HoTTLean

Formalizing the Meta-Theory of HoTT in Lean

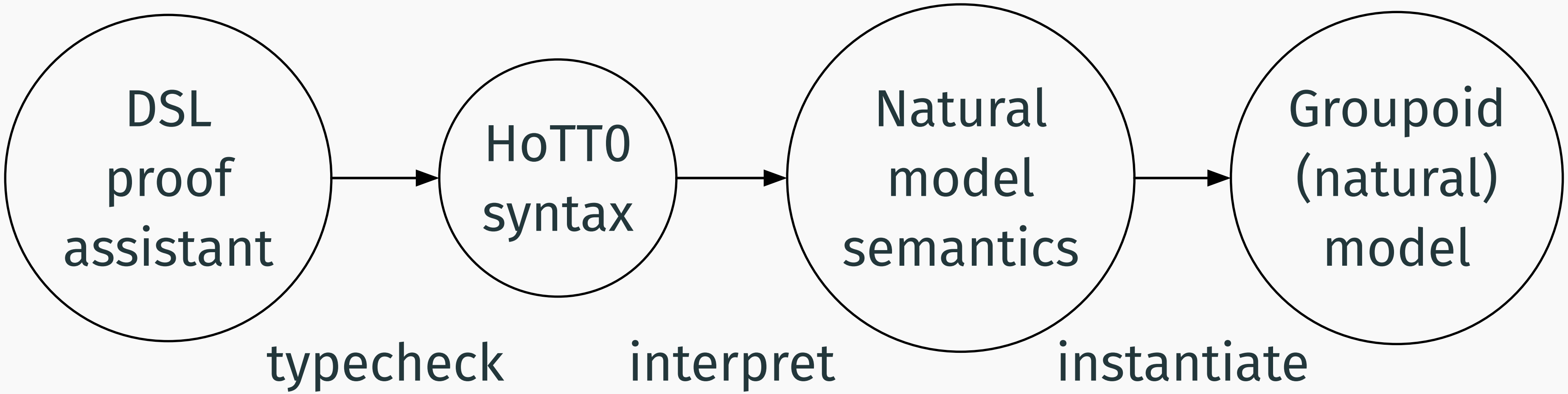
Joseph Hua [1] with Steve Awodey[1], Mario Carneiro[2], Sina Hazratpour[3], Wojciech Nawrocki[1], Spencer Woolfson[1], and Yiming Xu[4]

TYPES conference | University of Strathclyde | Friday 13 June 2025

[1] Carnegie Mellon University. [2] Chalmers University of Technology. [3] Stockholm University. [4] LMU Munich.

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0009.

Project overview



Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.

Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.
  - Apply synthetic homotopy theory results to concrete constructions (e.g. pushforward of an isofibration of groupoids is an isofibration)

Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.
  - Apply synthetic homotopy theory results to concrete constructions (e.g. pushforward of an isofibration of groupoids is an isofibration)
  - Apply internal language reasoning for sheaves. (Yiming Xu and Kenji Maillard 2025. Geometric reasoning in Lean)
  -
-
  -

Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.
  - Apply synthetic homotopy theory results to concrete constructions (e.g. pushforward of an isofibration of groupoids is an isofibration)
  - Apply internal language reasoning for sheaves. (Yiming Xu and Kenji Maillard 2025. Geometric reasoning in Lean)
  - Apply model theoretic results to classical algebra (e.g. Ax-Grothendieck.) (Flypitch/Mathlib.)
-
  -

Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.
  - Apply synthetic homotopy theory results to concrete constructions (e.g. pushforward of an isofibration of groupoids is an isofibration)
  - Apply internal language reasoning for sheaves. (Yiming Xu and Kenji Maillard 2025. Geometric reasoning in Lean)
  - Apply model theoretic results to classical algebra (e.g. Ax-Grothendieck.) (Flypitch/ Mathlib.)
- Semantics for **HoTT** formalised in Lean.
  -

Project aims

- Develop technology for **domain-specific languages (DSLs) in Lean** to extract mathlib-relevant proofs from **synthetic reasoning**.
  - Apply synthetic homotopy theory results to concrete constructions (e.g. pushforward of an isofibration of groupoids is an isofibration)
  - Apply internal language reasoning for sheaves. (Yiming Xu and Kenji Maillard 2025. Geometric reasoning in Lean)
  - Apply model theoretic results to classical algebra (e.g. Ax-Grothendieck.) (Flypitch/ Mathlib.)
- Semantics for **HoTT** formalised in Lean.
  - Sozeau and Tabareau 2014. Towards an internalization of the groupoid model of type theory

HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

•

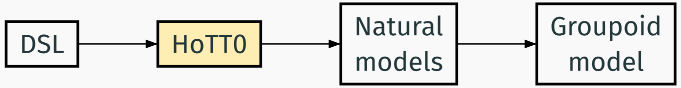
•

Noting that

•

•

•



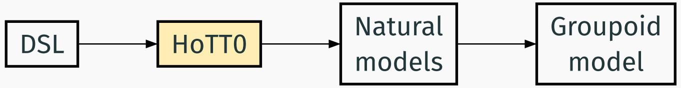
HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

- Martin-Löf Type Theory (MLTT) with universes $U_0 : U_1 : \cdots : U_n$ and Π, Σ, Id for each universe.
-

Noting that

-
-
-



HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

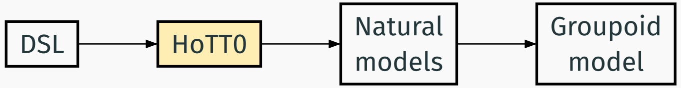
- Martin-Löf Type Theory (MLTT) with universes $U_0 : U_1 : \cdots : U_n$ and Π, Σ, Id for each universe.
- with two axioms: **univalence** for subuniverses of **sets** (0-truncated types), global functional extensionality.

Noting that

•

•

•



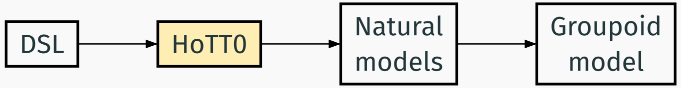
HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

- Martin-Löf Type Theory (MLTT) with universes $U_0 : U_1 : \cdots : U_n$ and Π, Σ, Id for each universe.
- with two axioms: **univalence** for subuniverses of **sets** (0-truncated types), global functional extensionality.

Noting that

- HoTT0 admits the **structure identity principle** for sets-level mathematics.



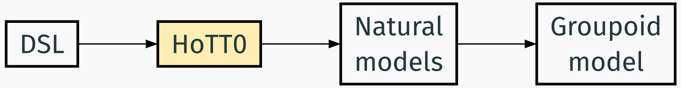
HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

- Martin-Löf Type Theory (MLTT) with universes $U_0 : U_1 : \cdots : U_n$ and Π, Σ, Id for each universe.
- with two axioms: **univalence** for subuniverses of **sets** (0-truncated types), global functional extensionality.

Noting that

- HoTT0 admits the **structure identity principle** for sets-level mathematics.
- Universes are not cumulative.



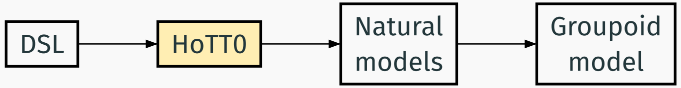
HoTT0 syntax

HoTT0 is a fragment of HoTT, meaning

- Martin-Löf Type Theory (MLTT) with universes $U_0 : U_1 : \cdots : U_n$ and Π, Σ, Id for each universe.
- with two axioms: **univalence** for subuniverses of **sets** (0-truncated types), global functional extensionality.

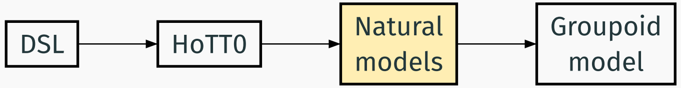
Noting that

- HoTT0 admits the **structure identity principle** for sets-level mathematics.
- Universes are not cumulative.
- Finitely many universes.



Natural model semantics - universes

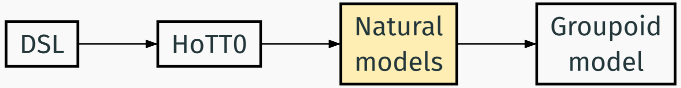
In a presheaf category $\mathsf{Set}^{\mathsf{Ctx}^{\mathrm{op}}}$



Natural model semantics - universes

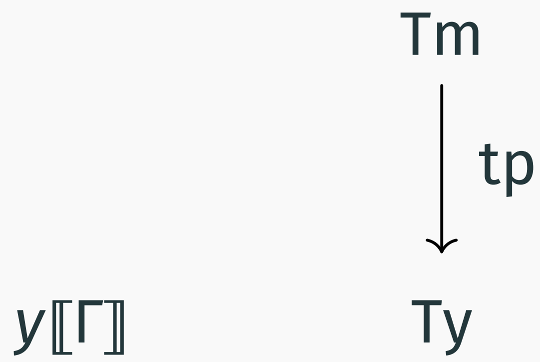
In a presheaf category $\mathbf{Set}^{\mathrm{Ctx}^{\mathrm{op}}}$

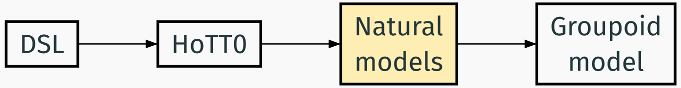
$$\begin{array}{c} \mathrm{Tm} \\ \big\downarrow \mathrm{tp} \\ \mathrm{Ty} \end{array}$$



Natural model semantics - universes

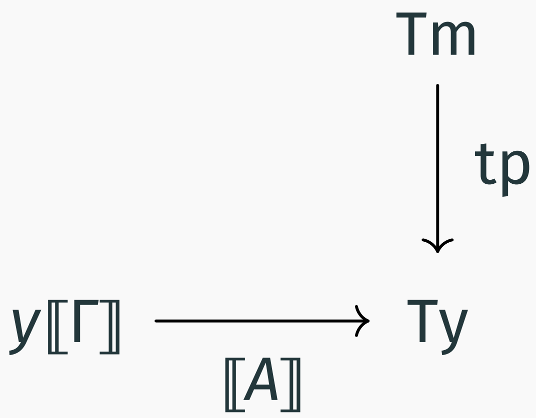
In a presheaf category $\mathrm{Set}^{\mathrm{Ctx}^{\mathrm{op}}}$

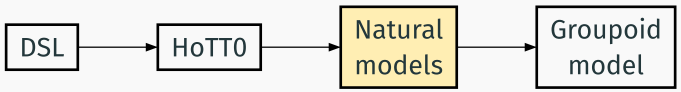




Natural model semantics - universes

In a presheaf category $\mathbf{Set}^{\mathrm{Ctx}^{\mathrm{op}}}$

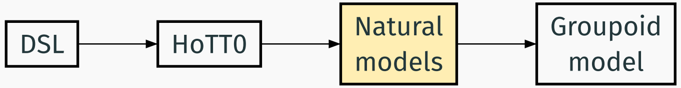




Natural model semantics - universes

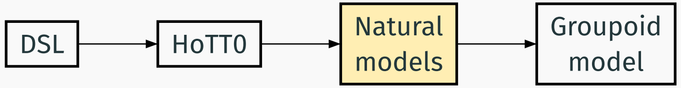
In a presheaf category $\mathbf{Set}^{\mathbf{Ctx}^{\mathrm{op}}}$

$$\begin{array}{ccc} y[\![\Gamma.A]\!] & \longrightarrow & \mathrm{Tm} \\ \downarrow & \lrcorner & \downarrow \mathrm{tp} \\ y[\![\Gamma]\!] & \xrightarrow[{[\![A]\!]}]{} & \mathrm{Ty} \end{array}$$



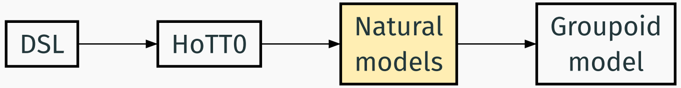
Natural model semantics - Π types

$$
\begin{array}{ccc}
P_{\mathrm{tp}}\,\mathrm{Tm} & & \mathrm{Tm} \\
\Big\downarrow{\scriptstyle P_{\mathrm{tp}}\,\mathrm{tp}} & & \Big\downarrow{\scriptstyle \mathrm{tp}} \\
P_{\mathrm{tp}}\,\mathrm{Ty} & & \mathrm{Ty}
\end{array}
$$



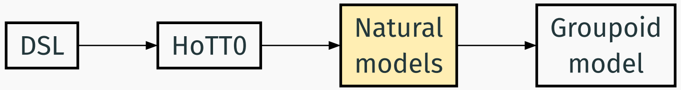
Natural model semantics - Π types

$$\begin{array}{ccc} P_{\mathrm{tp}}\,\mathrm{Tm} & \xrightarrow{\lambda} & \mathrm{Tm} \\ {\scriptstyle P_{\mathrm{tp}}\,\mathrm{tp}}\big\downarrow & & \big\downarrow{\scriptstyle \mathrm{tp}} \\ P_{\mathrm{tp}}\,\mathrm{Ty} & \xrightarrow[\Pi]{} & \mathrm{Ty}\end{array}$$



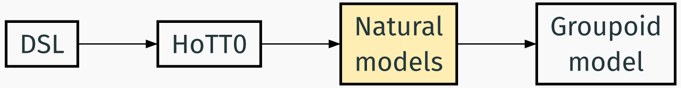
Natural model semantics - Π types

$$
\begin{array}{ccc}
P_{\mathrm{tp}}\,\mathrm{Tm} & \xrightarrow{\lambda} & \mathrm{Tm} \\
{\scriptstyle P_{\mathrm{tp}}\,\mathrm{tp}}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{tp}} \\
P_{\mathrm{tp}}\,\mathrm{Ty} & \xrightarrow[\Pi]{} & \mathrm{Ty}
\end{array}
$$



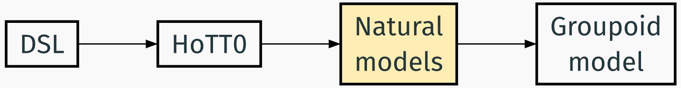
Theorems for Mathlib

- Lean formalisation of polynomial endofunctors (a.k.a containers). See github.com/sinhp/Poly project.
-
-



Theorems for Mathlib

- Lean formalisation of polynomial endofunctors (a.k.a containers). See github.com/sinhp/Poly project.

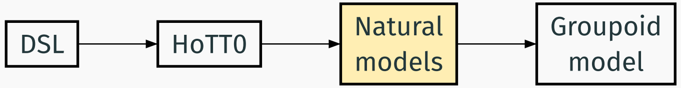


Theorems for Mathlib

- Lean formalisation of polynomial endofunctors (a.k.a containers). See github.com/sinhp/Poly project.
- Lean formalisation of profunctors. Universal property of polynomial endofunctors is composition of profunctor isomorphisms

```
/-- `C(Γ, PₚX) ≅ Σ(b : Γ → B), C(b*p, X)` -/
def iso_Sigma (P : UvPoly E B) :
    P.functor ⋙₂ coyoneda (C := C) ≅ P.partProdsOver :=
  … ≅ … ≅ … ≅ …
```

-

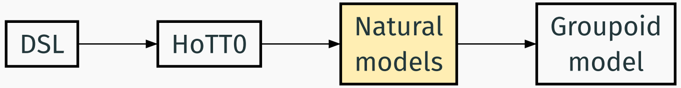


Theorems for Mathlib

- Lean formalisation of polynomial endofunctors (a.k.a containers). See github.com/sinhp/Poly project.
- Lean formalisation of profunctors. Universal property of polynomial endofunctors is composition of profunctor isomorphisms

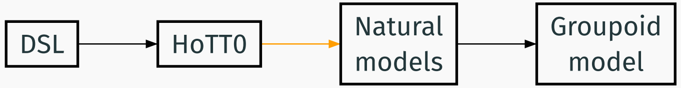
```
/-- `C(Γ, P_pX) ≅ Σ(b : Γ → B), C(b*p, X)` -/
def iso_Sigma (P : UvPoly E B) :
    P.functor ⋙₂ coyoneda (C := C) ≅ P.partProdsOver :=
  ... ≅ ... ≅ ... ≅ ...
```

- However this currently has significant performance issues (due to heavy `rfl` proofs).



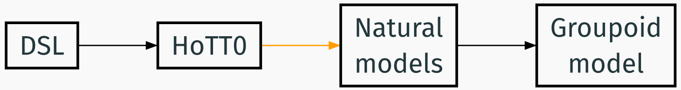
Interpretation of HoTT0 into natural model semantics

- Interpretation is a partial function on raw terms, that is defined on well-formed types and terms.



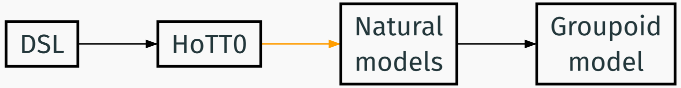
Interpretation of HoTT0 into natural model semantics

- Interpretation is a partial function on raw terms, that is defined on well-formed types and terms.
- We have constructed a sound interpretation of a fragment (with only Σ and Π types) into a class of natural models.
-



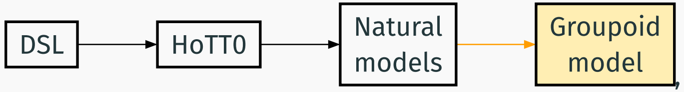
Interpretation of HoTT0 into natural model semantics

- Interpretation is a partial function on raw terms, that is defined on well-formed types and terms.
- We have constructed a sound interpretation of a fragment (with only Σ and Π types) into a class of natural models.
- Modular approach: we can plug in any natural model to this abstract interpretation.



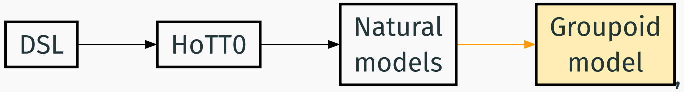
Groupoid model of HoTT0

- Category of contexts Ctx = Grpd is category of "large" groupoids (with (Type 5)-sized objects and arrows).
-
-



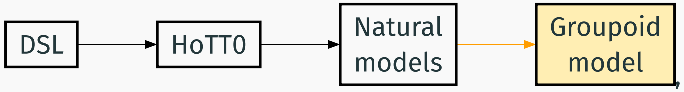
Groupoid model of HoTT0

- Category of contexts Ctx = Grpd is category of “large” groupoids (with (Type 5)-sized objects and arrows).
- $\mathrm{Ty}_0 = y(\mathrm{Grpd}_{\cong})$ is (Yoneda of) the core of the category of “small” groupoids (with (Type 0)-sized objects and arrows).
-



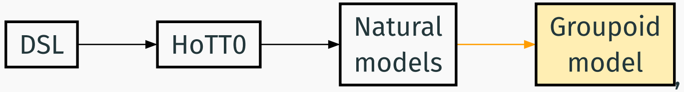
Groupoid model of HoTT0

- Category of contexts Ctx = Grpd is category of "large" groupoids (with (Type 5)-sized objects and arrows).
- $\mathrm{Ty}_0 = y\left(\mathrm{Grpd}_{\cong}\right)$ is (Yoneda of) the core of the category of "small" groupoids (with (Type 0)-sized objects and arrows).
- A type $A : y(\Gamma) \to \mathrm{Ty}_0$ is equivalent to a functor $\Gamma \to \mathrm{Grpd}$ from the "large" groupoid Γ into the category of "small" groupoids.



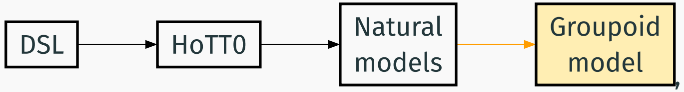
Necessary evil

- Mathlib convention is to avoid “evil” category theory: equal objects, equal functors, isomorphic categories…

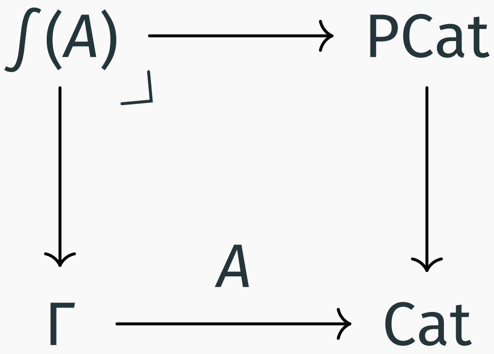


Necessary evil

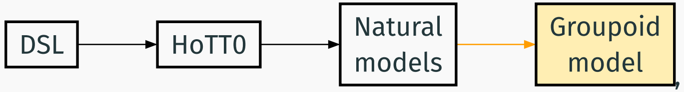
- Mathlib convention is to avoid “evil” category theory: equal objects, equal functors, isomorphic categories...
- Groupoid model **necessitates “evil” constructions**.
-
-



Necessary evil

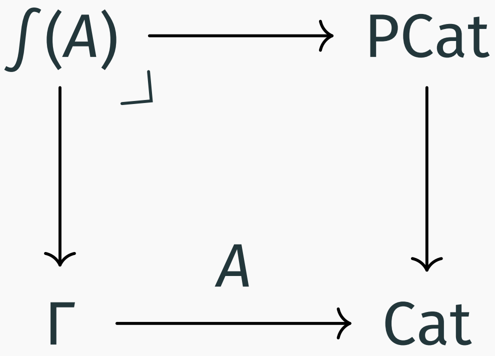


- Mathlib convention is to avoid “evil” category theory: equal objects, equal functors, isomorphic categories...
- Groupoid model **necessitates “evil” constructions.**
- Proven Grothendieck construction $\int(A)$ is a strict pullback of PCat. Developed API for pullbacks of categories.
-



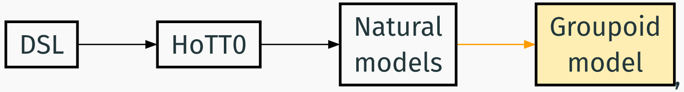
Necessary evil

- Mathlib convention is to avoid "evil" category theory: equal objects, equal functors, isomorphic categories...
- Groupoid model **necessitates "evil" constructions.**



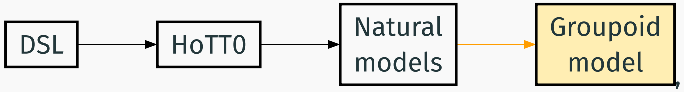
- Proven Grothendieck construction $\int(A)$ is a strict pullback of PCat. Developed API for pullbacks of categories.
- Note: Mathlib definitions are not general enough: categories in the pullback square are **not in the same category** due to universe levels.

```
Γ : Type u
Cat : Type (u+1)
```



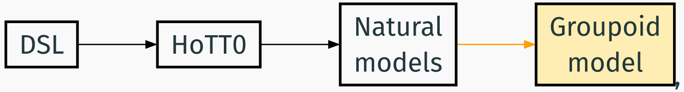
DTT hell - rewrite along paths over paths.

- Causes difficulties in formalising “evil” category theory (but a more general problem).
-
-
-



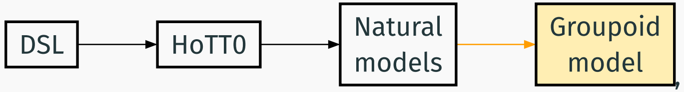
DTT hell - rewrite along paths over paths.

- Causes difficulties in formalising “evil” category theory (but a more general problem).
- If `x = y : A` and `B : A → Type` then `B x = B y : Type`, but cannot naively rewrite `p x : B x` for `p y : B y`.
-
-



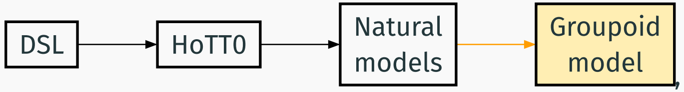
DTT hell - rewrite along paths over paths.

- Causes difficulties in formalising “evil” category theory (but a more general problem).
- If `x = y : A` and `B : A → Type` then `B x = B y : Type`, but cannot naively rewrite `p x : B x` for `p y : B y`.
- Developed (credit to Aaron Liu) Lean tactic `rw!` which can often (not always) rewrite such expressions.
-



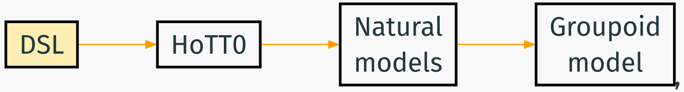
DTT hell - rewrite along paths over paths.

- Causes difficulties in formalising “evil” category theory (but a more general problem).
- If `x = y : A` and `B : A → Type` then `B x = B y : Type`, but cannot naively rewrite `p x : B x` for `p y : B y`.
- Developed (credit to Aaron Liu) Lean tactic `rw!` which can often (not always) rewrite such expressions.
- Towards a tactic for rewriting along Lean’s heterogeneous equality `HEq`.



Domain-specific language for HoTT0

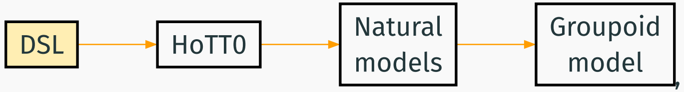
```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a
```



Domain-specific language for HoTT0

```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a

-- { l := 1,
--   val := lam 1 0 (univ 0) (lam 0 0 (el (bvar 0)) (bvar 0)),
--   tp := pi 1 0 (univ 0) (pi 0 0 (el (bvar 0)) (el (bvar 1))),
--   wf := (… : [] ⊢[l] val ≡ val : tp) }
#eval! idfun.checked -- type checker
```

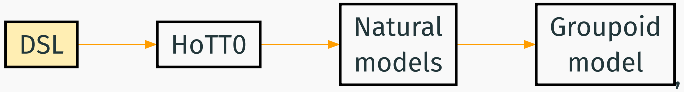


Domain-specific language for HoTT0

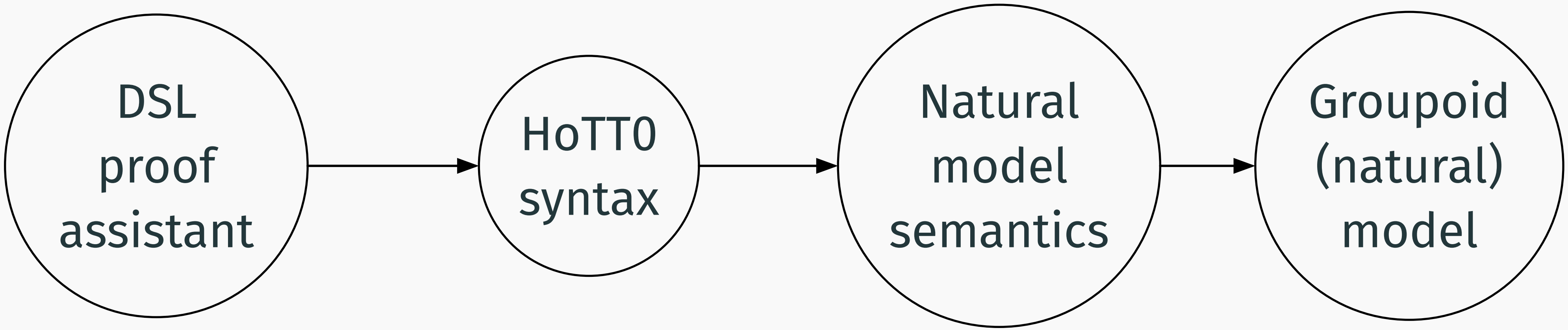
```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a

-- { l := 1,
--   val := lam 1 0 (univ 0) (lam 0 0 (el (bvar 0)) (bvar 0)),
--   tp := pi 1 0 (univ 0) (pi 0 0 (el (bvar 0)) (el (bvar 1))),
--   wf := (… : [] ⊢[l] val ≡ val : tp) }
#eval! idfun.checked -- type checker

-- type interpreted in groupoid model
noncomputable def GroupoidModel.idfun.interpType :
    ⊤_ _ ⟶ GroupoidModel.Ctx.ofCategory.{1,4} Grpd.{1,1} :=
  (uHomSeqPis.interpType …
    idfun.checked.wf.wf_tp … uHomSeqPis.nilCObj …).app (.op ◁ ⊤_ _)
(𝟙 _)
```



Project progress



typechecker	$U_n, \Sigma, \Pi, \mathrm{Id}$	$U_n, \Sigma, \Pi, \mathrm{Id}$	$U_n, \Sigma, \Pi, \mathrm{Id}$

\+ Set Univalence + Function Extensionality

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
-
-
-
-

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
- Rewriting tactics (e.g. `HEq`)
-
-
-

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
- Rewriting tactics (e.g. `HEq`)
- Formalisation of identity types
-
-

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
- Rewriting tactics (e.g. `HEq`)
- Formalisation of identity types
- User interface for DSLs
-

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
- Rewriting tactics (e.g. `HEq`)
- Formalisation of identity types
- User interface for DSLs
- Profunctor library performance issues

Next steps and ways to contribute

Repository: github.com/sinhp/groupoid_model_in_lean4

Possible ways to contribute:

- Polynomial functor library
- Rewriting tactics (e.g. `HEq`)
- Formalisation of identity types
- User interface for DSLs
- Profunctor library performance issues

LMU student thesis project.

Bibliography

[1] M. Sozeau and N. Tabareau, “Towards an internalization of the groupoid model of type theory,” *Types for Proofs and Programs 20th Meething (TYPES 2014), Book of Abstracts*, 2014.

[2] T. M. Community, “The Lean mathematical library,” in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, in CPP 2020. New Orleans, LA, USA: Association for Computing Machinery, 2020, pp. 367–381. doi: 10.1145/3372885.3373824.

[3] T. M. Community, “Mathlib.”

[4] T. Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: \url{https://homotopytypetheory.org/book}, 2013.

[5] M. Hofmann and T. Streicher, “The Groupoid Interpretation of Type Theory,” *Twenty-five years of constructive type theory (Venice, 1995)*, vol. 36. in Oxford Logic Guides, vol. 36. Oxford Univ. Press, New York, pp. 83–111, 1998.

[6] S. Ullrich and L. de Moura, “Beyond Notations: Hygienic Macro Expansion for Theorem Proving Languages,” in *Automated Reasoning*, N. Peltier and V. Sofronie-Stokkermans, Eds., Cham: Springer International Publishing, 2020, pp. 167–182.

[7] S. Awodey, “Natural models of homotopy type theory,” *Mathematical Structures in Computer Science*, vol. 28, no. 2, pp. 241–286, 2018.

[8] N. Veltri and N. van der Weide, “Constructing Higher Inductive Types as Groupoid Quotients,” *Logical Methods in Computer Science*, Apr. 2021, doi: 10.23638/lmcs-17(2:8)2021.

[9] T. Altenkirch, P. Capriotti, G. Dijkstra, N. Kraus, and F. Nordvall Forsberg, "Quotient Inductive-Inductive Types," in *Foundations of Software Science and Computation Structures*, Springer International Publishing, 2018, pp. 293–310. doi: 10.1007/978-3-319-89366-2_16.

[10] J. Vidmar, "Polynomial functors and W-types for groupoids," 2018.

[11] P. R. North, "Towards a directed homotopy type theory," *Electronic Notes in Theoretical Computer Science*, vol. 347, pp. 223–239, 2019.

[12] B. Ahrens, P. R. North, and N. van der Weide, "Bicategorical type theory: semantics and syntax," *Mathematical Structures in Computer Science*, vol. 33, no. 10, pp. 868–912, 2023.

[13] P. Aczel, "On Voevodsky's univalence axiom," *Mathematical Logic: Proof Theory, Constructive Mathematics, Samuel R. Buss, Ulrich Kohlenbach, and Michael Rathjen (Eds.). Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach*, p. 2967, 2011.

[14] J. M. Han and F. van Doorn, "A formal proof of the independence of the continuum hypothesis," in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, 2020.

[15] Y. Xu and K. Maillard, "Geometric Reasoning in Lean: from Algebraic Structures to Presheaves," in *31st International Conference on Types for Proofs and Programs*, 2025. [Online]. Available: https://openreview.net/forum?id=H6crYKvzHI

[16] M. Carneiro and E. Riehl, "Formalizing colimits in Cat," *arXiv preprint arXiv:2503.20704*, 2025.