PSSL 109
Leiden University & DutchCats
the 17^{th} of November 2024

Higher dimensional semantics of axiomatic dependent type theories

Matteo Spadetto
University of Udine

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*
- **term judgements**, $t : A$, read as t *is a proof of the statement* A

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*
- **term judgements**, $t : A$, read as t *is a proof of the statement* A
- **type equality judgements**, $A \equiv B$

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*
- **term judgements**, $t : A$, read as t *is a proof of the statement* A
- **type equality judgements**, $A \equiv B$
- **term equality judgements**, $t \equiv t' : A$

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*
- **term judgements**, $t : A$, read as t *is a proof of the statement* A
- **type equality judgements**, $A \equiv B$
- **term equality judgements**, $t \equiv t' : A$

Type constructors. Groups of deduction rules that encode pieces of logic.

Theories of dependent types

Theories that can make:

- **type judgements**, $A : \textsc{Type}$ (possibly releative to a context of variables, $\Gamma \vdash A : \textsc{Type}$), read as A *is a statement*
- **term judgements**, $t : A$, read as t *is a proof of the statement* A
- **type equality judgements**, $A \equiv B$
- **term equality judgements**, $t \equiv t' : A$

Type constructors. Groups of deduction rules that encode pieces of logic.

E.g.
identity type constructor, $t = t'$, for a notion of equality
dependent sum type constructor, $\Sigma_{x:A} B(x)$, for a notion of existential quantification (that we will focus on today)

Semantics of these theories

Semantics consists of *category theoretic copies* - formulated e.g. as **display map categories** - of a given theory, that *encode as morphisms* and properties between morphisms these type constructors.

Semantics of these theories

Semantics consists of *category theoretic copies* - formulated e.g. as **display map categories** - of a given theory, that *encode as morphisms* and properties between morphisms these type constructors.

There are essentially two approaches:

- a **syntactic** approach, *encoding type constructors in alignment with the syntax*
- a **categorical** approach, *characterising type constructors categorically*

Semantics of these theories

Semantics consists of *category theoretic copies* - formulated e.g. as **display map categories** - of a given theory, that *encode as morphisms* and properties between morphisms these type constructors.

There are essentially two approaches:

- a **syntactic** approach, *encoding type constructors in alignment with the syntax*
- a **categorical** approach, *characterising type constructors categorically*

The syntactic formulation can be used to **prove** things of the theory, while the categorical one to find specific models that can be used to **disprove** things of the theory.

Extensional theories (where identity proofs are irrelevant)

Extensional identity types

$$\frac{\vdash A : \textsc{Type}}{\begin{array}{c} x, x' : A \vdash x = x' : \textsc{Type} \\ x : A \vdash r(x) : x = x \end{array}} \qquad \frac{\vdash A : \textsc{Type}}{\begin{array}{c} x, x' : A, p : x = x' \vdash x \equiv x' \\ x, x' : A, p : x = x' \vdash p \equiv r(x) \end{array}}$$

Dependent sum types

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \end{array}}{\begin{array}{c} \vdash \Sigma_{x:A} B(x) : \textsc{Type} \\ x : A, y : B(x) \vdash \langle x, y \rangle : \Sigma_{x:A} B(x) \end{array}} \qquad \frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \\ u : \Sigma_{x:A} B(x) \vdash C(u) : \textsc{Type} \\ x : A;\ y : B(x) \vdash c(x, y) : C(\langle x, y \rangle) \end{array}}{\begin{array}{c} u : \Sigma_{x:A} B(x) \vdash \mathrm{split}(c, u) : C(u) \\ x : A;\ y : B(x) \vdash \quad \mathrm{split}(c, \langle x, y \rangle) \equiv c(x, y) \end{array}}$$

Intensional theories (with computation rules)

Intensional identity types

$$\frac{\vdash A : \textsc{Type}}{\begin{array}{c} x, x' : A \vdash x = x' : \textsc{Type} \\ x : A \vdash r(x) : x = x \end{array}}$$

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x, x' : A;\ p : x = x' \vdash C(x, x', p) : \textsc{Type} \\ x : A \vdash q(x) : C(x, x, r(x)) \end{array}}{\begin{array}{l} x, x' : A;\ p : x = x' \vdash J(q, x, x', p) : C(x, x', p) \\ x : A \vdash \qquad J(q, x, x, r(x)) \equiv q(x) \end{array}}$$

Dependent sum types

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \end{array}}{\begin{array}{c} \vdash \Sigma_{x:A} B(x) : \textsc{Type} \\ x : A, y : B(x) \vdash \langle x, y \rangle : \Sigma_{x:A} B(x) \end{array}}$$

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \\ u : \Sigma_{x:A} B(x) \vdash C(u) : \textsc{Type} \\ x : A;\ y : B(x) \vdash c(x, y) : C(\langle x, y \rangle) \end{array}}{\begin{array}{l} u : \Sigma_{x:A} B(x) \vdash \mathrm{split}(c, u) : C(u) \\ x : A;\ y : B(x) \vdash \qquad \mathrm{split}(c, \langle x, y \rangle) \equiv c(x, y) \end{array}}$$

Axiomatic theories (with computation *axioms*)

Axiomatic identity types

$$\frac{\vdash A : \textsc{Type}}{\begin{array}{c} x, x' : A \vdash x = x' : \textsc{Type} \\ x : A \vdash r(x) : x = x \end{array}}
\qquad
\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x, x' : A;\ p : x = x' \vdash C(x, x', p) : \textsc{Type} \\ x : A \vdash q(x) : C(x, x, r(x)) \end{array}}{\begin{array}{c} x, x' : A;\ p : x = x' \vdash J(q, x, x', p) : C(x, x', p) \\ x : A \vdash \qquad J(q, x, x, r(x)) \not\equiv q(x) \end{array}}$$

Axiomatic dependent sum types

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \end{array}}{\begin{array}{c} \vdash \Sigma_{x:A} B(x) : \textsc{Type} \\ x : A, y : B(x) \vdash \langle x, y \rangle : \Sigma_{x:A} B(x) \end{array}}
\qquad
\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \\ u : \Sigma_{x:A} B(x) \vdash C(u) : \textsc{Type} \\ x : A;\ y : B(x) \vdash c(x, y) : C(\langle x, y \rangle) \end{array}}{\begin{array}{c} u : \Sigma_{x:A} B(x) \vdash \mathrm{split}(c, u) : C(u) \\ x : A;\ y : B(x) \vdash \qquad \mathrm{split}(c, \langle x, y \rangle) \not\equiv c(x, y) \end{array}}$$

Axiomatic theories (with computation *axioms*)

Axiomatic identity types

$$\frac{\vdash A : \textsc{Type}}{\begin{array}{c} x, x' : A \vdash x = x' : \textsc{Type} \\ x : A \vdash r(x) : x = x \end{array}}$$

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x, x' : A;\ p : x = x' \vdash C(x, x', p) : \textsc{Type} \\ x : A \vdash q(x) : C(x, x, r(x)) \end{array}}{\begin{array}{c} x, x' : A;\ p : x = x' \vdash J(q, x, x', p) : C(x, x', p) \\ x : A \vdash H(q, x) : J(q, x, x, r(x)) = q(x) \end{array}}$$

Axiomatic dependent sum types

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \end{array}}{\begin{array}{c} \vdash \Sigma_{x:A} B(x) : \textsc{Type} \\ x : A, y : B(x) \vdash \langle x, y \rangle : \Sigma_{x:A} B(x) \end{array}}$$

$$\frac{\begin{array}{c} \vdash A : \textsc{Type} \\ x : A \vdash B(x) : \textsc{Type} \\ u : \Sigma_{x:A} B(x) \vdash C(u) : \textsc{Type} \\ x : A;\ y : B(x) \vdash c(x, y) : C(\langle x, y \rangle) \end{array}}{\begin{array}{c} u : \Sigma_{x:A} B(x) \vdash \mathrm{split}(c, u) : C(u) \\ x : A;\ y : B(x) \vdash \sigma(c, x, y) : \mathrm{split}(c, \langle x, y \rangle) = c(x, y) \end{array}}$$

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

To have a model of a type constructor:

- In the **syntactic approach** one copies the type constructor into a display map category by means of a choice function in the language of the display map category.

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

To have a model of a type constructor:

- In the **syntactic approach** one copies the type constructor into a display map category by means of a choice function in the language of the display map category.

  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$ there is a choice of a display map $\Gamma.A.A'.(x = x') \to \Gamma.A.A$ (formation rule) together with a choice of a section $\Gamma.A \to \Gamma.A.(x = x)$ of $\Gamma.A.(x = x) \to \Gamma.A$ (introduction rule), etc..
  - Dependent sum types (in presence of extensional identities). Analogously.

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

To have a model of a type constructor:

- In the **syntactic approach** one copies the type constructor into a display map category by means of a choice function in the language of the display map category.
  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$ there is a choice of a display map $\Gamma.A.A'.(x = x') \to \Gamma.A.A$ (formation rule) together with a choice of a section $\Gamma.A \to \Gamma.A.(x = x)$ of $\Gamma.A.(x = x) \to \Gamma.A$ (introduction rule), etc..
  - Dependent sum types (in presence of extensional identities). Analogously.
- In the **category theoretic approach** one looks for a 1-dimensional categorical property to give to display maps that *characterises* the type constructor, allowing *a choice function as in the syntactic approach to be induced* by this property.

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

To have a model of a type constructor:

- In the **syntactic approach** one copies the type constructor into a display map category by means of a choice function in the language of the display map category.
  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$ there is a choice of a display map $\Gamma.A.A'.(x = x') \to \Gamma.A.A$ (formation rule) together with a choice of a section $\Gamma.A \to \Gamma.A.(x = x)$ of $\Gamma.A.(x = x) \to \Gamma.A$ (introduction rule), etc..
  - Dependent sum types (in presence of extensional identities). Analogously.
- In the **category theoretic approach** one looks for a 1-dimensional categorical property to give to display maps that *characterises* the type constructor, allowing *a choice function as in the syntactic approach to be induced* by this property.
  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$, the unique diagonal arrow $\Gamma.A \to \Gamma.A.A'$ is itself a display map.
  - Dependent sum types (in presence of extensional identities). Up to isomorphism, display maps are closed under composition.

How semantics works

In a **display map category** we are given a family of display maps (notion introduced by **Paul Taylor**), denoted as $\Gamma.A \to \Gamma$ that interpret type judgements $\Gamma \vdash A : \textsc{Type}$. Term judgements $\Gamma \vdash t : A$ are interpreted as sections $\Gamma \to \Gamma.A$ of the corresponding display map.

To have a model of a type constructor:

- In the **syntactic approach** one copies the type constructor into a display map category by means of a choice function in the language of the display map category.

  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$ there is a choice of a display map $\Gamma.A.A'.(x = x') \to \Gamma.A.A$ (formation rule) together with a choice of a section $\Gamma.A \to \Gamma.A.(x = x)$ of $\Gamma.A.(x = x) \to \Gamma.A$ (introduction rule), etc..
  - Dependent sum types (in presence of extensional identities). Analogously.
- In the **category theoretic approach** one looks for a 1-dimensional categorical property to give to display maps that *characterises* the type constructor, allowing *a choice function as in the syntactic approach to be induced* by this property.

  Example:
  - Extensional identity types. For every display map $\Gamma.A \to \Gamma$, the unique diagonal arrow $\Gamma.A \to \Gamma.A.A'$ is itself a display map.
  - Dependent sum types (in presence of extensional identities). Up to isomorphism, display maps are closed under composition.

  Way easier to formulate!

How semantics works

For **extensional** dependent type theories, the categorical approach is clear and conceptually simple to formulate.

This is not the case for **intensional**, and **axiomatic**, dependent type theories: there aren't obvious categorical properties to characterise intensional and propositional inference rules.

How semantics works

For **extensional** dependent type theories, the categorical approach is clear and conceptually simple to formulate.

This is not the case for **intensional**, and **axiomatic**, dependent type theories: there aren't obvious categorical properties to characterise intensional and propositional inference rules.

Garner's approach: in order to characterise intensional type constructors, we can use **2-dimensional models**, *that still can be converted into ordinary models according to the syntactic approach*, and 2-dimensional - e.g. weakly universal - categorical properties.

How semantics works

For **extensional** dependent type theories, the categorical approach is clear and conceptually simple to formulate.

This is not the case for **intensional**, and **axiomatic**, dependent type theories: there aren't obvious categorical properties to characterise intensional and propositional inference rules.

Garner's approach: in order to characterise intensional type constructors, we can use **2-dimensional models**, *that still can be converted into ordinary models according to the syntactic approach*, and 2-dimensional - e.g. weakly universal - categorical properties.

This approach can also be used for axiomatic theories.

Goal. Having a 2-dimensional structure with natural categorical conditions that allow to interpret axiomatic theories.

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. The class of display maps is closed under **2-dimensional re-indexing**.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. The class of display maps is closed under **2-dimensional re-indexing**.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, "\Rightarrow", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, "=", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. The class of display maps is closed under **2-dimensional re-indexing**.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, "\Rightarrow", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, "=", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
$$

3. Every display map has an **arrow object**.

$$
\begin{tikzcd}
\Omega \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad \text{s.t.} \quad
\begin{array}{c}
\hom_\Gamma(\Delta, \Omega) \\
\cong \\
\hom_\Gamma(\Delta, \Gamma.A)^{\to}
\end{array}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. The class of display maps is closed under **2-dimensional re-indexing**.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] & \Gamma.A \arrow[d] \arrow[dl, phantom, "\Rightarrow"] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] & \Gamma.A \arrow[d] \arrow[dl, phantom, "="] \\
 & \Gamma
\end{tikzcd}
$$

3. Every display map has an **arrow object**.

$$
\begin{tikzcd}
\Omega \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad \text{s.t.} \quad
\begin{array}{c}
\hom_\Gamma(\Delta, \Omega) \\
\cong \\
\hom_\Gamma(\Delta, \Gamma.A)^{\rightarrow}
\end{array}
$$

4. The class of display maps is closed under composition, up to **homotopy equiv.**.

$$
\begin{tikzcd}
\Gamma.A.B \arrow[d] & \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Gamma.A.B \arrow[d] & \simeq & \Gamma.C \arrow[d] \\
\Gamma.A \arrow[rr] & & \Gamma
\end{tikzcd}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. To substitute into types and terms.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d, two heads] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d, two heads] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, "\Rightarrow", shift left=3] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, "=", shift left=3] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
$$

3. Every display map has an **arrow object**.

$$
\begin{tikzcd}
\Omega \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad \text{s.t.} \quad
\begin{array}{c}
\hom_\Gamma(\Delta, \Omega) \\
\cong \\
\hom_\Gamma(\Delta, \Gamma.A)^{\rightarrow}
\end{array}
$$

4. The class of display maps is closed under composition, up to **homotopy equiv.**.

$$
\begin{tikzcd}
\Gamma.A.B \arrow[d, two heads] & \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Gamma.A.B \arrow[d, two heads] \arrow[r, phantom, "\simeq"] & \Gamma.C \arrow[d, two heads] \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. To substitute into types and terms.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, "\Rightarrow", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, "=", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
$$

3. To have identity types with *pseudo-elimination*.

$$
\begin{tikzcd}
\Omega \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad \text{s.t.} \quad
\begin{array}{c}
\hom_\Gamma(\Delta, \Omega) \\
\wr\| \\
\hom_\Gamma(\Delta, \Gamma.A)^{\rightarrow}
\end{array}
\qquad
\begin{array}{c}
\Gamma.A.A'.(x = x') \\
\vdots\| \\
\Omega
\end{array}
$$

4. The class of display maps is closed under composition, up to **homotopy equiv.**.

$$
\begin{tikzcd}
\Gamma.A.B \arrow[d] & \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Gamma.A.B \arrow[r, phantom, "\simeq"] \arrow[d] & \Gamma.C \arrow[d] \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. To substitute into types and terms.

$$
\begin{tikzcd}
 & \Gamma.A \arrow[d] \\
\Delta \arrow[r, dashed, "f"] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\
\Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}
$$

2. Every display map is a **cloven isofibration**.

$$
\begin{tikzcd}
\Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, "\Rightarrow", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, "=", shift left=2] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
$$

3. To have identity types with *pseudo-elimination*.

$$
\begin{tikzcd}
\Omega \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\
 & \Gamma
\end{tikzcd}
\quad \text{s.t.} \quad
\begin{array}{c}
\hom_\Gamma(\Delta, \Omega) \\
\wr\| \\
\hom_\Gamma(\Delta, \Gamma.A)^{\to}
\end{array}
\qquad
\begin{array}{c}
\Gamma.A.A'.(x = x') \\
\| \\
\Omega
\end{array}
$$

4. To have dependent sum types with *pseudo-elimination*.

$$
\begin{tikzcd}
\Gamma.A.B \arrow[d] & \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
\Gamma.A.B \arrow[r, phantom, "\simeq"] \arrow[d] & \Gamma.C \arrow[d] \\
\Gamma.A \arrow[r] & \Gamma
\end{tikzcd}
\qquad
\begin{array}{c}
\Sigma_A^B \\
\| \\
C
\end{array}
$$

2-dimensional semantics of propositional theories

Display map 2-categories. (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

1. To substitute into types and terms.

$$\begin{tikzcd} & \Gamma.A \arrow[d] \\ \Delta \arrow[r, dashed, "f"] & \Gamma \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} \Delta.A[f] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma.A \arrow[d] \\ \Delta \arrow[r, "f"] & \Gamma \end{tikzcd}$$

2. To strictify eliminations in 3-4 in change of producing computation *axioms*.

$$\begin{tikzcd} \Delta \arrow[r] \arrow[dr] \arrow[dr, phantom, shift left=3, "\Rightarrow"] & \Gamma.A \arrow[d] \\ & \Gamma \end{tikzcd} \quad = \quad \begin{tikzcd} \Delta \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Uparrow"] \arrow[dr] \arrow[dr, phantom, shift left=3, "="] & \Gamma.A \arrow[d] \\ & \Gamma \end{tikzcd}$$

3. To have identity types with *pseudo-elimination*.

$$\begin{tikzcd} \Omega \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] \arrow[dr] & \Gamma.A \arrow[d] \\ & \Gamma \end{tikzcd} \quad \text{s.t.} \quad \begin{tikzcd} \hom_\Gamma(\Delta, \Omega) \arrow[d, phantom, "\cong" rotate=90] \\ \hom_\Gamma(\Delta, \Gamma.A)^\to \end{tikzcd} \qquad \begin{tikzcd} \Gamma.A.A'.(x = x') \arrow[d, phantom, "\rotatebox{90}{$=:$}"] \\ \Omega \end{tikzcd}$$

4. To have dependent sum types with *pseudo-elimination*.

$$\begin{tikzcd} \Gamma.A.B \arrow[d] & \\ \Gamma.A \arrow[r] & \Gamma \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} \Gamma.A.B \arrow[d] \arrow[r, phantom, "\simeq"] & \Gamma.C \arrow[d] \\ \Gamma.A \arrow[r] & \Gamma \end{tikzcd} \qquad \begin{tikzcd} \Sigma_A^B \arrow[d, phantom, "\rotatebox{90}{$=:$}"] \\ C \end{tikzcd}$$

2-dimensional semantics of propositional theories

Main theorem. *Display map 2-categories are models of axiomatic dependent type theory.*

2-dimensional semantics of propositional theories

Main theorem. *Display map 2-categories are models of axiomatic dependent type theory.*

An application:

Theorem. *The judgemental computation rule for intensional identity type constructor is independent of the axiomatic dependent type theory.*

Proof

2-dimensional semantics of propositional theories

Main theorem. *Display map 2-categories are models of axiomatic dependent type theory.*

An application:

Theorem. *The judgemental computation rule for intensional identity type constructor is independent of the axiomatic dependent type theory.*

Proof i.e. a revisitation of the groupoid model.

We consider the (2,1)-category GRPD of groupoids, functors, and natural transformations (i.e. natural isomorphisms) with **Grothendieck constructions of *pseudofunctors*** $\mathbf{\Gamma \to GRPD}$ as display maps over Γ.

2-dimensional semantics of propositional theories

Main theorem. *Display map 2-categories are models of axiomatic dependent type theory.*

An application:

Theorem. *The judgemental computation rule for intensional identity type constructor is independent of the axiomatic dependent type theory.*

Proof i.e. a revisitation of the groupoid model.

We consider the (2,1)-category GRPD of groupoids, functors, and natural transformations (i.e. natural isomorphisms) with **Grothendieck constructions of *pseudofunctors* $\mathbf{\Gamma \to GRPD}$** as display maps over Γ.
The model of axiomatic theory induced by this display map 2-category does not believe the judgemental computation rule, so the statement follows by soundness.

Do we obtain *every* model?

Do we obtain *every* model?

No,

Do we obtain *every* model?

No, because every such display map 2-category believes the following rule:

Discreteness

$$\frac{\vdash A : \textsc{Type}}{x, y : A;\ p, q : x = y;\ \alpha : p = q \vdash p \equiv q}$$

Do we obtain *every* model?

No, because every such display map 2-category believes the following rule:

Discreteness

$$\frac{\vdash A : \textsc{Type}}{x, y : A;\ p, q : x = y;\ \alpha : p = q \vdash p \equiv q}$$

Theorem. *The display map 2-categories are precisely the models (as in the syntactic formulation) of the axiomatic theory* **extended with the discreteness rule.**

Do we obtain *every* model?

No, because every such display map 2-category believes the following rule:

Discreteness

$$\frac{\vdash A : \textsc{Type}}{x, y : A;\ p, q : x = y;\ \alpha : p = q \vdash p \equiv q}$$

Theorem. *The display map 2-categories are precisely the models (as in the syntactic formulation) of the axiomatic theory* **extended with the discreteness rule.** *Therefore, this notion of semantics is* **sound** *w.r.t. the axiomatic theory of dependent types, and it is* **sound and complete** *w.r.t. the axiomatic theory of dependent types extended with the discreteness rule.*