The Identity Type

and Applications

Part I

PROBLEM: Definitional (Judgmental) Equality : Insufficient

$$\forall n,\ 0 + n \equiv n \qquad \text{(OK)}$$

$$\forall n,\ n + 0 \equiv n \qquad \text{(not OK)}$$

SOLUTION Identity type

Formation

$$\frac{a : A \quad b : A}{a =_A b}$$

Introduction

$$\frac{a : A}{\mathrm{refl}_a : a =_A a}$$

x is propositionally equal to y if $x =_A y$ is inhabited.

FACT Definitional Equality $\Rightarrow$ Propositional Equality

Converse not true

Ex. $2 =_{\mathbb{N}} 2$ is inhabited

$2+2 =_{\mathbb{N}} 4$ is inhabited:

$2+2 \equiv 4$, so $2+2 =_{\mathbb{N}} 4 \equiv \underbrace{4 =_{\mathbb{N}} N}_{\text{inhabited}}$

equal types!

What else? Will need properties of identity type

Properties of Identity Type

Reflexivity : by construction

Symmetry

$$\mathrm{sym} : (A : \mathcal{U}) \to (x : A) \to (y : A) \to (x =_A y) \to (y =_A x)$$

$$\mathrm{sym} : (A : \mathcal{U}) \to (x\ y : A) \to (x =_A y) \to (y =_A x)$$

Transitivity

$$\mathrm{trans} : (A : \mathcal{U}) \to (x\ y\ z : A) \to (x =_A y) \to (y =_A z) \to (x =_A z)$$

Congruence

$$\mathrm{cong} : (A\ B : \mathcal{U}) \to (f : A \to B) \to (x, y : A) \to (p : x =_A y) \to (f\,x =_A f\,y)$$

Propositional equality is an equivalence relation, is also a congruence.

How to prove these? ... Elimination Rule (J)

Given

1) A type family $C : (x\, y : A) \to (x =_A y) \to \mathcal{U}$

2) A function $c : (x : A) \to C(x, x, \mathrm{refl}_x)$

There is a function

$$f : (x : A) \to (p : x =_A y) \to C(x, y, p)$$

such that

$$f(x, x, \mathrm{refl}_x) \equiv c(x)$$

Package input + output into very long rule for the inductor (elimination rule J).

Proof of congruence

To construct

$$\mathrm{cong}\, f : (x\ y : A) \to (p : x =_A y) \to (f x =_B f y)$$

Choose $\quad C\ x\ y\ p = (f x =_B f y)$

Set $\quad c\ x = \mathrm{refl}_{(f x)} : C\ x\ x\ p$

This works because $C\ x\ x\ p = f x =_B f x$ ... inhabited by refl_x

---

Note cong is constructed using only two inputs: refl and the eliminator (J) for the identity type. This is a special case of the principle that functions out of a given type are constructed using the eliminator for that type.

Alternate proof style: Prove for $x=y$, refl_x:

$$\mathrm{cong}\ f\ x\ x\ \mathrm{refl}_x : \underline{(f\,x =_B f\,y)}$$ inhabited by $\mathrm{refl}_{(f\,x)}$

$$\mathrm{cong}\ f\ x\ x\ \mathrm{refl}_x = \mathrm{refl}_{(f\,x)}$$

Symmetry

$$\mathrm{sym} : (A : U) \to (x\ y : A) \to (x =_A y) \to (y =_A x)$$

$$\mathrm{sym}\ A\ x\ x\ \mathrm{refl}_x : (x =_A x)$$ $\leftarrow$ inhabited by refl

$$\mathrm{sym}\ A\ x\ x\ \mathrm{refl}_x = \mathrm{refl}_x$$

Agda

```
module IdentityType where

open import Agda.Primitive

variable
  ℓ : Level
  A : Set ℓ

data _≡_ {ℓ : Level} {A : Set ℓ} (x : A) : Set ℓ where

sym : {x y : A} → x ≡ y → y ≡ x
sym refl = refl
```

defines implicit parameters that are shared across multiple definitions.

Implicit args

Transitivity

$$\text{trans} : (A : U) \to (x\,y\,z : A) \to (x =_A y) \to (y =_A z) \to (x =_A z)$$

$$\text{trans}\ A\ x\ x\ z\ \text{refl}_x : (x =_A z) \to (x =_A z)$$

$$\text{trans}\ A\ x\ x\ x\ \text{refl}_x\ \text{refl}_x : (x =_A x)$$ ← inhabited by refl_x

$$\text{trans}\ A\ x\ x\ x\ \text{refl}_x\ \text{refl}_x = \text{refl}_x$$

Claim: there is a function $\text{lemma} : (n : \mathbb{N}) \to (n+0) =_{\mathbb{N}} n$

Proof (induction) $\text{lemma}\ 0 = \text{refl}_0$

because $\text{lemma}\ 0 = (0+0 =_{\mathbb{N}} 0) \equiv (0 =_{\mathbb{N}} 0)$ ← inhabited by refl_0

I.H. Assume $\text{lemma}\ n = (n+0) =_{\mathbb{N}} n$

Apply congsucc to I.H., where

$$\text{cong}\ f : (x\ y : A) \to (p : x =_A y) \to (f x =_B f y)$$

$$\begin{aligned} \text{cong}\ \text{succ}\ (\text{lemma}\ n) &: (\text{succ}\ (n+0) =_{\mathbb{N}} \text{succ}\ n) \\ &: ((\text{succ}\ n) + 0) =_{\mathbb{N}} \text{succ}\ n) \\ &: \text{lemma}\ (\text{succ}\ n) \end{aligned}$$

Because $\text{succ}\ (n+0) \equiv (\text{succ}\ n) + 0$

Conclude: $\text{lemma}\ (\text{succ}\ n) = \text{cong}\ \text{succ}\ (\text{lemma}\ n)$

$$\text{lemma}\ (\text{succ}^n\ 0) = (\text{cong}\ \text{succ})^n\ \text{refl}_0$$

$$\boxed{\text{lemma}\ n = (\text{cong}\ \text{succ})^n\ \text{refl}_0}$$

Our lemma in Agda

$$\text{lemma} : (n : \mathbb{N}) \to n + 0 \equiv n$$
$$\text{lemma zero} = \text{refl}$$
$$\text{lemma (succ } n) = \text{cong succ (lemma } n)$$

Familiar scheme

- pattern matching (converted into J internally)
- two clauses, one for each constructor of $\mathbb{N}$
- proof by induction
- no mention of $\mathbb{N}$ in "$n + 0 \equiv n$": <u>type inference</u>
- <u>note</u>: "$\equiv$" instead of $=$ (An Agda peculiarity)
- <u>Alternate syntax</u>: $\forall (n : \mathbb{N}) \to n + 0 \equiv n$

Review <u>lemma</u> is a <u>dependent function</u>:

$$\text{lemma} : (n : \mathbb{N}) \to n + 0 \equiv n$$

↑ we can't say "$\mathbb{N}$" in place of "$(n:\mathbb{N})$" because n appears to the right in the formula $n+0 \equiv n$.

- Dependent functions are used to express universal quantification. Hence the alternate notation $\forall (n:\mathbb{N}) \to n+0 \equiv n$

- <u>Input</u> $n : \mathbb{N}$ <u>Output</u>: a witness to the proposition $n+0 \equiv n$.

Alternate notation $\Pi_{(n:\mathbb{N})}\, n+0 = n$ ("Pi" types)

Review, continued

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma, x : A \vdash B(x) : \mathcal{U}}{\Gamma \vdash (x : A) \to B(x) : \mathcal{U}} \quad \text{formation}$$

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x . b(x) : (x : A) \to B(x)} \quad \text{intro}$$

$$\frac{\Gamma \vdash f : (x : A) \to B(x) \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B(a)} \quad \text{elim}$$

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{(\lambda x . b(x))(a) \equiv b(a)} \quad \text{comp.}$$

context Γ extended by $x : A$

Context $\Gamma = x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$

- a list of variable declarations where $x_j : A$ is defined in the context $x_1 : A_1, \ldots x_{j-1} : A_{j-1}$

Let's think about this:

$$\boxed{\text{lemma } n = (\text{cong succ})^n \, \text{refl}_0}$$

$$\text{lemma } n : (n+0) =_{\mathbb{N}} n$$

inhabited by

$(\text{cong succ})^n \, \text{refl}_0$

Examples: (1) View $\mathbb{Z}$ as a cyclic $\mathbb{Z}$-module:

$\mathbb{Z}_{\text{as module}} = \{n \cdot 1 \mid n \in \mathbb{Z}\}$.

(2) View $\mathbb{Z}/7$ as a cyclic $\mathbb{Z}$-module:

$(\mathbb{Z}/7)_{\text{as module}} = \{n \cdot 1 \mid n \in \mathbb{Z}\}$

$\mathbb{Z}$ is a free cyclic module. While $\mathbb{Z}/11$ is a cyclic $\mathbb{Z}$-module it is not free.

The HoTT book says that the identity type is "freely generated" by refl. The correct analogy is that of a free cyclic module M over a ring R: $M = \langle m \rangle = \{ rm \mid r \in R \}$. The module element m is the free generator. There are as many elements of $\langle m \rangle$ as there are of R.

Remarks

- It is because the identity type is freely generated by refl, that it suffices to provide proofs in the case $x = x$, $p = \mathrm{refl}_x$.

- The only way to produce a term of the identity type is to apply the eliminato (J) to refl. This is what we did in the example when we applied $(\mathrm{cong\ succ})^n$ to refl_0. Recall that cong was built using J.

- A type such as $(n+0=0)$ is not of the form $(a=a)$. Yet it is inhabited.
  If only $(a=a)$ were inhabited, propositional equality would be the same as definitional equality.

<u>Question</u>: Suppose $(a =_A b)$ is inhabited by terms p and q. Is it the case that $p = q$?

What do we mean by equality? Note that $(a =_A b)$ is a type, so it has its own identity type. The proper frame of reference for our question is

"Are p and q propositionally equal,"

means

"Is $p =_{(a =_A b)} q$ inhabited."

Affirmative answer to the question is

<u>UIP</u> : <u>Uniqueness of Identity Proofs</u>

$$\mathrm{UIP} : (A : \mathcal{U}) \to (x\ y : A) \to (p\ q : x =_A y) \to (p = q)$$

UIP is not provable in MLTT, but it is consistent with it — can be assumed as an axiom. (People did try!)

However, UIP does hold for certain types, e.g $\mathbb{N}$:

$$\mathrm{UIP}_{\mathbb{N}} = \mathrm{UIP}\ \mathbb{N} = (x\ y : \mathbb{N}) \to (p\ q : x =_{\mathbb{N}} y) \to (p = q)$$

<u>UIP = isSet</u>

$$\mathrm{isSet}\ A :\equiv (x\ y : A) \to (p\ q : x = y) \to (p = q)$$

$\text{isSet}\mathbb{N} :\equiv (m\, n : \mathbb{N}) \to (p\, q : m = n) \to (p = q)$

(double)

We proceed by identity induction on $p, q : m = n$.

By the eliminator for the identity type, it suffices to consider the case $p \equiv \text{refl} : m = m$. We are reduced to considering

$$\text{isSet}\mathbb{N} : (m : \mathbb{N}) \to (q : m = m) \to (\text{refl} = q)$$

Now we apply identity induction on q, reducing to the case $q \equiv \text{refl} : m = m$, so that we are left to consider

$$(m : \mathbb{N}) \to (\text{refl} = \text{refl})$$

The right hand side is inhabited by refl

Q.E.D.

<u>Conclusion: $\mathbb{N}$ is a set.</u>

<u>Corollary</u>: The <u>only</u> inhabitant of $(n + 0 =_{\mathbb{N}} n)$ is

$$\boxed{(\text{cong succ})^n\, \text{refl}_0}$$

Computability and UIP (The downside of adding axioms)

- Computability means that all terms have <u>canonical forms</u> and computation corresponds to normalization.
- <u>Canonical forms</u>: term is in normal form <u>and</u> has the shape expected for a term of its type
  - zero, succ zero, etc for $\mathbb{N}$
  - true, false for Bool
  - $\lambda x \to x$ is a canonical form for a function type
- Canonical forms contain only constructors and λ-abstractions (for function types) — no <u>stuck terms</u>, no redexes

Examples

- $(\lambda x \to \text{succ } x)$ zero $\quad$ reducible

  $\rightsquigarrow$ succ zero $\quad$ normal form, canonical

- identity : $\mathbb{N} \to \mathbb{N}$
  identity $x = x$

  foo : $\mathbb{N} \to \mathbb{N}$
  foo = identity $\quad$ normal form, not canonical
  - it is variable headed, not constructor or λ-abstraction

- bar : $\mathbb{N} \to \mathbb{N}$
  bar = $\lambda x \to x$ $\quad$ this is canonical

Adding an Axiom

Say this in Agda:

$$\textbf{postulate}\ \mathrm{UIP} : \forall \{A : \mathrm{Set}\}\ \{x\ y : A\}\ (p\ q : x \equiv y) \to p \equiv q$$

Now try to normalize

$$\mathrm{UIP}\ \mathrm{refl}\ \mathrm{refl} \to \mathrm{UIP}\ \mathrm{refl}\ \mathrm{refl}$$

There are no rules to reduce this to canonical form.

$\langle$ DEMO $\rangle$

Advantages of UIP

- Reasoning is more like classical mathematics
- Simplifies proofs:
  - pattern match on equalities, even self-equalities.
  - no higher identity types
- Some lemmas become trivial, e.g.

  $(A : \mathcal{U}) \to (x\ y : A) \to (p : x = y) \to \mathrm{sym}\,(\mathrm{sym}\ p) = p$

  with UIP, proof is refl.

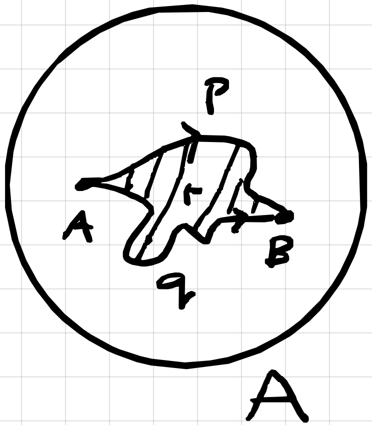
Disadvantages

- no Computation with UIP
- no higher inductive types (no circle )
- everything is a set — boring!

Homotopy Type Theory

Types as Spaces: Another Dictionary

$A : \mathcal{U}$	space
$a : A$	point
$p : (a =_A b)$	path from a to b
$r : p =_{(a =_A b)} q$	path from p to q (2-cell)



---

higher dimen. struct. $\left\{ \begin{array}{l} \bullet\ p : (a =_A b) \text{ is a path from } a \text{ to } b \text{ (one-cell)} \\ \bullet\ r : p =_{(a =_A b)} q \text{ is a path from } p \text{ to } q \text{ (two-cell)} \end{array} \right.$

Towards HoTT : Homotopies

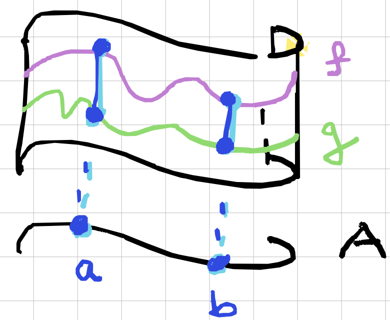
Defn Consider a type family $P : A \to \mathcal{U}$, and two dependent functions ("sections") of P, $f, g : (x:A) \to P(x)$. A homotopy from f to g is a dependent function

$$(f \sim g) : (x:A) \to \underbrace{(f(x) = g(x))}_{\text{path from } f(x) \text{ to } g(x)}$$

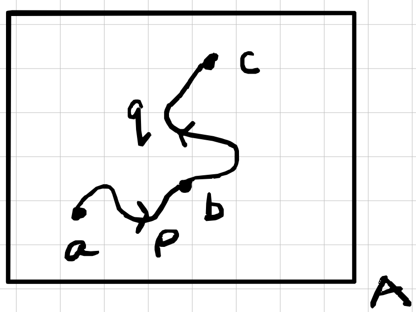
A homotopy $f \sim g$ gives a family of paths

$(f \sim g)(x) : (f(x) = g(x))$.

Just like a real homotopy in topology.



Groupoid Structure



In topology we concatenate paths whose endpoints join

$p : a =_A b \;,\; q : b =_A c$

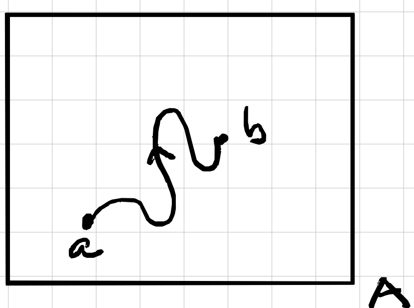
$p \cdot q : a =_A c$

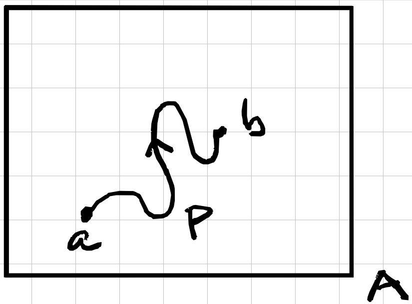
$p \cdot q = \text{trans } p \; q$

What is p^{-1}?

$p^{-1} = \text{sym } p$

(switch endpoints)





Are there identity elements?

$$p \cdot \mathrm{refl}_a \sim p$$

$$\mathrm{refl}_b \cdot p \sim p$$

Associative Law

$$(p \cdot q) \cdot r \sim p \cdot (q \cdot r)$$

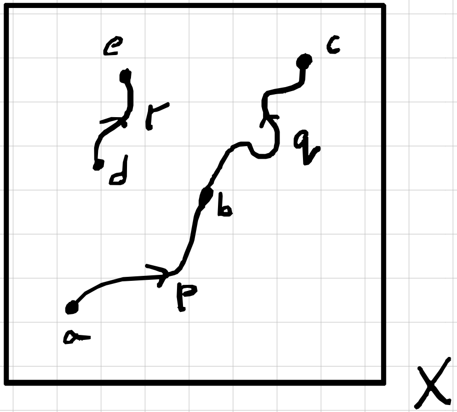
... + higher identities.

This structure — 1-paths, 2-paths, ...
with composition laws and identities,
is called an <u>∞-groupoid</u>, written $\Pi(A)$.

$\Omega(A, a)$ = the identity type $a =_A a$
with its higher structure: paths, paths between paths, ...
concatenation, etc.

Then $\Omega(A, a)$ is an <u>∞-group</u>.

<u>Something simpler</u>: Let X be a topological space, and let $\Pi_1(X)$ be the space of paths in X up to homotopy with fixed endpoints, and consider $\Pi_1(X, x)$, the paths up to homotopy based at x (loops).



$\Pi_1(X)$ is a <u>groupoid</u>

(category where all morphisms are isomorphisms)

$\Pi_1(X, x)$ is a <u>group</u> –

and a category with just one object.

The Homotopy Hypothesis

∞-groupoids and homotopy types are equivalent structures. (equivalence of ∞-categories)

... Idea goes back to Grothendieck in Pursuing Stacks, though he never formalized it.

Top_h : nice topological spaces up to homotopy

∞-Gpd : ∞-cat. of groupoids

$$Top_h \simeq \infty\text{-gpd}$$ (∞-cat where all morphisms are invertible)

Amazing fact In MLTT, a theory of logic, homotopical ideas arise naturally.

Sets and non-Sets

Revisiting Sets in MLTT

It may happen for a particular type A that there is no higher dimensional structure. In that case, we say that A is a set:

$$\text{isSet } A :\equiv (x\ y : A) \to (p\ q : x = y) \to (p = q)$$

Example: $\mathbb{N}$ is a set

Claim: There are types which are not sets.
This requires HoTT: postulate univalence (Voevodsky)

Note: both univalence and UIP are consistent with MLTT (but inconsistent with eachother).

A type which is not a set

- Define a type S^1 (the circle) with constructors

  - base : S^1 — point constructor
  - loop : base = base — path constructor

- The circle is a <u>higher inductive type</u>, meaning that it has higher-dimensional constructors such as <u>loop</u>.

- One can <u>compose</u> terms in base = base :

$$p \circ q = \text{trans } p \; q$$

For example, $\text{loop}^2 = \text{loop} \circ \text{loop} = \text{trans loop loop}$

S^1 continued

- One can invert terms $p : \mathrm{base} = \mathrm{base}$:

$$p^{-1} = \mathrm{sym}\ p$$

- $p \circ p^{-1} = p^{-1} \circ p = \mathrm{refl}_{\mathrm{base}}$

$$\mathrm{trans}\ p\ (\mathrm{sym}\ p) = \mathrm{refl}_{\mathrm{base}} \quad ?$$

$\dots$

- Conclude $\mathrm{base} = \mathrm{base}$ is a group.
- Definition $\Omega(S^1, \mathrm{base}) = (\mathrm{base} = \mathrm{base})$

the loop space of S^1

- Theorem (Shulman) $\Omega^1(S^1, \mathrm{base}) = \mathbb{Z}$

a higher inductive type (HIT)

Univalence: | equality, not isomorphism! |

Equivalence of Types : $A \simeq B$

Lemma: Homotopy is an equivalence relation

Definition Let $f : A \to B$. We say that f is an <u>equivalence of types</u> if there exist functions $g, h : B \to A$ such that $f \circ g \sim id_B$ and $h \circ f \sim id_A$. That is, f has left and right <u>homotopy inverses</u>.

$$\mathrm{isequiv}(f) :\equiv \left(\sum_{g : B \to A} (f \circ g) \sim id_B \right) \times \left(\sum_{h : B \to A} (g \circ f) \sim id_A \right)$$

Defn

$$(A \simeq B) :\equiv \sum_{f : A \to B} \mathrm{isequiv}(f)$$

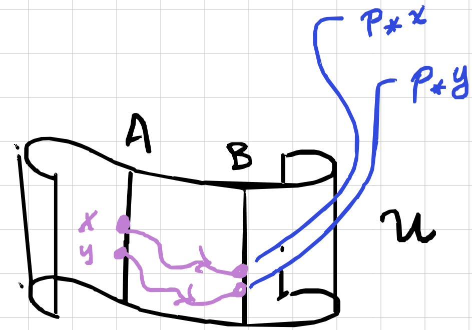
Lemma If $e_1, e_2 : \mathrm{isequiv}(f)$, then $e_1 = e_2$

$\rightsquigarrow$ If f defines an equivalence, we may write $f : A \simeq B$ instead of $(f, e) : A \simeq B$.

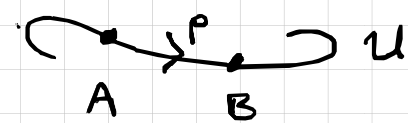
Univalence

There is a canonical map

$$\mathrm{idtoequiv} : A =_{\mathcal{U}} B \longrightarrow A \simeq B$$



If $p : A =_{\mathcal{U}} B$ is a path and $C : \mathcal{U} \to \mathcal{U}$ is a family of types, then there is a "transport" function



$$p_* : \text{fiber over } A \to \text{fiber over } B$$

Consider the universal family $\mathrm{id} : \mathcal{U} \to \mathcal{U}$.

Then fiber over $A = A$, fiber over $B = B$,

and

$$p_* : A \to B$$

<u>Define</u>

$$\boxed{\mathrm{idtoequiv}\ p = p_*}$$

Theorem (Voevodsky)

$\boxed{\text{idtoequiv is an equivalence}}$

Corollary The universe of small types is not a set.

Proof. Consider the type $\mathrm{Bool} : \mathcal{U}_0$. Let $f : \mathrm{Bool} \to \mathrm{Bool}$ be the unique involution of this type: the map such that $f\,\mathrm{false} = \mathrm{true}$ and $f\,\mathrm{true} = \mathrm{false}$. Since $f(f\,x) = x$ for all x, f is an equivalence. By univalence, f corresponds to a path p. If $p = \mathrm{refl}_{\mathrm{Bool}}$, then by univalence $p = \mathrm{id}_{\mathrm{Bool}}$. But

then true = false,

a contradiction. Note that $p : A =_{\mathcal{U}} B$. Thus $\mathcal{U}$ has identity types with $p \neq \mathrm{ref}$

QED

The End

(of this lecture)

Appendix

Part II : Proof that $\mathbb{N}$ is a set

Preliminaries : isSet ($\mathbb{1}$) unit type

- The unit type $\dfrac{}{\mathbb{1} : \mathcal{U}_0}$ $\dfrac{}{\ast : \mathbb{1}}$

$$\mathrm{ind}_{\mathbb{1}} \, (C : \mathbb{1} \to \mathcal{U}) \to (C \ast) \to (x : \mathbb{1} \to C x)$$

Define : $\mathrm{ind}_{\mathbb{1}} : (x : \mathbb{1}) \to (x = \ast)$

by the induction principle, setting $f \ast = \mathrm{refl}_{\ast}$

The function f is a proof that $\forall_{x : \mathbb{1}} \; x =_{\mathbb{1}} \ast$

Now consider arbitrary $x, y : \mathbb{1}$

Then $(x = y) \equiv (* = *)$ ← inhabited by refl_*

$\forall p : * = *, \; p = \mathrm{refl}_*$

$\forall q : * = *, \; q = \mathrm{refl}_*$

$\therefore \; p = q$

$\therefore \; \forall x, y : \mathbb{1}, \forall p, q : x = y, \; p = q$ <u>QED</u>

$\therefore \; \mathrm{isSet}(\mathbb{1})$

<u>Note</u>: <u>The empty type is also a set</u> ← trivially!

We will write the empty type as $\mathbb{0}$.

Example

Theorem For any $x, y : \mathbb{1}$, $(x = y) \cong \mathbb{1}$

Corollary If $p, q : x = y$, where $x, y : \mathbb{1}$, then $p = q$.

Proof $p, q : x = y \Rightarrow f(p), f(q) : \mathbb{1}$

$$\Rightarrow f(p) = f(q)$$

$$\Rightarrow g(f(p)) = g(f(q))$$

$$\Rightarrow p = q$$

Corollary: $\mathrm{isSet}(\mathbb{1})$

Preliminaries: Mere Propositions

In MLTT, a proposition P may have many proofs (terms). A <u>mere proposition</u> behaves like a truth value. Either it is inhabited (true) or uninhabited (false). That is, mere propositions are like an oracle: we know whether the proposition is proved, but we don't know why it is true. <u>No insight</u>

<u>Definition</u>: P is a mere proposition if $\forall x, y : P,\ x = y$. That is, any two elements are <u>propositionally</u> equal. (P is empty or has just one element).

Back to the proof: isSet $\mathbb{N}$

Recall:

$$\text{isSet } A :\equiv (x\ y : A) \to (p\ q : x = y) \to (p = q)$$

$\Leftrightarrow \forall x, y : A.\ \forall p\ q : x = y$

$\Leftrightarrow \forall x, y : A,\ x, y : A : x =_A y$ is a mere propositon

Proof by induction:

- zero = zero by refl
- succ m = succ n iff m = n by injectivity of constructors
- zero ≠ succ n by disjointness of constructors