2-Dimensional Directed Type Theory

Dan Licata
with Robert Harper

Carnegie Mellon University

Martin-Löf Type Theory

A type is specified by

Elements $\Gamma \vdash M : A$ Equality $\Gamma \vdash M \equiv N : A$

Families of types respect equality:

$x{:}A \vdash C : \text{type}$

$M \equiv N : A$

$P : C[M/x]$

$P : C[N/x]$

Martin-Löf Type Theory

A type is specified by

Elements $\Gamma \vdash M : A$ Equality $\Gamma \vdash M \equiv N : A$

Families of types respect equality:

$$\frac{x{:}A \vdash C : \text{type} \qquad M \equiv N : A \qquad P : C[M/x]}{P : C[N/x]}$$

Martin-Löf Type Theory

A type is specified by

Elements $\Gamma \vdash M : A$ Equality $\Gamma \vdash M \equiv N : A$

Families of types respect equality:

$$\frac{x{:}A \vdash C : \text{type} \qquad M \equiv N : A \qquad P : C[M/x]}{P : C[N/x]}$$

universe : elements of type are classifiers

Type Isomorphisms

list A ≅ Σ n:nat. vec A n

Type Isomorphisms

list A ≅ Σ n:nat. vec A n

Monoid : type → type
Monoid X = Σ m : X → X → X.
  Σ u : X.
    (Πx,y,z:X. Id (m x (m y z)) (m (m x y) z))
  * (Πx.Id (m x u) x)
  * (Πx.Id (m u x) x)

Type Isomorphisms

list A ≅ Σ n:nat. vec A n

Monoid : type → type
Monoid X = Σ m : X → X → X.
  Σ u : X.
    (Πx,y,z:X. Id (m x (m y z)) (m (m x y) z))
  * (Πx.Id (m x u) x)
  * (Πx.Id (m u x) x)

Want type families to respect iso, so

Monoid(list A) ≅ Monoid(Σ n:nat. vec A n)

Type Isomorphisms

list A ≅ Σ n:nat. vec A n

Monoid : type → type
Monoid X = Σ m : X → X → X.
  Σ u : X.
    (Πx,y,z:X. Id (m x (m y z)) (m (m x y) z))
  * (Πx.Id (m x u) x)
  * (Πx.Id (m u x) x)

Want type families to respect iso, so

Monoid(list A) ≅ Monoid(Σ n:nat. vec A n)

[cf. Voevodsky's univalence axiom]

Type Isomorphisms

Cannot **equate** isomorphic types:

- Different representations!
- Types can be isomorphic in different ways: isomorphism is structure, not property

Bool ≅ Bool by identity and by not

2-Dimensional Type Theory

Elements $\Gamma \vdash M : A$

Equivalence $\Gamma \vdash \alpha : M \simeq_A N$

+ equality of M's and α's

All families of types respect equivalence, but this has computational content:

$$\frac{x{:}A \vdash C : \mathrm{type} \qquad \alpha : M \simeq_A N \qquad P : C[M/x]}{\mathrm{map}_{x.C}\ \alpha\ P : C[N/x]}$$

a generic proof/program defined for each family C (e.g. Monoid)

2-Dimensional Type Theory

Elements $\Gamma \vdash M : A$

Equivalence $\Gamma \vdash \alpha : M \simeq_A N$

+ equality of M's and α's

All families of types respect equivalence, but this has computational content:

$$\frac{x{:}A \vdash C : \mathrm{type} \qquad \alpha : M \simeq_A N \qquad P : C[M/x]}{\mathrm{map}_{x.C}\ \alpha\ P : C[N/x]}$$

a generic proof/program defined for each family C (e.g. Monoid)

essentially C[α/x]

Outline

1. 2-Dimensional Type Theory
2. 2-Dimensional Directed Type Theory
    a. Application to functorial syntax
    b. Semantics in Cat
3. Higher Dimensions

Outline

1. 2-Dimensional Type Theory

2. 2-Dimensional Directed Type Theory

   a. Application to functorial syntax

   b. Semantics in Cat

3. Higher Dimensions

2-Dimensional Type Theory

Define equivalence for each A

$$\Gamma \vdash \alpha : M \simeq_{\mathbf{A}} N$$

Define map for each C

$$\frac{x{:}A \vdash C : \mathrm{type} \qquad \alpha : M \simeq_A N \qquad P : C[M/x]}{\mathrm{map}_{x.\mathbf{C}}\ \alpha\ P : C[N/x]}$$

Equivalence for types

$$\frac{f : A \to B \quad g : B \to A \quad \alpha : g \circ f \simeq id \quad \beta : f \circ g \simeq id}{iso(f,g,\alpha,\beta) : A \simeq_{\textbf{type}} B}$$

map for type

$$\frac{\mathrm{iso}(f,g,\alpha,\beta) : A \simeq_{\mathrm{type}} B}{\mathrm{map}_{(a:\mathrm{type}.a)}\,(\mathrm{iso}(f,g,\alpha,\beta)) : A \to B}$$

Computation rule: deploy the isomorphism!

$$\mathrm{map}_{(a.a)}\,(\mathrm{iso}(f,g,\alpha,\beta)) \equiv f$$

map for Functions

$\mathsf{map}_{\mathbf{x:A.B \to C}}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$

$\lambda\ x{:}B[M_2].\ \mathsf{map}_{\mathbf{x:A.C}}\ \alpha\ (f\ (\mathsf{map}_{\mathbf{x:A.B}}\ (\mathsf{sym}\ \alpha)\ x))$

map for Functions

$\mathsf{map}_{\mathbf{x:A.B \to C}}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$

$\lambda\ x{:}B[M_2].\ \mathsf{map}_{\mathbf{x:A.C}}\ \alpha\ (f\ (\mathsf{map}_{\mathbf{x:A.B}}\ (\mathsf{sym}\ \alpha)\ x))$

map for Functions

$\mathrm{map}_{\mathbf{x:A.B \to C}}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$

Goal: B[M_2] → C[M_2]

$\lambda\ x{:}B[M_2].\ \mathrm{map}_{\mathbf{x:A.C}}\ \alpha\ (f\ (\mathrm{map}_{\mathbf{x:A.B}}\ (\mathrm{sym}\ \alpha)\ x))$

map for Functions

B[M_1] → C[M_1]

map**x:A.B → C** (α : M_1 ≃A M_2)) f ≡ **Goal: B[M_2] → C[M_2]**

λ x:B[M_2]. map**x:A.C** α (f (map**x:A.B** (sym α) x))

Needs to be contravariant:

B[M_2] → B[M_1]

map for Functions

$B[M_1] \to C[M_1]$

$\text{map}_{\mathbf{x:A.B \to C}}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$ **Goal: $B[M_2] \to C[M_2]$**

$\lambda\ x{:}B[M_2].\ \text{map}_{\mathbf{x:A.C}}\ \alpha\ (f\ (\text{map}_{\mathbf{x:A.B}}\ (\text{sym}\ \alpha)\ x))$

Needs to be covariant:
$C[M_1] \to C[M_2]$

Needs to be contravariant:
$B[M_2] \to B[M_1]$

map for Functions

$B[M_1] \to C[M_1]$

$\mathsf{map}_{\mathbf{x:A.B \to C}}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$

Goal: $B[M_2] \to C[M_2]$

$\lambda\ x{:}B[M_2].\ \mathsf{map}_{\mathbf{x:A.C}}\ \alpha\ (f\ (\mathsf{map}_{\mathbf{x:A.B}}\ (\mathsf{sym}\ \alpha)\ x))$

Needs to be covariant:
$C[M_1] \to C[M_2]$

Needs to be contravariant:
$B[M_2] \to B[M_1]$

Requires $\dfrac{\alpha : M_1 \simeq M_2}{\mathsf{sym}\ \alpha : M_2 \simeq M_1}$

Symmetry for $\simeq_{type}$

$f : A \to B$
$g : B \to A$
$\alpha : g \circ f \simeq id$
$\beta : f \circ g \simeq id$

$iso(f,g,\alpha,\beta) : A \simeq_{type} B$

symmetry requires backwards map!

$$sym\ (iso(f,g,\alpha,\beta)) \equiv iso(g,f,\beta,\alpha)$$

Semantics

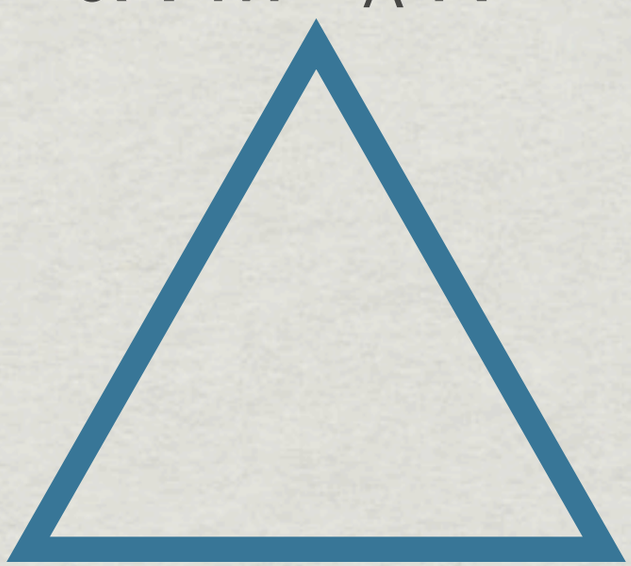
[Hofmann&Streicher,Awodey, Warren,Lumsdaine,Garner, Voevodsky]

type theory

A : type

M, N : A

α : M $\simeq_A$ N



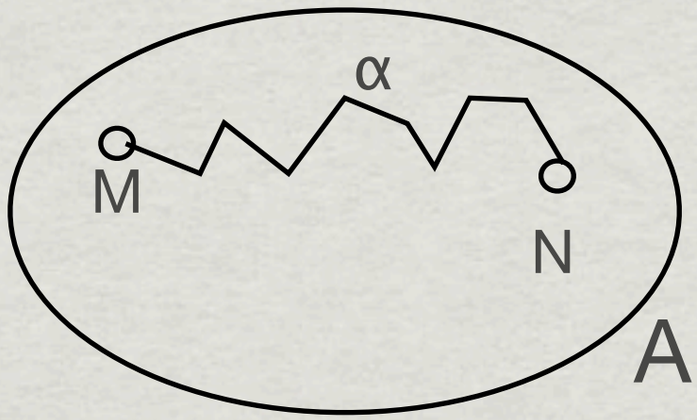
category theory

A is a groupoid

M,N objects

α : M → N in A

homotopy theory



Semantics

[Hofmann&Streicher,Awodey, Warren,Lumsdaine,Garner, Voevodsky]

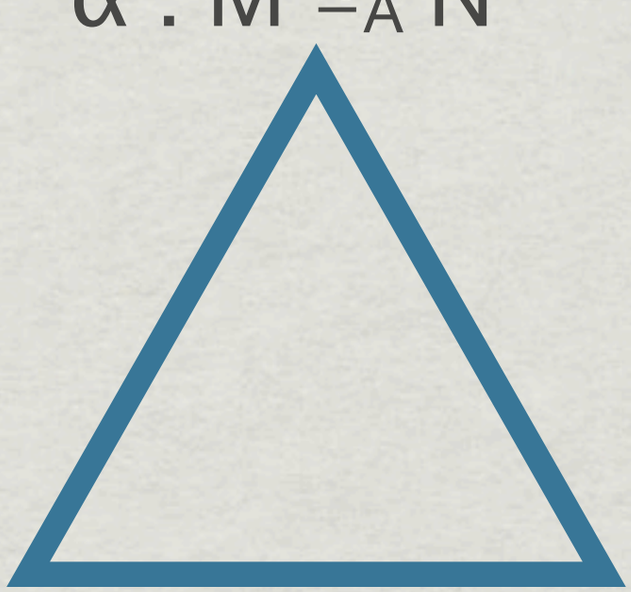
type theory

$$\frac{\alpha : M \simeq_A N}{\mathrm{sym}(\alpha) : N \simeq_A M}$$

$A : \mathrm{type}$

$M, N : A$

$\alpha : M \simeq_A N$



category theory

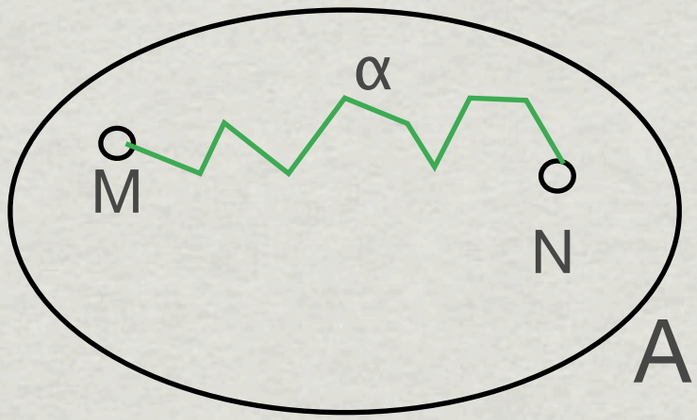
A is a groupoid

M,N objects

$\alpha : M \to N$ in A

$\alpha^{-1} : N \to M$ in A

homotopy theory



Summary

map is a generic program definable for all types:

- above implementation for → extends to Π
- can also be defined for Σ, (co)inductives, …

Let's put it to use!

Functorial Abstract Syntax

[FPT'99,AR'99,H'99]

A family of types Form[Ψ:Ctx] for formulas in Ψ

Extends to a functor:

$$\frac{\Psi' \vdash \sigma : \Psi}{\mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

that implements the structural properties:

Functorial Abstract Syntax

[FPT'99,AR'99,H'99]

A family of types Form[Ψ:Ctx] for formulas in Ψ

Extends to a functor:

$$\frac{\Psi' \vdash \sigma : \Psi}{\mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

that implements the structural properties:

- σ is bijections: exchange

Functorial Abstract Syntax

[FPT'99,AR'99,H'99]

A family of types Form[Ψ:Ctx] for formulas in Ψ

Extends to a functor:

$$\frac{\Psi' \vdash \sigma : \Psi}{\mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

that implements the structural properties:

- σ is bijections: exchange
- σ is var-for-var: weakening, exchange, contraction

Functorial Abstract Syntax

[FPT'99,AR'99,H'99]

A family of types Form[Ψ:Ctx] for formulas in Ψ

Extends to a functor:

$$\frac{\Psi' \vdash \sigma : \Psi}{\mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

that implements the structural properties:

- σ is bijections: exchange
- σ is var-for-var: weakening, exchange, contraction
- σ is term-for-var: substitution, too

Idea

Given a datatype for syntax/judgement:

```
data Form : Ctx → type where
  all : Form[Ψ,i] → Form[Ψ]
  ...

data nd : (Ψ : Ctx) → Form[Ψ] → type
  allR : ∀ {Ψ A} → nd (Ψ,i) A → nd Ψ (all A)
```

automatically implement structural properties using map

Exchange

x,y,z vs. y,z,x

Take Ctx to be a 2D type:

Terms	$\Psi : \mathrm{Ctx}$
Equivalences	$\Psi \simeq_{\mathrm{Ctx}} \Psi'$
	bijection between vars

$$\frac{\alpha : \Psi \simeq_{\mathrm{Ctx}} \Psi'}{\mathrm{map}_{\Psi.\mathrm{Form}[\Psi]}\, \alpha : \mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

Exchange

x,y,z vs. y,z,x

Take Ctx to be a 2D type:

Terms	$\Psi : \mathrm{Ctx}$
Equivalences	$\Psi \simeq_{\mathrm{Ctx}} \Psi'$

for de Bruijn form, not merely a relation

bijection between vars

$$\frac{\alpha : \Psi \simeq_{\mathrm{Ctx}} \Psi'}{\mathrm{map}_{\Psi.\mathrm{Form}[\Psi]}\, \alpha : \mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

Exchange

x,y,z vs. y,z,x

Take Ctx to be a 2D type:

Terms $\Psi : \mathrm{Ctx}$

Equivalences $\Psi \simeq_{\mathrm{Ctx}} \Psi'$

for de Bruijn form, not merely a relation

bijection between vars

$$\frac{\alpha : \Psi \simeq_{\mathrm{Ctx}} \Psi'}{\mathrm{map}_{\Psi.\mathrm{Form}[\Psi]}\ \alpha : \mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

derived from map for definition of Form

Weakening

x,y vs. x,z,y

Take Ctx to be a 2D type:

Terms $\Psi : \mathrm{Ctx}$

Equivalences $\Psi \simeq_{\mathrm{Ctx}} \Psi'$

variable-for-variable substitutions

$x,z,y \vdash x/x,y/y : x,y$

Weakening

x,y vs. x,z,y

Take Ctx to be a 2D type:

Terms $\Psi : \mathrm{Ctx}$

Equivalences $\Psi \simeq_{\mathrm{Ctx}} \Psi'$

variable-for-variable substitutions

$x,z,y \vdash x/x,y/y : x,y$

Problem: not symmetric!

Outline

1. 2-Dimensional Type Theory

2. 2-Dimensional Directed Type Theory

   a. Application to functorial syntax

   b. Semantics in Cat

3. Higher Dimensions

2-Dimensional Type Theory

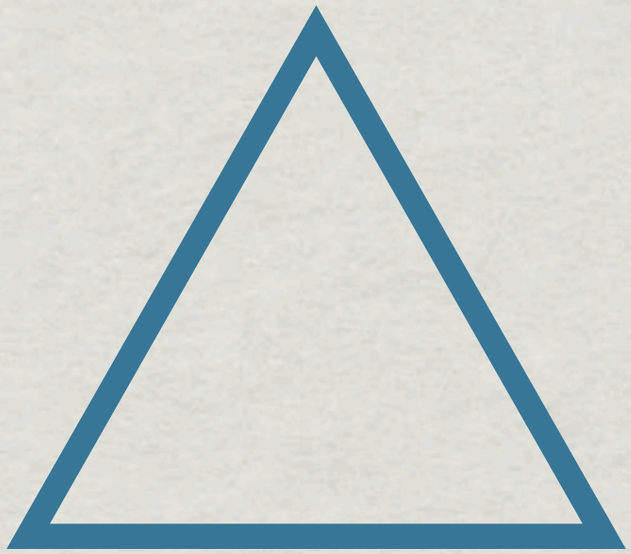
type theory

A : type

M, N : A

α : M ≃$_A$ N

$$\frac{\alpha : M \simeq_A N}{\mathrm{sym}(\alpha) : N \simeq_A M}$$



category theory

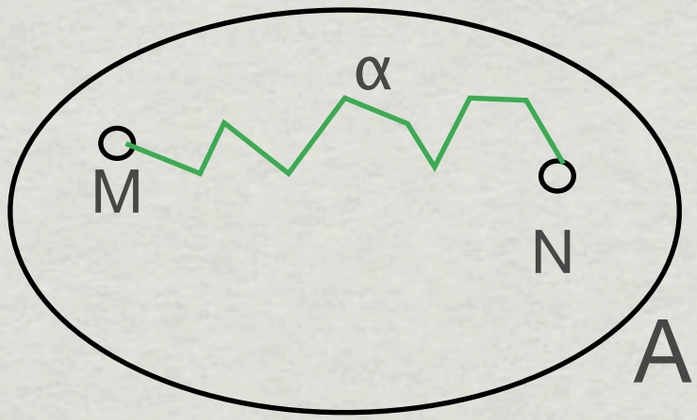
A is a groupoid

M,N objects

α : M → N in A

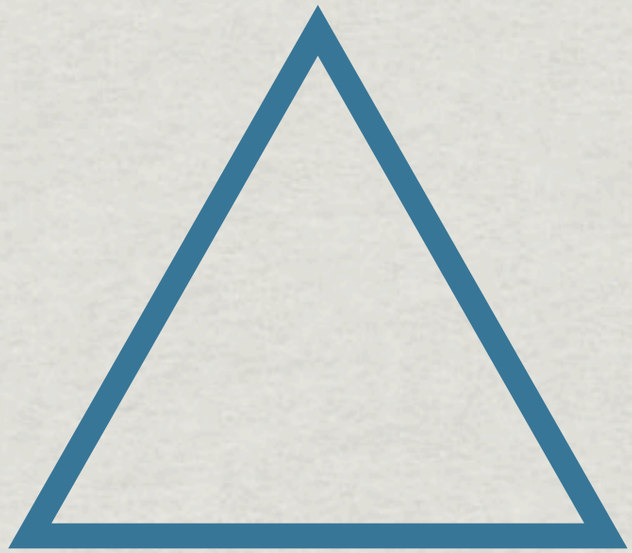
α^{-1} : N → M in A

homotopy theory



2-Dimensional Directed Type Theory

types in directed type theory



categories in category theory

homotopy types in directed homotopy theory

Directed Type Theory

What is a type?

Elements $\Gamma \vdash M : A$

Transformation $\Gamma \vdash \alpha : M \lesssim_A N$

not necessarily symmetric!

Weakening

x,y vs. x,z,y

Take Ctx to be a directed type:

Terms $\Psi : \mathrm{Ctx}$

Transformations $\Psi \lesssim_{\mathrm{Ctx}} \Psi'$

variable for variable substitutions

$\Psi' \vdash \sigma : \Psi$

$$\frac{\alpha : \Psi \lesssim_{\mathrm{Ctx}} \Psi'}{\mathrm{map}_{\Psi.\mathrm{Form}[\Psi]}\, \alpha : \mathrm{Form}[\Psi] \to \mathrm{Form}[\Psi']}$$

What about map for →?

$\mathsf{map}_{x:A.B \to C}\ (\alpha : M_1 \simeq_A M_2))\ f \equiv$

$\lambda\ x{:}B[M_2].\ \mathsf{map}_{x:A.C}\ \alpha\ (f\ (\mathsf{map}_{x:A.B}\ (\mathsf{sym}\ \alpha)\ x))$

Variances

$$\frac{\Gamma \ \mathrm{ctx}}{\Gamma^{\mathrm{op}} \ \mathrm{ctx}}$$

Contravariant **Covariant**

$$\frac{\Gamma^{\mathrm{op}} \vdash A : \mathrm{type} \quad \Gamma \vdash B : \mathrm{type}}{\Gamma \vdash A \to B : \mathrm{type}}$$

Variances

Covariant **Contravariant**

$$\Gamma ::= \cdot \mid \Gamma, x{:}A^{+} \mid \Gamma, x{:}A^{-}$$

$$\frac{}{\Gamma, x{:}A^{+} \vdash x : A}$$

only + assumptions can be used in a term

$$\frac{\Gamma \ \mathsf{ctx}}{\Gamma^{\mathsf{op}} \ \mathsf{ctx}}$$

interchanges + and -

Map

Covariant:

$$\frac{\Gamma, x{:}A^{+} \vdash B \text{ type} \quad \Gamma \vdash \alpha : M_1 \lesssim_A M_2 \quad \Gamma \vdash M : B[M_1/x]}{\Gamma \vdash \mathsf{map}_{x:A^{+}.B}\ \alpha\ M : B[M_2/x]}$$

Contravariant:

$$\frac{\Gamma, x{:}A^{-} \vdash B \text{ type} \quad \Gamma^{\mathsf{op}} \vdash \alpha : M_2 \lesssim_A M_1 \quad \Gamma \vdash M : B[M_1/x]}{\Gamma \vdash \mathsf{map}_{x:A^{-}.B}\ \alpha\ M : B[M_2/x]}$$

Functions

$$\frac{\Gamma^{op} \vdash A : \mathsf{type} \quad \Gamma \vdash B : \mathsf{type}}{\Gamma \vdash A \to B : \mathsf{type}}$$

$$\frac{\Gamma, x{:}A^{-} \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma^{op} \vdash N : A}{\Gamma \vdash M\ N : B}$$

map (→)

$$\frac{x{:}A^{-} \vdash B : \text{type} \quad x{:}A^{+} \vdash C : \text{type}}{x{:}A^{+} \vdash B \to C : \text{type}}$$

$B[M_1] \to C[M_1]$

$$\text{map}_{x:A+.B \to C}\ (\alpha : M_1 \lesssim_A M_2)\ f \equiv$$
$$\lambda\ x{:}B[M_2]^{-}.\ \text{map}_{x:A+.C}\ \alpha\ (f\ (\text{map}_{x:A-.B}\ \alpha\ x))$$

Goal: $B[M_2] \to C[M_2]$

Covariant: $C[M_1] \to C[M_2]$

Contravariant: $B[M_2] \to B[M_1]$

map (→)

$$\frac{x{:}A^{-} \vdash B : \mathsf{type} \quad x{:}A^{+} \vdash C : \mathsf{type}}{x{:}A^{+} \vdash B \to C : \mathsf{type}}$$

$B[M_1] \to C[M_1]$

$$\mathsf{map}_{x:A+.B \to C}\ (\alpha : M_1 \lesssim_A M_2)\ f \equiv$$
$$\lambda\ x{:}B[M_2]^{-}.\ \mathsf{map}_{x:A+.C}\ \alpha\ (f\ (\mathsf{map}_{x:A-.B}\ \alpha\ x))$$

Goal: $B[M_2] \to C[M_2]$

Covariant: $C[M_1] \to C[M_2]$

Contravariant: $B[M_2] \to B[M_1]$

$$\Gamma,\quad x{:}B[M_2]^{-} \vdash f\ (\mathsf{map}_{x:A-.B}\ \alpha\ x) : C[M_1]$$
$$\text{if } \Gamma^{op},\ x{:}B[M_2]^{+} \vdash (\mathsf{map}_{x:A-.B}\ \alpha\ x) : B[M_1]$$
$$\text{if } (\Gamma^{op})^{op} \vdash \alpha : M_1 \lesssim_A M_2$$

map (→)

$$\frac{x{:}A^{-} \vdash B : \mathsf{type} \quad x{:}A^{+} \vdash C : \mathsf{type}}{x{:}A^{+} \vdash B \to C : \mathsf{type}}$$

$B[M_1] \to C[M_1]$

$$\mathsf{map}_{x:A+.B \to C}\ (\alpha : M_1 \lesssim_A M_2)\ f \equiv$$
$$\lambda\ x{:}B[M_2]^{-}.\ \mathsf{map}_{x:A+.C}\ \alpha\ (f\ (\mathsf{map}_{x:A-.B}\ \alpha\ x))$$

Goal: $B[M_2] \to C[M_2]$

Covariant: $C[M_1] \to C[M_2]$

Contravariant: $B[M_2] \to B[M_1]$

$$\Gamma,\quad x{:}B[M_2]^{-} \vdash f\ (\mathsf{map}_{x:A-.B}\ \alpha\ x) : C[M_1]$$
$$\text{if } \Gamma^{\mathrm{op}},\ x{:}B[M_2]^{+} \vdash (\mathsf{map}_{x:A-.B}\ \alpha\ x) : B[M_1]$$
$$\text{if } (\Gamma^{\mathrm{op}})^{\mathrm{op}} \vdash \alpha : M_1 \lesssim_A M_2$$

involution

Semantics in Cat

Context	Γ ctx	category
Substitution	$\Gamma \vdash \theta : \Delta$	functor
Transformation	$\Gamma \vdash \delta : \theta \lesssim_\Delta \theta'$	natural transform.
Type	$\Gamma \vdash A$ type	functor $\Gamma \to$ Cat
Term	$\Gamma \vdash M : A$	“dependent functor”
Term Trans.	$\Gamma \vdash \alpha : M \lesssim_A N$	“dependent n.t.”

Identity and Composition

Transformation: $\Gamma \vdash \delta : \theta \lesssim_\Delta \theta'$

$$\frac{}{\Gamma \vdash \mathrm{refl} : \theta \lesssim_\Delta \theta} \qquad \frac{\Gamma \vdash \delta_1 : \theta_1 \lesssim_\Delta \theta_2 \quad \Gamma \vdash \delta_2 : \theta_2 \lesssim_\Delta \theta_3}{\Gamma \vdash \delta_2 \circ \delta_1 : \theta_1 \lesssim_\Delta \theta_3}$$

also horizontal composition: transformation respects transformation

Identity and Composition

Transformation: $\Gamma \vdash \delta : \theta \lesssim_\Delta \theta'$

$$\frac{}{\Gamma \vdash \mathrm{refl} : \theta \lesssim_\Delta \theta} \qquad \frac{\Gamma \vdash \delta_1 : \theta_1 \lesssim_\Delta \theta_2 \quad \Gamma \vdash \delta_2 : \theta_2 \lesssim_\Delta \theta_3}{\Gamma \vdash \delta_2 \circ \delta_1 : \theta_1 \lesssim_\Delta \theta_3}$$

also horizontal composition: transformation respects transformation

Equations hold strictly:

$$\delta_3 \circ (\delta_2 \circ \delta_1) \equiv (\delta_3 \circ \delta_2) \circ \delta_1$$

$$\mathrm{map}_C\ (\delta_2 \circ \delta_1)\ M \equiv \mathrm{map}_C\ \delta_3\ (\mathrm{map}_C\ \delta_2\ M)$$

Application

Generalizes functorial syntax to

- judgements/dependent types, via **map** for Σ and Π
- admissibility premises

(see paper/thesis for details)

Outline

1. 2-Dimensional Type Theory
2. 2-Dimensional Directed Type Theory
    a. Application to functorial syntax
    b. Semantics in Cat
3. **Higher Dimensions**

Higher Dimensions

More generally, equations may hold weakly:

$$\delta_3 \circ (\delta_2 \circ \delta_1) \simeq (\delta_3 \circ \delta_2) \circ \delta_1 : \theta_1 \lesssim \theta_4 : \Gamma \vdash \Delta$$

Example: $\mathbf{type}_1$ where $\mathbf{type} : \mathbf{type}_1$
equivalence should be “iso up to iso”

Higher Dimensions

More generally, equations may hold weakly:

$$\delta_3 \circ (\delta_2 \circ \delta_1) \simeq (\delta_3 \circ \delta_2) \circ \delta_1 : \theta_1 \lesssim \theta_4 : \Gamma \vdash \Delta$$

Example: **type**$_1$ where **type** : **type**$_1$
equivalence should be “iso up to iso”

Elements	$\Gamma \vdash M : A$
Equivalence	$\Gamma \vdash \alpha : M \simeq N$
Equivalence^2	$\Gamma \vdash \varphi : \alpha \simeq \alpha'$
...	

Higher Dimensions

In symmetric type theory, can exploit the identity type

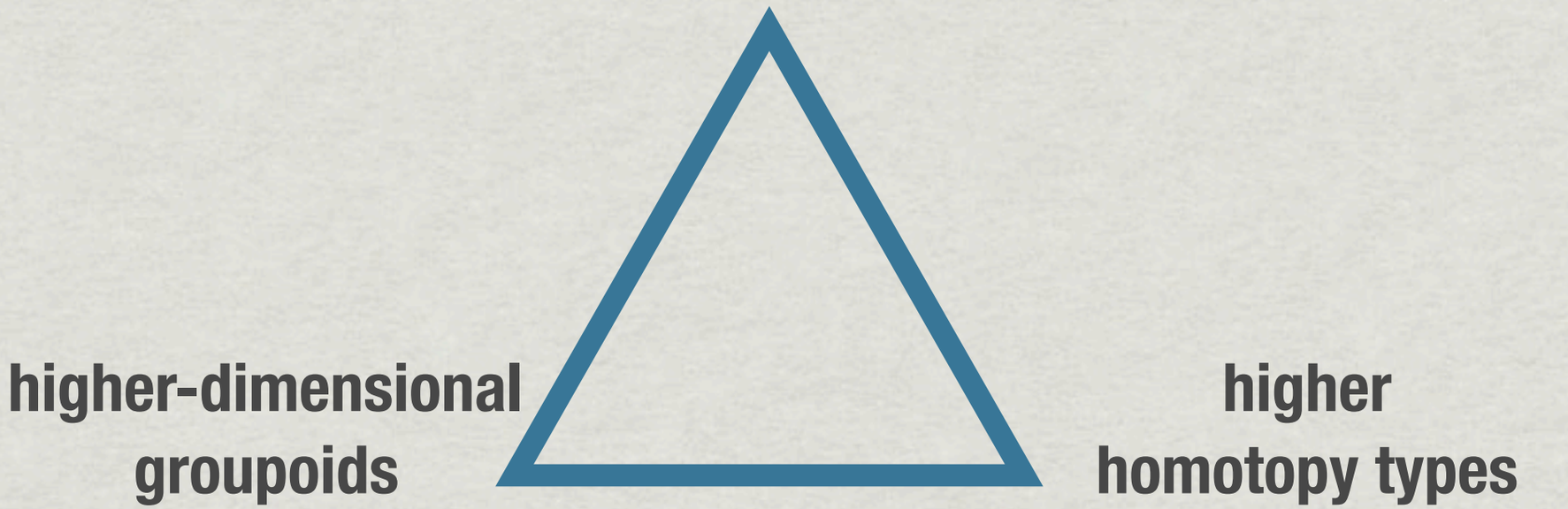
Elements	$M : A$	
Equivalence	$\alpha : M \simeq N$	$\alpha : Id_A(M,N)$
Equivalence^2	$\varphi : \alpha \simeq \alpha'$	$\varphi : Id_{Id(M,N)}(\alpha,\alpha')$

Interpret Id_A as hom-functor A x A → [type]

Standard rules for Id are sound for higher-dimensions, but miss computation rules for **map**

Homotopy Type Theory

[homotopytypetheory.org]



[Hofmann&Streicher,Awodey,
Warren,Lumsdaine,Garner,Voevodsky]

Higher Dimensions

In symmetric type theory, can exploit the identity type

Elements	$M : A$	
Equivalence	$\alpha : M \simeq N$	$\alpha : \mathrm{Id}_A(M,N)$
Equivalence^2	$\varphi : \alpha \simeq \alpha'$	$\varphi : \mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha,\alpha')$

But in directed type theory, a Hom type is harder:

Mixed variance of Hom : $C^{op} \times C \to$ [type]

Outline

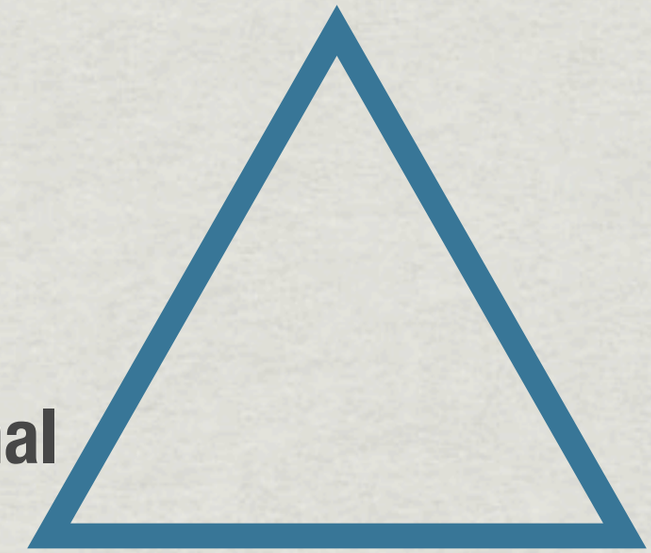
1. 2-Dimensional Type Theory
2. 2-Dimensional Directed Type Theory
    a. Application to functorial syntax
    b. Semantics in Cat
3. Higher Dimensions

Future Work

- Hom type, paralleling identity type Id
- Inductive, co-inductive types
- Quotients by internal descriptions of categories
- Integration with symmetric type theory
- Semantics in more general 2-categories
- Higher dimensions and their semantics
- Computational interpretation:
  N-canonicity, decidable definitional equality

Directed Type Theory

types in
directed type theory



higher-dimensional
categories
in category theory

higher
homotopy types
in directed homotopy theory

Blog: homotopytypetheory.org