITY

About the syntax of type theory.
But, by soundness, also about any model of type theory.

DEDUCTION

$$\frac{\Gamma \vdash A \equiv A' \ \mathit{type} \qquad \Gamma \vdash B \equiv B' \ \mathit{type}}{\Gamma \vdash (A \to B) \equiv (A' \to B') \ \mathit{type}}$$

ADMISSIBILITY

About the syntax of type theory.
(Only the syntax of type theory!)

INVERSION

$$\begin{array}{c}\Gamma \vdash (A \to B) \equiv (A' \to B') \ \mathit{type} \\ \hdashline \Gamma \vdash A \equiv A' \ \mathit{type} \qquad \Gamma \vdash B \equiv B' \ \mathit{type}\end{array}$$

Why do we care?

Admissibilities like this are why it is even possible to implement type checkers!

Most metatheorems important for implementation are consequences of *normalization*.

Even for basic type theory, normalization is hard to prove rigorously: 100-200 pages(*) of single-use technical lemmas that seem to have nothing to do with the matter at hand.

Someone should stop us

from making new type theories

Wouldn’t be so bad if we only had to do it once; but type theory is in its infancy and we keep making better ones.

Someone should stop us

from making new type theories

Wouldn’t be so bad if we only had to do it once; but type theory is in its infancy and we keep making better ones.
Today: cubical type theory.

Why not stick with ITT, ETT, or Nuprl?

Why not stick with ITT, ETT, or Nuprl?

Function extensionality and **quotient types** either break computation or they break type checking.

Why not stick with ITT, ETT, or Nuprl?

Function extensionality and **quotient types** either break computation or they break type checking.

Worse: well-behaved notions of **equivalence relation** and **quotient type** are inconsistent with standard(†) PER semantics of type theory. ***Non-starter for mathematical applications!***

Why not stick with ITT, ETT, or Nuprl?

Function extensionality and **quotient types** either break computation or they break type checking.

Worse: well-behaved notions of **equivalence relation** and **quotient type** are inconsistent with standard(†) PER semantics of type theory. ***Non-starter for mathematical applications!***

Universes don't have universal properties (unlike every other type connective).

Why not stick with ITT, ETT, or Nuprl?

Function extensionality and **quotient types** either break computation or they break type checking.

Worse: well-behaved notions of **equivalence relation** and **quotient typ**… R semantics of type theo… ***ications!***

All these are solved by cubical type theory (*).

Universes d… every other type connective).

(*) Angiuli, Hou (Favonia), and Harper [AHH17], Angiuli, Brunerie, Coquand, Hou (Favonia), Harper, and Licata [Ang+17], Awodey [Awo18a], Cohen, Coquand, Huber, and Mörtberg [Coh+17], Huber [Hub18], and Orton and Pitts [OP16], …

Implementations of cubical type theory!

Several variants of cubical type theory have been implemented.

Gen.	Style	Implementation
0	evaluator	`cubical`
1	typechecker PRL	`cubicaltt`, `yacctt` **RedPRL**
2	proof assistant	Cubical Agda, `redtt`, `cooltt`

[Ang+18b; Coh+15; Coh+18; MA18; Red18; Red20; VMA19]

Ours: **RedPRL**, `redtt`, `cooltt`.

Our implementations: RedPRL, redtt, cooltt

Each implementation was tied to a scientific experiment!

	Premise	Result
RedPRL	The PRL methodology benefits HTT implementation.	PRL impedes HTT implementation.
redtt	Interactive cubical refinement + decidable(?) jdg.eq. increases usability.	Confirmed.
cooltt	LF formulation + more extensional equality on $\mathbb{F}$ & systems suitable for efficient implementation.	Early positive indications.

My contributions: generalization of interactive proof with holes to account for cubical boundaries; more efficient algorithms for cubical evaluation.

What is cubical type theory?

An extension of Martin-Löf type theory!

1. Interval object $\mathbb{I}$, classifying "dimensions":

$$\frac{}{0 : \mathbb{I}} \qquad \frac{}{1 : \mathbb{I}} \qquad \frac{0 \equiv 1 : \mathbb{I}}{\mathcal{J}}$$

   Idea: A term $a : \prod_{i:\mathbb{I}} A(i)$ is an *identification* between $a(0)$ and $a(1)$.

2. Universe $\mathbb{F}$ of propositions closed under at least:
   - extensional equality $(r =_{\mathbb{I}} s)$ of dimensions $r, s : \mathbb{I}$
   - conjunction $\phi \wedge \psi$ and (extensional) disjunction $\phi \vee \psi$
   - universal quantification over the interval $\forall i : \mathbb{I}.\phi(i)$

   Idea: A term $a : \phi \to A$ is a *partial element* of A, defined only when ϕ is true.

New computations! :-)

Cubical type theory extends MLTT with new generic functions

$$\frac{A : \mathbb{I} \to \mathcal{U} \qquad \phi : \mathbb{F} \qquad r, s : \mathbb{I} \qquad f : \prod_{i:\mathbb{I}} \prod_{p:(i =_{\mathbb{I}} r) \vee \phi} A(i)}{\mathbf{com}_A^{r \rightsquigarrow s}(f) : \{\tilde{f}_s : A(s) \mid \forall p. \tilde{f}_s = f(s, p)\}}$$

Coercion/transport, symmetry, and transitivity are all special cases of com_A.

Resulting theory of equality much easier to use than ITT+J!

New computations! :-(

New computation rules branch on *non-equality* of r, s as well as the the value of $A(i)$ for a fresh dimension $i : \mathbb{I}$.

New computations! :-(

New computation rules branch on *non-equality* of r, s as well as the the value of $A(i)$ for a fresh dimension $i : \mathbb{I}$.

- Neither $M \ \mathsf{val}$ nor $M \mapsto^* N$ closed under substitution!

New computations! :-(

New computation rules branch on *non-equality* of r, s as well as the the value of $A(i)$ for a fresh dimension $i : \mathbb{I}$.

- Neither $M \text{ val}$ nor $M \mapsto^* N$ closed under substitution!
- Classic head expansion principle for operational model doesn't make sense, needs to be generalized significantly.

New computations! :-(

New computation rules branch on *non-equality* of r, s as well as the the value of $A(i)$ for a fresh dimension $i : \mathbb{I}$.

- Neither M val nor $M \mapsto^* N$ closed under substitution!
- Classic head expansion principle for operational model doesn't make sense, needs to be generalized significantly.

$\implies$ Operational metatheory very hard: possible, with great exertion, using *coherent expansion* principle contributed independently by Angiuli [Ang19] and Huber [Hub18].

New computations! :-(

New computation rules branch on *non-equality* of r, s as well as the the value of $A(i)$ for a fresh dimension $i : \mathbb{I}$.

- Neither M val nor $M \mapsto^* N$ closed under substitution!
- Classic head expansion principle for operational model doesn't make sense, needs to be generalized significantly.

$\implies$ Operational metatheory very hard: possible, with great exertion, using *coherent expansion* principle contributed independently by Angiuli [Ang19] and Huber [Hub18].

Canonicity at the limits of operational tractability: normalization will require new techniques (this thesis!).

New computations! (-:

Simpler alternative to operational semantics + PERs with coherent expansion: **equational theory + Artin gluing**, as proposed by Awodey in 2015.

> S., Angiuli, Gratzer (2019). **"Cubical Syntax for Reflection-Free Extensional Equality."** FSCD 2019.

S., Angiuli, and Gratzer [SAG19] present an **easy and complete** proof of canonicity for a version of cubical type theory in less than 30 pages. *Trial run of the "objective metatheory."*

What is the objective metatheory?

objective metatheory = local invariance + global invariance + proof relevance.

What is the objective metatheory?

objective metatheory = local invariance + global invariance + proof relevance.

1. **local invariance:**
   - raw syntax / op-sem $\Longrightarrow$ typed syntax in equational LF
   - work invariantly over equiv. classes of typed terms

What is the objective metatheory?

objective metatheory = local invariance + global invariance + proof relevance.

1. **local invariance:**
   - raw syntax / op-sem $\Longrightarrow$ typed syntax in equational LF
   - work invariantly over equiv. classes of typed terms
2. **global invariance:**
   - “objective syntax” defined only up to categorical equivalence
   - all statements respect weak equivalence of categories: freedom to choose presentations at will

What is the objective metatheory?

objective metatheory = local invariance + global invariance + proof relevance.

1. **local invariance:**
   - raw syntax / op-sem $\Longrightarrow$ typed syntax in equational LF
   - work invariantly over equiv. classes of typed terms
2. **global invariance:**
   - “objective syntax” defined only up to categorical equivalence
   - all statements respect weak equivalence of categories: freedom to choose presentations at will
3. **proof relevance:**
   - “property of raw syntax” $\Longrightarrow$ “structure over objective syntax”
   - generalization proof-relevant logical relations is forced!

Local invariance of objective syntax

Syntax of type theory with function types expressed as an equational LF signature:[1]

$$\mathbf{Tp} : \mathbf{Kind} \qquad \mathbf{Tm} : \mathbf{Tp} \to \mathbf{Type} \qquad \mathbf{Fn} : \mathbf{Tp} \times \mathbf{Tp} \to \mathbf{Tp}$$

$$\alpha_{\mathbf{Fn}} : \textstyle\prod_{A,B:\mathbf{Tp}} \mathbf{Tm}(\mathbf{Fn}(A,B)) \cong (\mathbf{Tm}(A) \to \mathbf{Tm}(B))$$

Above: introduction, elimination, computation, and uniqueness rules bundled in $\alpha_{\mathbf{Fn}}$.

Local invariance: impossible to utter a distinction between judgmentally equal terms. (Anti-bureaucratic power move!)

[1] Equational LF due to Uemura [Uem19] with universes $\mathbf{Type} \subseteq \mathbf{Kind}$; kinds closed under dependent products along types as in Harper, Honsell, and Plotkin [HHP93].

Global invariance of objective syntax

$$\mathbf{Tp} : \mathbf{Kind} \qquad \mathbf{Tm} : \mathbf{Tp} \to \mathbf{Type} \qquad \mathbf{Fn} : \mathbf{Tp} \times \mathbf{Tp} \to \mathbf{Tp}$$
$$\alpha_{\mathbf{Fn}} : \textstyle\prod_{A,B:\mathbf{Tp}} \mathbf{Tm}(\mathbf{Fn}(A, B)) \cong (\mathbf{Tm}(A) \to \mathbf{Tm}(B))$$

The signature Σ above involves a *choice* of LF encoding, but metatheorems don't depend on how we set up the function type (which is uniquely determined up to iso).

Global invariance: Σ presents a *classifying category* $\mathcal{C}_\Sigma$ of judgments and deductions, which we work with *up to weak equivalence of categories*.

Proof relevance in the objective metatheory

Let’s remember how logical relations work…

(Unary) logical relations on closed terms is: for each sort A *sort*$_\Sigma$, a subset $\tilde{A} \subseteq \{a \mid \cdot \vdash_\Sigma a : A\}/\equiv_\Sigma$ respecting all the operations of Σ.

(Unary) logical relations on closed terms is: for each sort $A : \mathcal{C}_\Sigma$, a subset $\tilde{A} \subseteq \mathsf{Hom}_{\mathcal{C}_\Sigma}(\cdot, A)$ respecting all the operations of Σ.

(Unary) logical relations on closed terms is: for each sort $A : \mathcal{C}_\Sigma$, a subset $\tilde{A} \subseteq \rho(A)$ respecting all the operations of Σ, where $\rho : \mathcal{C}_\Sigma \longrightarrow \mathbf{Set}$ is the global sections functor.

(Unary) logical relations on closed terms is: for each sort $A : \mathcal{C}_\Sigma$, a subset $\tilde{A} \subseteq \rho(A)$ respecting all the operations of Σ, where $\rho : \mathcal{C}_\Sigma \longrightarrow \mathbf{Set}$ is the global sections functor.

Elements $a \in \tilde{A}$ are called *computable*.

(Unary) logical relations on closed terms is: for each sort $A : \mathcal{C}_\Sigma$, a subset $\tilde{A} \subseteq \rho(A)$ respecting all the operations of Σ, where $\rho : \mathcal{C}_\Sigma \longrightarrow \mathbf{Set}$ is the global sections functor.

Elements $a \in \tilde{A}$ are called *computable*.

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by *Artin gluing*.

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by *Artin gluing*.

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by *Artin gluing*.

A morphism $(B, \tilde{B}) \longrightarrow (A, \tilde{A}) : \mathcal{G}_\Sigma$ is a term/deduction $a : B \longrightarrow A$ that preserves computability, i.e. sends closed terms in $\tilde{B}$ to closed terms in $\tilde{A}$.

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by *Artin gluing*.

A morphism $(B, \tilde{B}) \longrightarrow (A, \tilde{A}) : \mathcal{G}_\Sigma$ is a term/deduction $a : B \longrightarrow A$ that preserves computability, i.e. sends closed terms in $\tilde{B}$ to closed terms in $\tilde{A}$.

Thinking of subsets as injective functions:

$$\begin{tikzcd}
\tilde{B} \arrow[r] \arrow[d, hook] & \tilde{A} \arrow[d, hook] \\
\rho(B) \arrow[r, "\rho(a)"'] & \rho(A)
\end{tikzcd}$$

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by ***Artin gluing***.

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by ***Artin gluing***.

$$\begin{tikzcd}
\mathcal{G}_\Sigma \arrow[r, dotted] \arrow[d, dotted] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Sub}(\mathbf{Set}) \arrow[d] \\
\mathcal{C}_\Sigma \arrow[r, "\rho"'] & \mathbf{Set}
\end{tikzcd}$$

Logical relations $(A, \tilde{A} \subseteq \rho(A))$ have the structure of a category $\mathcal{G}_\Sigma$ defined by ***Artin gluing***.

$$\begin{tikzcd}
\mathcal{G}_\Sigma \arrow[r, dotted] \arrow[d, dotted] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Sub}(\mathbf{Set}) \arrow[d] \\
\mathcal{C}_\Sigma \arrow[r, "\rho"'] & \mathbf{Set}
\end{tikzcd}$$

Fundamental Theorem of Logical Relations: prove that $\mathcal{G}_\Sigma$ is a model of the theory Σ!

FTLR Idea: interpret each type into $\mathcal{G}_\Sigma$ in such a way that an element carries the proof of the desired metatheorem, e.g. canonicity at base type:

$$\begin{tikzcd}
\mathcal{G}_\Sigma \arrow[r, dotted] \arrow[d, dotted] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Sub}(\mathbf{Set}) \arrow[d] \\
\mathcal{C}_\Sigma \arrow[r, "\rho"'] & \mathbf{Set}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{\mathbf{yes}, \mathbf{no}\} \arrow[d, hook, "\mathcal{G}_\Sigma[\![\mathbf{ans}]\!]"] \\
\rho(\mathbf{ans})
\end{tikzcd}$$

Connectives that have β/η laws are uniquely determined! No need (or ability) to be clever.

Proof-relevant logical relations!

What is the logical relation for the collection of types $\mathbf{Tp}$?
Ought to be a “relation of relations”, but that’s nonsense.

Proof-relevant logical relations!

What is the logical relation for the collection of types $\mathbf{Tp}$?
Ought to be a “relation of relations”, but that’s nonsense.

Old solution: parameterize in complex inductive definition, prove dozens of technical lemmas [All87]

Proof-relevant logical relations!

What is the logical relation for the collection of types $\mathbf{Tp}$? Ought to be a "relation of relations", but that's nonsense.

Old solution: parameterize in complex inductive definition, prove dozens of technical lemmas [All87]

New solution: generalize from *property* to *structure*[2]

[1]Application to universes due to Shulman [Shu15], developed further by Coquand [Coq19]; main ingredients invented by the Grothendieck school in the 1970s [AGV72].

Proof-relevant logical relations!

What is the logical relation for the collection of types $\mathbf{Tp}$? Ought to be a “relation of relations”, but that’s nonsense.

Old solution: parameterize in complex inductive definition, prove dozens of technical lemmas [All87]

New solution: generalize from *property* to *structure*[2]

$$\begin{tikzcd}
\mathcal{G}_\Sigma \arrow[r, dotted] \arrow[d, dotted] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Sub}(\mathbf{Set}) \arrow[d] \\
\mathcal{C}_\Sigma \arrow[r, "\rho"'] & \mathbf{Set}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
??? \arrow[d, hook, "\mathcal{G}_\Sigma[\![\mathbf{Tp}]\!]"] \\
\rho(\mathbf{Tp})
\end{tikzcd}$$

[1]Application to universes due to Shulman [Shu15], developed further by Coquand [Coq19]; main ingredients invented by the Grothendieck school in the 1970s [AGV72].

Proof-relevant logical relations!

What is the logical relation for the collection of types $\mathbf{Tp}$? Ought to be a "relation of relations", but that's nonsense.

Old solution: parameterize in complex inductive definition, prove dozens of technical lemmas [All87]

New solution: generalize from *property* to *structure*[2]

$$\begin{tikzcd}
\mathcal{G}_\Sigma \arrow[r, dotted] \arrow[d, dotted] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Fam}(\mathbf{Set}) \arrow[d] \\
\mathcal{C}_\Sigma \arrow[r, "\rho"'] & \mathbf{Set}
\end{tikzcd}
\qquad
\begin{tikzcd}
\sum_{A:\rho(\mathbf{Tp})} (\rho(\mathbf{Tm})(A) \to \mathbf{Set}_{small}) \arrow[d, "\mathcal{G}_\Sigma[\![\mathbf{Tp}]\!]"] \\
\rho(\mathbf{Tp})
\end{tikzcd}$$

---

[1]Application to universes due to Shulman [Shu15], developed further by Coquand [Coq19]; main ingredients invented by the Grothendieck school in the 1970s [AGV72].

Abstract Artin gluing: logical relations as types!

Constructing the logical relation in $\mathcal{G}_\Sigma$ over $\mathcal{C}_\Sigma$ still very technical! **Trivialized by “synthetic Tait computability” (STC).**[3]

[3] Based on an idea of Shulman [Shu11], **STC** is an “Orton-Pitts method” [OP16] for syntactic metatheory [SG20; SH20].

Abstract Artin gluing: logical relations as types!

Constructing the logical relation in $\mathcal{G}_\Sigma$ over $\mathcal{C}_\Sigma$ still very technical! **Trivialized by "synthetic Tait computability" (STC).**[3]

Idea: replace "A logical relation is given by …" with "A logical relation can be used to…".

[3] Based on an idea of Shulman [Shu11], **STC** is an "Orton-Pitts method" [OP16] for syntactic metatheory [SG20; SH20].

Abstract Artin gluing: logical relations as types!

Constructing the logical relation in $\mathcal{G}_\Sigma$ over $\mathcal{C}_\Sigma$ still very technical! **Trivialized by "synthetic Tait computability" (STC).**[3]

Idea: replace "A logical relation is given by ..." with "A logical relation can be used to...".

Result: a rich type theory whose types denote generalized logical relations. Syntax and semantics accessed via **modalities** à la Rijke, Shulman, and Spitters [RSS17].

[3] Based on an idea of Shulman [Shu11], **STC** is an "Orton-Pitts method" [OP16] for syntactic metatheory [SG20; SH20].

Abstract Artin gluing: logical relations as types!

Constructing the logical relation in $\mathcal{G}_\Sigma$ over $\mathcal{C}_\Sigma$ still very technical! **Trivialized by "synthetic Tait computability" (STC).**[3]

Idea: replace "A logical relation is given by ..." with "A logical relation can be used to...".

Result: a rich type theory whose types denote generalized logical relations. Syntax and semantics accessed via **modalities** à la Rijke, Shulman, and Spitters [RSS17].

$$A \sim \text{"A synthetic logical relation"}$$

$$\bigcirc A \sim \text{"The syntactic part of } A\text{" (acts like a sort)}$$

$$\bullet A \sim \text{"The semantic part of } A\text{" (acts like a set)}$$

[3] Based on an idea of Shulman [Shu11], **STC** is an "Orton-Pitts method" [OP16] for syntactic metatheory [SG20; SH20].

Synthetic Tait computability

Extensional type theory **STC** with an *open modality* $\bigcirc A$, and a complementary *closed modality* $\bullet A$, both monadic.

$$\begin{aligned} A &\sim \text{“A synthetic logical relation”} \\ \bigcirc A &\sim \text{“The syntactic part of } A\text{” (acts like a sort)} \\ \bullet A &\sim \text{“The semantic part of } A\text{” (acts like a set)} \end{aligned}$$

Synthetic Tait computability

Extensional type theory **STC** with an *open modality* $\bigcirc A$, and a complementary *closed modality* $\bullet A$, both monadic.

$$A \sim \text{“A synthetic logical relation”}$$
$$\bigcirc A \sim \text{“The syntactic part of } A\text{” (acts like a sort)}$$
$$\bullet A \sim \text{“The semantic part of } A\text{” (acts like a set)}$$

Facts/Axioms:

1. The syntactic part of the semantic part of a logical relation is trivial: $\bigcirc\bullet A \cong \mathbf{1}$.

Synthetic Tait computability

Extensional type theory **STC** with an *open modality* $\bigcirc A$, and a complementary *closed modality* $\bullet A$, both monadic.

$$\begin{aligned} A &\sim \text{“A synthetic logical relation”} \\ \bigcirc A &\sim \text{“The syntactic part of } A\text{” (acts like a sort)} \\ \bullet A &\sim \text{“The semantic part of } A\text{” (acts like a set)} \end{aligned}$$

Facts/Axioms:

1. The syntactic part of the semantic part of a logical relation is trivial: $\bigcirc \bullet A \cong \mathbf{1}$.
2. A logical relation can be reconstructed from its syntactic and semantic parts [AGV72; RSS17].

Synthetic Tait computability

Extensional type theory **STC** with an *open modality* $\bigcirc A$, and a complementary *closed modality* $\bullet A$, both monadic.

$$\begin{aligned} A &\sim \text{“A synthetic logical relation”} \\ \bigcirc A &\sim \text{“The syntactic part of } A\text{” (acts like a sort)} \\ \bullet A &\sim \text{“The semantic part of } A\text{” (acts like a set)} \end{aligned}$$

Facts/Axioms:

1. The syntactic part of the semantic part of a logical relation is trivial: $\bigcirc\bullet A \cong \mathbf{1}$.
2. A logical relation can be reconstructed from its syntactic and semantic parts [AGV72; RSS17].
3. A logical relation can be realigned to have different (but isomorphic) syntactic part [OP16].

Idea: axiomatize a Σ-algebra $\mathcal{A}^{\circ}_{\Sigma}$ in the $\bigcirc$-modal fragment of **STC**, then construct a Σ-algebra $\mathcal{A}^{*}_{\Sigma}$ in **STC** such that $\bigcirc(\mathcal{A}^{*} = \mathcal{A}^{\circ})$ holds.

Idea: axiomatize a Σ-algebra $\mathcal{A}^{\circ}_{\Sigma}$ in the $\bigcirc$-modal fragment of **STC**, then construct a Σ-algebra $\mathcal{A}^{*}_{\Sigma}$ in **STC** such that $\bigcirc(\mathcal{A}^{*} = \mathcal{A}^{\circ})$ holds.

Examples:

$$\mathbf{Tp}^{*} \cong \textstyle\sum_{A^{\circ}:\mathbf{Tp}^{\circ}} \{A^{*} : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^{\circ}(A^{\circ}) = A^{*})\}$$

$$\mathbf{ans}^{*} \cong \left(\mathbf{ans}^{\circ}, \textstyle\sum_{\mathrm{b}^{\circ}:\mathbf{Tm}^{\circ}(\mathbf{ans}^{\circ})} \bullet\{\mathrm{b}^{\bullet} : \mathbf{2} \mid \mathrm{b}^{\circ} = \mathbf{if}\ \mathrm{b}^{\bullet}\ \mathbf{then}\ \mathbf{yes}^{\circ}\ \mathbf{else}\ \mathbf{no}^{\circ}\}\right)$$

Idea: axiomatize a Σ-algebra $\mathcal{A}^{\circ}_{\Sigma}$ in the $\bigcirc$-modal fragment of **STC**, then construct a Σ-algebra $\mathcal{A}^{*}_{\Sigma}$ in **STC** such that $\bigcirc(\mathcal{A}^{*} = \mathcal{A}^{\circ})$ holds.

Examples:

$$\mathbf{Tp}^{*} \cong \textstyle\sum_{A^{\circ}:\mathbf{Tp}^{\circ}}\{A^{*} : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^{\circ}(A^{\circ}) = A^{*})\}$$
$$\mathbf{ans}^{*} \cong \left(\mathbf{ans}^{\circ}, \textstyle\sum_{\mathrm{b}^{\circ}:\mathbf{Tm}^{\circ}(\mathbf{ans}^{\circ})} \bullet\{\mathrm{b}^{\bullet} : \mathbf{2} \mid \mathrm{b}^{\circ} = \mathbf{if}\ \mathrm{b}^{\bullet}\ \mathbf{then}\ \mathbf{yes}^{\circ}\ \mathbf{else}\ \mathbf{no}^{\circ}\}\right)$$

Payoff: painful construction of (e.g.) dependent product in logical relations made trivial, because open modalities commute with dependent products. *No more technical lemmas!*

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\bigcirc\mathbf{Tp}^*$$

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\bigcirc\sum_{A^\circ:\mathbf{Tp}^\circ}\{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$$

Show $\bigcirc\mathbf{Tp}^* \simeq \mathbf{Tp}^\circ$.

$$\sum_{A^\circ : \bigcirc\mathbf{Tp}^\circ} \bigcirc\{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$$

Show $\bigcirc\mathbf{Tp}^* \simeq \mathbf{Tp}^\circ$.

$$\sum_{A^\circ : \bigcirc\mathbf{Tp}^\circ} \bigcirc\{A^* : \mathcal{U} \mid \mathbf{Tm}^\circ(A^\circ) = A^*\}$$

Show $\bigcirc \mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\sum_{A^\circ : \bigcirc \mathbf{Tp}^\circ} \bigcirc \mathbf{1}$$

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\sum_{A^\circ : \bigcirc\mathbf{Tp}^\circ} \mathbf{1}$$

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\bigcirc\mathbf{Tp}^\circ$$

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\mathbf{Tp}^\circ$$

Show $\bigcirc\mathbf{Tp}^* \cong \mathbf{Tp}^\circ$.

$$\mathbf{Tp}^\circ$$

$\square$

Closure under dependent products.

Construct $\Pi^* : (A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*) \to \mathbf{Tp}^*$ with $\bigcirc \Pi^*(A, B) = \Pi^\circ(A^\circ, \lambda x. B^\circ(x))$.

Closure under dependent products.

Fix $A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*$ to construct $\Pi^*(A, B)$ with syntactic part $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Closure under dependent products.

Fix $A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*$ to construct $\Pi^*(A, B)$ with syntactic part $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Recall: $\mathbf{Tp}^* \cong \sum_{A^\circ : \mathbf{Tp}^\circ} \{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$

Closure under dependent products.

Fix $A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*$ to construct $\Pi^*(A, B)$ with syntactic part $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Recall: $\mathbf{Tp}^* \cong \sum_{A^\circ:\mathbf{Tp}^\circ}\{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$

1. Choose first component: $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Closure under dependent products.

Fix $A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*$ to construct $\Pi^*(A, B)$ with syntactic part $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Recall: $\mathbf{Tp}^* \cong \sum_{A^\circ : \mathbf{Tp}^\circ} \{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$

1. Choose first component: $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.
2. Choose second component: $(x : \mathbf{Tm}^*(A)) \to \mathbf{Tm}^*(B(x))$.

Closure under dependent products.

Fix $A : \mathbf{Tp}^*, B : A \to \mathbf{Tp}^*$ to construct $\Pi^*(A, B)$ with syntactic part $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.

Recall: $\mathbf{Tp}^* \cong \sum_{A^\circ : \mathbf{Tp}^\circ} \{A^* : \mathcal{U} \mid \bigcirc(\mathbf{Tm}^\circ(A^\circ) = A^*)\}$

1. Choose first component: $\Pi^\circ(A^\circ, \lambda x.B^\circ(x))$.
2. Choose second component: $(x : \mathbf{Tm}^*(A)) \to \mathbf{Tm}^*(B(x))$, *realigned by the following isomorphism:*

$$\bigcirc\big((x : \mathbf{Tm}^*(A)) \to \mathbf{Tm}^*(B(x))\big) \cong \mathbf{Tm}^\circ(\Pi^\circ(A^\circ, B^\circ(x)))$$

$$\bigcirc\Big((x : \mathbf{Tm}^*(A)) \to \mathbf{Tm}^*(B(x))\Big)$$

$$(x : \bigcirc\mathbf{Tm}^*(A)) \to \bigcirc\mathbf{Tm}^*(B(x))$$

$$(x : \mathbf{Tm}^{\circ}(A^{\circ})) \to \mathbf{Tm}^{\circ}(B^{\circ}(x))$$

$$(x : \mathbf{Tm}^{\circ}(A^{\circ})) \to \mathbf{Tm}^{\circ}(B^{\circ}(x))$$

$\square$

S. & Harper introduced **STC** to give a trivial proof of a non-trivial parametricity result for effectful ML modules:

> S., Harper (2020). **“Logical Relations as Types: Proof-Relevant Parametricity for Program Modules”**. Under review.

This thesis: use **STC** to prove normalization of Cartesian cubical type theory / $\mathbf{TT}_{\square}$.

Proposed work

Prove normalization for Cartesian cubical type theory ($\mathrm{TT}_{\square}$).

Proposed work

Prove normalization for Cartesian cubical type theory ($\mathbf{TT}_{\square}$).

1. Construct a model of **STC** extended by the syntax of $\mathbf{TT}_{\square}$ and its normal forms.

Proposed work

Prove normalization for Cartesian cubical type theory ($\mathbf{TT}_{\square}$).

1. Construct a model of **STC** extended by the syntax of $\mathbf{TT}_{\square}$ and its normal forms.
2. Define a "computability $\Sigma_{\square}$-algebra" in **STC** such that each sort A exhibits *Tait's Yoga* [Tai67]:

$$\begin{tikzcd}
A^{\mathbf{Ne}} \arrow[r, dotted, "\mathbf{reflect}_A"] \arrow[dr, "\eta^{\bigcirc}_{A^{\mathrm{Ne}}}"'] & A^{*} \arrow[r, dotted, "\mathbf{reify}_A"] \arrow[d, "\eta^{\bigcirc}_{A^{*}}"] & A^{\mathbf{Nf}} \arrow[dl, "\eta^{\bigcirc}_{A^{\mathrm{Nf}}}"] \\
 & A^{\circ} &
\end{tikzcd}$$

Proposed work

Prove normalization for Cartesian cubical type theory ($\mathbf{TT}_{\square}$).

1. Construct a model of **STC** extended by the syntax of $\mathbf{TT}_{\square}$ and its normal forms.
2. Define a “computability $\Sigma_{\square}$-algebra” in **STC** such that each sort A exhibits *Tait’s Yoga* [Tai67]:

$$
\begin{tikzcd}
A^{\mathbf{Ne}} \arrow[r, dotted, "\mathbf{reflect}_A"] \arrow[dr, "\eta^{\bigcirc}_{A^{\mathbf{Ne}}}"'] & A^{*} \arrow[r, dotted, "\mathbf{reify}_A"] \arrow[d, "\eta^{\bigcirc}_{A^{*}}"] & A^{\mathbf{Nf}} \arrow[dl, "\eta^{\bigcirc}_{A^{\mathbf{Nf}}}"] \\
& A^{\circ} &
\end{tikzcd}
$$

3. That does it! **Next: actually construct the model.**

Renamings & unstable normal forms, geometrically

Let $\mathcal{C}$ be the category of judgments and deductions.

Renamings & unstable normal forms, geometrically

Let $\mathcal{C}$ be the category of judgments and deductions.

- By Yoneda, have a space(*) $\widehat{\mathcal{C}}$ of (generalized) judgments and substitution functions.

Renamings & unstable normal forms, geometrically

Let $\mathcal{C}$ be the category of judgments and deductions.

- By Yoneda, have a space(*) $\widehat{\mathcal{C}}$ of (generalized) judgments and substitution functions.
- Likewise, have a space of (generalized) formal contexts and renaming functions $\widehat{\mathcal{R}}$ lying over $\widehat{\mathcal{C}}$. **Idea: Kripke logical relations varying in $\widehat{\mathcal{R}} \longrightarrow \widehat{\mathcal{C}}$.**

Renamings & unstable normal forms, geometrically

Let $\mathcal{C}$ be the category of judgments and deductions.

- By Yoneda, have a space(*) $\widehat{\mathcal{C}}$ of (generalized) judgments and substitution functions.
- Likewise, have a space of (generalized) formal contexts and renaming functions $\widehat{\mathcal{R}}$ lying over $\widehat{\mathcal{C}}$. **Idea: Kripke logical relations varying in $\widehat{\mathcal{R}} \longrightarrow \widehat{\mathcal{C}}$.**
- $\widehat{\mathcal{R}}$ has monoidal closed structure $(\otimes, \multimap)$ to implement *bunches* of non-equal dimensions and fresh binders:

$$\dfrac{\psi : \mathbb{I} \otimes \mathbb{I} \qquad A : \mathbb{I} \multimap \mathbf{Ne} \qquad M : \mathbf{Nf}}{\mathbf{coe}_A^{\psi}(M) : \mathbf{Ne}}\;\textsc{Neutral Coe}\ i \rightsquigarrow j$$

Renamings & unstable normal forms, geometrically

Let $\mathcal{C}$ be the category of judgments and deductions.

- By Yoneda, have a space(*) $\widehat{\mathcal{C}}$ of (generalized) judgments and substitution functions.
- Likewise, have a space of (generalized) formal contexts and renaming functions $\widehat{\mathcal{R}}$ lying over $\widehat{\mathcal{C}}$. **Idea: Kripke logical relations varying in $\widehat{\mathcal{R}} \longrightarrow \widehat{\mathcal{C}}$.**
- $\widehat{\mathcal{R}}$ has monoidal closed structure $(\otimes, \multimap)$ to implement *bunches* of non-equal dimensions and fresh binders:

$$\frac{\psi : \mathbb{I} \otimes \mathbb{I} \qquad A : \mathbb{I} \multimap \mathbf{Ne} \qquad M : \mathbf{Nf}}{\mathbf{coe}_A^{\psi}(M) : \mathbf{Ne}}\ \textsc{Neutral Coe } i \rightsquigarrow j$$

Foreseen difficulties: What to do with the inconsistent context $(\Gamma, p : 0 =_{\mathbb{I}} 1)$?

Kripke logical relations, geometrically

Kripke logical relations, geometrically

- **Sierpiński interval:** $\mathbb{S} = \{\varnothing, \{\circ\}, \{\circ, \bullet\}\}$ is the interface of a *family*.
  - open point $\circ$ = coordinate of the base/index
  - closed point $\bullet$ = coordinate of the total space

Kripke logical relations, geometrically

- **Sierpiński interval:** $\mathbb{S} = \{\varnothing, \{\circ\}, \{\circ, \bullet\}\}$ is the interface of a *family*.
  - open point $\circ$ = coordinate of the base/index
  - closed point $\bullet$ = coordinate of the total space
- **Sierpiński cylinder:** $\widehat{\mathcal{R}} \times \mathbb{S}$ is the interface of *families of generalized contexts and renamings*.

Kripke logical relations, geometrically

- **Sierpiński interval:** $\mathbb{S} = \{\varnothing, \{\circ\}, \{\circ, \bullet\}\}$ is the interface of a *family*.
  - open point $\circ$ = coordinate of the base/index
  - closed point $\bullet$ = coordinate of the total space
- **Sierpiński cylinder:** $\widehat{\mathcal{R}} \times \mathbb{S}$ is the interface of *families of generalized contexts and renamings*.
- **Sierpiński cone(*)/gluing:** $\mathcal{K} = \widehat{\mathcal{C}} \sqcup_{\widehat{\mathcal{R}}} \widehat{\mathcal{R}} \times \mathbb{S}$ is the interface of families of generalized contexts and renamings indexed in generalized judgments and substitutions, *i.e.* Kripke logical relations!

$$
\begin{tikzcd}
\widehat{\mathcal{R}} \arrow[r] \arrow[d, hook, "\widehat{\mathcal{R}} \times \circ"'] & \widehat{\mathcal{C}} \arrow[d, hook, dotted] \\
\widehat{\mathcal{R}} \times \mathbb{S} \arrow[r, dotted] & \mathcal{K} \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}
$$

Kripke logical relations, geometrically

- **Sierpiński interval:** $\mathbb{S} = \{\varnothing, \{\circ\}, \{\circ, \bullet\}\}$ is the interface of a *family*.
  - open point $\circ$ = coordinate of the base/index
  - closed point $\bullet$ = coordinate of the total space
- **Sierpiński cylinder:** $\widehat{\mathcal{R}} \times \mathbb{S}$ is the interface of *families of generalized contexts and renamings*.
- **Sierpiński cone(*)/gluing:** $\mathcal{K} = \widehat{\mathcal{C}} \sqcup_{\widehat{\mathcal{R}}} \widehat{\mathcal{R}} \times \mathbb{S}$ is the interface of families of generalized contexts and renamings indexed in generalized judgments and substitutions, *i.e.* Kripke logical relations!

$$\begin{tikzcd}
\widehat{\mathcal{R}} \arrow[r] \arrow[d, hook, "\widehat{\mathcal{R}} \times \circ"'] & \widehat{\mathcal{C}} \arrow[d, hook, dotted] \\
\widehat{\mathcal{R}} \times \mathbb{S} \arrow[r, dotted] & \mathcal{K} \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

$\mathbf{Sh}(\mathcal{K})$ a suitable model of **STC**.

Modeling neutral/normal forms

From now on, we work in the language of **STC**. Let A be a type of $\textbf{TT}_{\square}$.

1. We have an object $A^{\mathbf{Var}} : \mathcal{U}$ of variables with $\bigcirc(A^{\mathbf{Var}} = A^{\circ})$
2. Extend it to a definition of *neutral forms* $A^{\mathbf{Ne}}$ of type A
3. Extend it to a definition of *normal forms* $A^{\mathbf{Nf}}$ of type A.

Subtleties: representation of interval dimension binders, tensors of dimensions.

The computability algebra

From now on, we work abstractly in the (extended) **STC** substantiated above. Inspired by and improving on Coquand [Coq19], we define the computability algebra of types:

$$\mathbf{Tp}^* \cong \sum \left\{ \begin{array}{l} A^\circ : \mathbf{Tp}^\circ \\ \mathfrak{A} : \left\{\mathfrak{A} : \mathbf{Tp}^{\mathbf{Nf}} \;\middle|\; \bigcirc(\mathfrak{A} = A^\circ)\right\} \\ A^* : \{A^* : \mathcal{U} \mid \bigcirc(A^* = \mathbf{Tm}^\circ(A^\circ))\} \\ \mathbf{reflect}_A : \left\{\mathsf{f} : A^{\mathbf{Ne}} \to A^* \;\middle|\; \bigcirc(\mathsf{f} = \mathbf{id})\right\} \\ \mathbf{reify}_A : \left\{\mathsf{f} : A^* \to A^{\mathbf{Nf}} \;\middle|\; \bigcirc(\mathsf{f} = \mathbf{id})\right\} \end{array} \right\}$$

The computability algebra

From now on, we work abstractly in the (extended) **STC** substantiated above. Inspired by and improving on Coquand [Coq19], we define the computability algebra of types:

$$\mathbf{Tp}^* \cong \sum \left\{\begin{array}{l} A^\circ : \mathbf{Tp}^\circ \\ \mathfrak{A} : \left\{\mathfrak{A} : \mathbf{Tp}^{\mathbf{Nf}} \mid \bigcirc(\mathfrak{A} = A^\circ)\right\} \\ A^* : \{A^* : \mathcal{U} \mid \bigcirc(A^* = \mathbf{Tm}^\circ(A^\circ))\} \\ \mathbf{reflect}_A : \left\{\mathsf{f} : A^{\mathbf{Ne}} \to A^* \mid \bigcirc(\mathsf{f} = \mathbf{id})\right\} \\ \mathbf{reify}_A : \left\{\mathsf{f} : A^* \to A^{\mathbf{Nf}} \mid \bigcirc(\mathsf{f} = \mathbf{id})\right\} \end{array}\right\}$$

Foreseen difficulties: We must define a version of $\mathbf{Tp}^*$ that equips each *family* $\mathbb{I} \to \mathbf{Tp}^*$ with a composition operation; possible if the interval is atomic [Lic+18]. This is the main thing that could go wrong.

Timeline and fallback positions

The goal is a proof of normalization for cubical type theory with a univalent universe. Granular prediction impossible, but I estimate the following milestones to serve as fallback positions if part of this turns out to be intractable.

- **Now + 6 months:** a proof of β/η normalization for MLTT + $\mathbb{I}$ + $\mathbb{F}$ + extension types, no universes or HITs.
- **Now + 8 months:** a mathematical specification of the elaboration of a `redtt`/`cooltt`-style external language to the core type theory specified above.
- **Now + 12 months:** extension of proof to include a univalent universe.

Stretch goals

The following are things that I *do not* promise to do, but which might happen along the way if my progress is better than expected. If I don't do it, one of your students should!

- **Extensions of the `cooltt` implementation:** modules, higher inductive types, more sophisticated universe hierarchies, or support for **STC** modalities.
- **Extensions of synthetic Tait computability** to account for modal and substructural type theory.
- **Objective metatheory of effectful PLs:** Sterling and Harper [SH20] just a first step, more development needed!

References I

[AGV72] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *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, 1972.

[AHH17] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. *Computational Higher Type Theory III: Univalent Universes and Exact Equality*. 2017. arXiv: 1712.01800.

[AHS95] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. "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, 1995.

[AK16a] Thorsten Altenkirch and Ambrus Kaposi. "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, 2016. DOI: 10.4230/LIPIcs.FSCD.2016.6.

References II

[AK16b] Thorsten Altenkirch and Ambrus Kaposi. “Type Theory in Type Theory Using Quotient Inductive Types”. In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL ’16. St. Petersburg, FL, USA: ACM, 2016. DOI: 10.1145/2837614.2837638.

[All87] Stuart Frazier Allen. “A non-type-theoretic semantics for type-theoretic language”. PhD thesis. Ithaca, NY, USA: Cornell University, 1987.

[Ang+17] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. *Cartesian Cubical Type Theory*. Preprint. Dec. 2017.

[Ang+18a] Carlo Angiuli, Evan Cavallo, Kuen-Bang Hou (Favonia), Robert Harper, Anders Mörtberg, and Jonathan Sterling. redtt*: implementing Cartesian cubical type theory*. Dagstuhl Seminar 18341: Formalization of Mathematics in Type Theory. 2018.

[Ang+18b] Carlo Angiuli, Evan Cavallo, Kuen-Bang Hou (Favonia), Robert Harper, and Jonathan Sterling. “The RedPRL Proof Assistant (Invited Paper)”. In: *Proceedings of the 13th International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP@FSCD 2018, Oxford, UK, 7th July 2018*. 2018. DOI: 10.4204/EPTCS.274.1.

References III

[Ang+19] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. *Syntax and Models of Cartesian Cubical Type Theory*. Preprint. Feb. 2019.

[Ang19] Carlo Angiuli. "Computational Semantics of Cartesian Cubical Type Theory". PhD thesis. Carnegie Mellon University, 2019.

[AS19] Carlo Angiuli and Jonathan Sterling. "Lectures on type theory". Based on lectures given by Angiuli and Sterling in autumn of 2019. 2019.

[AW09] Steve Awodey and Michael A. Warren. "Homotopy theoretic models of identity types". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 146.1 (Jan. 2009). ISSN: 0305-0041. DOI: 10.1017/S0305004108001783.

[Awo18a] Steve Awodey. "A cubical model of homotopy type theory". In: *Annals of Pure and Applied Logic* 169.12 (2018). Logic Colloquium 2015. ISSN: 0168-0072. DOI: 10.1016/j.apal.2018.08.002.

[Awo18b] Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018). DOI: 10.1017/S0960129516000268.

[Car78] John Cartmell. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Oxford University, Jan. 1978.

References IV

[CCD17] Simon Castellan, Pierre Clairambault, and Peter Dybjer. "Undecidability of Equality in the Free Locally Cartesian Closed Category (Extended version)". In: *Logical Methods in Computer Science* 13.4 (2017).

[CH19] Evan Cavallo and Robert Harper. "Higher Inductive Types in Cubical Computational Type Theory". In: *Proc. ACM Program. Lang.* 3.POPL (Jan. 2019). ISSN: 2475-1421. DOI: 10.1145/3290314.

[CHM18] Thierry Coquand, Simon Huber, and Anders Mörtberg. "On Higher Inductive Types in Cubical Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. Oxford, United Kingdom: ACM, 2018. DOI: 10.1145/3209108.3209197.

[Coh+15] Cyril Cohen, Thierry Coquand, Simon Huber, and Simon Mörtberg. `cubical`*: Implementation of Univalence in Cubical Sets*. 2015.

[Coh+17] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. "Cubical Type Theory: a constructive interpretation of the univalence axiom". In: *IfCoLog Journal of Logics and their Applications* 4.10 (Nov. 2017).

[Coh+18] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. `cubicaltt`*: Experimental implementation of Cubical Type Theory*. 2018.

References V

[Coq19] Thierry Coquand. "Canonicity and normalization for dependent type theory". In: *Theoretical Computer Science* 777 (2019). In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2019.01.015. arXiv: 1810.09367.

[Dyb96] Peter Dybjer. "Internal type theory". In: *Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers*. Ed. by Stefano Berardi and Mario Coppo. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[Fio02] Marcelo Fiore. "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: ACM, 2002. DOI: 10.1145/571157.571161.

[Fio12] Marcelo Fiore. *Discrete generalised polynomial functors.* Slides from talk given at ICALP 2012. 2012.

[Fre78] Peter Freyd. "On proving that 1 is an indecomposable projective in various free categories". Unpublished manuscript. 1978.

[GSB19] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. "Implementing a Modal Dependent Type Theory". In: *Proceedings of the ACM on Programming Languages* 3.ICFP (July 2019). ISSN: 2475-1421. DOI: 10.1145/3341711.

References VI

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. “A Framework for Defining Logics”. In: *J. ACM* 40.1 (Jan. 1993). ISSN: 0004-5411. DOI: 10.1145/138027.138060.

[HS98] Martin Hofmann and Thomas Streicher. “The groupoid interpretation of type theory”. In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998.

[Hub18] Simon Huber. “Canonicity for Cubical Type Theory”. In: *Journal of Automated Reasoning* (June 13, 2018). ISSN: 1573-0670. DOI: 10.1007/s10817-018-9469-1.

[Joh02] Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium: Volumes 1 and 2*. Oxford Logical Guides 43. Oxford Science Publications, 2002.

[Law63] F. William Lawvere. “Functorial Semantics of Algebraic Theories”. PhD thesis. Columbia University, 1963.

[Lic+18] Daniel R. Licata, Ian Orton, Andrew M. Pitts, and Bas Spitters. “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*. 2018. DOI: 10.4230/LIPIcs.FSCD.2018.22.

[Lur09] Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009.

References VII

[MA18] Anders Mörtberg and Carlo Angiuli. yacctt*: Yet Another Cartesian Cubical Type Theory*. 2018.

[Mar79] Per Martin-Löf. "Constructive Mathematics and Computer Programming". In: *6th International Congress for Logic, Methodology and Philosophy of Science*. Published by North Holland, Amsterdam. 1982. Hanover, Aug. 1979.

[Mar84] Per Martin-Löf. *Intuitionistic type theory. Notes by Giovanni Sambin.* Vol. 1. Studies in Proof Theory. Bibliopolis, 1984.

[New18] Clive Newstead. "Algebraic Models of Dependent Type Theory". PhD thesis. Carnegie Mellon University, 2018.

[NPS90] Bengt Nordström, Kent Peterson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory*. Vol. 7. International Series of Monographs on Computer Science. NY: Oxford University Press, 1990.

[OP16] Ian Orton and Andrew M. Pitts. "Axioms for Modelling Cubical Type Theory in a Topos". In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*. 2016.

[Red18] The RedPRL Development Team. redtt. 2018.

[Red20] The RedPRL Development Team. cooltt. 2020.

References VIII

[RSS17] Egbert Rijke, Michael Shulman, and Bas Spitters. *Modalities in homotopy type theory*. 2017. arXiv: 1706.07526.

[SA20] Jonathan Sterling and Carlo Angiuli. “Gluing models of type theory along flat functors”. Unpublished draft. 2020.

[SAG19] Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. “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, 2019. DOI: 10.4230/LIPIcs.FSCD.2019.31. arXiv: 1904.08562.

[SAG20] Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. *A cubical language for Bishop sets*. 2020. arXiv: 2003.01491.

[SG20] Jonathan Sterling and Daniel Gratzer. “Lectures on Synthetic Tait Computability”. Based on lectures given by Sterling in the summer of 2020. 2020.

[SH18] Jonathan Sterling and Robert Harper. “Guarded Computational Type Theory”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. Oxford, United Kingdom: ACM, 2018. arXiv: 1804.09098.

References IX

[SH20] Jonathan Sterling and Robert Harper. “Logical Relations As Types: Proof-Relevant Parametricity for Program Modules”. Unpublished draft. 2020.

[Shu11] Michael Shulman. *Internalizing the External, or The Joys of Codiscreteness*. blog. 2011.

[Shu13] Michael Shulman. *Scones, Logical Relations, and Parametricity*. Blog. 2013.

[Shu15] Michael Shulman. “Univalence for inverse diagrams and homotopy canonicity”. In: *Mathematical Structures in Computer Science* 25.5 (2015). DOI: 10.1017/S0960129514000565.

[SS18] Jonathan Sterling and Bas Spitters. *Normalization by gluing for free λ-theories*. Sept. 2018. arXiv: 1809.08646 [cs.LO].

[Ste20] Jonathan Sterling. *Objective Metatheory of Cubical Type Theories*. Thesis Proposal. 2020.

[Str91] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Cambridge, MA, USA: Birkhauser Boston Inc., 1991.

References X

[Str98] Thomas Streicher. "Categorical intuitions underlying semantic normalisation proofs". In: *Preliminary Proceedings of the APPSEM Workshop on Normalisation by Evaluation*. Ed. by O. Danvy and P. Dybjer. Department of Computer Science, Aarhus University, 1998.

[Tai67] W. W. Tait. "Intensional Interpretations of Functionals of Finite Type I". In: *The Journal of Symbolic Logic* 32.2 (1967). ISSN: 00224812.

[Uem19] Taichi Uemura. *A General Framework for the Semantics of Type Theory*. 2019. arXiv: 1904.04097.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. "Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types". In: *Proceedings of the 24th ACM SIGPLAN International Conference on Functional Programming*. ICFP '19. Boston, Massachusetts, USA: ACM, 2019. DOI: 10.1145/3341691.