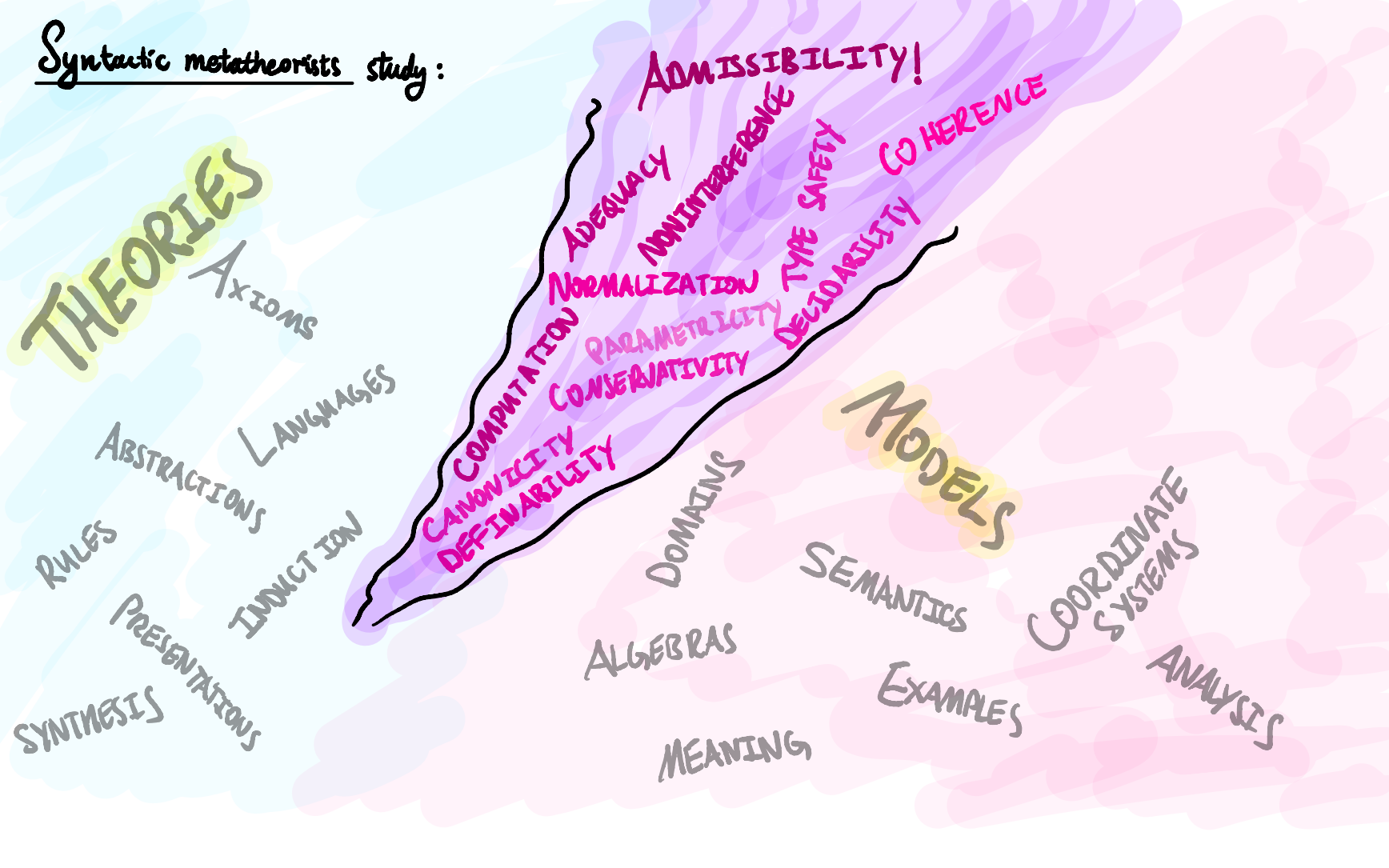
Towards a Geometry for syntax.

Jon Sterling
Aarhus University

"Σχῆμά ἐστι τὸ ὑπό τινος ἢ τινων ὅρου περιεχόμενον."

— Euclid


Syntactic metatheorists study:
ADMISSIBILITY!
THEORIES
AXIOMS
LANGUAGES
ABSTRACTIONS
RULES
INDUCTION
PRESENTATIONS
SYNTHESIS
ADEQUACY
NONINTERFERENCE
NORMALIZATION
PARAMETRICITY
CONSERVATIVITY
CANONICITY
DEFINABILITY
COMPUTATION
TYPE SAFETY
DECIDABILITY
COHERENCE
MODELS
DOMAINS
SEMANTICS
ALGEBRAS
EXAMPLES
MEANING
COORDINATE SYSTEMS
ANALYSIS

There are three kinds of facts that can be known about a theory:

Derivability

Statements that make sense inside the doctrine of a theory.

e.g. "$x:A \vdash \varphi_x$ type"

Can always be proved using the rules of the theory

Non-derivability

e.g. consistency.

Goes outside the theory.

Proved by means of

~ countermodels ~

(denotational semantics)

Geometrical / topological models

Admissibility

Positive statements about derivability that go outside the theory.

e.g. normalization, decidability, computational adequacy

Proved by means of

(Kripke) logical relations.

Non-derivability is studied geometrically

To show that $\not\vdash 0 \equiv 1 : \mathsf{nat}$ in PCF, we interpret PCF into a category of geometric spaces, e.g. pointed dcpos.

$$\begin{aligned} [\![\mathsf{nat}]\!] &= \mathbb{N}_\perp \\ [\![0]\!] &= 0 \\ [\![1]\!] &= 1 \end{aligned}$$

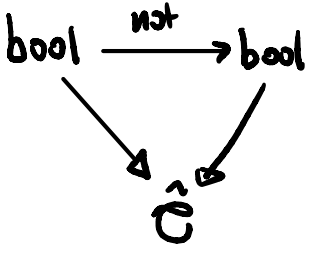
Suppose that $\vdash 0 \equiv 1 : \mathsf{nat}$. Then $[\![0]\!] = [\![1]\!] \in \mathbb{N}_\perp$, a contradiction. $\square$

But derivability can also be studied geometrically...

Example (STλC)

* The typed λ-calculus has two sorts (judgments): types and terms.
* We have a "category of contexts" $\mathcal{C}$, a free cartesian closed category.
* The category of presheaves $Pr(\mathcal{C}) = [\mathcal{C}^{op}, Set]$ allows us to treat each type of STλC as an étale space/sheaf over a topos (=generalized space) $\hat{\mathcal{C}}$. [Write $Set_{\hat{\mathcal{C}}} = Pr(\mathcal{C})$].

Derivability is captured by morphisms of étale spaces:

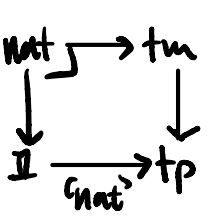


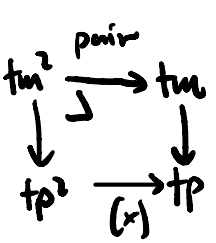
$\in \mathrm{Set}_{\hat{\mathcal{C}}} \subseteq \mathrm{Topos}/\hat{\mathcal{C}}$

$\in$ Topos

But $\mathrm{Set}_{\hat{\mathcal{C}}}$ contains many useful spaces beyond those that come from STLC. For instance, there is a "generic type" $(\mathrm{tm} \longrightarrow \mathrm{tp})$ in $\mathrm{Set}_{\hat{\mathcal{C}}}$ from which every type of STLC arises by pullback:

[[Lumsdaine–Warren, Awodey]
"local universe"





The <u>generalized</u> derivability embodied in $\hat{\mathcal{C}}$ allows us to express many useful concepts, such as schematic polymorphism:

e.g. $\prod_{\alpha:\mathrm{tp}} \mathrm{tm}[\alpha \Rightarrow \alpha]$

Interesting "noninterference" properties are reflected in the internal language of $\mathrm{Set}_{\hat{\mathcal{C}}}$. For instance:

$$\hat{\mathcal{C}} \models \forall A{:}\,\mathrm{bool} \Rightarrow \mathrm{tp}.\ \ulcorner A \text{ is constant} \urcorner.$$

(Because STλC doesn't have dependent types!)

Other "generalized" derivabilities provide useful reduction techniques

—See A. Kock's observation that the generic local ring is a field!

What about ADMISSIBILITY?

(What even is admissibility?)

Admissibility

In a formal system, the rule $\dfrac{\Gamma \vdash \mathcal{J}}{\Gamma' \vdash \mathcal{J}'}$ is "admissible" when the derivability of $(\Gamma \vdash \mathcal{J})$ implies the derivability of $(\Gamma' \vdash \mathcal{J}')$.

Very imprecise!

What actually are $\Gamma, \Gamma', \mathcal{J}, \mathcal{J}'$? Do they have "parameters"?

What kind of transformation $(\Gamma \vdash \mathcal{J}) \rightsquigarrow (\Gamma' \vdash \mathcal{J}')$ is allowed? Should it commute with replacement of "parameters"?

A more rational definition is needed. START FROM THE GEOMETRY!

Example: injectivity of type constructors.

Injectivity of $(\Rightarrow)$

$$\frac{\vdash A \to B \equiv A' \to B' : \mathcal{U}}{\vdash A \equiv A' : \mathcal{U} \quad \vdash B \equiv B' : \mathcal{U}} \qquad (*)$$

Over the syntactic space $\hat{\mathcal{C}}$, we first note that the following statement is <u>not</u> true, corresponding to the non-derivability of (*):

$$\hat{\mathcal{C}} \nvdash \ulcorner (\Rightarrow) : \mathcal{U}^2 \to \mathcal{U} \text{ is injective} \urcorner \qquad (\dagger)$$

Easily refuted using a countermodel. But what were the <u>syntactical</u> obstructions? Roughly that the "proof" of injectivity is not <u>natural</u> / does not commute with substitution.

New plan:

Find a way to "enlarge" $\hat{C}$ to make (†) true, without losing the connection to syntax.

Let $\mathcal{A}$ be the cartesian category of contexts and renamings of variables (not all substitutions!).

We have:

$$\mathcal{A} \xrightarrow{\alpha} \mathcal{C} : \mathrm{CAT}$$

hence:

$$\hat{\mathcal{A}} \xrightarrow{\hat{\alpha}} \hat{\mathcal{C}} : \mathrm{Topos}$$

given by:

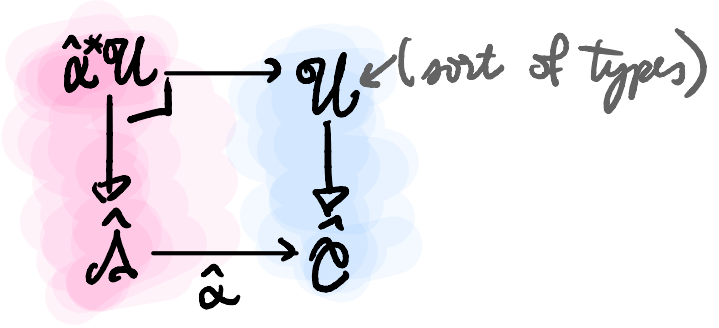
$$\hat{\alpha}^* := (E : \mathrm{Set}_{\hat{\mathcal{C}}}) \longmapsto (\lambda a : \mathcal{A}^{op}.\ E(\alpha(a)))$$

$\hat{\alpha}^* \dashv \hat{\alpha}_*$

precomposition

Idea: restrict the sorts of our language along $\hat{\alpha}: \hat{\mathcal{A}} \to \hat{\mathcal{C}}$:

e.g.



Theorem. We have:

$$\hat{\mathcal{A}} \models \ulcorner \hat{\alpha}^*(=) : \hat{\alpha}^* \mathcal{U}^2 \to \hat{\alpha}^* \mathcal{U} \text{ is injectiv} \urcorner$$

Theorem. We have:

$$\hat{\mathcal{A}} \vDash \ulcorner \hat{\alpha}^*(\Rightarrow) : \hat{\alpha}^* \mathcal{U}^2 \to \hat{\alpha}^* \mathcal{U} \quad \text{is injective} \urcorner$$

↑ Suggests a notion of "α-admissibility" for each $\alpha : \mathcal{A} \to \mathcal{C}$.

But how do we prove such admissibilities?

'Kripke Logical Predicates'

In essence, we need to do an 'inductive proof' over $\mathbb{C}$, where the motive of induction is valued in $\mathrm{Set}_{/\hat{A}}$. That's what Kripke logical relations do!

1) To each $C : \mathbb{C}$, assign a subobject of $\llbracket C \rrbracket \downarrow \hat{\alpha}^* y_{\mathbb{C}} C : \mathrm{Set}_{/\hat{A}}$

2) To each $c \xrightarrow{f} d : \mathbb{C}$, assign a diagram:

$$\begin{array}{ccc} \llbracket c \rrbracket & \xrightarrow{\llbracket f \rrbracket} & \llbracket d \rrbracket \\ \downarrow & & \downarrow \\ \hat{\alpha}^* y_{\mathbb{C}} c & \xrightarrow[\hat{\alpha}^* y_{\mathbb{C}} f]{} & \hat{\alpha}^* y_{\mathbb{C}} d \end{array}$$

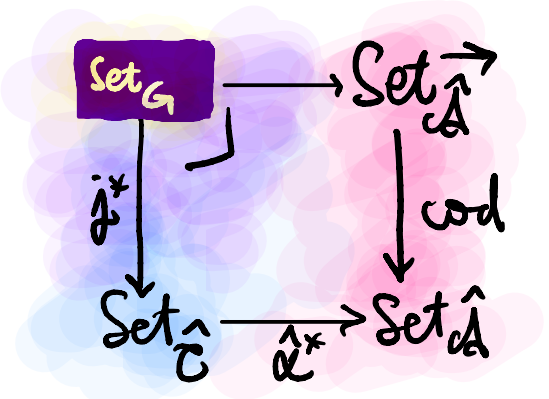
(Functorially)

But this is almost a comma category / Artin gluing!

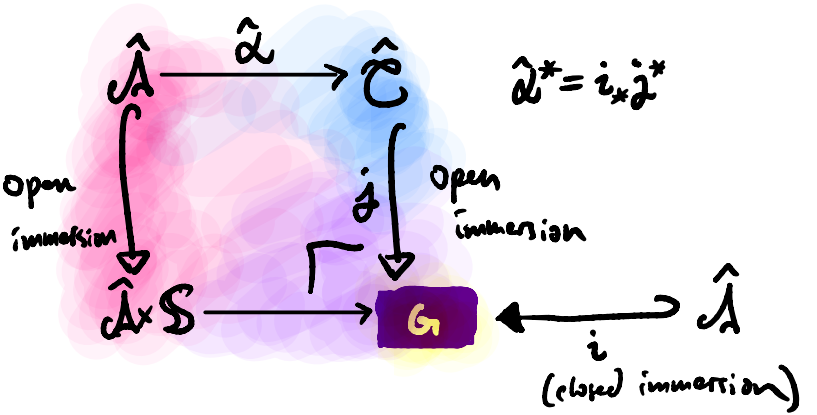
$$\begin{array}{ccc} \mathrm{KrPred}_{\mathbb{C}} & \longrightarrow & \mathrm{Sub}(\mathrm{Set}_{/\hat{A}}) \\ \downarrow & & \downarrow \\ \mathbb{C} & \xrightarrow{y_{\mathbb{C}}} \mathrm{Set}_{\mathbb{C}} \xrightarrow{\tilde{\alpha}^*} & \mathrm{Set}_{/\hat{A}} \end{array}$$

We may simplify things by relaxing the requirement that $\mathcal{C}\downarrow\hat{\alpha}^* Y_{\mathcal{C}^{\mathrm{op}}}$ be a mono.

We have a category (topos!) of generalized, proof-relevant Kripke Logical Predicates:

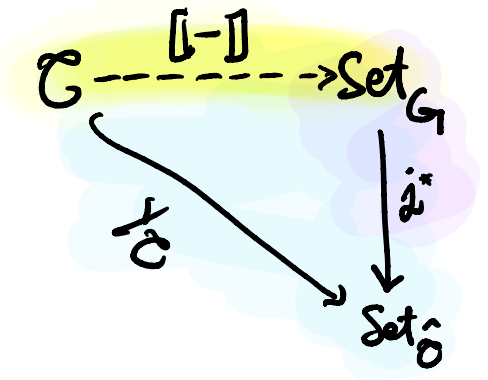


(in category theory)



(in topos geometry)

To make a Kripke logical predicate model, "just" define a structure preserving functor:



IDEA:

Just as the internal language of $Set_{\hat{\mathcal{O}}}$ has provided a rational/objective basis to study derivability,

we may study α-admissibility by grasping the internal language of Set_G.

I characterize the internal language of Set_G by a simple set of postulates called:

SYNTHETIC TAIT COMPUTABILITY

Euclid's Postulates for syntactic metatheory?

We abstract the following properties of Set_G:

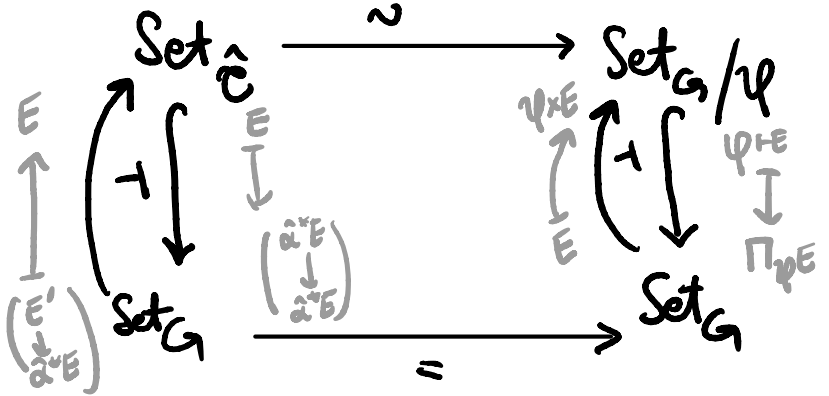
1) Ext. Martin-Löf type theory with universes $\mathcal{U}_i, \Omega$ and QITs.

2) There exists a proposition $\varphi : \Omega$.

Def.: A type E is "φ-modal" when $E \longrightarrow E^{\varphi}$ is iso
"φ-connected" " $E \times \varphi \longrightarrow \varphi$ is iso

In Set_G, φ is the subterminal $\left(\begin{array}{ccc} \emptyset & \longrightarrow & \amalg \\ \downarrow & & \| \\ \hat{\alpha}^* \amalg & = & \hat{\alpha}^* \amalg \end{array}\right)$

Then we reconstruct $\mathrm{Set}_{\hat{\mathcal{C}}}$ as $\mathrm{Set}_{G/\varphi}$. [Artin, Grothendieck, Ex. 1962]



3) Given a φ-modal type E and a family $x:E \vdash F[x]$ of φ-connected types, then exists a type $E \ltimes F$ and an iso. $\Sigma_E F \xrightarrow{\text{glue}} E \ltimes F$ s.t. $-:\varphi \vdash E \ltimes F \equiv E$ strictly and $-:\varphi \vdash \text{glue} \equiv \pi_1 : \Sigma_E F \to E$.

These axioms are called **Synthetic Tait Computability**, after Bill Tait's introduction of the method of computability / logical predicates in the late 1960s.

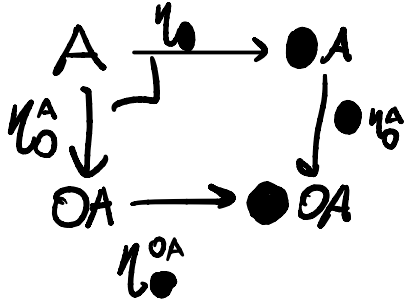
Two modalities:

[Rijke, Shulman, Spitters 2017]

$\bigcirc A := \varphi \Rightarrow A$ (open modality)

$\bullet A := \varphi \sqcup_{\varphi \times A} A$ [closed modality]

Facts: $\bigcirc \bullet A = \mathbb{1}$.



Example:

$$[\![\text{bool}]\!] = \sum_{b : \bigcirc \text{bool}} \bullet \left\{ i : 2 \;\middle|\; \begin{array}{l} i = 0 \to b = \text{true} \\ i = 1 \to b = \text{false} \end{array} \right\}$$

$[\![\text{true}]\!] = (\text{true}, \eta_\bullet\, 0)$

$[\![\text{false}]\!] = (\text{false}, \eta_\bullet\, 1)$

With just (*) these axioms, we can reconstruct most Kripke Logical Predicate arguments in a simpler, less error-prone way:

* Canonicity and normalization for MLTT (S.)
* Parametricity and data abstraction for ML modules (S., Harper)
* Computational adequacy for denotational semantics à la Plotkin (S.)

But we have also proved <u>new</u> theorems that were previously out of reach:

* Normalization & decidability of type-checking for cubical type theory (S., Angiuli)
* Normalization & decidability of type checking for multi-modal type theory (Gratzer)

Prospects for language design

STC is not only a tool to nuke hard theorems.

The STC modalities express a form of non-interference or phase distinction, e.g. between syntax & semantics:

Any map $\bullet A \to \bigcirc B$ is constant.

But syntax/semantics is not the only useful phase distinction!

Phase Distinctions & Noninterference for language design

1) Static/dynamic phase separation in ML Modules (Harper, Mitchell, Moggi 1990) reconstructed by S. & Harper in our 2021 J.ACM paper.

2) Behavior/complexity: Niu, S., Grodin, Harper (POPL 2022)

3) Public vs. private: STC for security typing / IFC.
Termination-insensitive noninterference for security modalities.

Thanks!

See my website jonmsterling.com for links to papers, including my PhD thesis:

> First Steps in Synthetic Tait Computability: the Objective Metatheory of Cubical Type Theory.