Models for Axiomatic Type Theory

Daniël Otten and Matteo Spadetto

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Contents

We explain and motivate Axiomatic Type Theory (ATT).
(type theory without reductions)

We compare two semantics for a minimal version of ATT:

- comprehension categories: more traditional and well-studied; closely follow the syntax and intricacies of type theory.
- path categories (Van den Berg, Moerdijk 2017): more concise; take inspiration from homotopy theory.

However, both specify substitutions only up to isomorphism.
Luckily, we can turn comprehension categories into actual models.

Our Contributions

Path categories are equivalent to certain comprehension categories. This allows us to turn path categories into actual models as well.

We introduce a more fine-grained notion: display path categories, and show a similar equivalence.

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=3, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=3, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"] \arrow[u, shift left=3, "U"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=3, "F"]
\end{tikzcd}$$

Our Contributions

Path categories are equivalent to certain comprehension categories.
This allows us to turn path categories into actual models as well.

We introduce a more fine-grained notion: display path categories, and show a similar equivalence.

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left, "C"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left, "U"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"] \arrow[u, shift left, "U"] \arrow[u, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left, "F"] \arrow[u, phantom, "\dashv"]
\end{tikzcd}$$

Our Contributions

Path categories are equivalent to certain comprehension categories. This allows us to turn path categories into actual models as well.

We introduce a more fine-grained notion: display path categories, and show a similar equivalence.

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"] \arrow[u, shift left, "U"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left, "F"]
\end{tikzcd}$$

Our Contributions

Path categories are equivalent to certain comprehension categories. This allows us to turn path categories into actual models as well.

We introduce a more fine-grained notion: display path categories, and show a similar equivalence.

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[u, shift left=2, "U"] \arrow[r, "\sim"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Our Contributions

Path categories are equivalent to certain comprehension categories. This allows us to turn path categories into actual models as well.

We introduce a more fine-grained notion: display path categories, and show a similar equivalence.

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=3, "C"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=3, "U"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"] \arrow[u, shift left=3, "U"] \arrow[u, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=3, "F"] \arrow[u, phantom, "\dashv"]
\end{tikzcd}$$

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger $\rightsquigarrow$ work in the system,
- smaller $\rightsquigarrow$ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger $\rightsquigarrow$ work in the system,
- smaller $\rightsquigarrow$ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger $\rightsquigarrow$ work in the system,
- smaller $\rightsquigarrow$ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger $\rightsquigarrow$ work in the system,
- smaller $\rightsquigarrow$ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Equality

Intensional Type Theory (ITT) has two notions of equality:

definitional ($\equiv$)	judgement	reductions	decidable,
propositional ($=$)	type	proofs	undecidable.

Definitional eq is a fragment of propositional eq.

Other fragments:

- larger ⇝ work in the system,
- smaller ⇝ find models.

Two extremes:

- Extensional Type Theory (ETT): everything is definitional,
- Axiomatic Type Theory (ATT): nothing is definitional.

Other Fragments

Larger:

- If we define
$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$
then we can prove
$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$
But these proven eq are not definitional.
`Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define
$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$
then we can prove
$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$
But these proven eq are not definitional.
`Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define

$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$

  then we can prove

$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$

  But these proven eq are not definitional.
  `Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define

$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$

  then we can prove

$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$

  But these proven eq are not definitional.

  `Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define
$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\,m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$
then we can prove
$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\,n) &= \mathsf{S}\,(m + n). \end{aligned}$$
But these proven eq are not definitional.
`Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define
  $$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$
  then we can prove
  $$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$
  But these proven eq are not definitional.
  `Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define
$$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\, m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$
  then we can prove
$$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\, n) &= \mathsf{S}\,(m + n). \end{aligned}$$
  But these proven eq are not definitional.

  `Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Other Fragments

Larger:

- If we define

  $$\begin{aligned} 0 + n &\equiv n, \\ (\mathsf{S}\,m) + n &\equiv \mathsf{S}\,(m + n), \end{aligned}$$

  then we can prove

  $$\begin{aligned} m + 0 &= m, \\ m + (\mathsf{S}\,n) &= \mathsf{S}\,(m + n). \end{aligned}$$

  But these proven eq are not definitional.

  `Agda` allows you to make them definitional.

Smaller:

- Cubical Type Theory: only propositional β-rule for $=$-types.
- Coinductive Types: only propositional β-rule as otherwise definitional eq becomes undecidable.

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.
(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.
(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.

(Winterhalter 2019)

Complexity and Conservativity

The complexity of type checking:

- ETT: undecidable,
- ITT: nonelementary,
- ATT: quadratic.

Does ETT prove more than ATT? Yes, namely:

- binder extensionality (bindext),
- uniqueness of identity proofs (uip).

However, these are the only additional things we can prove.
(Winterhalter 2019)

Minimal ATT

Lets start by considering the normal rules for $=$-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type}\\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type}\\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x', \mathsf{refl}_x/p]} d}(=\beta_{\mathrm{red}}).$$

Minimal ATT

Lets start by considering the normal rules for =-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x', \mathsf{refl}_x/p]} d}(=\beta_{\mathrm{red}}).$$

Minimal ATT

Lets start by considering the normal rules for $=$-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x', \mathsf{refl}_x/p]} d}(=\beta_{\text{red}}).$$

Minimal ATT

Lets start by considering the normal rules for =-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type}\\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type}\\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x',\mathsf{refl}_x/p]} d}(=\beta_{\mathrm{red}}).$$

Minimal ATT

Lets start by considering the normal rules for =-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \\ \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x', \mathsf{refl}_x/p]} d}(=\beta_{\text{red}}).$$

Minimal ATT

Lets start by considering the normal rules for =-types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \qquad \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]}{\Gamma, x, x' : A, p : x =_A x' \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\Gamma, x, x' : A, p : x =_A x' \vdash C \text{ type} \qquad \Gamma, x : A \vdash d : C[x/x', \mathsf{refl}_x/p]}{\Gamma, x : A \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x', \mathsf{refl}_x/p]} d}(=\beta_{\text{red}}).$$

Minimal ATT

Without Π-types, we have to strengthen the rules:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x', \Delta \vdash C \text{ type}\\ \Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x', \Delta \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x', \Delta \vdash C \text{ type}\\ \Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} \equiv_{C[x/x',\mathsf{refl}_x/p]} d}(=\beta_{\mathrm{red}}).$$

Minimal ATT

In ATT, we change the reduction to an axiom:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x, x' : A \vdash x =_A x' \text{ type}}(=\mathcal{F}),$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}(=\mathcal{I}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x', \Delta \vdash C \text{ type}\\ \Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x, x' : A, p : x =_A x', \Delta \vdash \mathsf{ind}^{=}_{C,d,p} : C}(=\mathcal{E}),$$

$$\frac{\begin{array}{l}\Gamma, x, x' : A, p : x =_A x', \Delta \vdash C \text{ type}\\ \Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash d : C[x/x', \mathsf{refl}_x/p]\end{array}}{\Gamma, x : A, \Delta[x/x', \mathsf{refl}_x/p] \vdash \beta^{=}_{C,d,x} : \mathsf{ind}^{=}_{C,d,\mathsf{refl}_x} =_{C[x/x',\mathsf{refl}_x/p]} d}(=\beta_{\mathrm{ax}}).$$

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
  - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
  - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
  - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
  - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
  - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
  - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
    - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
    - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
    - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
    - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
  - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
  - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Models

How do we model this minimal ATT?

Two options:

- Follow the syntax and rules. (comprehension category)
  - We require: $=_A$, refl_A, $\mathsf{ind}^{=}_{C,c,p}$, and $\beta^{=}_{C,c,x}$.
- Use intuition from homotopy theory. (path category)
  - We require: $=_A$, refl_A, and that refl_A is an equivalence.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathrm{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathrm{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl$_A$, ind$^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Comprehension Categories

In a comprehension category we have:

- a category of contexts with terminal object ϵ,
- a category of types,
- for every type A a context map $p_A : \Gamma.A \to \Gamma$. (display map)
- for every type A in context Γ and context map $\sigma : \Delta \to \Gamma$, a type $A[\sigma]$ in context Δ. (substitution)
- some other requirements.

The terms of A are the maps $a : \Gamma \to \Gamma.A$ such that $p_A \circ a = \mathrm{id}_\Gamma$.

Each type former gives additional requirements. For equality:

- $=$-types: for A a type $=_A$ and terms refl_A, $\mathsf{ind}^{=}_{A,C,d}$, $\beta^{=}_{A,C,d}$,
- weak stability: for σ we have that $=_A[\sigma]$ is also an $=$-type.

Strict Models

To model ATT, we need choices that are split:

$$A[\mathrm{id}_\Gamma] = A,$$
$$A[\tau \circ \sigma] = A[\sigma][\tau].$$

And strongly stable:

$$=_A[\sigma] = \ =_{A[\sigma]},$$
$$\mathsf{refl}_A[\sigma] = \mathsf{refl}_{A[\sigma]},$$
$$\mathsf{ind}^{=}_{A,C,d}[\sigma] = \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]},$$
$$\beta^{=}_{A,C,d}[\sigma] = \ \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}.$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Strict Models

To model ATT, we need choices that are split:

$$A[\mathrm{id}_\Gamma] = A,$$
$$A[\tau \circ \sigma] = A[\sigma][\tau].$$

And strongly stable:

$$=_A[\sigma] = \; =_{A[\sigma]},$$
$$\mathsf{refl}_A[\sigma] = \mathsf{refl}_{A[\sigma]},$$
$$\mathsf{ind}^{=}_{A,C,d}[\sigma] = \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]},$$
$$\beta^{=}_{A,C,d}[\sigma] = \; \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}.$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Strict Models

To model ATT, we need choices that are split:

$$A[\mathrm{id}_\Gamma] = A,$$
$$A[\tau \circ \sigma] = A[\sigma][\tau].$$

And strongly stable:

$$=_A[\sigma] = \ =_{A[\sigma]},$$
$$\mathsf{refl}_A[\sigma] = \mathsf{refl}_{A[\sigma]},$$
$$\mathsf{ind}^{=}_{A,C,d}[\sigma] = \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]},$$
$$\beta^{=}_{A,C,d}[\sigma] = \ \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}.$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Strict Models

To model ATT, we need choices that are split:

$$\begin{aligned} A[\mathrm{id}_\Gamma] &= A, \\ A[\tau \circ \sigma] &= A[\sigma][\tau]. \end{aligned}$$

And strongly stable:

$$\begin{aligned} =_A[\sigma] &= \ =_{A[\sigma]}, \\ \mathsf{refl}_A[\sigma] &= \mathsf{refl}_{A[\sigma]}, \\ \mathsf{ind}^{=}_{A,C,d}[\sigma] &= \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]}, \\ \beta^{=}_{A,C,d}[\sigma] &= \ \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}. \end{aligned}$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Strict Models

To model ATT, we need choices that are split:

$$A[\mathrm{id}_\Gamma] = A,$$
$$A[\tau \circ \sigma] = A[\sigma][\tau].$$

And strongly stable:

$$=_A[\sigma] = \ =_{A[\sigma]},$$
$$\mathsf{refl}_A[\sigma] = \mathsf{refl}_{A[\sigma]},$$
$$\mathsf{ind}^{=}_{A,C,d}[\sigma] = \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]},$$
$$\beta^{=}_{A,C,d}[\sigma] = \ \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}.$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Strict Models

To model ATT, we need choices that are split:

$$A[\mathrm{id}_\Gamma] = A,$$
$$A[\tau \circ \sigma] = A[\sigma][\tau].$$

And strongly stable:

$$=_A[\sigma] = \ =_{A[\sigma]},$$
$$\mathsf{refl}_A[\sigma] = \mathsf{refl}_{A[\sigma]},$$
$$\mathsf{ind}^{=}_{A,C,d}[\sigma] = \mathsf{ind}^{=}_{A[\sigma],C[\sigma],d[\sigma]},$$
$$\beta^{=}_{A,C,d}[\sigma] = \ \beta^{=}_{A[\sigma],C[\sigma],d[\sigma]}.$$

We can turn a comprehension category into one that satisfies this:

- (Lumsdaine, Warren 2014): Local Universe Construction.
- (Bocquet 2021): Generic Contexts.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, "g"] & C \arrow[r, "h"] & D \end{tikzcd}$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, "g"] & C \arrow[r, "h"] & D \end{tikzcd}$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, "g"] & C \arrow[r, "h"] & D \end{tikzcd}$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, "g"] & C \arrow[r, "h"] & D \end{tikzcd}$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

where $g \circ f$ and $h \circ g$ are equivalences,
then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Categories

A path category is a category $\mathcal{C}$ with two classes of maps:

- fibrations: closed under pullbacks and compositions,
- (weak) equivalences: satisfying 2-out-of-6, so, if we have

  $$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

  where $g \circ f$ and $h \circ g$ are equivalences,
  then f, g, h, and $h \circ g \circ h$ are equivalences.

If a map is both then we call it a trivial fibration. We require that:

- isomorphisms are trivial fibration,
- trivial fibrations are closed under pullbacks,
- every trivial fibration has a section.

$\mathcal{C}$ has a terminal object 1 and every map $A \to 1$ is a fibration.

Path Objects

Lastly, a path category has a path object for every object A:

- a factorisation of the diagonal $\delta_A = (\mathrm{id}_A, \mathrm{id}_A)$:

$$\begin{tikzcd} A \arrow[rr, "\delta_A"] \arrow[dr, "r_A"', "\sim"] & & A \times A \\ & PA \arrow[ur, two heads, "(s_A{,}t_A)"'] & \end{tikzcd}$$

  into a weak equivalence r_A followed by a fibration (s_A, t_A).

Path Objects

Lastly, a path category has a path object for every object A:

- a factorisation of the diagonal $\delta_A = (\mathrm{id}_A, \mathrm{id}_A)$:

$$\begin{tikzcd} A \arrow[rr, "\delta_A"] \arrow[dr, "r_A"', "\sim"] & & A \times A \\ & PA \arrow[ur, two heads, "(s_A, t_A)"'] & \end{tikzcd}$$

  into a weak equivalence r_A followed by a fibration (s_A, t_A).

Path Objects

Lastly, a path category has a path object for every object A:

- a factorisation of the diagonal $\delta_A = (\mathrm{id}_A, \mathrm{id}_A)$:

$$\begin{tikzcd} A \arrow[rr, "\delta_A"] \arrow[dr, "r_A"', "\sim"] & & A \times A \\ & PA \arrow[ur, two heads, "(s_A{,}t_A)"'] & \end{tikzcd}$$

into a weak equivalence r_A followed by a fibration (s_A, t_A).

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Homotopy Theory

We call two maps $f, g : A \to B$ homotopic, written $f \simeq g$, if there exists a map $h : A \to PB$ such that $s_B \circ h = f$ and $t_B \circ h = g$.

We call $f : A \to B$ an homotopy equivalence, if there exists a map $g : B \to A$ such that $g \circ f \simeq \mathrm{id}_A$ and $f \circ g \simeq \mathrm{id}_B$.

The homotopy equivalences are precisely the weak equivalences.

In addition, we have a lifting theorem: for a commutative square

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "w"', "\wr"] & C \arrow[d, two heads, "p"] \\
B \arrow[r] \arrow[ur, dashed] & D
\end{tikzcd}$$

where w is an equivalence and p is a fibration, there is a map $d : B \to C$ unique up to homotopy such that the lower triangle commutes and the upper triangle commutes up to homotopy.

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category ⇝ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\to}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category ⇝ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\to}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\mathsf{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\mathrm{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category ⇝ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\mathrm{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\mathrm{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\to}$,
- the display map for $p \in \mathcal{C}^{\mathrm{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\mathsf{fib}} \subseteq \mathcal{C}^{\to}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Path Category $\rightsquigarrow$ Comprehension Category

We can view a path category $\mathcal{C}$ as a comprehension category:

- the contexts are given by $\mathcal{C}$,
- the types are given by the full subcategory $\mathcal{C}^{\text{fib}} \subseteq \mathcal{C}^{\rightarrow}$,
- the display map for $p \in \mathcal{C}^{\text{fib}}$ is p itself,
- the substitution $p[\sigma]$ is the pullback $\sigma^* p$.

We will show that it has additional structure:

- weakly stable $=$-types,
- weakly stable Σ-types with β and η reductions,
- contextuality (contexts are finite).

Weakly Stable =-Types

For a type A we define:

$$=_A \coloneqq (s_A, t_A) : P_A \twoheadrightarrow A \times A, \qquad \text{(formation)}$$
$$\mathsf{refl}_A \coloneqq r_A : A \xrightarrow{\sim} PA. \qquad \text{(introduction)}$$

The elimination and β-axiom follow from our lifting theorem and the fact that r_A is an equivalence.

We get weak stability because we can show that path objects are preserved by taking pullbacks.

Weakly Stable =-Types

For a type A we define:

$$=_A \coloneqq (s_A, t_A) : P_A \twoheadrightarrow A \times A, \qquad \text{(formation)}$$
$$\mathsf{refl}_A \coloneqq r_A : A \xrightarrow{\sim} PA. \qquad \text{(introduction)}$$

The elimination and β-axiom follow from our lifting theorem and the fact that r_A is an equivalence.

We get weak stability because we can show that path objects are preserved by taking pullbacks.

Weakly Stable =-Types

For a type A we define:

$$\begin{aligned} =_A &:= (s_A, t_A) : P_A \twoheadrightarrow A \times A, && \text{(formation)} \\ \mathsf{refl}_A &:= r_A : A \xrightarrow{\sim} PA. && \text{(introduction)} \end{aligned}$$

The elimination and β-axiom follow from our lifting theorem and the fact that r_A is an equivalence.

We get weak stability because we can show that path objects are preserved by taking pullbacks.

Weakly Stable =-Types

For a type A we define:

$$\begin{aligned} =_A &:= (s_A, t_A) : P_A \twoheadrightarrow A \times A, && \text{(formation)} \\ \mathsf{refl}_A &:= r_A : A \xrightarrow{\sim} PA. && \text{(introduction)} \end{aligned}$$

The elimination and β-axiom follow from our lifting theorem and the fact that r_A is an equivalence.

We get weak stability because we can show that path objects are preserved by taking pullbacks.

Weakly Stable Σ-Types with β and η

We obtain Σ-types because path categories do not distinguish between a single extension $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

The requirements on a comprehension category can be simplified:

for $\Gamma.A.B$ we have a type $\Sigma_A B$ and an iso $\Gamma.A.B \xrightarrow{\cong} \Gamma.\Sigma_A B$ making the square commute:

$$\begin{tikzcd}
\Gamma.A.B \arrow[r, "\cong"] \arrow[d, "p_B"'] & \Gamma.\Sigma_A B \arrow[d, "p_{\Sigma_A B}"] \\
\Gamma.A \arrow[r, "p_A"'] & \Gamma
\end{tikzcd}$$

Holds in path categories: fibrations are closed under composition.

Weakly Stable Σ-Types with β and η

We obtain Σ-types because path categories do not distinguish between a single extension $\Gamma.A$ and $\Gamma.A_0.\ldots.A_{n-1}$.

The requirements on a comprehension category can be simplified:

for $\Gamma.A.B$ we have a type $\Sigma_A B$ and an iso $\Gamma.A.B \xrightarrow{\cong} \Gamma.\Sigma_A B$ making the square commute:

$$\begin{tikzcd}
\Gamma.A.B \arrow[r, "\cong"] \arrow[d, "p_B"'] & \Gamma.\Sigma_A B \arrow[d, "p_{\Sigma_A B}"] \\
\Gamma.A \arrow[r, "p_A"'] & \Gamma
\end{tikzcd}$$

Holds in path categories: fibrations are closed under composition.

Weakly Stable Σ-Types with β and η

We obtain Σ-types because path categories do not distinguish between a single extension $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

The requirements on a comprehension category can be simplified:

for $\Gamma.A.B$ we have a type $\Sigma_A B$ and an iso $\Gamma.A.B \xrightarrow{\cong} \Gamma.\Sigma_A B$ making the square commute:

$$\begin{tikzcd}
\Gamma.A.B \arrow[r, "\cong"] \arrow[d, "p_B"'] & \Gamma.\Sigma_A B \arrow[d, "p_{\Sigma_A B}"] \\
\Gamma.A \arrow[r, "p_A"'] & \Gamma
\end{tikzcd}$$

Holds in path categories: fibrations are closed under composition.

Weakly Stable Σ-Types with β and η

We obtain Σ-types because path categories do not distinguish between a single extension $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

The requirements on a comprehension category can be simplified:

for $\Gamma.A.B$ we have a type $\Sigma_A B$ and an iso $\Gamma.A.B \xrightarrow{\cong} \Gamma.\Sigma_A B$ making the square commute:

$$\begin{tikzcd}
\Gamma.A.B \arrow[r, "\cong"] \arrow[d, "p_B"'] & \Gamma.\Sigma_A B \arrow[d, "p_{\Sigma_A B}"] \\
\Gamma.A \arrow[r, "p_A"'] & \Gamma
\end{tikzcd}$$

Holds in path categories: fibrations are closed under composition.

Weakly Stable Σ-Types with β and η

We obtain Σ-types because path categories do not distinguish between a single extension $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

The requirements on a comprehension category can be simplified:

for $\Gamma.A.B$ we have a type $\Sigma_A B$ and an iso $\Gamma.A.B \xrightarrow{\cong} \Gamma.\Sigma_A B$ making the square commute:

$$\begin{tikzcd}
\Gamma.A.B \arrow[r, "\simeq"] \arrow[d, "p_B"'] & \Gamma.\Sigma_A B \arrow[d, "p_{\Sigma_A B}"] \\
\Gamma.A \arrow[r, "p_A"'] & \Gamma
\end{tikzcd}$$

Holds in path categories: fibrations are closed under composition.

Contextuality

A comprehension category is contextual if for every Γ we have:

- a type A_0 in context ϵ,
- a type A_1 in context $\epsilon.A_0$,
- a type A_2 in context $\epsilon.A_0.A_1$,

  $\vdots$
- a type A_{n-1} in context $\epsilon.A_0.\dots.A_{n-2}$,

such that $\Gamma \cong \epsilon.A_0.\dots.A_{n-1}$.

Holds in path categories: every map $\Gamma \to 1$ is a fibration.

Contextuality

A comprehension category is contextual if for every Γ we have:

- a type A_0 in context ϵ,
- a type A_1 in context $\epsilon.A_0$,
- a type A_2 in context $\epsilon.A_0.A_1$,

  $\vdots$
- a type A_{n-1} in context $\epsilon.A_0.\dots.A_{n-2}$,

such that $\Gamma \cong \epsilon.A_0.\dots.A_{n-1}$.

Holds in path categories: every map $\Gamma \to 1$ is a fibration.

Contextuality

A comprehension category is contextual if for every Γ we have:

- a type A_0 in context ϵ,
- a type A_1 in context $\epsilon.A_0$,
- a type A_2 in context $\epsilon.A_0.A_1$,

  $\vdots$
- a type A_{n-1} in context $\epsilon.A_0.\ldots.A_{n-2}$,

such that $\Gamma \cong \epsilon.A_0.\ldots.A_{n-1}$.

Holds in path categories: every map $\Gamma \to 1$ is a fibration.

Comprehension Category $\rightsquigarrow$ Path Category

We can turn a comprehension category $\mathcal{C}$ with weakly stable $=$, $\Sigma_{\beta,\eta}$, and contextuality into a path category by taking:

- the fibrations as the compositions of display maps and isos,
- the weak equivalences as the homotopy equivalences.
- the path objects as the $=$-types.

Comprehension Category $\rightsquigarrow$ Path Category

We can turn a comprehension category $\mathcal{C}$ with weakly stable $=$, $\Sigma_{\beta,\eta}$, and contextuality into a path category by taking:

- the fibrations as the compositions of display maps and isos,
- the weak equivalences as the homotopy equivalences.
- the path objects as the $=$-types.

Comprehension Category $\rightsquigarrow$ Path Category

We can turn a comprehension category $\mathcal{C}$ with weakly stable $=$, $\Sigma_{\beta,\eta}$, and contextuality into a path category by taking:

- the fibrations as the compositions of display maps and isos,
- the weak equivalences as the homotopy equivalences.
- the path objects as the $=$-types.

Comprehension Category $\rightsquigarrow$ Path Category

We can turn a comprehension category $\mathcal{C}$ with weakly stable $=$, $\Sigma_{\beta,\eta}$, and contextuality into a path category by taking:

- the fibrations as the compositions of display maps and isos,
- the weak equivalences as the homotopy equivalences.
- the path objects as the $=$-types.

Comprehension Category $\rightsquigarrow$ Path Category

We can turn a comprehension category $\mathcal{C}$ with weakly stable $=$, $\Sigma_{\beta,\eta}$, and contextuality into a path category by taking:

- the fibrations as the compositions of display maps and isos,
- the weak equivalences as the homotopy equivalences.
- the path objects as the $=$-types.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\ldots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\ldots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\dots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\ldots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Display Path Categories

In a display path category we distinguish $\Gamma.A$ and $\Gamma.A_0.\ldots.A_{n-1}$.

Instead of fibrations we use display maps as a primitive notion.

Fibrations are compositions of display maps and isomorphisms.

In addition, we replace path objects for objects Γ with a seemingly weaker notion: path objects for display maps $A \to \Gamma$.

This is sufficient: we can use a lifting theorem and transport to inductively construct path objects for objects.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\text{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] \arrow[d, phantom, "\dashv"] & \text{ComprehensionCat}_{\text{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\text{DisplayPathCat} \arrow[r, "\sim"'] \arrow[u, shift left=2, "U"] & \text{ComprehensionCat}_{\text{Contextual},=} \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"'] \arrow[u, shift left=2, "U"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"'] \arrow[u, shift left=2, "U"] \arrow[u, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=2, "F"] \arrow[u, phantom, "\dashv"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[r, "\sim"'] \arrow[u, shift left=2, "U"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=2, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[u, shift left=2, "U"] \arrow[r, "\sim"'] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

Equivalence

We obtain the following diagram of 2-categories:

$$\begin{tikzcd}
\mathrm{PathCat} \arrow[r, "\sim"] \arrow[d, shift left=3, "C"] \arrow[d, phantom, "\dashv"] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=,\Sigma_{\beta\eta}} \arrow[d, shift left=3, "U"] \arrow[d, phantom, "\dashv"] \\
\mathrm{DisplayPathCat} \arrow[u, shift left=3, "U"] \arrow[r, "\sim"'] & \mathrm{ComprehensionCat}_{\mathrm{Contextual},=} \arrow[u, shift left=3, "F"]
\end{tikzcd}$$

Here the U's are forgetful, F is a free, and C is a cofree.

We end this talk with some open questions:

- Can we simplify other type formers as we did with $=$-types?
- In particular, are propositional Σ-types and Π-types homotopical weakenings of left and right adjoints of the pullback functor.

References

- Benno van den Berg and Ieke Moerdijk (2017): Exact completion of path categories.
- Benno van den Berg (2018): Path categories and propositional identity types.
- Rafaël Bocquet (2020): Coherence of strict equalities in dependent type theories.
- Rafaël Bocquet (2021): Strictification of weakly stable type-theoretic structures using generic contexts.
- Martin Hofmann (1995): On the interpretation of type theory in locally Cartesian closed categories.
- Peter Lumsdaine and Michael Warren (2014): The local universes model, an overlooked coherence construction for dependent type theories.
- Nicolas Oury (2005): Extensionality in the calculus of constructions.
- Matteo Spadetto (2023): A conservativity result for homotopy elementary types in dependent type theory.
- Theo Winterhalter (2019): Formalisation and meta-theory of type theory.