A Type Theory for Comprehension Categories

Niyousha Najmaei, Niels van der Weide, Benedikt Ahrens, Paige Randall North

HoTTEST, February 2026

Slides slightly adapted from Niyousha's — thanks a lot!

Setting

- Martin-Löf type theory (MLTT) serves as a foundation for proof assistants and programming languages
- Several well-established categorical semantics: contextual categories, categories with families, display map categories, natural models, ...
- Comprehension categories as a general framework to organize these notions [ALN24]:

  "We take comprehension categories as a unifying language and show how almost all established notions of model embed as sub-2-categories (usually full) of the 2-category of comprehension categories."

Motivation

Interpretation of MLTT in comprehension categories:

$$\text{MLTT} \xrightarrow{\text{semantics}} \begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \text{cod}} & \\ & \mathcal{C} & \end{array}$$

Semantics does not use all the features of $\begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \text{cod}} & \\ & \mathcal{C} & \end{array}$

Motivation

Interpretation of MLTT in comprehension categories:

$$\text{MLTT} \xrightarrow{\text{semantics}} \begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \mathsf{cod}} & \\ & \mathcal{C} & \end{array}$$

Semantics does not use all the features of $\begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \mathsf{cod}} & \\ & \mathcal{C} & \end{array}$

Two options:

1. Restrict the comprehension categories to 'simple' ones (fully faithful or discrete)

Motivation

Interpretation of MLTT in comprehension categories:

$$\text{MLTT} \xrightarrow{\text{semantics}} \begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \mathsf{cod}} & \\ & \mathcal{C} & \end{array} \xrightarrow{\text{language}} \text{CCTT}$$

Semantics does not use all the features of $\begin{array}{ccc} \mathcal{T} & \xrightarrow{\chi} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \quad \swarrow{\scriptstyle \mathsf{cod}} & \\ & \mathcal{C} & \end{array}$

Two options:

1. Restrict the comprehension categories to ‘simple’ ones (fully faithful or discrete)
2. **Make the type theory more expressive: CCTT**

Why not Restrict the Models

- Are there interesting examples we would miss?
- Are there interesting features that we would lose?

More on this after some preliminaries

Outline

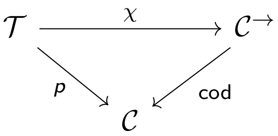
Outline

Comprehension Categories

Comprehension Category [Jac93, Definition 4.1]

1. a category $\mathcal{C}$,
2. a (cloven) fibration $p : \mathcal{T} \to \mathcal{C}$,
3. a functor $\chi : \mathcal{T} \to \mathcal{C}^{\to}$ preserving cartesian arrows,

such that the following diagram commutes.



Comprehension Categories

Comprehension Category [Jac93, Definition 4.1]

1. a category $\mathcal{C}$,
2. a (cloven) fibration $p : \mathcal{T} \to \mathcal{C}$,
3. a functor $\chi : \mathcal{T} \to \mathcal{C}^{\to}$ preserving cartesian arrows,

such that the following diagram commutes.

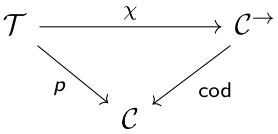
$$\begin{array}{ccc} \mathcal{T} & \xrightarrow{\quad \chi \quad} & \mathcal{C}^{\to} \\ & {\scriptstyle p}\searrow \qquad \swarrow{\scriptstyle \text{cod}} & \\ & \mathcal{C} & \end{array}$$

A comprehension category is

full if χ is full and faithful;

split if p is a split fibration.

Interpreting MLTT in a Comprehension Category



Category $\mathcal{C}$ models contexts

Fibre $\mathcal{T}_\Gamma$ models types in context Γ

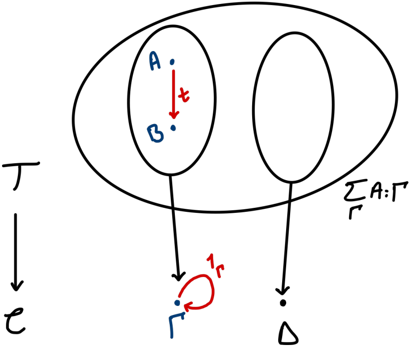
Reindexing models substitution

Comprehension χ models context extension $(\Gamma, A) \mapsto (\Gamma.A \xrightarrow{\chi(A)} \Gamma)$

Sections of $\chi(A)$ model terms $\Gamma \vdash t : A$

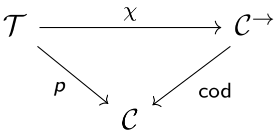
Vertical Morphisms

What about **morphisms in a fibre** $\mathcal{T}_\Gamma$?



Outline

Back to Our Motivation



- A comprehension category can express both
  1. morphisms between contexts and
  2. morphisms between types.
- Interpreting MLTT does not make use of morphisms of types, hence these are often taken to be trivial ($\mathcal{T}$ discrete) or coming from $\mathcal{C}$ (χ fully faithful)
- Restriction 'kills off' this 'extra dimension' of morphisms.

Later we will see that this extra dimension captures **coercive subtyping**.

Examples of Non-Full Comprehension Categories I

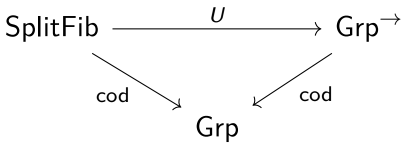
- Intensional type theories are often given semantics in an algebraic weak factorisation system (AWFS) [GL23]
- AWFSs give rise to **non-full** comprehension categories

$$\begin{array}{ccc} \mathrm{EM}(R) & \xrightarrow{U} & \mathcal{C}^{\to} \\ & \searrow \quad \swarrow_{\mathrm{cod}} & \\ & \mathcal{C} & \end{array}$$

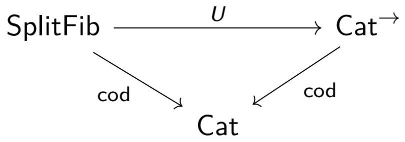
We capture this extra semantic structure in CCTT.

Examples of Non-Full Comprehension Categories II

Groupoids

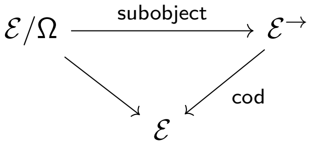


Categories

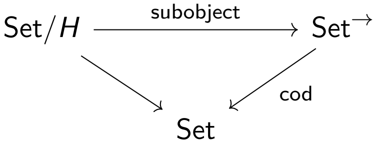


Examples of Non-Full Comprehension Categories III

Topos $\mathcal{E}$



Heyting algebra H



In This Work...

1. We design rules of a type theory that reflect the structure of comprehension categories: CCTT
2. We show how some rules of CCTT can be seen as rules for coercive subtyping, extending work by Coraglia and Emmenegger [CE24]
3. Extend CCTT with Π-, Σ- and Id-types and their compatibility with subtyping

Based on *From Semantics to Syntax: A Type Theory for Comprehension Categories*

Outline

CCTT: Judgements

1. Γ ctx
2. $\Gamma \vdash s : \Delta$
3. $\Gamma \vdash s \equiv s' : \Delta$
4. $\Gamma \vdash A$ type
5. $\Gamma | A \vdash t : B$
6. $\Gamma | A \vdash t \equiv t' : B$

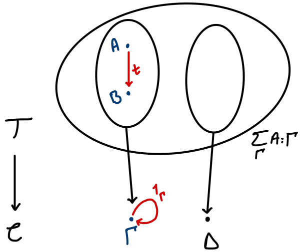
CCTT: Judgements

1. Γ ctx
2. $\Gamma \vdash s : \Delta$
3. $\Gamma \vdash s \equiv s' : \Delta$
4. $\Gamma \vdash A$ type
5. $\Gamma | A \vdash t : B$
6. $\Gamma | A \vdash t \equiv t' : B$

(5–6) $\Big\}$ $\Gamma \vdash t : A$ & $\Gamma \vdash t \equiv t' : A$ in MLTT

CCTT: Judgements

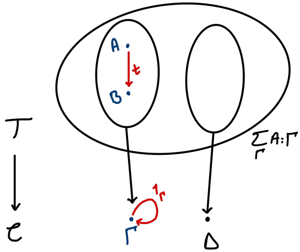
1. Γ ctx
2. $\Gamma \vdash s : \Delta$
3. $\Gamma \vdash s \equiv s' : \Delta$
4. $\Gamma \vdash A$ type
5. $\Gamma | A \vdash t : B$
6. $\Gamma | A \vdash t \equiv t' : B$



- Judgement 5: a morphism $[\![t]\!] : [\![A]\!] \to [\![B]\!]$ in the fibre $\mathcal{T}_{[\![\Gamma]\!]}$

CCTT: Judgements

1. Γ ctx
2. $\Gamma \vdash s : \Delta$
3. $\Gamma \vdash s \equiv s' : \Delta$
4. $\Gamma \vdash A$ type
5. $\Gamma | A \vdash t : B$
6. $\Gamma | A \vdash t \equiv t' : B$



- Judgement 5: a morphism $[\![t]\!] : [\![A]\!] \to [\![B]\!]$ in the fibre $\mathcal{T}_{[\![\Gamma]\!]}$
- Term judgement $\Gamma \vdash a : A$ as a macro for $\Gamma \vdash a : \Gamma.A$ and $\Gamma \vdash \pi_a \circ a \equiv 1$

CCTT: Structural Rules

Structural rules regarding the category of contexts:

$$\frac{\Gamma \ \mathsf{ctx}}{\Gamma \vdash 1_\Gamma : \Gamma} \ \text{ctx-mor-id}$$

$$\frac{\Gamma \vdash s : \Delta \qquad \Delta \vdash s' : \Theta}{\Gamma \vdash s' \circ s : \Theta} \ \text{ctx-mor-comp}$$

$$\frac{\Gamma \vdash s : \Delta}{\begin{array}{c}\Gamma \vdash s \circ 1_\Gamma \equiv s : \Delta \\ \Gamma \vdash 1_\Delta \circ s \equiv s : \Delta\end{array}} \ \text{ctx-id-unit}$$

$$\frac{\Gamma \vdash s : \Delta \qquad \Delta \vdash s' : \Theta \qquad \Theta \vdash s'' : \Phi}{\Gamma \vdash s'' \circ (s' \circ s) \equiv (s'' \circ s') \circ s : \Phi} \ \text{ctx-comp-assoc}$$

We have similar rules for the category of types.

CCTT: Structural Rules

See the paper for the rest of the structural rules: substitution, context extension, etc

Theorem (Soundness)

Every comprehension category models the rules of CCTT.

Next, we discuss some of the rules through the lens of subtyping.

Outline

Subtyping in CCTT

Coraglia and Emmenegger [CE24] observe that the vertical morphisms can be thought of as **witnesses for coercive subtyping**.

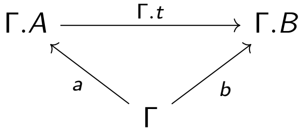
$$\Gamma | A \vdash t : B \qquad \rightsquigarrow \qquad \Gamma \vdash A \leq_t B$$

Subtyping: Subsumption

Theorem (Subsumption)

From the rules of CCTT, we can derive the following rule.

$$\frac{\Gamma \vdash A, B \text{ type} \quad \Gamma \vdash A \leq_t B \quad \Gamma \vdash a : A}{\Gamma \vdash \Gamma.t \circ a : B}$$



Γ.t is like a **coercion function for $A \leq_t B$.**

Subtyping: Weakening and Substitution

We postulate **substitution for subtyping**:

$$\frac{\Delta \vdash A, B \text{ type} \quad \Delta \vdash A \leq_t B \quad \Gamma \vdash s : \Delta}{\Gamma \vdash A[s] \leq_{t[s]} B[s]}$$

Theorem (Weakening for Subtyping)

From the rules of CCTT, we can derive the following rule.

$$\frac{\Gamma \vdash A, A', B \text{ type} \quad \Gamma \vdash A \leq_t A'}{\Gamma.B \vdash A[\pi_B] \leq_{t[\pi_B]} A'[\pi_{B'}]}$$

Outline

Subtyping for Type Formers

Type Formers Are Functors

- Type formers make new types from old
- Also have term formers; here: context morphism formers

$\rightsquigarrow$ Type formers should also act on **morphisms of types**

To Do

1. Extend CCTT with a type former (e.g. Σ-types) and show soundness.
2. Extend CCTT with subtyping for the type former and show soundness

We look at Σ-types to keep things simple.

Rules for Σ-types

Extend CCTT with Σ-types, e.g.:

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma.A \vdash B \text{ type}}{\Gamma \vdash \Sigma_A B \text{ type}} \text{ sigma-form}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma.A \vdash B \text{ type}}{\Gamma.A.B \vdash \mathsf{pair}_{\Sigma_A B} : \Gamma.\Sigma_A B} \text{ sigma-intro}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma.A \vdash B \text{ type}}{\Gamma.\Sigma_A B \vdash \mathsf{proj}_{\Sigma_A B} : \Gamma.A.B} \text{ sigma-elim}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma.A \vdash B \text{ type}}{\begin{array}{c}\Gamma.A.B \vdash \mathsf{proj}_{\Sigma_A B} \circ \mathsf{pair}_{\Sigma_A B} \equiv 1_{\Gamma.A.B} : \Gamma.A.B \\ \Gamma.\Sigma_A B \vdash \mathsf{pair}_{\Sigma_A B} \circ \mathsf{proj}_{\Sigma_A B} \equiv 1_{\Gamma.\Sigma_A B} : \Gamma.\Sigma_A B\end{array}} \text{ sigma-beta-eta}$$

$$\frac{\Delta \vdash A \text{ type} \qquad \Delta.A \vdash B \text{ type} \qquad \Gamma \vdash s : \Delta}{\Gamma \mid \Sigma_{A[s]} B[s.A] \overset{\sim}{\vdash} i_{\Sigma_A B, s} : (\Sigma_A B)[s]} \text{ subst-sigma}$$

Rules for Subtyping for Σ-types

1. We want to have the following rule:

$$\frac{\Gamma \vdash A, A' \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A' \vdash B' \text{ type} \qquad \Gamma \vdash A \leq_f A' \quad \Gamma.A \vdash B \leq_g B'[\Gamma.f]}{\Gamma \vdash \Sigma_A B \leq_{\Sigma(f,g)} \Sigma_{A'} B'}$$

   Σ acts covariantly on both arguments.

Rules for Subtyping for Σ-types

1. We want to have the following rule:

$$\frac{\Gamma \vdash A, A' \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A' \vdash B' \text{ type} \quad \Gamma \vdash A \leq_f A' \quad \Gamma.A \vdash B \leq_g B'[\Gamma.f]}{\Gamma \vdash \Sigma_A B \leq_{\Sigma(f,g)} \Sigma_{A'} B'}$$

   Σ acts covariantly on both arguments.

2. The coercion function for $\Sigma_A B \leq_{\Sigma(f,g)} \Sigma_{A'} B'$ should act as follows:

$$\Gamma.\Sigma_A B \xrightarrow{\text{proj}_{\Sigma_A B}} \Gamma.A.B \xrightarrow{\chi_0 g} \Gamma.A.B'[\chi_0 f] \xrightarrow{\chi_0 f.B'} \Gamma.A'.B' \xrightarrow{\text{pair}_{\Sigma_{A'} B'}} \Gamma.\Sigma_{A'} B'$$

Rules for Subtyping for Σ-types

1. We want to have the following rule:

$$\frac{\Gamma \vdash A, A' \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A' \vdash B' \text{ type} \quad \Gamma \vdash A \leq_f A' \quad \Gamma.A \vdash B \leq_g B'[\Gamma.f]}{\Gamma \vdash \Sigma_A B \leq_{\Sigma(f,g)} \Sigma_{A'} B'}$$

   Σ acts covariantly on both arguments.

2. The coercion function for $\Sigma_A B \leq_{\Sigma(f,g)} \Sigma_{A'} B'$ should act as follows:

$$\Gamma.\Sigma_A B \xrightarrow{\text{proj}_{\Sigma_A B}} \Gamma.A.B \xrightarrow{\chi_0 g} \Gamma.A.B'[\chi_0 f] \xrightarrow{\chi_0 f.B'} \Gamma.A'.B' \xrightarrow{\text{pair}_{\Sigma_{A'} B'}} \Gamma.\Sigma_{A'} B'$$

3. Rules for functoriality for $\Sigma(-,-)$

Semantic Structure for Subtyping for Σ–Types

Definition
$(\mathcal{C}, \mathcal{T}, p, \chi)$ **has subtyping for Σ-types** if it has

1. dependent sums and
2. for each $f : A \to A'$ in $\mathcal{T}_\Gamma$ and $g : B \to B'[\chi_0 f]$ in $\mathcal{T}_{\Gamma.A}$, a morphism in $\mathcal{T}_\Gamma$
$$\Sigma_f g : \Sigma_A B \to \Sigma_{A'} B'$$
3. $\chi_0(\Sigma_f g)$ is the following composite
$$\Gamma.\Sigma_A B \xrightarrow{\text{proj}_{\Sigma_A B}} \Gamma.A.B \xrightarrow{\chi_0 g} \Gamma.A'.B'[\chi_0 f] \xrightarrow{\chi_0 f.B'} \Gamma.A'.B' \xrightarrow{\text{pair}_{\Sigma_{A'} B'}} \Gamma.\Sigma_{A'} B'$$
4. $\Sigma_{(-)}(-)$ preserves identities and composition

Interpretation and Sanity Check

Theorem

Any comprehension category with subtyping for Σ-types models CCTT extended with subtyping for Σ-types.

Sanity Check

When χ is fully faithful,

- our Σ-structures are equivalent to Lumsdaine and Warren's [LW15]
- Jacobs' structure for Σ–types [Jac93] gives rise to ours

Outline

Some Related Work

- Melliès and Zeilberger [MZ15] give a fibrational view of **subsumptive** subtyping
- Coraglia and Emmenegger [CE24]
  - study "generalized categories with families", equivalent to non-full comprehension categories
  - study type morphisms as witnesses for coercive subtyping
  - specify some of the rules for Σ and Π

Some More Related Work

- Laurent, Lennon–Bertrand and Maillard [LLM24]
  - extend MLTT to a type theory with definitionally functorial type formers
  - extend MLTT to two type theories with coercive (MLTTcoe) and subsumptive subtyping
  - MLTTcoe has at most one coercion between any two types, subsitution is strictly functorial (see our CCTTsplit)
- Adjedj, Lennon–Bertrand, Benjamin, and Maillard [Adj+26]
  - develop a type theory AdapTT modelled by split generalized categories with families and
  - provide a general framework AdapTT2 for defining type formers that are automatically functorial

Conclusion

Summary

- CCTT reflects the structure of a comprehension category.
- Gain back the 'extra dimension' of type morphisms which captures coercive subtyping.

Future: Type Morphisms $\rightsquigarrow$ Definitional Equalities

- In models of MLTT from AWFSs: type morphisms are morphisms preserving transport of structure along an identity **strictly**, up to **definitional** equality.

$\rightsquigarrow$ Could add rules to CCTT that express this strict preservation

- Example of a commonly used function in MLTT that is a type morphisms in these models: the first projection of a Σ-type.

Thank you for your attention!

References I

[Adj+26] Arthur Adjedj et al. "AdapTT: Functoriality for Dependent Type Casts". In: *Proc. ACM Program. Lang.* 10.POPL (2026), pp. 628–658. DOI: 10.1145/3776664. URL: https://doi.org/10.1145/3776664.

[ALN24] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Paige Randall North. "Comparing Semantic Frameworks for Dependently-Sorted Algebraic Theories". In: *Programming Languages and Systems: 22nd Asian Symposium, APLAS 2024, Kyoto, Japan, October 22-24, 2024, Proceedings*. Kyoto, Japan: Springer-Verlag, 2024, pp. 3–22. ISBN: 978-981-97-8942-9. DOI: 10.1007/978-981-97-8943-6_1. URL: https://doi.org/10.1007/978-981-97-8943-6_1.

[CE24] Greta Coraglia and Jacopo Emmenegger. "Categorical Models of Subtyping". In: *29th International Conference on Types for Proofs and Programs (TYPES 2023)*. Ed. by Delia Kesner, Eduardo Hermo Reyes, and Benno van den Berg. Vol. 303. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 3:1–3:19. ISBN: 978-3-95977-332-4. DOI: 10.4230/LIPIcs.TYPES.2023.3. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TYPES.2023.3.

[GL23] Nicola Gambino and Marco Federico Larrea. "Models of Martin-Löf Type Theory from Algebraic Weak Factorisation Systems". In: *The Journal of Symbolic Logic* 88.1 (2023), pp. 242–289. ISSN: 0022-4812,1943-5886. DOI: 10.1017/jsl.2021.39.

[Jac93] Bart Jacobs. "Comprehension Categories and the Semantics of Type Dependency". In: *Theor. Comput. Sci.* 107.2 (1993), pp. 169–207. DOI: 10.1016/0304-3975(93)90169-T. URL: https://doi.org/10.1016/0304-3975(93)90169-T.

[LLM24] Théo Laurent, Meven Lennon-Bertrand, and Kenji Maillard. "Definitional Functoriality for Dependent (Sub)Types". In: ed. by Stephanie Weirich. Vol. 14576. Lecture Notes in Computer Science. Springer, 2024, pp. 302–331. DOI: 10.1007/978-3-031-57262-3_13. URL: https://doi.org/10.1007/978-3-031-57262-3%5C_13.

[LW15] Peter Lefanu Lumsdaine and Michael A. Warren. "The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories". In: *ACM Transactions on Computational Logic* 16.3 (July 2015), pp. 1–31. ISSN: 1557-945X. DOI: 10.1145/2754931. URL: http://dx.doi.org/10.1145/2754931.

References II

[MZ15] Paul-André Melliès and Noam Zeilberger. "Functors are Type Refinement Systems". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*. Ed. by Sriram K. Rajamani and David Walker. ACM, 2015, pp. 3–16. DOI: 10.1145/2676726.2676970. URL: https://doi.org/10.1145/2676726.2676970.