Higher Structures in Homotopy Type Theory

Antoine Allioux
Université Paris Cité
Institut de Recherche en Informatique Fondamentale (IRIF)

Séminaire Logique et Interactions
Institut de Mathématiques de Marseille

9 November 2023

Joint work with Eric Finster

Plan

1. Algebra in HoTT
2. A universe of polynomial monads
3. Opetopic types and their applications

Sets

Mathematics are classically based on sets: collections of discrete elements.

Sets

Mathematics are classically based on sets: collections of discrete elements.

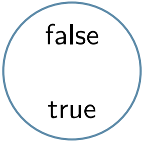


Figure: the set of booleans

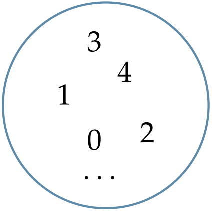


Figure: the set of natural numbers

Sets

Mathematics are classically based on sets: collections of discrete elements.

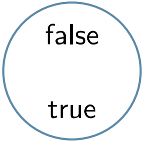


Figure: the set of booleans

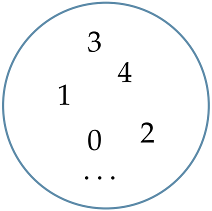


Figure: the set of natural numbers

Two elements of a set are *equal* if they have the same *definition*.

Principle of equivalence

The principle of equivalence states that mathematical reasoning should be invariant under the proper notion of equivalence.

Principle of equivalence

The principle of equivalence states that mathematical reasoning should be invariant under the proper notion of equivalence.

In a foundation respecting this principle, two mathematical objects should be equal if they have the same *properties*.

Principle of equivalence

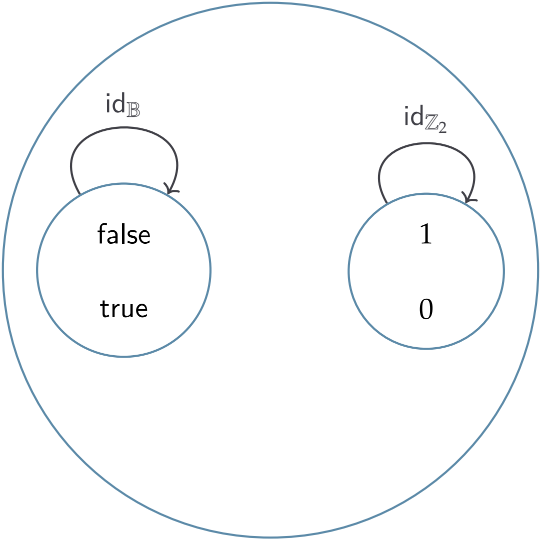
The principle of equivalence states that mathematical reasoning should be invariant under the proper notion of equivalence.

In a foundation respecting this principle, two mathematical objects should be equal if they have the same *properties*.

Set theory does not respect this principle (e.g., two bijective sets are not necessarily equal).

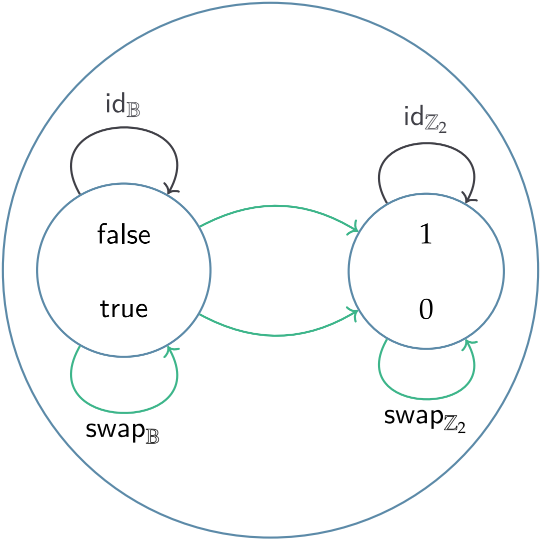
Principle of equivalence

Equality in sets does not account for all equivalences.



Principle of equivalence

Equality in sets does not account for all equivalences.



Homotopy type theory

Homotopy type theory is a foundation for constructive mathematics in which this principle holds.

Homotopy type theory

Homotopy type theory is a foundation for constructive mathematics in which this principle holds.

In type theory, logical propositions are defined as the types of their proofs (BHK interpretation of intuitionistic logic).

Homotopy type theory

Homotopy type theory is a foundation for constructive mathematics in which this principle holds.

In type theory, logical propositions are defined as the types of their proofs (BHK interpretation of intuitionistic logic).

Type theory is a language rich enough to unify mathematical constructions and logical propositions.

Proposition-as-types paradigm

The correspondence goes as follows.

Logic	Type theory
$\bot$	$\mathbf{0}$
$A \wedge B$	$A \times B$
$A \vee B$	$A + B$
$A \implies B$	$A \to B$
$\exists (x \in A).B(x)$	$(x : A) \times B(x)$
$\forall (x \in A).B(x)$	$(x : A) \to B(x)$

Identity types

Given two elements $x, y : A$, their identity type is $x =_A y$.

Identity types

Given two elements $x, y : A$, their identity type is $x =_A y$.

For any identity $p : x =_A y$ and any type family $B : A \to \mathcal{U}$, there is a transport function between fibres

$$p_* : B(x) \to B(y)$$

Identity types

Given two elements $x, y : A$, their identity type is $x =_A y$.

For any identity $p : x =_A y$ and any type family $B : A \to \mathcal{U}$, there is a transport function between fibres

$$p_* : B(x) \to B(y)$$

Formal version of Leibniz's identity of indiscernibles.

Univalence

In HoTT, the equality between two types X and Y is equivalent to the type of equivalences between these two types:

$$(X =_{\mathcal{U}} Y) \simeq (X \simeq Y)$$

Univalence

In HoTT, the equality between two types X and Y is equivalent to the type of equivalences between these two types:

$$(X =_{\mathcal{U}} Y) \simeq (X \simeq Y)$$

Not only are equivalent types identified, but the different ways they are identified are recorded.

Univalence

In HoTT, the equality between two types X and Y is equivalent to the type of equivalences between these two types:

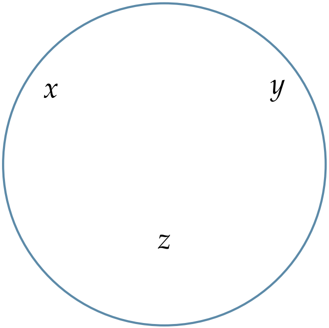
$$(X =_{\mathcal{U}} Y) \simeq (X \simeq Y)$$

Not only are equivalent types identified, but the different ways they are identified are recorded.

In general, no uniqueness of identity proofs (UIP).

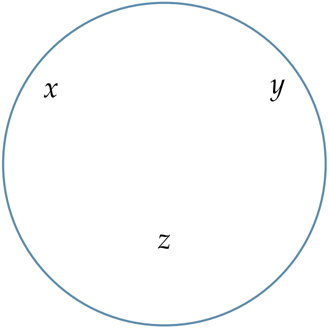
Types

Types contain elements—like sets.

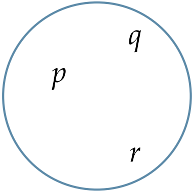


Types

Types contain elements—like sets.

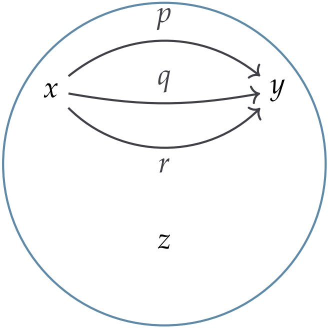


The proposition $x = y$ is a type.

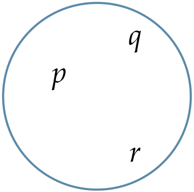


Types

Types contain elements—like sets.

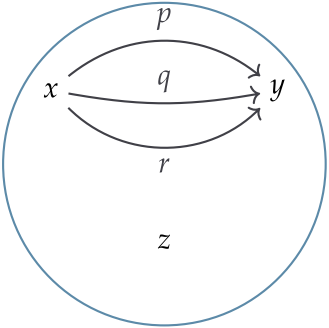


The proposition $x = y$ is a type.

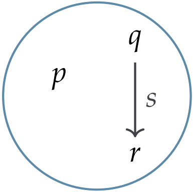


Types

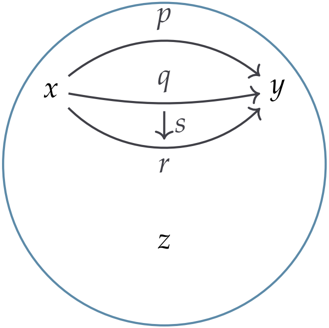
Types contain elements—like sets.



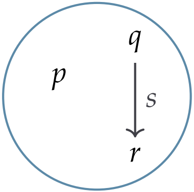
The proposition $x = y$ is a type.



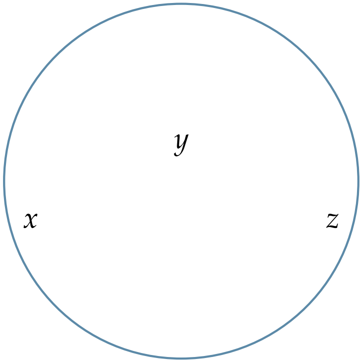
Types contain elements—like sets.



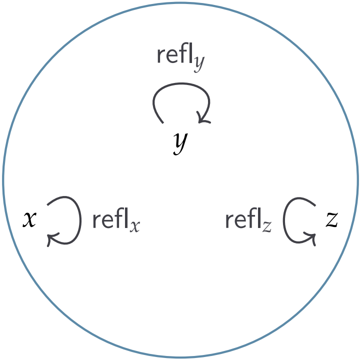
The proposition $x = y$ is a type.



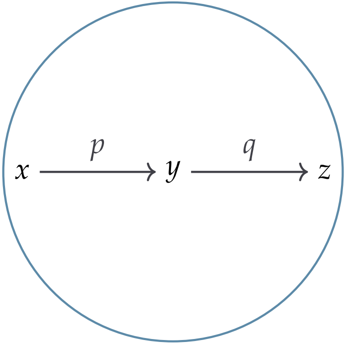
Any element comes with a distinguished loop called *reflexivity*.



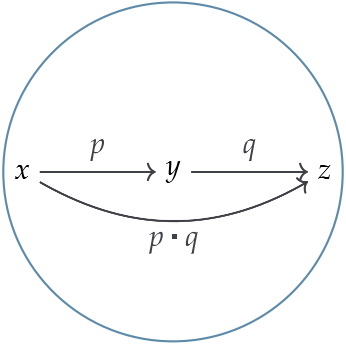
Any element comes with a distinguished loop called *reflexivity*.



Identities can be composed (transitivity of equality).

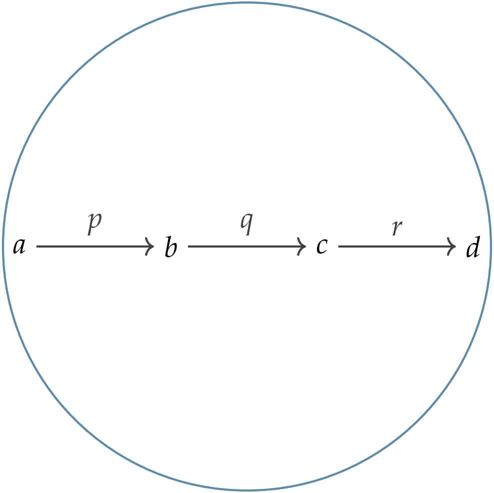


Identities can be composed (transitivity of equality).



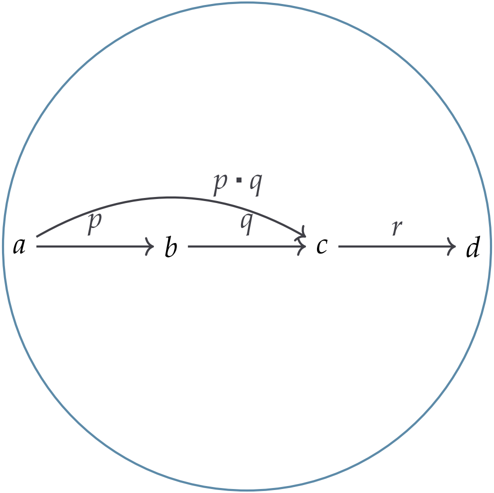
Algebra of paths

Composition of identities is associative up to a *higher identity*.

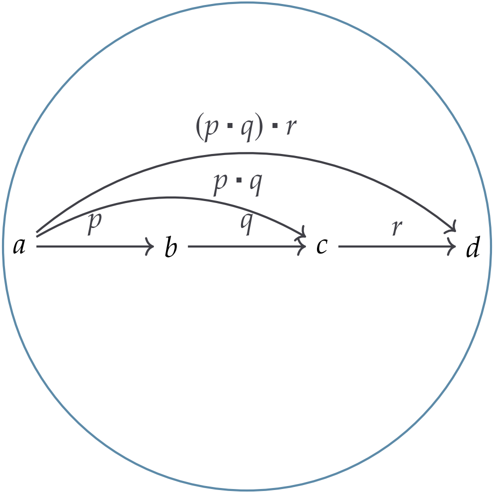


Algebra of paths

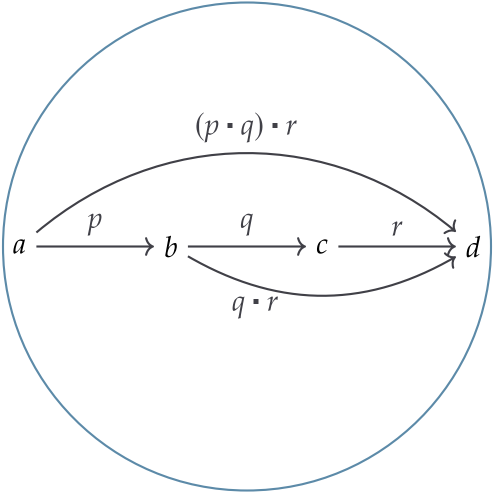
Composition of identities is associative up to a *higher identity*.



Composition of identities is associative up to a *higher identity*.

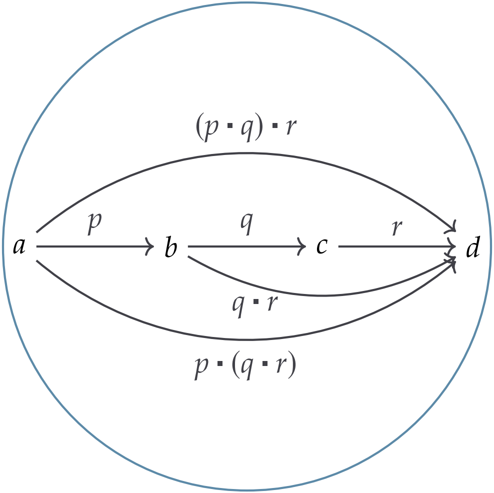


Composition of identities is associative up to a *higher identity*.

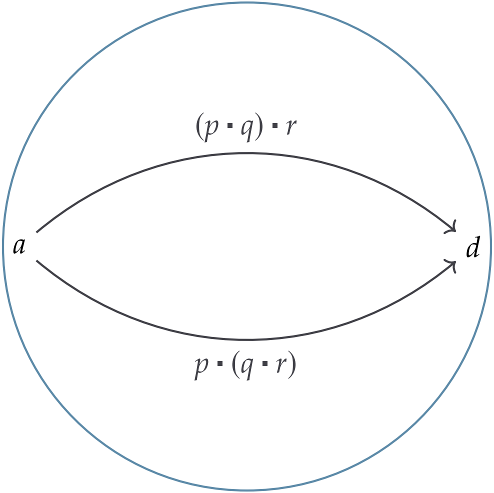


Algebra of paths

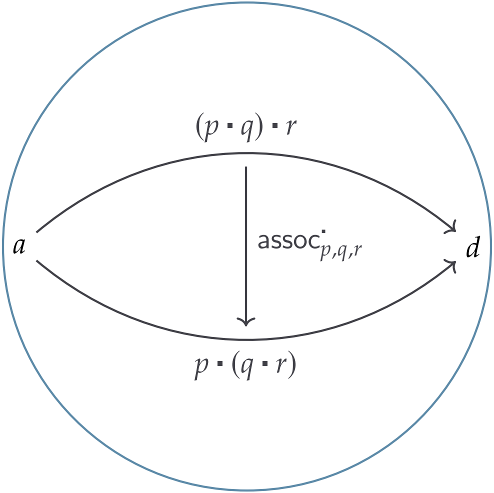
Composition of identities is associative up to a *higher identity*.



Composition of identities is associative up to a *higher identity*.

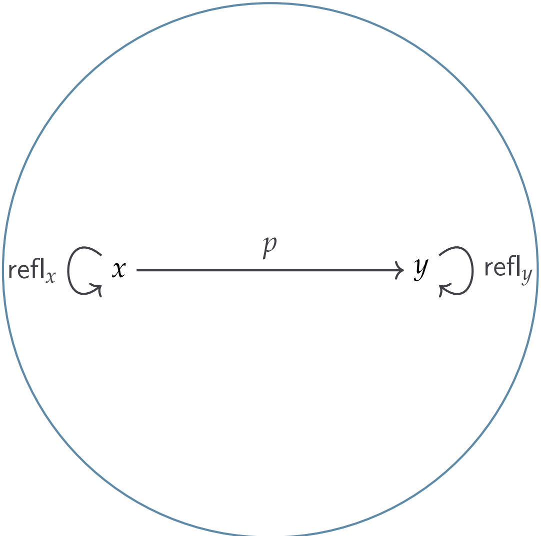


Composition of identities is associative up to a *higher identity*.



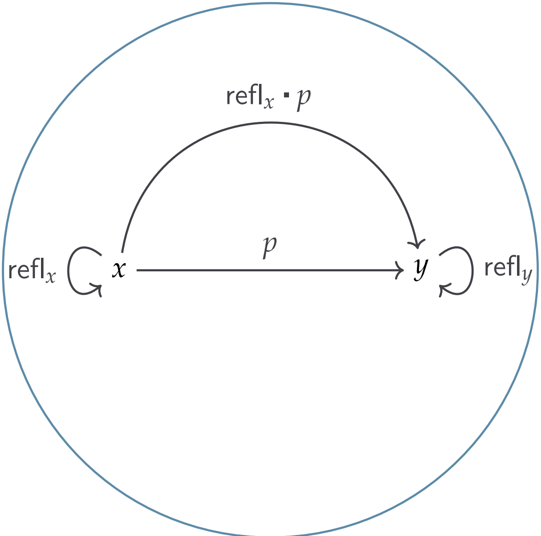
Algebra of paths

Composition of identities is left and right unital up to a *higher identity*.



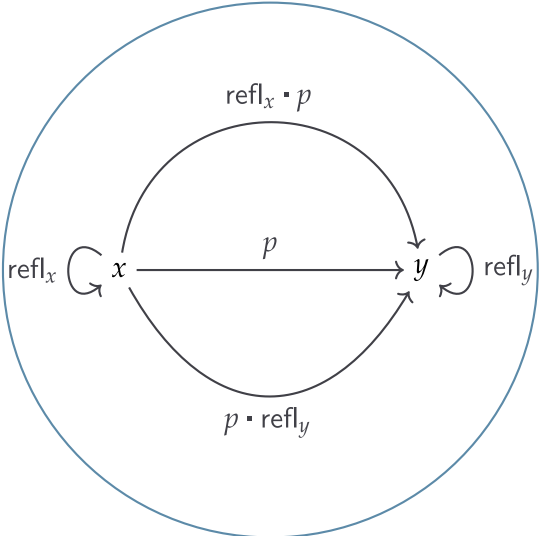
Algebra of paths

Composition of identities is left and right unital up to a *higher identity*.



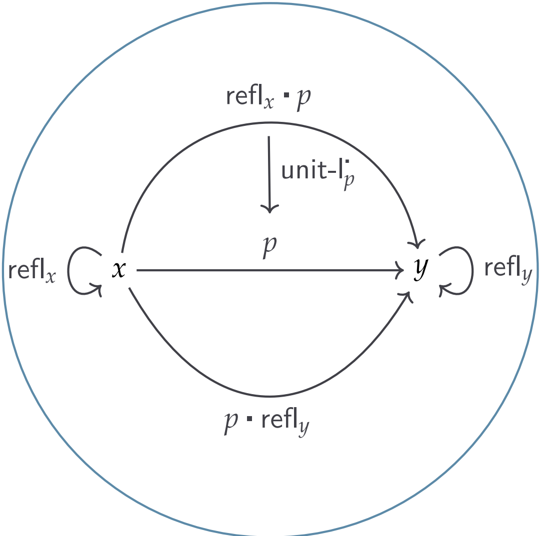
Algebra of paths

Composition of identities is left and right unital up to a *higher identity*.

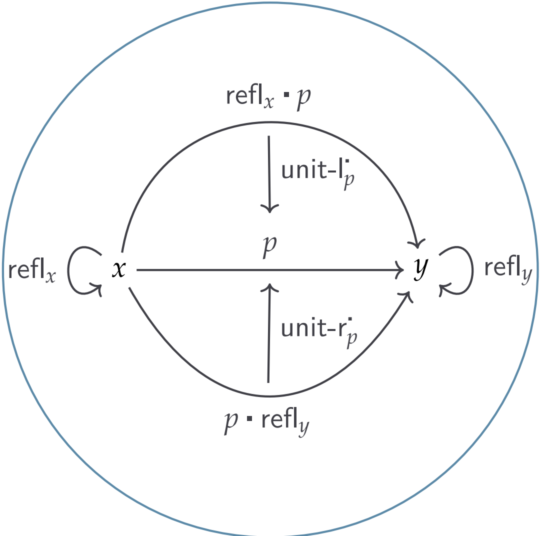


Algebra of paths

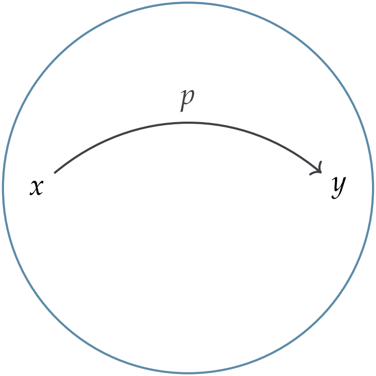
Composition of identities is left and right unital up to a *higher identity*.



Composition of identities is left and right unital up to a *higher identity*.

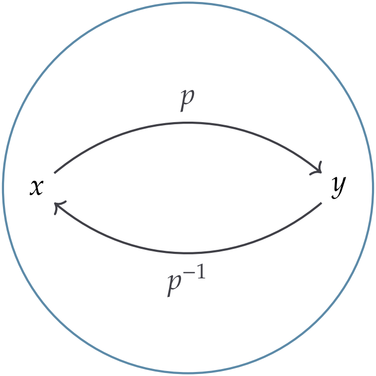


Identities can be inverted.

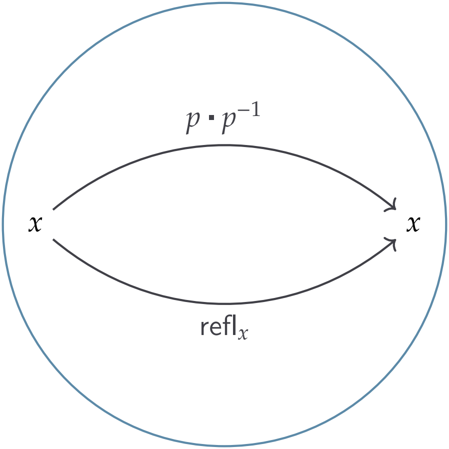


Algebra of paths

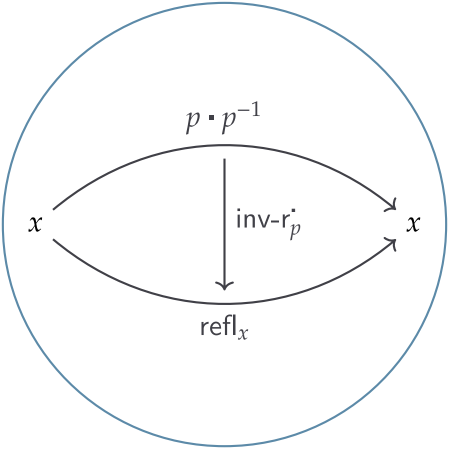
Identities can be inverted.



Identities can be inverted.

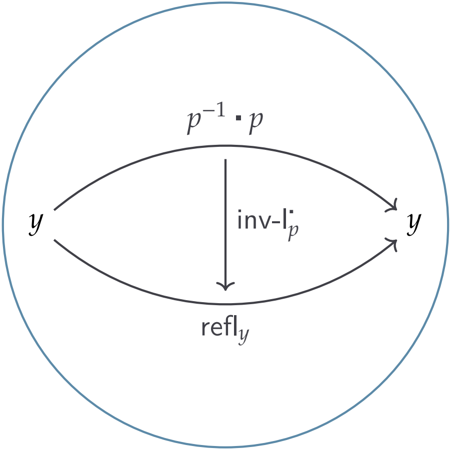


Identities can be inverted.



Algebra of paths

Similarly, there is an identity



In addition, the identities witnessing the laws satisfy *coherence* conditions. More on that later.

Algebra of paths

In addition, the identities witnessing the laws satisfy *coherence* conditions. More on that later.

This structure we just described is the one of ∞-groupoid.

Algebra on types

We want to generalise usual algebraic structures (groups, monoids, rings, …) to types.

Algebra on types

We want to generalise usual algebraic structures (groups, monoids, rings, …) to types.

An algebraic structure on a type is an operation acting on the elements of this type satisfying *coherent* laws expressed in terms of identities.

Algebra on a type

Example

An associative magma on a type X is the data of a binary operation

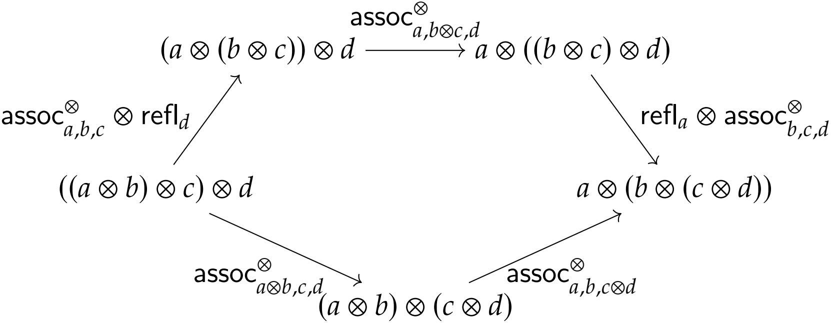
$$\_ \otimes \_ : X \times X \to X$$

along with an identity

$$(a \otimes b) \otimes c \xrightarrow{\mathsf{assoc}^{\otimes}_{a,b,c}} a \otimes (b \otimes c)$$

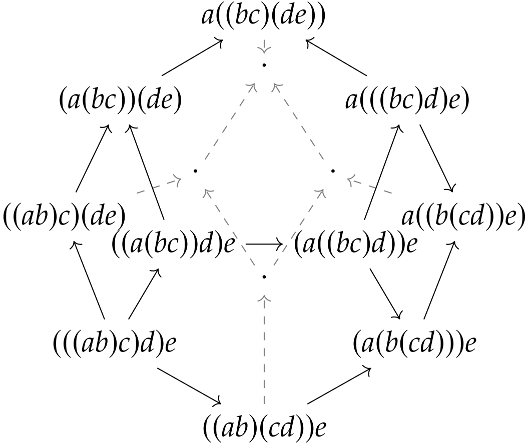
for any $a, b, c : X$ witnessing that the multiplication is associative.

In addition, we require this identity to be such that the following diagram commutes up to a higher identity:



Coherence

In turn, this new data has to satisfy its own coherence conditions leading to an infinite tower of data described by Stasheff's associahedra K_n.



Coherence

Without these coherence laws, some expected mathematical results do not hold.

Coherence

Without these coherence laws, some expected mathematical results do not hold.

For instance, we cannot define the slice category of a category if its laws are not coherent.

Coherence

Without these coherence laws, some expected mathematical results do not hold.

For instance, we cannot define the slice category of a category if its laws are not coherent.

Sets are degenerate types whose only identities are reflexivities. Laws of algebraic structures on sets are therefore trivially coherent.

In classical mathematics, spaces are *encoded* in terms of sets.

Algebra in type theory

In classical mathematics, spaces are *encoded* in terms of sets.

Algebras on spaces are then presented using algebraic structures on sets (operads, presheaves, …).

Algebra in type theory

In classical mathematics, spaces are *encoded* in terms of sets.

Algebras on spaces are then presented using algebraic structures on sets (operads, presheaves, …).

Spaces are primitive in type theory, any algebraic structure must be stated coherently in the first place.

A theory of structures

By embracing the principle of equivalence in type theory, we lost the ability to do algebra on types.

A theory of structures

By embracing the principle of equivalence in type theory, we lost the ability to do algebra on types.

Type theory seems to be missing a theory of structures.

Proposal

Extension of type theory with a universe of cartesian polynomial monads.

Proposal

Extension of type theory with a universe of cartesian polynomial monads.

Presentation of types and their higher structures as *opetopic types*.

Proposal

Extension of type theory with a universe of cartesian polynomial monads.

Presentation of types and their higher structures as *opetopic types*.

Allows the definition of higher algebraic structures on arbitrary types (∞-groupoids, $(\infty, 1)$-categories).

Proposal

Extension of type theory with a universe of cartesian polynomial monads.

Presentation of types and their higher structures as *opetopic types*.

Allows the definition of higher algebraic structures on arbitrary types (∞-groupoids, $(\infty, 1)$-categories).

This approach is compatible with univalence.

Applications

In this type theory, the following results have been established:

- Fibrant opetopic types are equivalent to Baez-Dolan coherent algebras whose morphisms are invertible.
- The internal ∞-groupoid associated to a type.
- The $(\infty, 1)$-category of types.
- Adjunctions between $(\infty, 1)$-categories.
- Fibrant opetopic types are closed under dependent sums.

Setting

The base type theory is book HoTT with Agda's features (coinductive records, inductive-recursive types, …).

Setting

The base type theory is book HoTT with Agda's features (coinductive records, inductive-recursive types, ...).

Most of this work has been formalised in Agda using postulates and rewrite rules to define the universe of polynomial monads.

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

- $\mathsf{Idx}_M : \mathcal{U}$

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$

Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$
- $\mathsf{Typ}_M : \{i : \mathsf{Idx}_M\}\ \{c : \mathsf{Cns}_M(i)\} \to \mathsf{Pos}_M(c) \to \mathsf{Idx}_M$

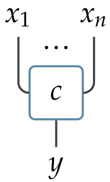
Polynomial monads

Type theory is extended with a universe of cartesian polynomial monads $\mathcal{M} : \mathcal{U}$.

The base data of a monad $M : \mathcal{M}$ is defined by the following data:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$
- $\mathsf{Typ}_M : \{i : \mathsf{Idx}_M\}\ \{c : \mathsf{Cns}_M(i)\} \to \mathsf{Pos}_M(c) \to \mathsf{Idx}_M$

Constructors depicted as corollas:



Polynomial monads

The structure of cartesian polynomial monad is defined by a unit operation η and a multiplication operation μ:

$$\eta_M : (i : \mathsf{Idx}_M) \to \mathsf{Cns}_M(i)$$

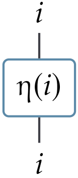
$$\mu_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i)) \to \overrightarrow{\mathsf{Cns}_M}(c) \to \mathsf{Cns}_M(i)$$

Polynomial monads

The unit

$$\eta_M : (i : \mathsf{Idx}_M) \to \mathsf{Cns}_M(i)$$

Units $\eta(i)$ are *unary* constructors whose source and target have the same sort:

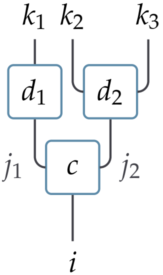


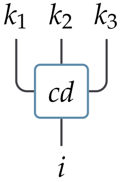
Polynomial monads

The multiplication

$$\mu_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i))\ (d : \overrightarrow{\mathsf{Cns}_M}(c)) \longrightarrow \mathsf{Cns}_M(i)$$

The multiplication "contracts" a tree of constructors while preserving the type of positions and their typing.





The operation μ_M is associative and unital with units η_M:

$$\mu_M(c, \lambda\, p \to \eta_M(\mathsf{Typ}_M(c, p))) \equiv c$$

$$\mu_M(\eta_M(i), d) \equiv d(\eta\text{-}\mathsf{pos}\,(i))$$

$$\mu_M(\mu_M(c, d), e) \equiv \mu_M(c, (\lambda\, p \to \mu_M(d(p), (\lambda\, q \to e(\mathsf{pair}^{\mu}(p, q))))))$$

Identity monad

The identity monad $\mathsf{Id} : \mathcal{M}$ has a single unary constructor.

Its monad structure is trivial.

Baez-Dolan slice construction

The universe $\mathcal{M}$ is closed under the Baez-Dolan slice construction. For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$ with

$$\mathsf{Fam}_M :\equiv \mathsf{Idx}_M \to \mathcal{U}$$

there is a monad $\boxed{M/X : \mathcal{M}}$.

Baez-Dolan slice construction

The universe $\mathcal{M}$ is closed under the Baez-Dolan slice construction. For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$ with

$$\mathsf{Fam}_M :\equiv \mathsf{Idx}_M \to \mathcal{U}$$

there is a monad $\boxed{M/X : \mathcal{M}}$.

If X is a carrier of a M-algebra, the monad M/X captures the different ways to compose a particular configuration of sources.

Baez-Dolan slice construction

The universe $\mathcal{M}$ is closed under the Baez-Dolan slice construction. For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$ with

$$\mathsf{Fam}_M :\equiv \mathsf{Idx}_M \to \mathcal{U}$$

there is a monad $\boxed{M/X : \mathcal{M}}$.

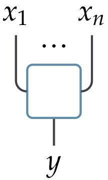
If X is a carrier of a M-algebra, the monad M/X captures the different ways to compose a particular configuration of sources.

Iterating this construction, we capture the combinatorics of

- the composition of X-cells
- its laws
- the coherences satisfied by the laws
- …

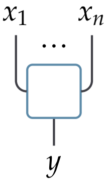
Baez-Dolan slice construction

The indices of M/X are *frames*: constructors of M decorated with elements in X:



Baez-Dolan slice construction

The indices of M/X are *frames*: constructors of M decorated with elements in X:

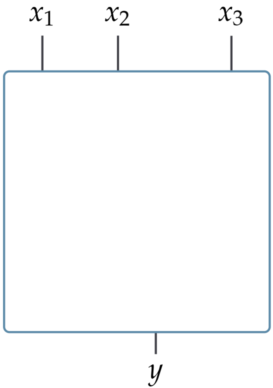


Defined as quadruplets $(i, y) \triangleleft (c, x)$ of type

$$\mathsf{Idx}_{M/X} :\equiv (i : \mathsf{Idx}_M) \times (y : X(i)) \times (c : \mathsf{Cns}_M(i)) \times (x : \overrightarrow{X}(c))$$

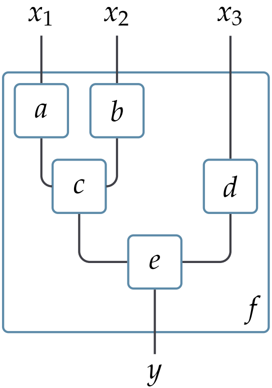
Baez-Dolan slice construction

Constructors of M/X are well-founded trees of frames which multiply to their indexing frame.



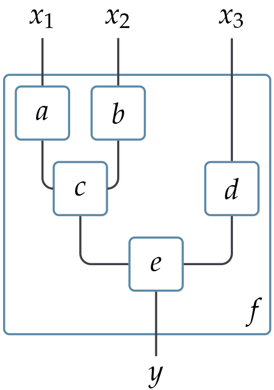
Baez-Dolan slice construction

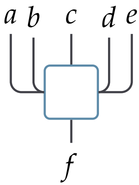
Constructors of M/X are well-founded trees of frames which multiply to their indexing frame.



Baez-Dolan slice construction

Constructors of M/X are well-founded trees of frames which multiply to their indexing frame.





Baez-Dolan slice construction

Trees are defined as an inductive type.

Baez-Dolan slice construction

Trees are defined as an inductive type.

Provide a notion of pasting diagram.

Baez-Dolan slice construction

Trees are defined as an inductive type.

Provide a notion of pasting diagram.

A lot of proofs in this framework go by induction on pasting diagrams.

Baez-Dolan slice construction

Trees are defined as an inductive type.

Provide a notion of pasting diagram.

A lot of proofs in this framework go by induction on pasting diagrams.

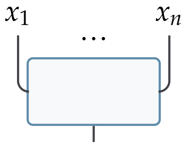
Particularly suited to type theory.

0-ALGEBRAS

A 0-algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$, there exists is a unique pair composed of



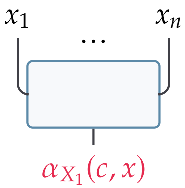
0-ALGEBRAS

A 0-algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$, there exists is a unique pair composed of

- a composite $\alpha_{X_1}(c, x) : X_0(i)$,



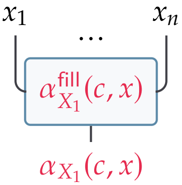
0-ALGEBRAS

A 0-algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$, there exists is a unique pair composed of

- a composite $\alpha_{X_1}(c, x) : X_0(i)$,
- a filler $\alpha^{\mathsf{fill}}_{X_1}(c, x) : X_1((i, \alpha_{X_1}(c, x)) \triangleleft (c, x))$.



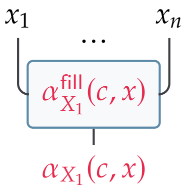
0-ALGEBRAS

A 0-algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$, there exists is a unique pair composed of

- a composite $\alpha_{X_1}(c, x) : X_0(i)$,
- a filler $\alpha^{\mathsf{fill}}_{X_1}(c, x) : X_1((i, \alpha_{X_1}(c, x)) \triangleleft (c, x))$.



X_1 is an *entire* and *functional* relation.

Fundamental thm. of identity types

Theorem (Fundamental thm. of identity types)

Let $A : \mathcal{U}$ *and* $B : A \to \mathcal{U}$ *such that* $(x : A) \times B(x)$ *is contractible with centre of contraction* (x, p)*, then for any* $y : A$*,*

$$B(y) \simeq (x = y)$$

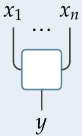
Theorem (Fundamental thm. of identity types)

Let $A : \mathcal{U}$ *and* $B : A \to \mathcal{U}$ *such that* $(x : A) \times B(x)$ *is contractible with centre of contraction* (x, p)*, then for any* $y : A$*,*

$$B(y) \simeq (x = y)$$

Corollary

Let (X_0, X_1) *be a* M*-0-algebra, for any constructor* $c : \mathsf{Cns}_M(i)$*, values* $x : \overrightarrow{X_0}(c)$*, and value* $y : X_0(i)$*,*

$X_1($ 
$) \simeq (\alpha_{X_1}(c, x) = y)$

Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

A M-opetopic type X is *fibrant* if it satisfies the following coinductive property:

- (X_0, X_1) is an algebra.
- $X_{>0}$ is a fibrant opetopic type.

Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

A M-opetopic type X is *fibrant* if it satisfies the following coinductive property:

- (X_0, X_1) is an algebra.
- $X_{>0}$ is a fibrant opetopic type.

$\mathcal{O}_M$ denotes the type of M-opetopic types.

Opetopic types

Some definitions of higher algebraic structures:

- $\infty\text{-}\mathsf{Grp} = (X : \mathcal{O}_{\mathsf{Id}}) \times \mathsf{is\text{-}fibrant}(X)$
- $(\infty, 1)\text{-}\mathsf{Cat} = (X : \mathcal{O}_{\mathsf{Id}}) \times \mathsf{is\text{-}fibrant}(X_{>0})$

Let X be a fibrant Id-opetopic type (i.e., an ∞-groupoid).

Let X be a fibrant Id-opetopic type (i.e., an ∞-groupoid).

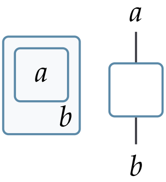
The family $X_0 : \mathsf{Fam}_{\mathsf{Id}}$ is equivalent to a type.

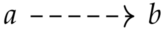
Let X be a fibrant Id-opetopic type (i.e., an ∞-groupoid).

The family $X_0 : \mathsf{Fam}_{\mathsf{Id}}$ is equivalent to a type.

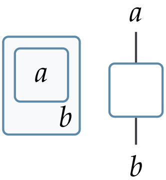
X_0 is the type of objects.

The family of 1-cells $X_1 : \mathsf{Fam}_{\mathsf{Id}/X_0}$ is a binary relation on X_0.



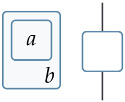


The family of 1-cells $X_1 : \mathsf{Fam}_{\mathsf{Id}/X_0}$ is a binary relation on X_0.

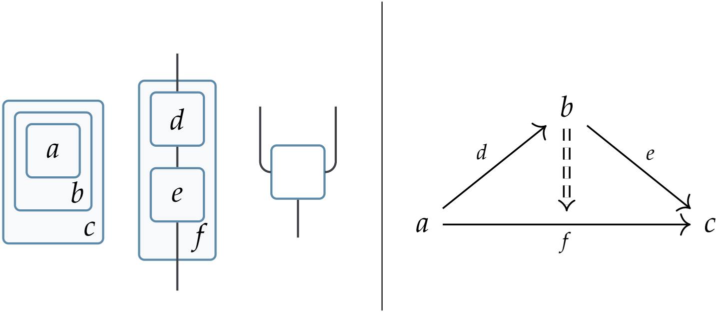


$$a \dashrightarrow b$$

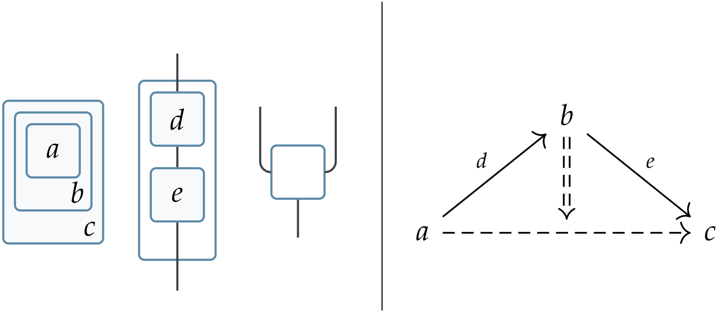
X being fibrant,

$X_1($  $) \simeq (a = b)$

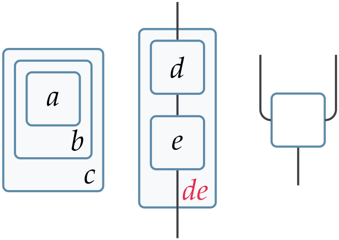
The family of 2-cells $X_2 : \mathsf{Fam}_{\mathsf{Id}/X_0/X_1}$ relates a source pasting diagram of 1-cells with a target *parallel* 1-cell.

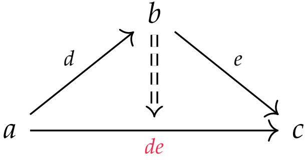


X being fibrant, pasting diagrams of 1-cells can be composed.

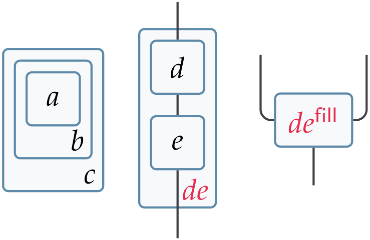


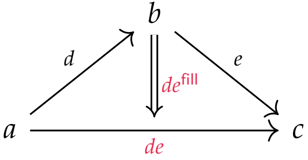
X being fibrant, pasting diagrams of 1-cells can be composed.





X being fibrant, pasting diagrams of 1-cells can be composed.





The family of 3-cells $X_3 : \mathsf{Fam}_{\mathsf{Id}/X_0/X_1/X_2}$ relates a source pasting diagram of 2-cells to a target 2-cell.

Fibrancy makes the of composition of 1-cells associative and unital.

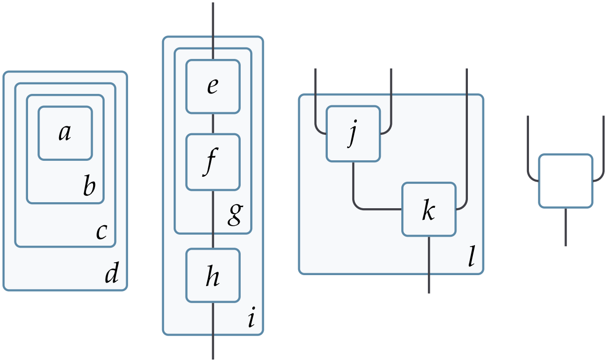
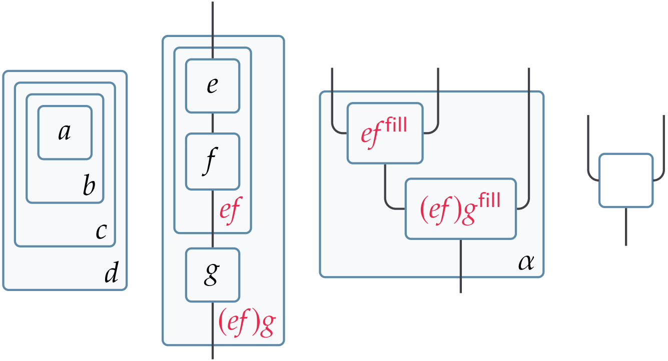


Figure: A 3-dimensional frame

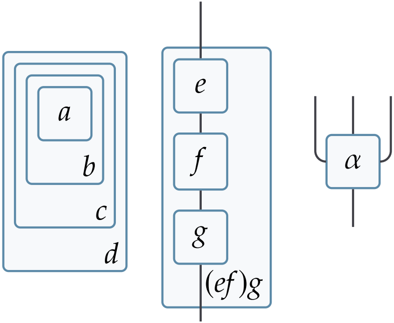
∞-GROUPOIDS

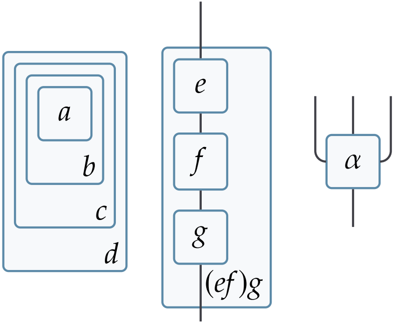
3-CELLS



∞-GROUPOIDS

3-CELLS





X is fibrant therefore

$$(ef)g = efg$$

The universe

0-cells

Types and their fibrant relations assemble into the $(\infty, 1)$-category

$$\mathcal{U}^o : \mathcal{O}_{\mathsf{Id}}$$

Types and their fibrant relations assemble into the $(\infty,1)$-category

$$\mathcal{U}^o : \mathcal{O}_{\mathsf{Id}}$$

Its family of objects $\mathcal{U}^o_0$ is the universe of types $\mathcal{U}$:

$$\mathcal{U}^o_0(*) \equiv \mathcal{U}$$

The family of 1-cells

$$\mathcal{U}_1^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o} \to \mathcal{U}$$

is a binary relation on $\mathcal{U}$.

The universe

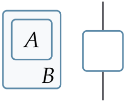
1-cells

The family of 1-cells

$$\mathcal{U}_1^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o} \to \mathcal{U}$$

is a binary relation on $\mathcal{U}$.

For example,

$\mathcal{U}_1^o($  $) \simeq (R : (a : A)\,(b : B) \to \mathcal{U}) \times \text{is-fibrant}(R)$

The universe

2-cells

The family of 2-cells

$$\mathcal{U}_2^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o/\mathcal{U}_1^o} \to \mathcal{U}$$

relates a *source* pasting diagram of 1-cells to a *target* 1-cell.

The universe
2-cells

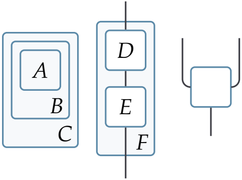
The family of 2-cells

$$\mathcal{U}_2^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o/\mathcal{U}_1^o} \to \mathcal{U}$$

relates a *source* pasting diagram of 1-cells to a *target* 1-cell.

For example,

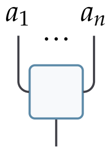
$\mathcal{U}_2^o($

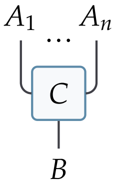


$) \simeq (R : (a : A)\,(b : B)\,(c : C)$
$\to (d : D(a,b))\,(e : E(b,c))\,(f : F(a,c)) \to \mathcal{U})$
$\times\ \text{is-fibrant}(R))$

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

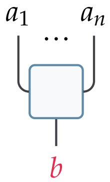


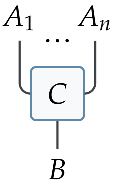


The universe

Fibrant relations

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

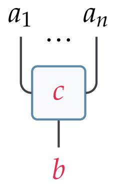


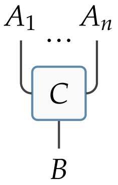


The universe

Fibrant relations

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.





Conclusion

Fibrant opetopic types are internal presentations of types which enables the definition of higher algebraic structures on arbitrary types.

Conclusion

Fibrant opetopic types are internal presentations of types which enables the definition of higher algebraic structures on arbitrary types.

The geometry of opetopes is particularly suited to a type-theoretical approach.

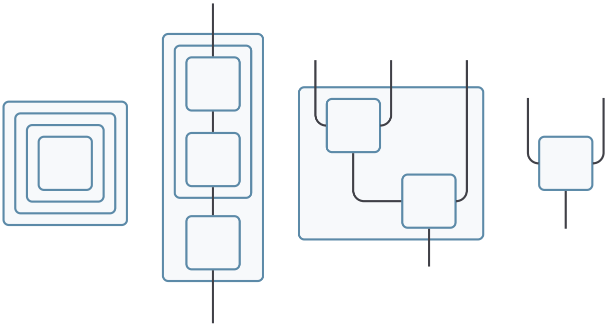
Conclusion

Fibrant opetopic types are internal presentations of types which enables the definition of higher algebraic structures on arbitrary types.

The geometry of opetopes is particularly suited to a type-theoretical approach.

Paves the way for the development of higher category theory in univalent opetopic foundations.

Thank you for your attention.



Slice monad constructors

The type of constructors of the slice monad is an inductive type with two constructors:

$$\mathsf{lf} : (x : \mathsf{Idx}_M) \to \mathsf{Cns}_{M/}\ (x \triangleleft \eta_M\ x)$$

$$\mathsf{nd} : (x : \mathsf{Idx}_M)\ (y : \mathsf{Cns}_M\ x)\ \{z : \overrightarrow{\mathsf{Cns}_M}\ y\}$$
$$\to (t : \overrightarrow{\mathsf{Cns}_{M/}}\ (y \blacktriangleleft z))$$
$$\to \mathsf{Cns}_{M/}\ (x \triangleleft \mu_M\ y\ z)$$