The Leibniz adjunction in homotopy type theory, with an application to simplicial type theory

Tom de Jong, Nicolai Kraus, Axel Ljungström

University of Nottingham, UK

TYPES 2026

32nd International Conference on Types for Proofs and Programs

8 May 2026 — Gothenburg, Sweden

Introduction & main results (informally)

- Riehl and Shulman introduced simplicial type theory (STT) as a synthetic framework for higher categories.[1]
- In homotopy type theory (HoTT) types behave like ∞-groupoids.
  STT additionally equips types with a notion of directed morphisms, via an interval.

[1] Emily Riehl and Michael Shulman (2017). 'A type theory for synthetic ∞-categories'. In: *Higher Structures* 1.1, pp. 147–224. DOI: 10.21136/hs.2017.06.

Introduction & main results (informally)

- Riehl and Shulman introduced simplicial type theory (STT) as a synthetic framework for higher categories.[1]
- In homotopy type theory (HoTT) types behave like ∞-groupoids.
  STT additionally equips types with a notion of directed morphisms, via an interval.
- Segal types are those types in which such morphisms compose.
- Composition should be associative, identities should be neutral w.r.t. composition, and higher coherences should be satisfied, e.g. the different ways of rewriting $\mathrm{id} \circ (f \circ \mathrm{id})$ to f should be the same, up to a higher morphism.

[1] Emily Riehl and Michael Shulman (2017). 'A type theory for synthetic ∞-categories'. In: *Higher Structures* 1.1, pp. 147–224. DOI: 10.21136/hs.2017.06.

Introduction & main results (informally)

- Riehl and Shulman introduced simplicial type theory (STT) as a synthetic framework for higher categories.[1]
- In homotopy type theory (HoTT) types behave like ∞-groupoids.
  STT additionally equips types with a notion of directed morphisms, via an interval.
- Segal types are those types in which such morphisms compose.
- Composition should be associative, identities should be neutral w.r.t. composition, and higher coherences should be satisfied, e.g. the different ways of rewriting $\mathrm{id} \circ (f \circ \mathrm{id})$ to f should be the same, up to a higher morphism.

🏁 For Segal types, all higher coherences can be derived.

🏁 We give a slick proof using some category theory and families instead of maps.

[1] Emily Riehl and Michael Shulman (2017). 'A type theory for synthetic ∞-categories'. In: *Higher Structures* 1.1, pp. 147–224. DOI: 10.21136/hs.2017.06.

Foundational setup

- Instead of simplicial type theory à la Riehl & Shulman, we follow Gratzer, Weinberger and Buchholtz[2] and work in an axiomatic extension of HoTT:

$$\text{HoTT} + \text{bounded distributive lattice } (I, 0, 1, \wedge, \vee).$$

For this work, we do not need I to be linearly ordered.

[2] Daniel Gratzer, Jonathan Weinberger and Ulrik Buchholtz (2024). 'Directed univalence in simplicial homotopy type theory'. arXiv: 2407.09146 [cs.LO].

Foundational setup

- Instead of simplicial type theory à la Riehl & Shulman, we follow Gratzer, Weinberger and Buchholtz[2] and work in an axiomatic extension of HoTT:

  $$\text{HoTT} + \text{bounded distributive lattice } (I, 0, 1, \wedge, \vee).$$

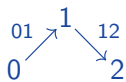
  For this work, we do not need I to be linearly ordered.

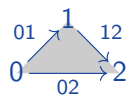
- Use I to construct the objects necessary for simplicial combinatorics (next slide).

[2] Daniel Gratzer, Jonathan Weinberger and Ulrik Buchholtz (2024). 'Directed univalence in simplicial homotopy type theory'. arXiv: 2407.09146 [cs.LO].

First main result (more precisely)

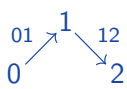
- Simplicial combinatorics: composition in X is given by extending along the horn inclusion $\Lambda^2_1 \hookrightarrow \Delta^2$.

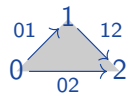
$\Lambda^2_1 =$ 

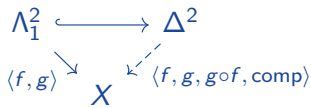
$\Delta^2 =$ 

First main result (more precisely)

- Simplicial combinatorics: composition in X is given by extending along the horn inclusion $\Lambda^2_1 \hookrightarrow \Delta^2$.

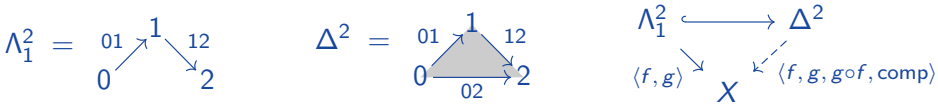
$\Lambda^2_1 =$ 

$\Delta^2 =$ 



First main result (more precisely)

- Simplicial combinatorics: composition in X is given by extending along the horn inclusion $\Lambda^2_1 \hookrightarrow \Delta^2$.

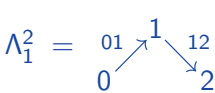


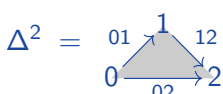
Def. A type X is a Segal type if every map $\Lambda^2_1 \to X$ has a unique (in the contractible/HoTT sense) extension to a map $\Delta^2 \to X$.

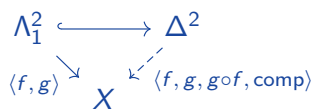
We also say that a Segal type has unique fillers for (2,1)-horns.

First main result (more precisely)

- Simplicial combinatorics: composition in X is given by extending along the horn inclusion $\Lambda^2_1 \hookrightarrow \Delta^2$.







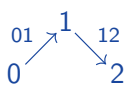
Def. A type X is a Segal type if every map $\Lambda^2_1 \to X$ has a unique (in the contractible/HoTT sense) extension to a map $\Delta^2 \to X$.

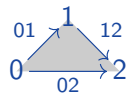
We also say that a Segal type has unique fillers for (2,1)-horns.

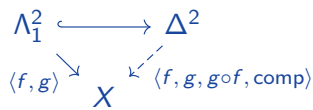
- We can similarly consider higher dimensional simplices Δ^n and horns Λ^n_k which encode higher coherences.

First main result (more precisely)

- Simplicial combinatorics: composition in X is given by extending along the horn inclusion $\Lambda^2_1 \hookrightarrow \Delta^2$.

$\Lambda^2_1 =$ 

$\Delta^2 =$ 



Def. A type X is a Segal type if every map $\Lambda^2_1 \to X$ has a unique (in the contractible/HoTT sense) extension to a map $\Delta^2 \to X$.

We also say that a Segal type has unique fillers for (2,1)-horns.

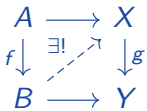
- We can similarly consider higher dimensional simplices Δ^n and horns Λ^n_k which encode higher coherences.

Thm. Any Segal type has unique fillers for all (n,k)-horns where $0 < k < n$.

- This generalizes a result of Riehl & Shulman for $n = 3$ and $k \in \{1, 2\}$, and is a STT version of a result by Lurie.

Reduction to the Leibniz adjunction

- Segal types can be characterized by orthogonality



$(f \perp g)$

Reduction to the Leibniz adjunction

- Segal types can be characterized by orthogonality

$(f \perp g)$ $\rightsquigarrow$

$(X \text{ Segal})$

Reduction to the Leibniz adjunction

- Segal types can be characterized by orthogonality

- General fact: $f \perp g$ if and only if their pullback-hom $f \pitchfork g$ is an equivalence. (Definition on the next slide.)

Reduction to the Leibniz adjunction

- Segal types can be characterized by orthogonality

- General fact: $f \perp g$ if and only if their pullback-hom $f \pitchfork g$ is an equivalence. (Definition on the next slide.)
- For 1-categories: the pullback-hom is right adjoint to the pushout-product $f \mathbin{\widehat{\times}} g$. Sometimes known as the Leibniz adjunction.

Reduction to the Leibniz adjunction

- Segal types can be characterized by orthogonality

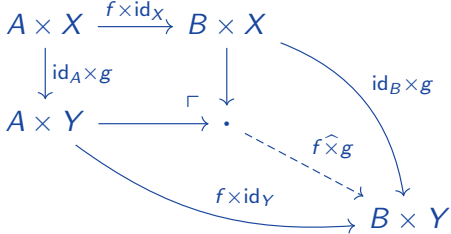
- General fact: $f \perp g$ if and only if their pullback-hom $f \pitchfork g$ is an equivalence. (Definition on the next slide.)
- For 1-categories: the pullback-hom is right adjoint to the pushout-product $f \mathbin{\widehat{\times}} g$. Sometimes known as the Leibniz adjunction.

What we did:

- Reduce simplicial comb. to formal arguments about $\pitchfork$ and $\mathbin{\widehat{\times}}$ as much as possible.
- Prove the Leibniz adjunction for maps in HoTT.

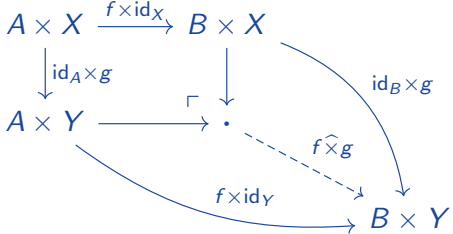
Pushout-product and pullback-hom

Given $f : A \to B$ and $g : X \to Y$, we define

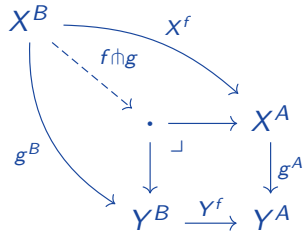


Pushout-product and pullback-hom

Given $f : A \to B$ and $g : X \to Y$, we define



and



Second main result: the Leibniz adjunction in HoTT

Def. For $f : A \to B$ and $g : X \to Y$, write

$$\mathrm{Map}(f, g) :\equiv \left\{ \begin{array}{ccc} A & \xrightarrow{u} & X \\ {\scriptstyle f}\downarrow & \swarrow^{\alpha} & \downarrow{\scriptstyle g} \\ B & \xrightarrow{v} & Y \end{array} \right\}$$

for the type of commutative squares with f on the left and g on the right.

Second main result: the Leibniz adjunction in HoTT

Def. For $f : A \to B$ and $g : X \to Y$, write

$$\mathsf{Map}(f, g) :\equiv \left\{ \begin{array}{ccc} A & \xrightarrow{u} & X \\ {\scriptstyle f}\downarrow & \swarrow\!\!\swarrow\ {\scriptstyle \alpha} & \downarrow{\scriptstyle g} \\ B & \xrightarrow{v} & Y \end{array} \right\}$$

for the type of commutative squares with f on the left and g on the right.

⚠ The commutativity is *data* in HoTT!

Second main result: the Leibniz adjunction in HoTT

Def. For $f : A \to B$ and $g : X \to Y$, write

$$\mathrm{Map}(f, g) :\equiv \left\{ \begin{array}{ccc} A & \xrightarrow{u} & X \\ {\scriptstyle f}\downarrow & \swarrow^{\alpha} & \downarrow{\scriptstyle g} \\ B & \xrightarrow{v} & Y \end{array} \right\}$$

for the type of commutative squares with f on the left and g on the right.

⚠ The commutativity is *data* in HoTT!

Thm. We have a natural equivalence

$$\mathrm{Map}(f \mathbin{\widehat{\times}} g, h) \simeq \mathrm{Map}(f, g \pitchfork h).$$

Towards a proof of the Leibniz adjunction

- Unfolding definitions, an element of $\mathrm{Map}(f \mathbin{\widehat{\times}} g, h)$ is a 7-tuple: three maps, three equations between maps and one additional coherence.

  Similarly for $\mathrm{Map}(f, g \pitchfork h)$.

Towards a proof of the Leibniz adjunction

- Unfolding definitions, an element of $\mathrm{Map}(f \mathbin{\widehat{\times}} g, h)$ is a 7-tuple: three maps, three equations between maps and one additional coherence.

  Similarly for $\mathrm{Map}(f, g \pitchfork h)$.

- With some squinting, a little hand-waving and enough whiteboard space, you can convince yourself that you can map such tuples across to get an equivalence.

Towards a proof of the Leibniz adjunction

- Unfolding definitions, an element of $\mathrm{Map}(f \mathbin{\widehat{\times}} g, h)$ is a 7-tuple: three maps, three equations between maps and one additional coherence.

  Similarly for $\mathrm{Map}(f, g \pitchfork h)$.

- With some squinting, a little hand-waving and enough whiteboard space, you can convince yourself that you can map such tuples across to get an equivalence.

- *But the details are annoying...*

Towards a proof of the Leibniz adjunction

- Unfolding definitions, an element of $\mathrm{Map}(f \mathbin{\widehat{\times}} g, h)$ is a 7-tuple: three maps, three equations between maps and one additional coherence.

  Similarly for $\mathrm{Map}(f, g \pitchfork h)$.

- With some squinting, a little hand-waving and enough whiteboard space, you can convince yourself that you can map such tuples across to get an equivalence.

- *But the details are annoying...*

  Use a proof assistant to help us!

Towards a proof of the Leibniz adjunction

- Unfolding definitions, an element of $\mathrm{Map}(f \mathbin{\widehat{\times}} g, h)$ is a 7-tuple: three maps, three equations between maps and one additional coherence.

  Similarly for $\mathrm{Map}(f, g \pitchfork h)$.

- With some squinting, a little hand-waving and enough whiteboard space, you can convince yourself that you can map such tuples across to get an equivalence.

- *But the details are annoying...*

  Use a proof assistant to help us!

  But the details are annoying...

Axel goes to Nottingham

- Axel Ljungström suggested to use families instead of maps.

Axel goes to Nottingham

- Axel Ljungström suggested to use families instead of maps.
- By univalence, the type of (small) maps

$$\sum(A : \mathcal{U}) \sum(B : \mathcal{U}) (A \to B)$$

is equivalent to the type of type families

$$\sum(X : \mathcal{U}) (X \to \mathcal{U}).$$

Axel goes to Nottingham

- Axel Ljungström suggested to use families instead of maps.
- By univalence, the type of (small) maps

$$\sum(A : \mathcal{U}) \sum(B : \mathcal{U}) (A \to B)$$

is equivalent to the type of type families

$$\sum(X : \mathcal{U}) (X \to \mathcal{U}).$$

A map $f : A \to B$ is sent to the family $b \mapsto \mathrm{fib}_f\ b :\equiv \sum(a : A) f\ a = b$.
Conversely, a family $Y \to \mathcal{U}$ is sent to the projection $\mathrm{pr}_1 : (\sum(x : X) Y\ x) \to X$.

Axel goes to Nottingham

- Axel Ljungström suggested to use families instead of maps.
- By univalence, the type of (small) maps

$$\sum(A : \mathcal{U})\sum(B : \mathcal{U})\,(A \to B)$$

is equivalent to the type of type families

$$\sum(X : \mathcal{U})\,(X \to \mathcal{U}).$$

A map $f : A \to B$ is sent to the family $b \mapsto \mathrm{fib}_f\ b :\equiv \sum(a : A)\, f\, a = b$.
Conversely, a family $Y \to \mathcal{U}$ is sent to the projection $\mathrm{pr}_1 : (\sum(x : X)\, Y\, x) \to X$.

- Given $Y : X \to \mathcal{U}$ and $Y' : X' \to \mathcal{U}$, write

$$\mathrm{Fam}((X, Y); (X', Y')) :\equiv \sum(m : X \to X')\prod(x : X)\,(Y\, x \to Y'(m\, x)).$$

Axel goes to Nottingham

- Axel Ljungström suggested to use families instead of maps.
- By univalence, the type of (small) maps

$$\sum(A : \mathcal{U})\sum(B : \mathcal{U})\,(A \to B)$$

  is equivalent to the type of type families

$$\sum(X : \mathcal{U})\,(X \to \mathcal{U}).$$

  A map $f : A \to B$ is sent to the family $b \mapsto \mathsf{fib}_f\ b :\equiv \sum(a : A)\, f\, a = b$.
  Conversely, a family $Y \to \mathcal{U}$ is sent to the projection $\mathsf{pr}_1 : (\sum(x : X)\, Y\, x) \to X$.
- Given $Y : X \to \mathcal{U}$ and $Y' : X' \to \mathcal{U}$, write

$$\mathsf{Fam}((X, Y); (X', Y')) :\equiv \sum(m : X \to X')\prod(x : X)\,(Y\, x \to Y'(m\, x)).$$

- The above equivalence χ extends to an equivalence

$$\mathsf{Map}(f, g) \simeq \mathsf{Fam}(\chi\, f, \chi\, g).$$

The merits of Fam

- Associativity of composition in Map requires some path algebra.

  In contrast, composition in Fam is *definitionally* associative.

The merits of Fam

- Associativity of composition in Map requires some path algebra.

  In contrast, composition in Fam is *definitionally* associative.

The merits of Fam

- Associativity of composition in Map requires some path algebra.

  In contrast, composition in Fam is *definitionally* associative.

1. Calculate what $\pitchfork$ and $\mathbin{\widehat{\times}}$ should be for families.

   Calculation is guided by the desired equations
   $\chi(f \mathbin{\widehat{\times}} g) = \chi f \mathbin{\widehat{\times}} \chi g$ and $\chi(f \pitchfork g) = \chi f \pitchfork \chi g$.

The merits of Fam

- Associativity of composition in Map requires some path algebra.

  In contrast, composition in Fam is *definitionally* associative.

1. Calculate what $\pitchfork$ and $\mathbin{\widehat{\times}}$ should be for families.

   Calculation is guided by the desired equations
   $\chi(f \mathbin{\widehat{\times}} g) = \chi f \mathbin{\widehat{\times}} \chi g$ and $\chi(f \pitchfork g) = \chi f \pitchfork \chi g$.

2. Construct the adjunction in Fam.

The merits of Fam

- Associativity of composition in Map requires some path algebra.

  In contrast, composition in Fam is *definitionally* associative.

1. Calculate what $\pitchfork$ and $\widehat{\times}$ should be for families.

   Calculation is guided by the desired equations
   $\chi(f \mathbin{\widehat{\times}} g) = \chi f \mathbin{\widehat{\times}} \chi g$ and $\chi(f \pitchfork g) = \chi f \pitchfork \chi g$.

2. Construct the adjunction in Fam.

3. Transfer the adjunction back to Map using
   - $\mathsf{Map}(f, g) \simeq \mathsf{Fam}(\chi f, \chi g)$,
   - that χ and its inverse preserve $\widehat{\times}$ and $\pitchfork$.

The merits of Fam

- Associativity of composition in Map requires some path algebra.
  In contrast, composition in Fam is *definitionally* associative.

1. Calculate what $\pitchfork$ and $\mathbin{\widehat{\times}}$ should be for families.
   Calculation is guided by the desired equations
   $\chi(f \mathbin{\widehat{\times}} g) = \chi f \mathbin{\widehat{\times}} \chi g$ and $\chi(f \pitchfork g) = \chi f \pitchfork \chi g$.
2. Construct the adjunction in Fam.
3. Transfer the adjunction back to Map using
   - $\mathsf{Map}(f, g) \simeq \mathsf{Fam}(\chi f, \chi g)$,
   - that χ and its inverse preserve $\mathbin{\widehat{\times}}$ and $\pitchfork$.

- Steps (1)–(3) turn out to be relatively easy and natural. ☺
  E.g. $(A, B) \mathbin{\widehat{\times}} (X, Y) \coloneqq (A \times X, (a, x) \mapsto B\, a * Y\, x)$,
  where $*$ denotes the join of types.

Conclusion

- Main results:
  1. Segal types have all higher coherences
  2. Leibniz adjunction in HoTT

Conclusion

- Main results:
  1. Segal types have all higher coherences
  2. Leibniz adjunction in HoTT
- The desire to formalize led to a much nicer proof.
- Families proved much more convenient than maps.

Conclusion

- Main results:
  1. Segal types have all higher coherences
  2. Leibniz adjunction in HoTT
- The desire to formalize led to a much nicer proof.
- Families proved much more convenient than maps.
- **?** Other illustrative examples where this perspective helps?
  Pointed maps; composition of pullbacks
- **?** Precise connection to Riehl & Shulman's type theory?

Conclusion

- Main results:
  1. Segal types have all higher coherences
  2. Leibniz adjunction in HoTT
- The desire to formalize led to a much nicer proof.
- Families proved much more convenient than maps.
- **?** Other illustrative examples where this perspective helps?
  Pointed maps; composition of pullbacks
- **?** Precise connection to Riehl & Shulman's type theory?

The Leibniz adjunction in homotopy type theory, with an application to simplicial type theory. TdJ, Nicolai Kraus, Axel Ljungström. January 2026. `arXiv:2601.21843`.

Fully formalized in Cubical Agda.

References

Gratzer, Daniel, Jonathan Weinberger and Ulrik Buchholtz (2024). 'Directed univalence in simplicial homotopy type theory'. arXiv: 2407.09146 [cs.LO].

Lurie, Jacob (2009). *Higher Topos Theory*. Vol. 170. Annals of Mathematics Studies. Princeton University Press. DOI: 10.1515/9781400830558. arXiv: math/0608040.

Riehl, Emily and Michael Shulman (2017). 'A type theory for synthetic ∞-categories'. In: *Higher Structures* 1.1, pp. 147–224. DOI: 10.21136/hs.2017.06.