A Type Theory for Comprehension Categories with Applications to Subtyping

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

Programming languages group seminar, TU Delft
19 March, 2025

Overview

$$\begin{tikzcd} \mathrm{MLTT} \arrow[r, shift left, "\text{semantics}"] & \mathcal{T} \arrow[l, dashed, shift left, "\text{language}"] \arrow[r, "\chi"] \arrow[dr] & \mathcal{C}^{\to} \arrow[d] \\ & & \mathcal{C} \end{tikzcd}$$

1. Restrict the comprehension categories
2. Make the type theory more expressive: CCTT

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Overview

1. Design rules of a type theory which reflect the structure of comprehension categories
2. Prove soundness by giving an interpretation of the type theory in any comprehension category
3. Extend Coraglia and Emmenegger's work [CE24] by giving rules that captures coercive subtyping
4. Develop rules for Π-, Σ- and Id-types and give soundness results wrt each suitable semantic structure
5. Extend the rules with subtyping for type formers
6. Define suitable semantic structure for subtyping for each type former and show soundness wrt to these

Preprint is available on arXiv: https://arxiv.org/abs/2503.10868

Outline

1. Review: MLTT and Comprehension Categories
2. Our Work: Core Syntax CCTT
3. CCTT Captures Subtyping
4. Extending CCTT with Subtyping for Type Formers

Outline

1. Review: MLTT and Comprehension Categories
2. Our Work: Core Syntax CCTT
3. CCTT Captures Subtyping
4. Extending CCTT with Subtyping for Type Formers

MLTT

From "Syntax and Semantics of Dependent Types"[1]:

$\vdash \Gamma$ *ctxt*	Γ is a valid context
$\Gamma \vdash \sigma$ type	σ is a type in context Γ
$\Gamma \vdash M : \sigma$	M is a term of type σ in context Γ
$\vdash \Gamma = \Delta$ *ctxt*	Γ and Δ are definitionally equal contexts
$\Gamma \vdash \sigma = \tau$ type	σ and τ are definitionally equal types in context Γ
$\Gamma \vdash M = N : \sigma$	M and N are def. equal terms of type σ in context Γ.

[1] Martin Hofmann. *Syntax and Semantics of Dependent Types*. Publications of the Newton Institute. Cambridge University Press, 1997.

MLTT

From "Syntax and Semantics of Dependent Types"[1]:

$\vdash \Gamma\ ctxt$	Γ is a valid context
$\Gamma \vdash \sigma$ type	σ is a type in context Γ
$\Gamma \vdash M : \sigma$	M is a term of type σ in context Γ
$\vdash \Gamma = \Delta\ ctxt$	Γ and Δ are definitionally equal contexts
$\Gamma \vdash \sigma = \tau$ type	σ and τ are definitionally equal types in context Γ
$\Gamma \vdash M = N : \sigma$	M and N are def. equal terms of type σ in context Γ.

- Rules for context formation:

$$\frac{}{\vdash \diamond\ ctxt}\ \text{C-Emp} \qquad \frac{\Gamma \vdash \sigma \text{ type}}{\vdash \Gamma, x{:}\sigma\ ctxt}\ \text{C-Ext}$$

$$\frac{\vdash \Gamma = \Delta\ ctxt \qquad \Gamma \vdash \sigma = \tau \text{ type}}{\vdash \Gamma, x{:}\sigma = \Delta, y{:}\tau\ ctxt}\ \text{C-Ext-Eq}$$

The variables x and y in rules C-Ext and C-Ext-Eq are assumed to be fresh.

- The variable rule

$$\frac{\vdash \Gamma, x{:}\sigma, \Delta\ ctxt}{\Gamma, x{:}\sigma, \Delta \vdash x : \sigma}\ \text{Var}$$

- Rules expressing that definitional equality is an equivalence relation:

$$\frac{\vdash \Gamma\ ctxt}{\vdash \Gamma = \Gamma\ ctxt}\ \text{C-Eq-R}$$

$$\frac{\vdash \Gamma = \Delta\ ctxt}{\vdash \Delta = \Gamma\ ctxt}\ \text{C-Eq-S}$$

$$\frac{\vdash \Gamma = \Delta\ ctxt \qquad \vdash \Delta = \Theta\ ctxt}{\vdash \Gamma = \Theta\ ctxt}\ \text{C-Eq-T}$$

$$\frac{\Gamma \vdash \sigma \text{ type}}{\Gamma \vdash \sigma = \sigma \text{ type}}\ \text{Ty-Eq-R}$$

$$\frac{\Gamma \vdash \sigma = \tau \text{ type}}{\Gamma \vdash \tau = \sigma \text{ type}}\ \text{Ty-Eq-S}$$

$$\frac{\Gamma \vdash \sigma = \tau \text{ type} \qquad \Gamma \vdash \tau = \rho \text{ type}}{\Gamma \vdash \sigma = \rho \text{ type}}\ \text{Ty-Eq-T}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma}\ \text{Tm-Eq-R}$$

$$\frac{\Gamma \vdash M = N : \sigma}{\Gamma \vdash N = M : \sigma}\ \text{Tm-Eq-S}$$

$$\frac{\Gamma \vdash M = N : \sigma \qquad \Gamma \vdash N = O : \sigma}{\Gamma \vdash M = O : \sigma}\ \text{Tm-Eq-T}$$

- Rules relating typing and definitional equality:

$$\frac{\Gamma \vdash M : \sigma \qquad \vdash \Gamma = \Delta\ ctxt \qquad \Gamma \vdash \sigma = \tau \text{ type}}{\Delta \vdash M : \tau}\ \text{Tm-Conv}$$

$$\frac{\vdash \Gamma = \Delta\ ctxt \qquad \Gamma \vdash \sigma \text{ type}}{\Delta \vdash \sigma \text{ type}}\ \text{Ty-Conv}$$

[1] Martin Hofmann. *Syntax and Semantics of Dependent Types*. Publications of the Newton Institute. Cambridge University Press, 1997.

Comprehension Categories

Comprehension Category [Jac93, Definition 4.1]

A *comprehension category* consists of a category $\mathcal{C}$, a (cloven) fibration $p : \mathcal{T} \to \mathcal{C}$, and a functor $\chi : \mathcal{T} \to \mathcal{C}^{\to}$ preserving cartesian arrows, such that the following diagram commutes.

$$\begin{tikzcd} \mathcal{T} \arrow[rr, "\chi"] \arrow[dr, "p"'] & & \mathcal{C}^{\to} \arrow[dl, "\mathsf{cod}"] \\ & \mathcal{C} & \end{tikzcd}$$

A comprehension category is *full* if χ is full and faithful.
A comprehension category is *split* if p is a split fibration.

Full split comprehension categories are models for MLTT.

Comprehension Categories

$$\begin{tikzcd} \mathcal{T} \arrow[rr, "\chi"] \arrow[dr, "p"'] & & \mathcal{C}^{\to} \arrow[dl, "\mathsf{cod}"] \\ & \mathcal{C} & \end{tikzcd}$$

1. $\mathcal{C}$: category of contexts and context morphisms
2. A fibre $\mathcal{T}_\Gamma$: category of types in context Γ
3. Substitution is captured by the reindexing functors
4. Context extension is given by $\mathsf{dom} \circ \chi : A \mapsto \Gamma.A$
5. Terms of type A in context Γ are interpreted as sections of projections $\chi(A) : \Gamma.A \to \Gamma$ in $\mathcal{C}$

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

Outline

CCTT: Goal

Goal: Design rules that reflect all structure of (not-necessarily full) comprehension categories.

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. $\Gamma | A \vdash t \equiv t' : B$

$\Big\}$ $\Gamma \vdash t : A$ and $\Gamma \vdash t \equiv t' : A$ 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: Structural Rules

See the paper for the structural rules.

In the next section, we discuss some rules through the lens of subtyping.

CCTT: Structural Rules

See the paper for the structural rules.

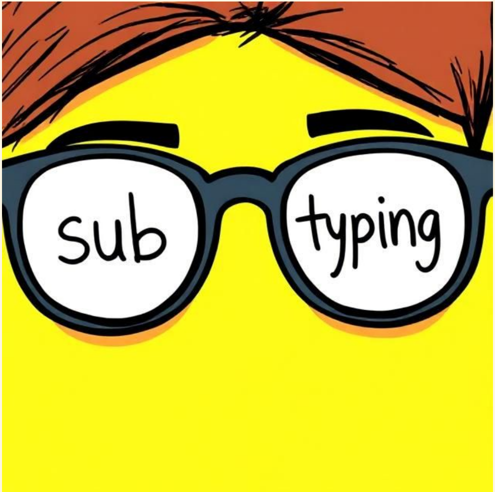
In the next section, we discuss some rules through the lens of sub-typing.

Theorem

Every comprehension category models the rules of CCTT.

Outline

We Put Our Subtyping Glasses on



Subtyping in CCTT

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

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

Proposition (Subsumption)

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

$$\frac{\Gamma \vdash A, B \text{ type} \quad \Gamma \vdash A \leq_t B \quad \Gamma \vdash a : \Gamma.A \quad \Gamma \vdash \pi_A \circ a \equiv 1_\Gamma : \Gamma}{\begin{array}{c}\Gamma \vdash b : \Gamma.B \\ \Gamma \vdash \pi_B \circ b \equiv 1_\Gamma : \Gamma\end{array}}$$

Subtyping: Subsumption

Proposition (Subsumption)

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

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

Subtyping: Subsumption

Proposition (Subsumption)

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

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

$$\begin{tikzcd}
\Gamma.A \arrow[rr, "\Gamma.t"] & & \Gamma.B \\
& \Gamma \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

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

Subtyping: Weakening and Substitution

Proposition (Weakening for Subtyping)

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

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

Proposition (Substitution for Subtyping)

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

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

Both follow from the functoriality of the reindexing functors.

Outline

Subtyping for Type formers

1. Extend CCTT with a type former (e.g. Σ-types) and show soundness: naturally, no rules involving judgements of the form $\Gamma \vdash A \leq_t B$ get added.
2. Extend CCTT with subtyping for the type former and show soundness: we see how through an example!

Subtyping for Type formers

1. Extend CCTT with a type former (e.g. Σ-types) and show soundness: naturally, no rules involving judgements of the form $\Gamma \vdash A \leq_t B$ get added.
2. Extend CCTT with subtyping for the type former and show soundness: we see how through an example!

Example: Σ-types

Extend CCTT with Σ-types, e.g.:

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

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

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

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

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

Example: Subtyping for Σ-types

1. We want to have the following rule:

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

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

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

Example: Subtyping for Σ-types

1. We want to have the following rule:

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

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

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

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

Example: Subtyping for Σ-types

1. We want to have the following rule:

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

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

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

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

Theorem

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

Summary

1. We presented CCTT
2. CCTT captures coercive subtyping
3. We extended CCTT with Π (resp. Σ, Id) and subtyping for Π (resp. Σ, Id)
4. At each step we showed soundness wrt to the suitable semantic structure

Summary

1. We presented CCTT
2. CCTT captures coercive subtyping
3. We extended CCTT with Π (resp. Σ, Id) and subtyping for Π (resp. Σ, Id)
4. At each step we showed soundness wrt to the suitable semantic structure

Thank you for your attention!

References I

[AL19] Benedikt Ahrens and Peter LeFanu Lumsdaine. "Displayed Categories". In: *Log. Methods Comput. Sci.* 15.1 (2019). DOI: 10.23638/LMCS-15(1:20)2019. URL: https://doi.org/10.23638/LMCS-15(1:20)2019.

[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.

[Hof97] Martin Hofmann. *Syntax and Semantics of Dependent Types*. Publications of the Newton Institute. Cambridge University Press, 1997.

[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.

[Str18] Thomas Streicher. *Fibered Categories à la Jean Bénabou*. 2018. arXiv: 1801.02927 [math.CT]. URL: https://arxiv.org/abs/1801.02927.