$\begin{array}{c} {\rm PSSL~109}\\ {\rm Leiden~University~\&~DutchCats}\\ {\rm the~17^{th}~of~November~2024} \end{array}$ 

Higher dimensional semantics of axiomatic dependent type theories

> Matteo Spadetto University of Udine

> > ◆□▶ ◆□▶ ◆目▶ ◆目▶ 目 の

Theories that can make:

**type judgements**, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : TYPE), read as A is a statement

◆□▶ < @ ▶ < @ ▶ < @ ▶ = 0</p>

Theories that can make:

▶ type judgements, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : TYPE), read as A is a statement

◆□▶ < @ ▶ < @ ▶ < @ ▶ = 0</p>

**term judgements**, t : A, read as t is a proof of the statement A

Theories that can make:

▶ type judgements, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : TYPE), read as A is a statement

◆□▶ ◆□▶ ◆□▶ ◆□▶ □ の

- **term judgements**, t : A, read as t is a proof of the statement A
- **•** type equality judgements,  $A \equiv B$

Theories that can make:

▶ type judgements, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : TYPE), read as A is a statement

◆□▶ ◆□▶ ◆目▶ ◆目▶ 目 の

- **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 that can make:

- ▶ type judgements, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : 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 that can make:

- ▶ type judgements, A: TYPE (possibly releative to a context of variables,  $\Gamma \vdash A$ : 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)

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ の

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 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

(ロ) (部) (注) (注) (注) ()

▶ a **categorical** approach, characterising type constructors categorically

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

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

Dependent sum types

## Intensional theories (with computation rules)

#### Intensional identity types

	$\vdash A : Type$
	$x, x': A; \ p: x = x' \vdash C(x, x', p): Type$
$\vdash A : Type$	$x:A\vdash q(x):C(x,x,r(x))$
$\overline{x, x' : A \vdash x = x' : \mathrm{Type}}$	$\overline{x,x':A;\ p:x=x'\vdash J(q,x,x',p):C(x,x',p)}$
$x:A\vdash r(x):x=x$	$x:A \vdash \qquad \qquad J(q,x,x,r(x)) \equiv q(x)$

Dependent sum types

$$\begin{array}{c} \vdash A: \mathrm{TypE} \\ x:A \vdash B(x): \mathrm{TypE} \\ \hline \\ \underbrace{x:A \vdash B(x): \mathrm{TypE}}_{k:A \vdash B(x): \mathrm{TypE}} \\ x:A \vdash B(x): \mathrm{TypE} \\ \hline \\ \underbrace{x:A \vdash B(x): \mathrm{TypE}}_{x:A \vdash B(x): \mathrm{TypE}} \\ x:A, y:B(x) \vdash \langle x, y \rangle: \Sigma_{x:A}B(x) \\ \hline \\ x:A, y:B(x) \vdash \langle x, y \rangle: \Sigma_{x:A}B(x) \\ \hline \\ \end{array}$$

# Axiomatic theories<sup>1</sup> (with computation axioms)

Axiomatic identity types

$$\begin{array}{c} \vdash A : \text{Type} \\ \hline x, x' : A \vdash x = x' : \text{Type} \\ x : A \vdash r(x) : x = x \end{array} \qquad \begin{array}{c} \vdash A : \text{Type} \\ x, x' : A; \ p : x = x' \vdash C(x, x', p) : \text{Type} \\ \hline x, x' : A; \ p : x = x' \vdash C(x, x, r(x)) \\ \hline x, x' : A; \ p : x = x' \vdash J(q, x, x', p) : C(x, x', p) \\ \hline x : A \vdash U(q, x, x, r(x)) \neq q(x) \end{array}$$

Axiomatic dependent sum types

$$\begin{array}{c} \vdash A: \mathrm{TypE} \\ x:A \vdash B(x): \mathrm{TypE} \\ \hline \\ x:A \vdash B(x): \mathrm{TypE} \\ \hline \\ x:A \vdash B(x): \mathrm{TypE} \\ \hline \\ x:A,y:B(x) \vdash \langle x,y \rangle: \Sigma_{x:A}B(x) \end{array} \qquad \begin{array}{c} \vdash A: \mathrm{TypE} \\ x:A \vdash B(x): \mathrm{TypE} \\ u: \Sigma_{x:A}B(x) \vdash C(u): \mathrm{TypE} \\ \hline \\ x:A; y:B(x) \vdash c(x,y): C(\langle x,y \rangle) \\ \hline \\ u: \Sigma_{x:A}B(x) \vdash \mathrm{split}(c,u): C(u) \\ x:A; y:B(x) \vdash \\ \end{array}$$

 $<sup>^1\</sup>mathrm{Also}$  known as weak, objective, propositional theories.

<sup>▲□▶ ▲</sup>圖▶ ▲臣▶ ▲臣▶ 二臣 → 約

# Axiomatic theories (with computation axioms)

Axiomatic identity types

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

Axiomatic dependent sum types

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

(ロ) (個) (目) (目) (E) (の)

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  $\Gamma \vdash A : \text{Type.}$  Term judgements  $\Gamma \vdash t : A$  are interpreted as sections  $\Gamma \to \Gamma.A$  of the corresponding display map.

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  $\Gamma \vdash A : \text{Type.}$  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.

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  $\Gamma \vdash A : \text{Type.}$  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 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  $\Gamma \vdash A$ : 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.

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  $\Gamma \vdash A$ : 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.

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  $\Gamma \vdash A : \text{Type.}$  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!

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 axiomatic inference rules.

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 axiomatic 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.

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 axiomatic 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.

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

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

◆□▶ ◆□▶ ◆目▶ ◆目▶ 目 の

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

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \to \Gamma.A \\ \downarrow & & \downarrow & \downarrow \\ \Delta - f \to \Gamma & & \Delta - f \to \Gamma \end{array}$$

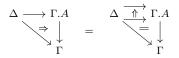
**Display map 2-categories.** (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

(ロ) (部) (注) (注) (注) ()

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

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.

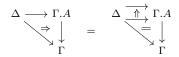


**Display map 2-categories.** (2,1)-dimensional categories with a specified class of 1-morphisms, called **display maps**, that satisfy the following conditions:

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

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.



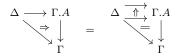
3. Every display map has an **arrow object**.

**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**.

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.



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

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

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ の

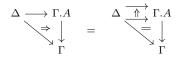
$$\begin{array}{ccc} \Gamma.A.B & \Rightarrow & \Gamma.A.B \simeq \Gamma.C \\ \downarrow & & \downarrow & \downarrow \\ \Gamma.A \longrightarrow \Gamma & & \Gamma.A \longrightarrow \Gamma \end{array}$$

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

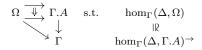
1. To substitute into types and terms.

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.



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



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

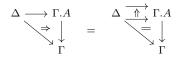
$$\begin{array}{ccc} \Gamma.A.B & \Rightarrow & \Gamma.A.B \simeq \Gamma.C \\ \downarrow & & \downarrow & \downarrow \\ \Gamma.A \longrightarrow \Gamma & & \Gamma.A \longrightarrow \Gamma \end{array}$$

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

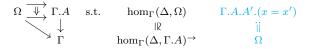
1. To substitute into types and terms.

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.



3. To have identity types with pseudo-elimination.



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

$$\begin{array}{ccc} \Gamma.A.B & \Rightarrow & \Gamma.A.B \simeq \Gamma.C \\ \downarrow & \downarrow & \downarrow \\ \Gamma.A \longrightarrow \Gamma & & \Gamma.A \longrightarrow \Gamma \end{array}$$

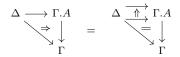
▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ 三臣 の

**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.

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

2. Every display map is a cloven isofibration.



3. To have identity types with pseudo-elimination.

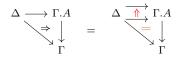
4. To have dependent sum types with *pseudo-elimination*.

**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.

$$\begin{array}{ccc} \Gamma.A & \Rightarrow & \Delta.A[f] \longrightarrow \Gamma.A \\ \downarrow & & \downarrow \\ \Delta - f \Rightarrow \Gamma & & \Delta - f \Rightarrow \Gamma \end{array}$$

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



3. To have identity types with pseudo-elimination.

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

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

◆□▶ ◆□▶ ◆□▶ ◆□▶ □ の

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

**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*  $\Gamma \rightarrow \mathbf{GRPD}$  as display maps over  $\Gamma$ .

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ の

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*  $\Gamma \rightarrow \mathbf{GRPD}$  as display maps over  $\Gamma$ .

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.

< ロ > < 個 > < 差 > < 差 > 差 の

No,

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

Discreteness

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

◆□▶ ◆□▶ ◆□▶ ◆□▶ □ の

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

Discreteness

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

< ロト < 団ト < 団ト < 団ト < 団ト < 団 ・ </li>

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

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

Discreteness

$$\vdash A : \text{Type} \\ \hline x, y : A; \ p, q : x = y; \ \alpha : p = q \vdash p \equiv q \\ \hline \end{array}$$

**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.

◆□▶ ◆□▶ ◆目▶ ◆目▶ 目 の