DISSERTATION OVERVIEW
First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory

Jonathan Sterling*

March 2022

1 Synopsis

1.1 Background: homotopy and cubical type theory

For more than four decades, *dependent type theory* has been positioned as the "common language" that can finally unify mathematics and computer programming [Mar79]: while it has never been controversial that a computer program is a form of mathematical construction, the running hypothesis of the type theoretic community has been the *converse* to this claim, namely that mathematical constructions should be viewed as programs that can in principle be executed by a physical machine — roughly, **sets = types** and **elements = programs**. Thus the struggle to realize this type theoretic hypothesis has been a two-way process, punctuated by moments at which the mathematical meaning of a programming construct is elucidated, or at which the computational content of a mathematical construct is uncovered.

In the current millennium, a new identification has been taking shape in which **types = ∞-groupoids** (homotopy types), which are an infinite-dimensional generalization of sets; the origins of this new perspective on type theory lie with Hofmann and Streicher's 1998 *groupoid interpretation* of type theory [HS98], combined with the revolutionary contributions of Voevodsky [Voe06] and Awodey and Warren [AW09] respectively. The main feature of the new language, dubbed **homotopy type theory** or **HoTT** [Uni13], is that isomorphisms between types are equipped with a new induction rule called *univalence* stating that all type theoretic constructs respect isomorphisms:

*Aarhus University, Department of Computer Science to a first approximation, if $A \cong B$ then $P(A) \cong P(B)$ for any P. The univalence principle is motivated by the phenomenon of *homotopy invariance* that pervades the large-scale structure of modern-day mathematics, from algebraic topology to algebraic geometry to mathematical physics; as a programming construct, univalence suggests new approaches to both generic and modular programming [Ang+21b].

Thus one of the main projects for the first decade of homotopy type theory was to substantiate the relationship between HoTT and mathematics on the one hand, and between HoTT and computer programming on the other hand. The question of whether homotopy type theoretic language can be interpreted in sheaves on arbitrary infinite-dimensional spaces (∞-topoi) has finally been resolved satisfactorily by Shulman [Shu19] in 2019. On the other hand, the computational interpretation of homotopy type theory has involved a reformulation of HoTT called **cubical type theory** [Ang+21a; AHH18; Coh+17] that reorganizes the higher-dimensional structure discussed by considering all the points, lines, squares, cubes, hypercubes, and so-on that one can draw in a given type. The computational interpretation of the new cubical type theory can be split into two different conjectures:

(C1) *Canonicity.* For any closed term $\cdot \vdash N : \mathsf{nat}$ of cubical type theory, there exists a unique natural number $n \in \mathbb{N}$ such that $\cdot \vdash N \equiv \overline{n} : \mathsf{nat}$ where $\overline{n}$ is the encoding of the number n as a term in the type theory.

(C2) *Decidability.* The assertions $\Gamma \vdash M : A$ and $\Gamma \vdash M \equiv N : A$ are decidable.

The canonicity conjecture **(C1)** states that terms written in cubical type theory can be thought of as computer programs, and was verified independently by Huber [Hub18] and Angiuli [Ang19] for different variants of cubical type theory. The decidability conjecture **(C2)** is no less important, as it is a necessary ingredient to implement a *typechecker* or a *compiler* for a programming language based on cubical type theory.

1.2 Contributions of this dissertation

This dissertation positively resolves the decidability conjecture **(C2)** for cubical type theory, the last remaining open question in its syntactic metatheory. Standard techniques proved inadequate for tackling this problem, so the bulk of this dissertation focuses on developing a new mathematical technique called **synthetic Tait computability** that generalizes and abstracts the method of *Tait computability* or *logical predicates*; in the past two years, synthetic Tait computability has played a central role in solving several problems in both type theory [Gra22; GB22; Ste21; SA21] and core programming languages [Niu+22; SH21; SH22], suggesting that this dissertation presents a lasting and transformative contribution to the state of the art.

2 Detailed Overview

This section contains a chapter-by-chapter overview; the novel contributions are contained in Chapters 4, 5, 7, and 8. The remaining Chapters 0–3 and 6 are primarily expository.

Part I: Dependent Type Theory

Ch. 0: Conspectus on type theory This chapter situates the motivations and applications of type theory in mathematics and computer science, and poses these against the semantic and syntactic properties of type theory that are needed to substantiate these applications. On the semantic side, type theory needs a number of properties including function extensionality, function comprehension, propositional univalence, effective quotients, *etc.*; on the syntactic side, type theory needs to be at least consistent, and many applications require both canonicity and decidability. Combining these syntactic and semantic properties into a single system has been a challenge, and cubical type theory was designed with the intention of satisfying them all. Prior to this dissertation, only the decidability conjecture remained open; thus with the present contribution, we regard the *Cubical Hypothesis* confirmed.

Ch. 1: Objective syntax of dependent types To state and prove theorems like canonicity and decidability **(C1,2)** for a type theory, we must have a mathematical definition of the *syntax* of type theory. Conventionally, the syntax of type theory has been studied in several layers: one starts with a definition of “raw” syntax as trees labeled by the names of the generating operations, quotients these trees under permutation of bound variables, and then layers on top of this an additional inductively defined formalism expressing the well-formedness of types, well-formedness of terms, definitional equality of types, and definitional equality of terms. After this, one verifies that definitional equivalence classes of well-formed types and terms can be used as the raw materials to construct a *universal model* of the type theory that has a universal property: any other model of type theory can be equipped with a unique structure-preserving homomorphism from the universal model. The described universal property determines the universal model up to *unique* isomorphism, if such a model exists.

We refer to the painstaking process described above as the **subjective metatheory**, building on the Hegel–Lawvere distinction between objective and subjective approaches to logic [Heg69; Law94; LS09]. The **objective metatheory**, in contrast, involves stating and proving results about type theories and programming languages relying *only* on the universal property of the universal model and not on any specifics of its presentation; the advantage of the objective metatheory is that it is simpler, more direct, more modular, and more composable.

Chapter 1 argues that the subjective metatheory in the sense described is redundant: the decidability conjecture can be stated with respect to *any* representative of the universal model and does not depend in any way on the raw syntax of type theory, and moreover, for all the type theories considered in this dissertation the existence of at least one representative of the universal model is guaranteed for somewhat trivial reasons that have nothing to do with the specifics of type theory. In this chapter, we develop a logical framework for specifying type theories modularly and working with their universal models in an objective fashion.

Part II: Mathematical Background

Ch. 2: The language of topoi Much of this dissertation is stated *and* proved by exploiting the language of (Grothendieck) topoi, a concept that leads a dual life as a kind of generalized logic *and* as a kind of generalized topology. The viewpoint of topos theory *qua* generalized topology plays an important role in this dissertation, and yet it nonetheless remains unfamiliar to most computer scientists and logicians. For this reason, Chapter 2 is provided as sufficient exposition to understand the use of topoi in the remainder of the dissertation, focusing on the *recollement* or *gluing* of a topos from a pair of complementary open and closed subtopoi, a classical construction that provides the geometrical basis for synthetic Tait computability.

Ch. 3: The theory of universes Universes started their life in the Grothendieck school of algebraic geometry [AGV72] as a technical device to circumvent the annoyance that there cannot be a “set of all sets”; a universe is a set of *enough* sets, and whilst a universe cannot contain itself, it may nonetheless lie within an even bigger universe. Several important developments in the theory of universes from the 1970s onward by Bénabou, Martin-Löf, Dybjer, Hofmann, Streicher, Awodey, and others [Awo18; Bén73; Dyb96; Hof97; JM95; Mar84; Str14] have collided to deliver the present-day understanding of the centrality of universes in semantics: a model of type theory is just a special kind of universe in a special kind of category. Chapter 3 provides expository background on the theory of universes, including a novel account of open and closed subuniverses to reflect the recollement theory of topoi from Chapter 2. These open and closed subuniverses will play a critical role in the development of synthetic Tait computability in subsequent chapters.

Part III: Synthetic Tait Computability

Ch. 4: Tait's method of computability It is simple enough to verify **negative** properties of a formal system, *e.g.* the non-derivability of a given assertion Φ: find a mathematical object that models all the rules of the formal system and yet refutes Φ. In contrast, it is much harder to verify any non-trivial **positive** property of a formal system (such as canonicity, normalization, decidability, *etc.*). To handle such results, new techniques were needed — and delivered in the late 1960s by Tait, Martin-Löf and others under the name of *Tait's method of computability* or *logical predicates.* Since its inception, Tait's method has been the primary tool for verifying positive properties of logics, programming languages, and type theories. Early on, Freyd [Fre78] noticed that the logical predicates arguments can be rephrased as model constructions that glue together geometrical objects corresponding to *syntax* (object) and *set theory* (meta), setting the stage for this thesis. Thus despite appearances, both positive and negative properties can both be proved using semantic methods.

In the subsequent development of the computability method for applications in computer science, *indexed* variants of the logical predicates have proved to be fundamental and a number of variations on indexed logical predicates have appeared including the Kripke logical predicates of Jung and Tiuryn [JT93] and the much more sophisticated *Grothendieck* logical predicates of Fiore and Simpson [FS99] as well as Altenkirch, Dybjer, Hofmann, and Scott [Alt+01]. This chapter points out that all of these forms of indexing arise in the same way from what is referred to as a **figure shape**, a continuous map into the classifying space of "Henkin models" of a given theory. Then the (Kripke, Grothendieck, *etc.*) logical predicates model is presented much more simply as the Artin gluing of this morphism's inverse image.

An explicit proof of canonicity for the simply typed λ-calculus motivates the abstraction and axiomatization of the geometry of figure shapes and their gluings as a new language for syntactic metatheory, namely **synthetic Tait computability**. The idea of synthetic Tait computability is to treat both object-level notions (*e.g.* the collection of terms of type bool) and meta-level notions (*e.g.* a normal form of a given term) in the same language by means of a pair of lex idempotent modalities. One strength of this presentation is that both object-level *and* meta-level notions can be treated using higher-order abstract syntax (HOAS) in the sense of Hofmann [Hof99], which greatly simplifies the manipulation of variables.

The first demonstration of the power and modularity of synthetic Tait computability is a new a proof of the canonicity property for Martin-Löf type theory. Unlike traditional proofs of canonicity via non-synthetic Tait computability, the synthetic version is completely modular and broken up into general-purpose lemmas that are stated at a high level of abstraction and can be reused in proofs of *different properties* for *different type theories*.[1] The modularization of syntactic metatheory is one of the main contributions of this dissertation.

Ch. 5: Synthetic normalization by evaluation This chapter develops a more sophisticated application of synthetic Tait computability, the proof of normalization and decidability of Martin-Löf's type theory with a cumulative hierarchy of universes. The synthetic argument contained in this chapter builds on the work of Fiore [Fio02] on categorical normalization by gluing for simply typed λ-calculus, and that of Coquand [Coq19] on a presheaf-theoretic version of normalization by evaluation for dependent types. Analogous to the external argument of Fiore [Fio02], we construe the syntax of normal and neutral forms as the initial algebra for an internal inductive definition in the language of synthetic Tait computability. The influence of Coquand [Coq19] is visible in the definition of the *Tait saturation yoga* for dependent types in the synthetic setting, an important closure condition for logical predicates that comprised one of the main innovations of Tait [Tai67] in the context of simply typed combinators. Although this chapter is intended only as "dry run" for the main result (to be exposed in Chapter 7), the normalization argument presented here has intrinsic value: it is the simplest and most direct proof of normalization and decidability for Martin-Löf type theory with η-laws and cumulative universes that has appeared in the literature so far.

Part IV: Cubical Type theory

Ch. 6: Cartesian cubical type theory This expository chapter introduces cubical type theory as an extension to Martin-Löf's type theory by an interval $\mathbb{I}$ with two distinct endpoints $0, 1 : \mathbb{I}$. The interval is a basic "figure" that defines a notion of *path* or *identification* $u \sim_A v$ between two elements of any type; for instance, to identify $u, v : A$ is the same as to construct a function $p : \mathbb{I} \to A$ such that $p(0) = u$ and $p(1) = v$. Terms involving *variables* of type $\mathbb{I}$ can exhibit complex computational behavior that is difficult to account

[1] Indeed, some of the constructions isolated in this chapter are used off the shelf in Chapter 7 to prove normalization for cubical type theory.

for: for instance, if $p : u \sim_A v$ and $i : \mathbb{I}$ are variables, then the application $p(i) : A$ is a normal form, but it must nonetheless reduce to either u or v when i is substituted for by a constant. Despite appearances, this scenario is fundamentally different from the way that $p(i)$ must reduce when p is replaced by a λ-abstraction, as normal forms must *a priori* be closed under arbitrary $\mathbb{I}$-substitutions — a necessity, because the normal form of an n-cube must be an n-cube of normal forms. One of the main technical contributions of this dissertation, introduced in the next chapter, is to generalization of the notion of *neutral form* and the Tait saturation yoga that smoothly accommodates the problematic computational behavior of the interval.

Ch. 7: Normalization for cubical type theory This chapter reports the main result of the dissertation, normalization for cubical type theory and its corollaries: injectivity of type constructors, and decidability of equality & typing **(C2)**. These results were first obtained by Sterling and Angiuli [SA21] for the fragment of cubical type theory *without* universes; the present chapter extends the results of *op. cit.* to support a cumulative hierarchy of universes.

The central innovation of this chapter is to generalize the notion of *neutral form* to accommodate the computational behavior of terms that have free variables of type $\mathbb{I}$ discussed in our synopsis of Chapter 6. In the conventional account of neutral and normal forms, neutrals $\mathbf{e}$ are built up inductively from $\mathsf{var}(x)$ for term variables $x : A$, function applications to normal forms $\mathbf{e} \cdot \mathbf{m}$ and projections from neutral pairs $\mathbf{e}.1, \mathbf{e}.2$; our account of neutrals is much the same, *except* that each neutral form $\mathbf{e}$ comes equipped with a **frontier of instability** $\partial\mathbf{e}$, a predicate on its free $\mathbb{I}$-variables that indicates when it "needs to compute further". We think of a neutral form for an n-cube as being *undefined* on its frontier of instability; the process of restricting a neutral to its frontier of instability is then referred to as *destabilization.*

When $x : A$ is a variable of an ordinary type, the frontier of instability $\partial(\mathsf{var}(x))$ is empty because variables never need to compute further. Where something new happens is the path type: given a neutral form $\mathbf{e} : \mathsf{ne}(u \sim_A v)$ of path type, we have for each term $r : \mathbb{I}$ a neutral form $\mathbf{e} @ r : \mathsf{ne}(A)$ whose frontier of instability is defined like so:

$$\partial(\mathbf{e} @ r) = \partial\mathbf{e} \vee (r = 0) \vee (r = 1)$$

In other words, the path neutral application $\mathbf{e}$ needs to compute as soon as $\mathbf{e}$ needs to compute, *and* as soon as the interval term $r : \mathbb{I}$ becomes equal to a constant. Prior to the introduction of the frontier of instability, the neutrals are embedded into the normals at base types unconditionally, *i.e.* for each neutral form $\mathbf{e} : \mathsf{ne}(\mathsf{bool})$, we have a normal form $\lfloor \mathbf{e} \rfloor : \mathsf{nf}(\mathsf{bool})$.

Now that neutrals are equipped with frontiers of instability, a more refined notion of normal form is needed: when e is a neutral form for a term e, the corresponding normal form should contain (recursively) normal forms for e that are defined under the frontier of instability $\partial\mathsf{e}$. To be more concrete, let $x : \mathsf{tt} \sim_{\mathsf{bool}} \mathsf{tt}$ be a variable of path type and $r : \mathbb{I}$ is a term; then $\mathsf{var}(x) @ r : \mathsf{ne}(\mathsf{tt} \sim_{\mathsf{bool}} \mathsf{tt})$ is a neutral form for the term $x(r)$ whose frontier of instability is the boundary $(r = 0) \vee (r = 1)$; the corresponding normal form must therefore glue onto $\mathsf{var}(x) @ r$ additional normal forms for $x(0)$ and $x(1)$. We refer to the process of completing a neutral with additional data defined on its frontier of instability as **stabilization**; the stabilized normal form of $x(r)$ is then written $\lfloor \mathsf{var}(x) @ r \mid r = 0 \hookrightarrow \mathsf{tt}, r = 1 \hookrightarrow \mathsf{tt} \rfloor$ where tt is the normal form representing the term tt.

Just as the embedding of neutrals into normals is "stabilized" by a compatible normal form defined on the neutral's frontier of instability, so too must the Tait saturation yoga be adjusted. Conventionally one requires the computability predicate for a type A to be equipped with a function that takes neutral forms e of terms $e : A$ to computability witnesses for the same term. In the **stabilized Tait saturation yoga**, we strengthen the induction hypothesis to require for each neutral form e a function that *extends* a computability witness defined only on the frontier of instability $\partial\mathsf{e}$ to a computability witnessed defined everywhere.

The twin innovations of *frontiers of instability* and *stabilization* then suffice to adapt the synthetic normalization argument of Chapter 5 to a proof of normalization (and thus decidability) for cubical type theory.

Part V: Prospects

Ch. 8: A plan for PL This dissertation has focused almost solely on the development and applications of synthetic Tait computability in the context of pure type theory, but the author originally invented synthetic Tait computability to solve problems in core programming languages, as part of Sterling and Harper's re-analysis [SH21] of the *phase distinction* in ML-style module systems [HMM90; Mog89] between *static* (compiletime) and *dynamic* (runtime) code. The purpose of this chapter is to identify several applications of synthetic Tait computability to core programming languages, and set an agenda for future work — some of which has been executed and published following the completion of this dissertation.

§ 8.1 A brief overview is given of the applications of synthetic Tait computability to program modules, material that is published in *J.ACM* [SH21].

§ 8.2 The modal language of synthetic Tait computability promises a new and more abstract account of *refinement types* and *program extraction* via a phase distinction between *computation* and *specification*. Refinement types are often thought of as a kind of subtype, but there is a fundamental difference: when $\phi \sqsubset A$ and $\psi \sqsubset B$ are refinements of types A and B respectively, then $\phi \to \psi$ refines $A \to B$. In contrast, subtyping laws for function spaces are *contravariant* in the domain. The refinements available in synthetic Tait computability are moreover *proof-relevant* in the sense that specification-level code can contain data in addition to properties. The application to proof-relevant refinement types is employed by Niu, Sterling, Grodin, and Harper [Niu+22] to develop a logical framework for simultaneously verifying behavior *and* complexity of functional programs.

§ 8.3 Finally, an application of synthetic Tait computability to *security* and *information flow control* is identified: a security class is a phase distinction between low and high security. The preliminary results presented in this section have been substantially improved upon by Sterling and Harper [SH22], adding support for general recursion and termination-insensitive noninterference by combining synthetic Tait computability with synthetic domain theory [Hyl91].

Bibliography of Synthetic Tait Computability

[Gra22] Daniel Gratzer. “Normalization for Multimodal Type Theory”. In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2022. DOI: 10.1145/3531130.3532398.

[GB22] Daniel Gratzer and Lars Birkedal. “A Stratified Approach to Löb Induction”. In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*. Ed. by Amy Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Aug. 2022. DOI: 10.4230/LIPIcs.FSCD.2022.3. URL: https://jozefg.github.io/papers/a-stratified-approach-to-lob-induction.pdf.

[Niu+22] Yue Niu, Jonathan Sterling, Harrison Grodin, and Robert Harper. "A Cost-Aware Logical Framework". In: *Proceedings of the ACM on Programming Languages* 6.POPL (Jan. 2022). DOI: 10.1145/3498670. arXiv: 2107.04663 [cs.PL].

[Ste21] Jonathan Sterling. "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory". CMU technical report CMU-CS-21-142. PhD thesis. Carnegie Mellon University, 2021. DOI: 10.5281/zenodo.5709838.

[SA21] Jonathan Sterling and Carlo Angiuli. "Normalization for Cubical Type Theory". In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2021, pp. 1–15. DOI: 10.1109/LICS52264.2021.9470719. arXiv: 2101.11479 [cs.LO].

[SH21] Jonathan Sterling and Robert Harper. "Logical Relations as Types: Proof-Relevant Parametricity for Program Modules". In: *Journal of the ACM* 68.6 (Oct. 2021). ISSN: 0004-5411. DOI: 10.1145/3474834. arXiv: 2010.08599 [cs.PL].

[SH22] Jonathan Sterling and Robert Harper. "Sheaf semantics of termination-insensitive noninterference". In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*. Ed. by Amy Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs) 15. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Aug. 2022. DOI: 10.4230/LIPIcs.FSCD.2022.15. arXiv: 2204.09421 [cs.PL].

References

[Alt+01] T. Altenkirch, P. Dybjer, M. Hofmann, and P. Scott. "Normalization by Evaluation for Typed Lambda Calculus with Coproducts". In: *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*. Washington, DC, USA: IEEE Computer Society, 2001. DOI: 10.1109/LICS.2001.932506.

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

[Ang+21a] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. "Syntax and models of Cartesian cubical type theory". In: *Mathematical Structures in Computer Science* 31.4 (2021), pp. 424–468. DOI: `10.1017/S0960129521000347`.

[Ang+21b] Carlo Angiuli, Evan Cavallo, Anders Mörtberg, and Max Zeuner. "Internalizing Representation Independence with Univalence". In: *Proceedings of the ACM on Programming Languages* 5.POPL (Jan. 2021), pp. 1–30. DOI: `10.1145/3434293`.

[AHH18] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities". In: *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*. Ed. by Dan Ghica and Achim Jung. Vol. 119. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 6:1–6:17. ISBN: 978-3-95977-088-0. DOI: `10.4230/LIPIcs.CSL.2018.6`. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9673`.

[AGV72] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *Théorie des topos et cohomologie étale des schémas.* Vol. 269, 270, 305. Lecture Notes in Mathematics. 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. Berlin: Springer-Verlag, 1972.

[Awo18] Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018), pp. 241–286. DOI: `10.1017/S0960129516000268`. arXiv: `1406.3219 [math.CT]`.

[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), pp. 45–55. ISSN: 0305-0041. DOI: `10.1017/S0305004108001783`.

[Bén73] Jean Bénabou. *Problèmes dans les topos : d'après le cours de Questions spéciales de mathématique.* Séminaires de mathématique pure : Rapport, no 34. 34. Louvain-la-Neuve : Institut de mathématique pure et appliquée, Université catholique de Louvain, 1973.

[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), pp. 3127–3169. arXiv: `1611.02108 [cs.LO]`.

[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, pp. 184–191. ISSN: 0304-3975. DOI: `10.1016/j.tcs.2019.01.015`. arXiv: `1810.09367 [cs.PL]`.

[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, pp. 120–134. ISBN: 978-3-540-70722-6.

[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: Association for Computing Machinery, 2002, pp. 26–37. ISBN: 1-58113-528-9. DOI: `10.1145/571157.571161`.

[FS99] Marcelo Fiore and Alex Simpson. "Lambda Definability with Sums via Grothendieck Logical Relations". In: *Typed Lambda Calculi and Applications.* Ed. by Jean-Yves Girard. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 147–161. ISBN: 978-3-540-48959-7.

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

[Gra22] Daniel Gratzer. "Normalization for Multimodal Type Theory". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science.* New York, NY, USA: Association for Computing Machinery, 2022. DOI: `10.1145/3531130.3532398`.

[GB22] Daniel Gratzer and Lars Birkedal. "A Stratified Approach to Löb Induction". In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022).* Ed. by Amy Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Aug. 2022. DOI: `10.4230/`

LIPIcs.FSCD.2022.3. URL: https://jozefg.github.io/papers/a-stratified-approach-to-lob-induction.pdf.

[HMM90] Robert Harper, John C. Mitchell, and Eugenio Moggi. “Higher-Order Modules and the Phase Distinction”. In: *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. San Francisco, California, USA: Association for Computing Machinery, 1990, pp. 341–354. ISBN: 0-89791-343-4. DOI: 10.1145/96709.96744.

[Heg69] Georg Wilhelm Friedrich Hegel. *Science of Logic*. Trans. by A. V. Miller. London: Allen & Unwin, 1969.

[Hof97] Martin Hofmann. “Syntax and Semantics of Dependent Types”. In: *Semantics and Logics of Computation*. Ed. by Andrew M. Pitts and Peter Dybjer. Cambridge University Press, 1997, pp. 79–130.

[Hof99] Martin Hofmann. “Semantical Analysis of Higher-Order Abstract Syntax”. In: *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 204–. ISBN: 0-7695-0158-3. URL: http://dl.acm.org/citation.cfm?id=788021.788940.

[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, pp. 83–111. DOI: 10.1093/oso/9780198501275.001.0001.

[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.

[Hyl91] J. M. E. Hyland. “First steps in synthetic domain theory”. In: *Category Theory*. Ed. by Aurelio Carboni, Maria Cristina Pedicchio, and Guiseppe Rosolini. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 131–156. ISBN: 978-3-540-46435-8.

[JM95] André Joyal and Ieke Moerdijk. *Algebraic Set Theory*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511752483.

[JT93] Achim Jung and Jerzy Tiuryn. “A new characterization of lambda definability”. In: *Typed Lambda Calculi and Applications*. Ed. by Marc Bezem and Jan Friso Groote. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 245–257. ISBN: 978-3-540-47586-6.

[Law94] F. William Lawvere. “Tools for the advancement of objective logic: closed categories and toposes”. In: *The logical foundations of cognition*. Ed. by John Macnamara and Gonzalo Reyes. 4. Oxford University Press on Demand, 1994.

[LS09] F. William Lawvere and Stephen H. Schanuel. *Conceptual Mathematics: A First Introduction to Categories*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 0-521-89485-9.

[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, pp. 153–175.

[Mar84] Per Martin-Löf. *Intuitionistic type theory. Notes by Giovanni Sambin*. Vol. 1. Studies in Proof Theory. Bibliopolis, 1984, pp. iv+91. ISBN: 88-7088-105-9.

[Mog89] Eugenio Moggi. “A Category-Theoretic Account of Program Modules”. In: *Category Theory and Computer Science*. Berlin, Heidelberg: Springer-Verlag, 1989, pp. 101–117. ISBN: 3-540-51662-X.

[Niu+22] Yue Niu, Jonathan Sterling, Harrison Grodin, and Robert Harper. “A Cost-Aware Logical Framework”. In: *Proceedings of the ACM on Programming Languages* 6.POPL (Jan. 2022). DOI: `10.1145/3498670`. arXiv: `2107.04663 [cs.PL]`.

[Shu19] Michael Shulman. *All* $(\infty, 1)$*-toposes have strict univalent universes*. Unpublished manuscript. Apr. 2019. arXiv: `1904.07004`.

[Ste21] Jonathan Sterling. “First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory”. CMU technical report CMU-CS-21-142. PhD thesis. Carnegie Mellon University, 2021. DOI: `10.5281/zenodo.5709838`.

[SA21] Jonathan Sterling and Carlo Angiuli. “Normalization for Cubical Type Theory”. In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2021, pp. 1–15. DOI: `10.1109/LICS52264.2021.9470719`. arXiv: `2101.11479 [cs.LO]`.

[SH21] Jonathan Sterling and Robert Harper. “Logical Relations as Types: Proof-Relevant Parametricity for Program Modules”. In: *Journal of the ACM* 68.6 (Oct. 2021). ISSN: 0004-5411. DOI: 10.1145/3474834. arXiv: 2010.08599 [cs.PL].

[SH22] Jonathan Sterling and Robert Harper. “Sheaf semantics of termination-insensitive noninterference”. In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*. Ed. by Amy Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs) 15. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Aug. 2022. DOI: 10.4230/LIPIcs.FSCD.2022.15. arXiv: 2204.09421 [cs.PL].

[Str14] Thomas Streicher. *Semantics of Type Theory Formulated in Terms of Representability.* Feb. 2014. URL: https://www2.mathematik.tu-darmstadt.de/~streicher/FIBR/natmod.pdf.

[Tai67] W. W. Tait. “Intensional Interpretations of Functionals of Finite Type I”. In: *The Journal of Symbolic Logic* 32.2 (1967), pp. 198–212. ISSN: 00224812. URL: http://www.jstor.org/stable/2271658.

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

[Voe06] Vladimir Voevodsky. “A very short note on homotopy λ-calculus”. Unpublished note. Sept. 2006. URL: https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Dynamic_logic/Stage_9_2012_09_01/2006_09_Hlambda.pdf.