(Cubical) Computability Structures

Jonathan Sterling

March 6, 2020

What is a type theory?

What is a type theory?

Answers I have espoused in the past:

A type theory is an indexed inductive definition of trees labeled by $\langle \Gamma\ \mathit{ctx} \rangle$, $\langle \Gamma \vdash A\ \mathit{type} \rangle$, $\langle \Gamma \vdash A \equiv B\ \mathit{type} \rangle$, $\langle \Gamma \vdash a : A \rangle$, and $\langle \Gamma \vdash a \equiv b : A \rangle$, satisfying some bizarre conditions.

What is a type theory?

Answers I have espoused in the past:

A type theory is an indexed inductive definition of trees labeled by $\langle \Gamma\ \mathit{ctx} \rangle$, $\langle \Gamma \vdash A\ \mathit{type} \rangle$, $\langle \Gamma \vdash A \equiv B\ \mathit{type} \rangle$, $\langle \Gamma \vdash a : A \rangle$, and $\langle \Gamma \vdash a \equiv b : A \rangle$, satisfying some bizarre conditions.

A type theory is a syntactic object in a logical framework (such as the theory GATs / *Generalized Algebraic Theories* [Car86]).

What is a type theory?

Answers I have espoused in the past:

A type theory is an indexed inductive definition of trees labeled by $\langle \Gamma\ \mathit{ctx} \rangle$, $\langle \Gamma \vdash A\ \mathit{type} \rangle$, $\langle \Gamma \vdash A \equiv B\ \mathit{type} \rangle$, $\langle \Gamma \vdash a : A \rangle$, and $\langle \Gamma \vdash a \equiv b : A \rangle$, satisfying some bizarre conditions.

A type theory is a syntactic object in a logical framework (such as the theory GATs / *Generalized Algebraic Theories* [Car86]).

A type theory is a category equipped with some kind of stable class of carrable maps [Tay99; Joy17; Shu15; Uem19].

What is a type theory?

Answers I have espoused in the past:

A type theory is an indexed inductive definition of trees labeled by $\langle \Gamma\ \mathit{ctx} \rangle$, $\langle \Gamma \vdash A\ \mathit{type} \rangle$, $\langle \Gamma \vdash A \equiv B\ \mathit{type} \rangle$, $\langle \Gamma \vdash a : A \rangle$, and $\langle \Gamma \vdash a \equiv b : A \rangle$, satisfying some bizarre conditions.

A type theory is a syntactic object in a logical framework (such as the theory GATs / *Generalized Algebraic Theories* [Car86]).

A type theory is a category equipped with some kind of stable class of carrable maps [Tay99; Joy17; Shu15; Uem19].

First two notions legitimate, but not good definitions of "a type theory"; confusing "has" for "is".

Functorial semantics of type theories

Definition (Uemura [Uem19])

A representable map category is a lex category together with a class of "representable maps" closed under isomorphism, composition, pullback, and along which pushforwards exist.

- Judgmental structure of (non-modal) strict type theories with many judgments == representable map category.
- Hypothetical judgment == pushforward (only available for representable maps!).

Example (Natural model)

Let $\mathbb{T}$ be the smallest type theory containing a representable map $\tilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$. A rep-map functor $\mathbb{T} \longrightarrow \mathcal{P}\mathrm{r}(\mathcal{C})$ is a natural model over $\mathcal{C}$ in the sense of Awodey [Awo18].

What is a type theorist?

What is a type theorist?

My answer: Someone who studies the properties of the term model that *are not* preserved by homomorphisms of models.

Why do we care about that? Necessary for *implementing* type theoretic languages in a computer (typechecking).

What is a type theorist?

My answer: Someone who studies the properties of the term model that *are not* preserved by homomorphisms of models.

Why do we care about that? Necessary for *implementing* type theoretic languages in a computer (typechecking).

Example (Injectivity of type constructors)

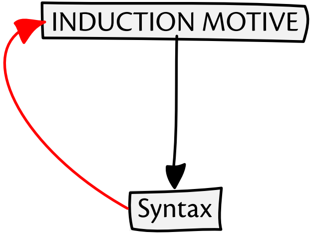
If $\Gamma \vdash A \to B \equiv C \to D\ \mathit{type}$, then $\Gamma \vdash A \equiv C\ \mathit{type}$ and $\Gamma \vdash B \equiv D\ \mathit{type}$.

- Necessary for completeness of algorithmic definitional equality, typechecking.
- Goes beyond the language of representable map categories, therefore not preserved by homomorphisms.

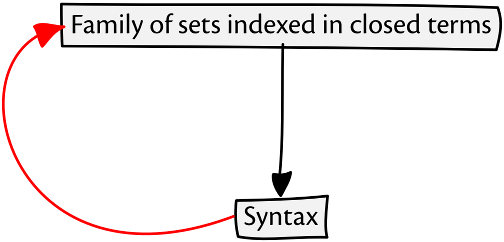
Studying syntax using semantics

First of all, syntax *is* semantics (Lawvere); special because it has a *universal property*.

Interesting theorems about syntax are proved by choosing a family lying over the term model (induction motive), and exhibiting a section using this universal property.

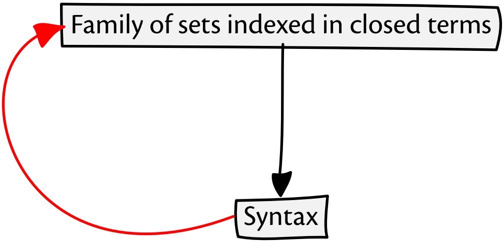


Example: Canonicity [Cro93; Shu15; Coq19]



To get the section, we must build a *model* of the type theory out of families of sets indexed in closed terms!

Example: Canonicity [Cro93; Shu15; Coq19]

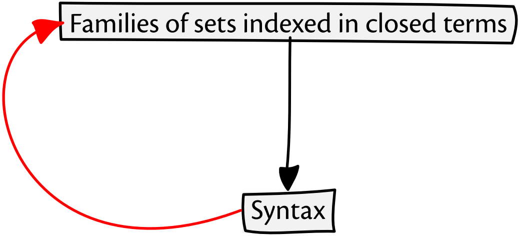


To get the section, we must build a *model* of the type theory out of families of sets indexed in closed terms!

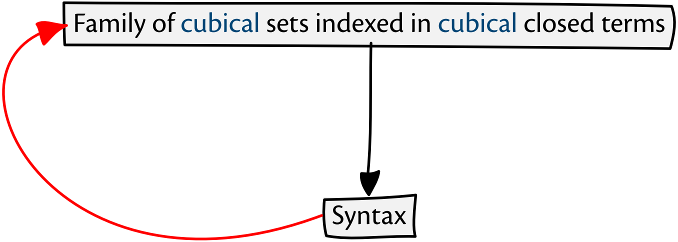
1. Contexts are families of sets indexed in the closed substitutions for a context. For instance, the canonical family $\mathbb{N} \longrightarrow \{\diamond \longrightarrow \mathsf{nat}\}$.
2. Substitutions are morphisms of families of sets, tracked by a syntactic substitution.

The section exhibits for each closed term $\diamond \vdash N : \mathsf{nat}$ a numeral $n \in \mathbb{N}$ encoded by N. □

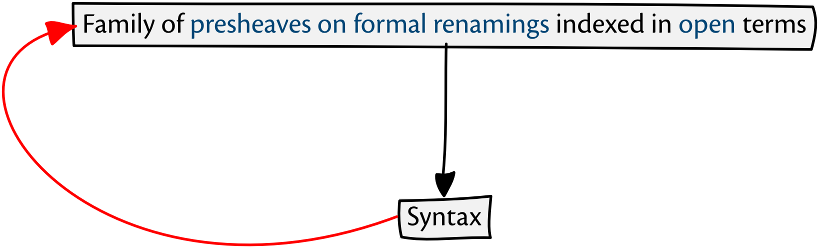
Example: Cubical Canonicity [Awo15; CHS19; SAG19; SAG20]



Example: Cubical Canonicity [Awo15; CHS19; SAG19; SAG20]



Example: Normalization [AHS95; Fio02; Coq19]



Artin gluing and Tait's method

In the 1960s and 1970s, both geometers and logicians were searching for useful ways to combine existing (spaces, theories, models) into new (spaces, theories, models). Unfortunately they didn't talk to each other much.

- In 1967, Tait develops his ingenious *method of computability* to prove a normalization theorem for typed λ-calculus [Tai67], by attaching *predicates* (proof-irrelevant families) to *raw* terms.
- In the early 1970s, the Grothendieck school develops what is now called *Artin gluing* [AGV72], a way to parameterize a topos in another topos.
- In 1978, Freyd rationalizes Tait's trick as an instance of Artin gluing [Fre78], passing from proof-irrelevant families on raw terms to proof-relevant families on abstract terms.
- In 2015–2020, a flurry of activity in applying gluing to type theory, most notably by Shulman and Coquand [Shu13; Shu15; AK16; SS18; Coq19; KHS19; KS19; CHS19; SAG19; SA20; SAG20].

Artin gluing

Let $\mathcal{C}$ be a structured category of some kind, and let $\mathcal{C} \xrightarrow{F} \mathcal{E}$ be a functor (typically, preserving some finite limits).

$$\begin{tikzcd}
\mathcal{G} \arrow[r] \arrow[d, "\mathrm{gl}"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathcal{E}^{\to} \arrow[d, "\mathrm{cod}"] \\
\mathcal{C} \arrow[r, "F"'] & \mathcal{E}
\end{tikzcd}$$

An "Artin gluing theorem" says that $\mathcal{G}$ can be equipped with the same structures, and that $\mathcal{G} \xrightarrow{\mathrm{gl}} \mathcal{C}$ preserves it. We have Artin gluing theorems for many kinds of theory [AGV72; CJ95; KHS19; SA20].

Gluing models of type theory along flat functors [SA20]

Different theorems can be proved by gluing along different functors:

- The global sections functor $\mathcal{C} \longrightarrow \mathbf{Set}$ proves canonicity.
- The Yoneda embedding $\mathcal{C} \longrightarrow \mathcal{P}\mathrm{r}(\mathcal{C})$ proves definability results.
- Let $\mathcal{R} \longrightarrow \mathcal{C}$ be the category of *formal contexts and formal renamings*; the resulting *renaming nerve* functor $\mathcal{C} \longrightarrow \mathcal{P}\mathrm{r}(\mathcal{R})$ proves normalization [Fio02; Coq19].
- Let $\mathcal{C}$ carry an *interval object*, i.e. a f.p. functor $\square \longrightarrow \mathcal{C}$. Resulting *renaming nerve* functor $\mathcal{C} \longrightarrow \mathcal{P}\mathrm{r}(\square)$ proves cubical canonicity [SAG19; SAG20], as suggested by Awodey and Fiore.

General theorem (S., Angiuli): you can glue along *flat functors* from a model of MLTT into a Grothendieck topos [SA20].

Less general than Kaposi, Huber, and Sattler [KHS19], but avoids pernicious/anti-categorical *equality of objects*.

Gluing models of type theory along flat functors [SA20]

Origin of this work was efforts to exploit the classical theory of Artin gluing for Grothendieck topoi, to simplify the corresponding proofs for models of type theory (not usually topoi!!).

Let $\mathcal{C}$ be the category of contexts of a model of MLTT, and let $\mathcal{E}$ be a Grothendieck topos. When does $\mathcal{G}$ carry a model of type theory?

$$\begin{tikzcd}
\mathcal{G} \arrow[r] \arrow[d, "\mathsf{gl}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\to} \arrow[d, "\mathsf{cod}"] \\
\mathcal{C} \arrow[r, "F"'] & \mathcal{E}
\end{tikzcd}$$

Probably not enough for F to preserve finite limits ($\mathcal{C}$ barely has any limits!). Generally very painful because we must show existence and preservation of type-theoretic structure simultaneously.

Power Move: factor through Yoneda

$$\begin{tikzcd}
\mathcal{G}_K \arrow[r, hook, "K"] \arrow[d, "\mathsf{gl}_K"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{G} \arrow[r] \arrow[d, "\mathsf{gl}"] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\to} \arrow[d, "\mathsf{cod}"] \\
\mathcal{C} \arrow[r, hook, "\mathsf{y}"'] & \mathcal{P}r(\mathcal{C}) \arrow[r, "\hat{F}"'] & \mathcal{E}
\end{tikzcd}$$

- **Artin, Grothendieck, Verdier:** if $\hat{F}$ lex & accessible, then $\mathcal{G}$ a topos and gl a logical morphism [AGV72]!
- **Diaconescu:** $\mathcal{P}r(\mathcal{C})$ is the classifying topos for the theory of flat functors out of $\mathcal{C}$; therefore $\hat{F}$ lex iff F is flat [Bor94].
- **S., Angiuli:** $\mathcal{G}_K \overset{K}{\hookrightarrow} \mathcal{G}$ dense, therefore the nerve $\mathcal{G} \longrightarrow \mathcal{P}r(\mathcal{G}_K)$ fully faithful.

Easier, more abstract! Work in the internal language of $\mathcal{G}$ (“general computability structures), then transfer to model of type theory over $\mathcal{G}_K$ (“compact computability structures”) via nerve.

Power Move: factor through Yoneda

$$\begin{tikzcd}
\mathcal{G}_{\mathsf{K}} \arrow[r, hook, "\mathsf{K}"] \arrow[d, "\mathsf{gl}_{\mathsf{K}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{G} \arrow[r] \arrow[d, "\mathsf{gl}"] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\rightarrow} \arrow[d, "\mathsf{cod}"] \\
\mathcal{C} \arrow[r, hook, "\mathsf{y}"'] & \mathcal{P}\mathrm{r}(\mathcal{C}) \arrow[r, "\hat{F}"'] & \mathcal{E}
\end{tikzcd}$$

Easier, more abstract! Work in the internal language of $\mathcal{G}$ ("general computability structures), then transfer to model of type theory over $\mathcal{G}_{\mathsf{K}}$ ("compact computability structures") via nerve.

Advantage of working in $\mathcal{G}$ atop $\mathcal{P}\mathrm{r}(\mathcal{C})$: the judgmental structure (natural model) of type theory lives in $\mathcal{P}\mathrm{r}(\mathcal{C})$, so we may define a computability structure lying over it directly.

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*.

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps!

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps! $\mathbf{comp}_{i.A}^{i \rightsquigarrow j}[k.k = i \to a]$ *is normal form, but* $\mathbf{comp}_{i.A}^{i \rightsquigarrow i}[k.k = i \to a]$ *must reduce. Therefore, diagonal substitution must be killed.*

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps! $\mathbf{comp}_{i.A}^{i \rightsquigarrow j}[k.k = i \rightarrow a]$ *is normal form, but* $\mathbf{comp}_{i.A}^{i \rightsquigarrow i}[k.k = i \rightarrow a]$ *must reduce. Therefore, diagonal substitution must be killed.*
2. Glue along change of base $\mathcal{P}\mathrm{r}(\mathcal{C}) \xrightarrow{\epsilon^*} \mathcal{P}\mathrm{r}(\mathcal{R})$ to get glued judgments (general computability structures), pull back along Yoneda to get glued contexts (compact computability structures); latter is gluing along cubical nerve $\mathcal{C} \xrightarrow{\mathsf{N}_\epsilon} \mathcal{P}\mathrm{r}(\mathcal{R})$.

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps! $\mathbf{comp}_{i.A}^{i \rightsquigarrow j}[k.k = i \rightarrow a]$ *is normal form, but* $\mathbf{comp}_{i.A}^{i \rightsquigarrow i}[k.k = i \rightarrow a]$ *must reduce. Therefore, diagonal substitution must be killed.*
2. Glue along change of base $\mathcal{P}\mathrm{r}(\mathcal{C}) \xrightarrow{\epsilon^*} \mathcal{P}\mathrm{r}(\mathcal{R})$ to get glued judgments (general computability structures), pull back along Yoneda to get glued contexts (compact computability structures); latter is gluing along cubical nerve $\mathcal{C} \xrightarrow{\mathsf{N}_\epsilon} \mathcal{P}\mathrm{r}(\mathcal{R})$.
3. **Characterize neutral & normal forms of both types and elements!** Distinguished (non-compact) computability in the gluing fibration.

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps! *$\mathbf{comp}_{i.A}^{i \rightsquigarrow j}[k.k = i \to a]$ is normal form, but $\mathbf{comp}_{i.A}^{i \rightsquigarrow i}[k.k = i \to a]$ must reduce. Therefore, diagonal substitution must be killed.*
2. Glue along change of base $\mathcal{P}\mathrm{r}(\mathcal{C}) \xrightarrow{\epsilon^*} \mathcal{P}\mathrm{r}(\mathcal{R})$ to get glued judgments (general computability structures), pull back along Yoneda to get glued contexts (compact computability structures); latter is gluing along cubical nerve $\mathcal{C} \xrightarrow{\mathsf{N}_\epsilon} \mathcal{P}\mathrm{r}(\mathcal{R})$.
3. **Characterize left-normal & right-normal forms of both types and elements!** Distinguished (non-compact) computability in the gluing fibration.

(Future) Application: Cubical NbE (NbG)

1. Choose notion $\mathcal{R} \xrightarrow{\epsilon} \mathcal{C}$ of *formal renaming*. "Interval" in $\mathcal{R}$ only symmetric monoidal, no finite products, no face maps! $\mathbf{comp}_{i.A}^{i \rightsquigarrow j}[k.k = i \rightarrow a]$ *is normal form, but* $\mathbf{comp}_{i.A}^{i \rightsquigarrow i}[k.k = i \rightarrow a]$ *must reduce. Therefore, diagonal substitution must be killed.*
2. Glue along change of base $\mathcal{P}\mathrm{r}(\mathcal{C}) \xrightarrow{\epsilon^*} \mathcal{P}\mathrm{r}(\mathcal{R})$ to get glued judgments (general computability structures), pull back along Yoneda to get glued contexts (compact computability structures); latter is gluing along cubical nerve $\mathcal{C} \xrightarrow{\mathsf{N}_\epsilon} \mathcal{P}\mathrm{r}(\mathcal{R})$.
3. **Characterize left-normal & right-normal forms of both types and elements!** Distinguished (non-compact) computability in the gluing fibration.

$$
\begin{tikzcd}
\tilde{\mathbf{L}} \arrow[r, "\boldsymbol{\lambda}"] \arrow[d] & \mathbf{L} \arrow[d] & \tilde{\mathbf{R}} \arrow[r, "\boldsymbol{\rho}"] \arrow[d] & \mathbf{R} \arrow[d] & \mathcal{G} \arrow[d] \\
\tilde{\mathbf{T}} \arrow[r, "\boldsymbol{\tau}"'] & \mathbf{T} & \tilde{\mathbf{T}} \arrow[r, "\boldsymbol{\tau}"'] & \mathbf{T} & \mathcal{P}\mathrm{r}(\mathcal{C})
\end{tikzcd}
$$

Saturation of computability structures

Tait Power Move: left-normal forms are computable, and computable elements are right-normal.

Saturation of computability structures

Tait Power Move: left-normal forms ~~are computable~~, and computable elements ~~are right-normal~~.

Saturation of computability structures

Tait–Altenkirch–Hofmann–Streicher–Fiore–Coquand Power Move: left-normal forms **have computability structures**, and **computability structures have normal forms**.

Saturation of computability structures

Tait–Altenkirch–Hofmann–Streicher–Fiore–Coquand Power Move: left-normal forms **have computability structures**, and **computability structures have normal forms**.

Require vertical "reify" / "reflect" maps in the language of the gluing fibration:

$$
\begin{tikzcd}
\mathbf{L} \arrow[r, "\uparrow"] \arrow[dr, mapsto] & \mathcal{G}\langle \mathbf{T} \rangle \arrow[r, "\downarrow"] \arrow[d, mapsto] & \mathbf{R} \arrow[dl, mapsto] & & \mathcal{G} \arrow[d] \\
& \mathbf{T} & & & \mathcal{P}\mathrm{r}(\mathcal{C})
\end{tikzcd}
$$

Saturation of computability structures

Tait–Altenkirch–Hofmann–Streicher–Fiore–Coquand Power Move: left-normal forms **have computability structures**, and **computability structures have normal forms**.

Require vertical "reify" / "reflect" maps in the language of the gluing fibration:

$$
\begin{tikzcd}
\uparrow_! \boldsymbol{\lambda} \arrow[r, "\Uparrow"] \arrow[dr, mapsto] & \mathcal{G}\langle \boldsymbol{\tau} \rangle \arrow[r, "\Downarrow"] \arrow[d, mapsto] & \downarrow^* \boldsymbol{\rho} \arrow[dl, mapsto] & \mathcal{G}_{/\mathcal{G}\langle \boldsymbol{\tau} \rangle} \arrow[d] \\
& \boldsymbol{\tau} & & \mathcal{P}\mathrm{r}(\mathcal{C})_{/\boldsymbol{\tau}}
\end{tikzcd}
$$

Applications to programming languages

Programming languages currently understood through transition systems instead of semantics — and it's not for lack of trying!

On the other hand, PL landscape has richer and more sophisticated use of Tait's method/computability than type theory. **Can we attempt reunification?**

Harper & Sterling: Study phase separation in ML module systems using Artin gluing at two levels:

1. Gluing appears to capture the general abstract content of static/dynamic phase separation (after Moggi [Mog89] and Harper, Mitchell, and Moggi [HMM89]).
2. Gluing can be used to prove representation independence / parametricity results for module languages.

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.

[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.

References II

[AK16] 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.

[Awo15] Steve Awodey. “Personal communication to Michael Shulman”. 2015.

[Awo18] Steve Awodey. “Natural models of homotopy type theory”. In: *Mathematical Structures in Computer Science* 28.2 (2018). DOI: 10.1017/S0960129516000268.

References III

[Bor94] Francis Borceux. *Handbook of Categorical Algebra*. Vol. 1. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511525858.

[Car86] John Cartmell. "Generalised Algebraic Theories and Contextual Categories". In: *Annals of Pure and Applied Logic* 32 (1986). ISSN: 0168-0072.

[CHS19] Thierry Coquand, Simon Huber, and Christian Sattler. "Homotopy canonicity for cubical type theory". 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.

References IV

[CJ95] Aurelio Carboni and Peter Johnstone. "Connected limits, familial representability and Artin glueing". In: *Mathematical Structures in Computer Science* 5.4 (1995). DOI: 10.1017/S0960129500001183.

[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.

[Cro93] R. L. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. New York: Cambridge University Press, 1993.

References V

[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.

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

[Geu19] Herman Geuvers, ed. Vol. 131. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

References VI

[HMM89] 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, 1989. DOI: 10.1145/96709.96744.

[Joy17] Andre Joyal. *Notes on Clans and Tribes*. 2017. arXiv: 1710.10238.

[KHS19] Ambrus Kaposi, Simon Huber, and Christian Sattler. “Gluing for type theory”. 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.

References VII

[KS19] Chris Kapulkin and Christian Sattler. *Homotopy canonicity of homotopy type theory*. Slides from a talk given at the International Conference on Homotopy Type Theory (HoTT 2019). Aug. 2019. URL: `https://hott.github.io/HoTT-2019/conf-slides/Sattler.pdf`.

[Mog89] Eugenio Moggi. "A Category-Theoretic Account of Program Modules". In: *Category Theory and Computer Science*. Berlin, Heidelberg: Springer-Verlag, 1989.

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

References VIII

[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". In preparation. 2020.

[Shu13] Michael Shulman. *Scones, Logical Relations, and Parametricity*. Blog. 2013. URL: `https://golem.ph.utexas.edu/category/2013/04/scones_logical_relations_and_p.html`.

References IX

[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].

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

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge studies in advanced mathematics. Cambridge, New York (N. Y.), Melbourne: Cambridge University Press, 1999.

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