The Univalent Notion of Equality

HÅKON R. GYLTERUD

"Every type of objects has a natural, extensional notion of equality"

"This notion is called the identity type"

"The Univalence Axiom lets us compute the identity type"

Why?

"This equality is preserved by all expressible operations giving some amount of correctness for free."

Why?

“This equality is preserved by all expressible operations giving some amount of correctness for free.”

Huh‽

In software, equality is hairy:

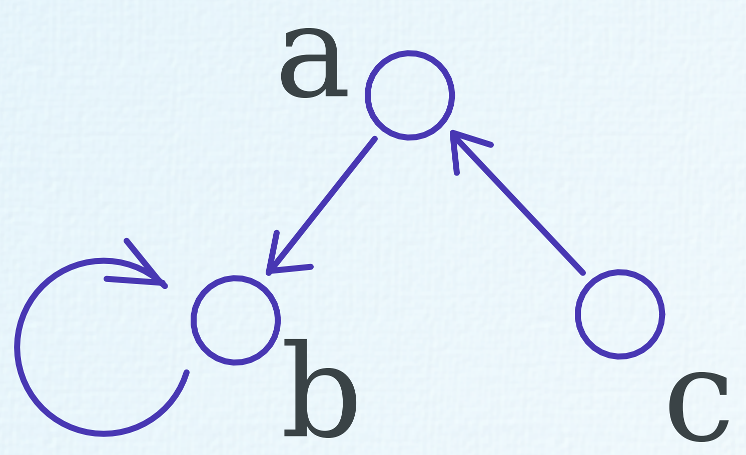
```
$ python
>>> (0.3 + 0.7) – 0.7 == 0.3 + (0.7 – 0.7)
        False
```

Programs work with representations:

[(a,b),(b,b),(c,a)]

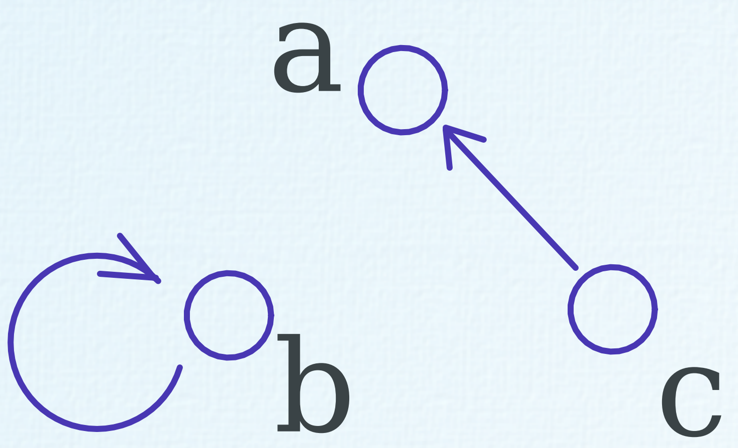
and could both represent:

[(c,a),(b,b),(a,b)]

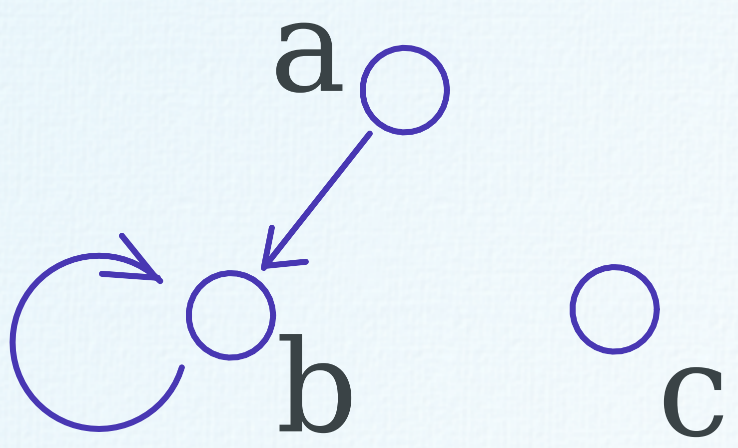


Not all operations on representations preserve semantic equality:

deleteFirstEdge([(a,b),(b,b),(c,a)])
= [(b,b),(c,a)]



But:
deleteFirstEdge([(c,a),(b,b),(a,b)])
= [(b,b),(a,b)]



Conventional solution: APIs

- Hide direct access to representations.
- Provide access to a set of “good” operations to work with objects.

Example:

filterEdges :: (Edge → Bool) → Graph → Graph

With APIs, we must hope and pray (or prove) that:

- The operations provided really respect equality.
- Everything we want can be done using the provided operations.

Often using APIs involve “pinky swearing”:

mapKeysMonotonic
    :: (k → k') → Map k a → Map k' a

Must be montone!

But there is no way to express this in the syntax.

Dependent type theory approaches this problem differently:

- Much more expressive type system
- Accurate representations:

  → All expressible operations preserve equality

- Univalence allows computing exactly what equality is.

Different notions of equality:

- Syntactic $1 = 1$ but not $1+1 = 2$
- Definitional $f(x) := x^2$
- Representational $0001010 = 0001010$
- Computational $1+2=3$ but not $\sum_{k=0}^{n} k = \frac{n^2 + n}{2}$
- Dynamic / equality of entities
- Extensional

Leibniz' Principle

(G.W. Leibniz, Discourse on metaphysics, 1686)

$$X = Y \quad \Leftrightarrow \quad \forall P.\ \ P(X) \ \Leftrightarrow \ P(Y)$$

Predicate / property

Fundamental property of extensional equality

Propositions as types

Brouwer — Heyting — Kolmogorov

Computational interpretation of logic

- L. E. J. Brouwer (1881–1966)
- Arend Heyting (1898–1980)

} 1934 (1908, 1924)

- Andrey Kolmogorov (1903–1987)

1932 (Propositional logic)

Logical connectives and quantifiers

$$\top \qquad\qquad \bot$$

$$A \land B \qquad A \lor B \qquad A \to B$$

$$\exists x \in D \;\; P(x) \qquad\qquad \forall x \in D \;\; P(x)$$

$$\neg A \coloneqq A \to \bot$$

BHK: What is a proof?

$*$ is a proof of $\top$

A constant!

(a,b) is a proof of $A \wedge B$ if $\begin{cases} \text{a is a proof of A} \\ \text{and} \\ \text{b is a proof of B} \end{cases}$

A pair!

l(a) is a proof of $A \vee B$ if a is a proof of A
r(b) is a proof of $A \vee B$ if b is a proof of B

Labels!

A proof of $A \to B$ is a <u>function f</u> which
transformes any proof a of A
into a proof f(a) of B

(d,p) is a proof of $\exists x \in D\ P(x)$
if $d \in D$ and p is a proof $P(d)$

f is a proof of $\forall x \in D\ P(x)$
if for each $d \in D$, $f(d)$ is a proof of $P(d)$

Again a function!

The rules of logic, justified:

Intuitionistic!

$$\frac{A \to B \qquad A}{B} \quad \text{Modus Ponens}$$

MP is justified since given f proving $A \to B$ and a proving A, we get f(a) proving B.

Propositions as types

Curry—Howard (1934,1958,1969)

Types:	Elements:
0 the empty type	
1 the unit type	$\ast : 1$
$A \times B$ (binary) product type	$(a,b) : A \times B$ when $a:A$ and $b:B$
$A + B$ (binary) sum type	$l(a) : A+B$ when $a:A$ $r(b) : A+B$ when $b:B$
$A \to B$ function type	$\lambda(x:A).b(x)$ when $b(x) : B$

Martin-Löf, 1972

Propositions as types

Curry—Howard (1934,1958,1969)

Types:		Elements:
$\sum(x:A)\ B(x)$	dependent sum	(a,b) where $a:A$ and $b:B(a)$
$\prod(x:A)\ B(x)$	dependent product	$\lambda(x:A).b(x)$ where $b(x) : B(x)$
$\mathbb{N}$, Vec A	inductive types	$S^n 0$, vectors
Type	universe type	Closed under all other type formers.

Propositions as types

Curry—Howard (1934,1958,1969)

A type	A is a proposition
a : A	a is a proof of A

Propositions as types

Curry—Howard (1934,1958,1969)

$0 , 1$	$\top , \bot$
$A \times B$	$A \wedge B$
$A + B$	$A \vee B$
$A \to B$	$A \to B$
$\sum(x:A)\ B(x)$	$\exists(x \in A)\ B(x)$
$\prod(x:a)\ B(x)$	$\forall(x \in a)\ B(x)$

Simplified for simplicity!

Example: Sorting

In Haskell: sort :: [ℕ] → [ℕ]

In type theory we can translate the prop.:

"Every list can be sorted"

$\forall s \in [\mathbb{N}]\ \exists \sigma \in \mathrm{Perm}\ \ \mathrm{Sorted}(s \circ \sigma)$

To a type:

$\prod(s : [\mathbb{N}])\ \sum(\sigma : \mathrm{Perm})\ \mathrm{Sorted}(s \circ \sigma)$

whose elements are sorting functions w/proof of correctness!

But what about equality?

The Identity Type

Martin-Löf (late 1970s)

For any type A and elements a,a' : A

the type $a =_A a'$ is "generated" by $\mathrm{refl}(a) : a =_A a$

Examples:

$0 =_{\mathbb{N}} 1$ is empty!

$2+2 =_{\mathbb{N}} 4$ has exactly one element: refl(4) (Hedberg 1998) rich!

The identity is suprisingly ~~complex~~.

$p : a =_A a'$ is a proof that a equals a'

but there can be more than one such proof!

But Leibniz' principle holds for the identity type!

Homotopy type theory:

The understanding of:

$$\left.\begin{array}{l} a =_A a' \quad \text{as a path type and} \\ \text{types as } \omega\text{-groupoids.} \end{array}\right\} \quad \text{Warren (2009)}$$

Defined using
the identity type!

The type $A \simeq B$ of equivalences from A to B

Voevodsky (2009)

Generalises isomorphisms from
category theory!

Now we can compute identity types:

$$(a,b) =_{A\times B} (a',b') \quad \simeq \quad a =_A a' \times b =_B b'$$

$$f =_{A\to B} g \quad \simeq \quad \prod(x:A)\ f(x) =_B g(x)$$

$$A =_{Type} B \quad \simeq \quad (A \simeq B)$$

Univalence!

Example: Graphs in Homotopy Type Theory

A graph consists of:

$$\begin{aligned}&\mathrm{Node} : \mathrm{Type}\\ &\mathrm{Edge} : \mathrm{Node} \to \mathrm{Node} \to \mathrm{Type}\end{aligned}$$

$$\mathrm{Graph} \coloneqq \textstyle\sum(\mathrm{N} : \mathrm{Type})\ (\mathrm{Node} \to \mathrm{Node} \to \mathrm{Type})$$

Example: Graphs in Homotopy Type Theory

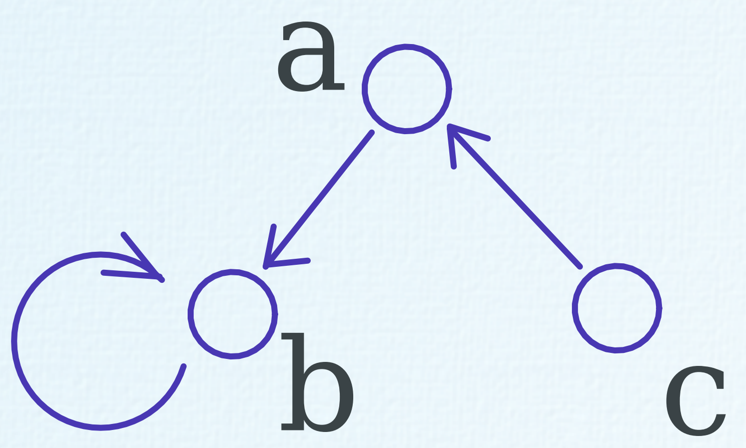
A graph consists of:

$$\begin{aligned}&\mathrm{Node} : \mathrm{Type}\\ &\mathrm{Edge} : \mathrm{Node} \to \mathrm{Node} \to \mathrm{Type}\end{aligned}$$

$$\mathrm{Graph} \coloneqq \sum(\mathrm{N} : \mathrm{Type})\ (\mathrm{Node} \to \mathrm{Node} \to \mathrm{Type})$$

Example!

$$\left(\{a,b,c\}\ ,\ \begin{cases} a\ ,\ b \mapsto 1 \\ b\ ,\ b \mapsto 1 \\ c\ ,\ a \mapsto 1 \\ \_\ ,\ \_ \mapsto 0 \end{cases} \right)$$



Example: Graphs in Homotopy Type Theory

A graph consists of:

$$\mathrm{Node} : \mathrm{Type}$$
$$\mathrm{Edge} : \mathrm{Node} \to \mathrm{Node} \to \mathrm{Type}$$

$$\mathrm{Graph} \coloneqq \textstyle\sum(N : \mathrm{Type})\ (\mathrm{Node} \to \mathrm{Node} \to \mathrm{Type})$$

Using univalence we can compute:

$$(N, E) =_{\mathrm{Graph}} (N', E')$$

$$\simeq \quad \textstyle\sum(\alpha : N \simeq N')\ \prod(n\ n' : N)\ E(n,n') \simeq E'(\alpha(n), \alpha(n'))$$

MORAL: An equality between graphs is a graph isomorphism!

Example: Graphs in Homotopy Type Theory

MORAL: An equality between graphs is a graph isomorphism!

Leibniz' principle: Any graph property
is invariant under isomorphism!

Also: Every operation preserves isomorphisms of graphs!

For instance any function Graph $\to \mathbb{N}$ is a graph invariant.

"Every type of objects has a natural, extensional notion of equality"

"This notion is called the identity type"

"The Univalence Axiom lets us compute the identity type"

Why?

“This equality is preserved by all expressible operations giving some amount of correctness for free.”

Applications

Homotopical Patch Theory

Angiuli, Morehouse, Licata, Harper (2016)

Like Git!

A formalisation of version control systems:

- The repository is a type

“commit” in Git-lingo

- A path in the repository is a patch
- Merging is a special kind of operation on patches:

$$
\begin{tikzcd}
& S_3 & \\
S_1 \arrow[ur, dashed] & & S_2 \arrow[ul, dashed] \\
& S_0 \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

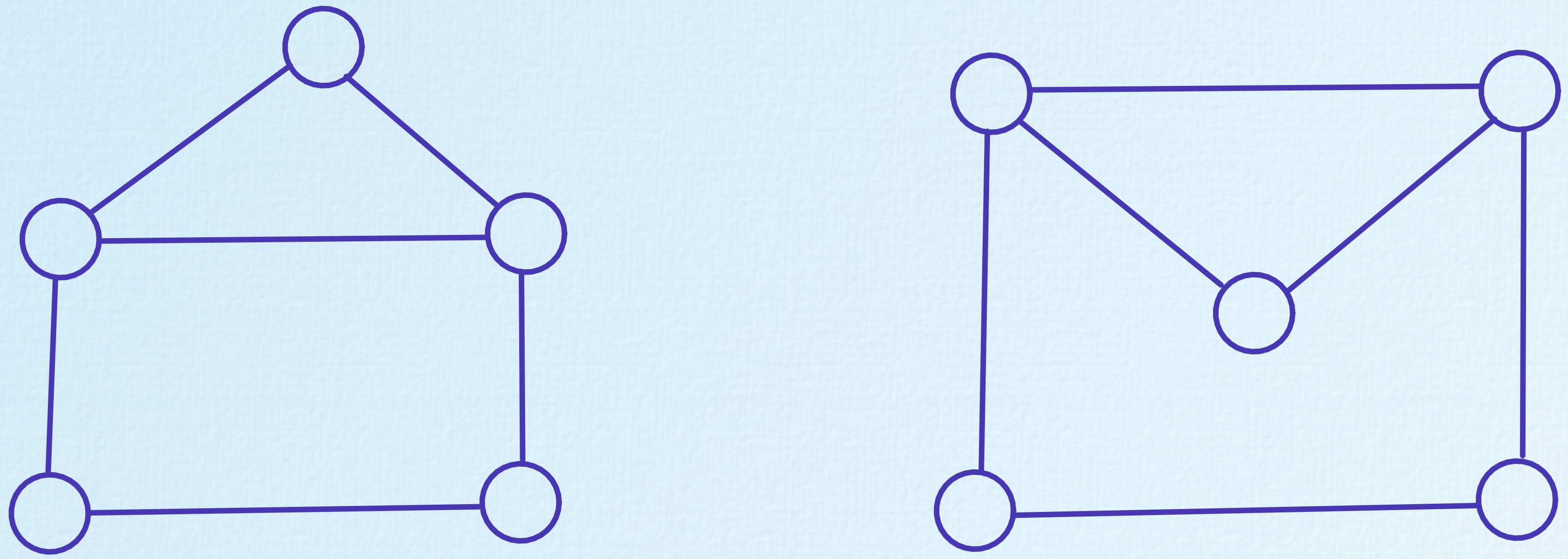
Planar Graphs in HoTT

Prieto Cubides, Gylterud (on-going)

Defining a notion of graph embedding in the plane.

Equality of embeddings is isotopy.

Continous deformation
w/o crossing edges!



Thank you!