Introduction to Homotopy Type Theory

Part 1

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.y. 2020/2021

Parts

- Introduction: from the λ-calculus to Martin-Löf type theory
- Homotopy type theory
- Classical logic
- Definition of equivalence

References

The history of the λ-calculus in details can be found in a paper by F. Cardone and J. R. Hindley: *History of Lambda-Calculus and Combinatory Logic*, Handbook of the History of Logic, vol. 5, 2006.

The rest of the lessons is from *Homotopy type theory: Univalent foundations of mathematics*, Institute for Advanced Study, Princeton, 2013. A pdf of the book is freely available at `https://homotopytypetheory.org/book`.

Syllabus

Introduction

- history of the λ-calculus
- simple theory of types
- propositions-as-types interpretation
- dependent types
- Martin-Löf type theory
- propositional equality

History

1928 Church defined the *λ-calculus*, a formal system whose purpose was to provide a foundation for logic;

1935 Kleene and Rosser found a *paradox* proving that λ-calculus, when viewed as a system of logic, is inconsistent;

1940 Church reformulated the *simple theory of types* on a λ-calculus base, to avoid the Kleene-Rosser paradox;

1958 Curry and Feyes described the *propositions-as-types isomorphism*;

1969 Howard extended the isomorphism to *proofs and terms*;

1972 Martin-Löf defined *Martin-Löf type theory*, a constructive type theory with dependent types;

2008 Voevodsky and a number of mathematicians joining the Univalent Foundations Program extended Martin-Löf type theory to *Homotopy type theory*.

Simple theory of types

The types in the simple theory of types are $\mathbf{1}$, $\mathbf{0}$, $A \times B$, $A + B$, $A \to B$, such that:

- $* : \mathbf{1}$;
- $\mathbf{0}$ is the empty type;
- if $a : A$ and $b : B$, then $(a, b) : A \times B$;
- if $a : A$, then $i_1^B(a) : A + B$; also, if $b : B$ then $i_2^A(b) : A + B$;
- if $b : B$ and $x : A$ is a variable, then $\lambda x : A.b : A \to B$.

We denote as $\mathrm{pr}_1 : A \times B \to A$ and $\mathrm{pr}_2 : A \times B \to B$ the two projections such that $\mathrm{pr}_1(a, b) \equiv a$ and $\mathrm{pr}_2(a, b) \equiv b$.

Propositions-as-types

There is a correspondence between the types and the propositions

types	propositions
$\mathbf{1}$	$\top$
$\mathbf{0}$	$\bot$
$A \times B$	$A \wedge B$
$A + B$	$A \vee B$
$A \to B$	$A \supset B$

which extends to a correspondence between the constructors for the terms of the types and the inference rules of logic, becoming a correspondence between the terms of a type and the proofs of the related proposition.

Propositions-as-types

The two introduction rules for the product type ($\times$–intro) and for logical conjunction ($\wedge$I) have the same structure:

$$\frac{a : A \quad b : B}{(a, b) : A \times B}\ \times\text{–intro} \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots \\ A & B\end{array}}{A \wedge B}\ \wedge\text{I}$$

Thus, a type is inhabited when the corresponding proposition is provable, and each distinct term of a type corresponds to a different proof of the related proposition.

Propositions-as-types

In the case of the introduction of the λ-term in an arrow type ($\rightarrow$ –intro) and implication ($\supset$ I), the correspondence becomes

$$\frac{x : A \quad b : B}{\lambda x : A.b : A \rightarrow B} \rightarrow\text{–intro} \qquad \frac{\begin{array}{c}[A]^{*} \\ \vdots \\ B\end{array}}{A \supset B} \supset \text{I}$$

where $x : A$ is a variable and the label $*$ stands for x.

Propositions-as-types

The same happens with the introduction rules for the coproduct type (+−intro) and logical disjuction (∨I):

$$\frac{a : A}{i_1^B(a) : A + B}\ +\text{−intro}_1 \qquad \frac{\begin{array}{c}\vdots\ \text{a}\\ A\end{array}}{A \vee B}\ \vee\text{I}_1$$

Notice that $i_1^B(a)$ has labels indicating to which type belongs the term a from which it is constructed, and which is the other type.

The type theory is constructive: each term is not only a statement saying that the corresponding type is inhabited, but also a witness showing how to construct an inhabitant.

Intuitionistic logic

If we choose the "right" logic, the above correspondence allows to identify it with a calculus system.

In this framework, the right logic is intuitionistic logic. It has the same inference rules of classical logic, but without the *law of excluded middle:*

$$\frac{}{A \vee \neg A}\,\mathrm{LEM}$$

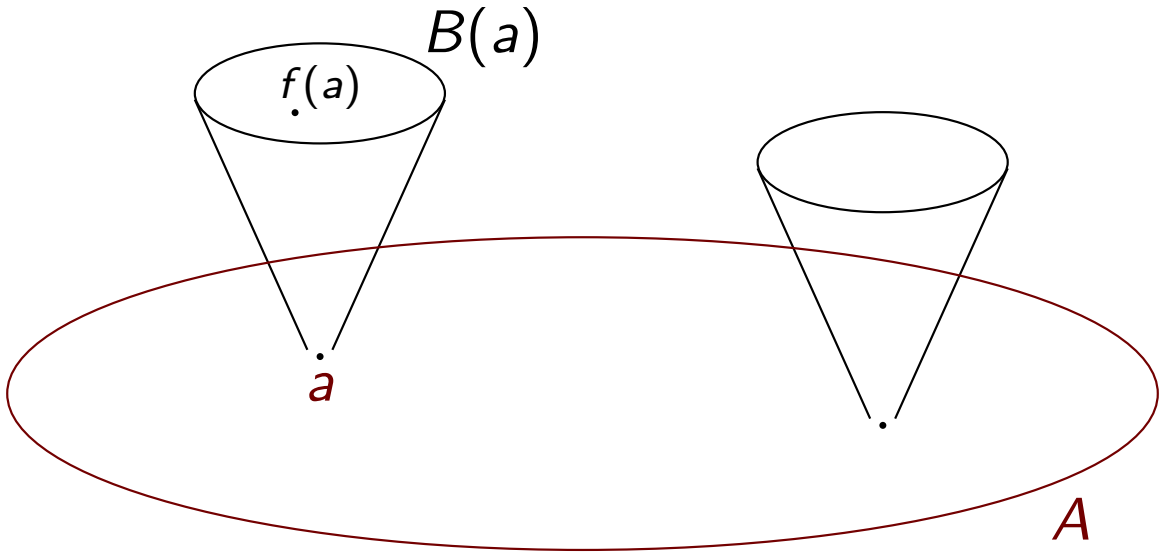
Thus, $\neg\neg A \supset A$ is not provable, and the *reductio ad absurdum* is not allowed.

Analogously to what happens with types, a proof of a proposition in intuitionistic logic does not only show that the proposition is true, but contains an algorithm allowing to obtain the content of the proposition.

Dependent types

The propositions-as-types interpretation can be extended to first order intuitionistic logic by introducing the notion of dependent types.

The concept of function $f : A \to B$ can be extended to the case in which B has a *dependent structure*: if B is a function with domain A, a function f from A to B can be defined such that $f(a) : B(a)$ for each $a : A$.



The same can be done for pairs.

Dependent types

Thus, are introduced

- $\Pi_{x:A} B(x)$ type of the dependent functions
- $\Sigma_{x:A} B(x)$ type of the dependent pairs

such that, if B does not depend on $x : A$, then $\Pi_{x:A} B(x) = A \to B$ and $\Sigma_{x:A} B(x) = A \times B$.

The canonical terms of a dependent type are defined as

- if $b : B$ depends on a variable $x : A$, then $\lambda x : A.b : \Pi_{x:A} B(x)$;
- if $a : A$ and $b : B(a)$, then $(a, b) : \Sigma_{x:A} B(x)$.

Dependent types

Example 1

A dependent pair is

$$\Sigma_{n:\,\mathbb{N}}\mathbb{N}^n$$

containing all the n-tuples of naturals for each $n:\mathbb{N}$.

Example 2

An example of term of dependent product is

$$\mathsf{zeros}:(\Pi_{n:\,\mathbb{N}}\mathbb{N}^n)$$

whose terms are such that $\mathsf{zeros}\,n$ is the n-component vector $(0,\ldots,0)$.

Propositions-as-types

This allows to extend the propositions-as-types correspondence as

types	propositions
$\mathbf{1}$	$\top$
$\mathbf{0}$	$\bot$
$A \times B$	$A \wedge B$
$A + B$	$A \vee B$
$A \to B$	$A \supset B$
$\Pi_{x:A} B(x)$	$\forall x \in A. B(x)$
$\Sigma_{x:A} B(x)$	$\exists x \in A. B(x)$

Propositions-as-types

In the case of the introduction rule for the Σ-type ($\Sigma-$intro) and the existential quantifier ($\exists$I):

$$\frac{a : A \quad b : B(a)}{(a, b) : \Sigma_{x:A} B(x)}\ \Sigma-\text{intro} \qquad\qquad \frac{\begin{array}{cc}\vdots\ \text{a} & \vdots\ \text{b} \\ A & B(a)\end{array}}{\exists x \in A.B(x)}\ \exists\text{I}$$

Notice that $(a, b) : \Sigma_{x:A} B(x)$ does not only say that a certain $B(x)$ is inhabited, but allows to reconstruct that the instance with a witness is $B(a)$. This is a consequence of the constructive nature of the type theory, as we have seen for coproducts.

Martin-Löf type theory

Martin-Löf type theory consists in the simple theory of types extended with Π, Σ, natural numbers and an equality type, plus an infinite hierarchy of universes.

Naturals are defined inductively:

- $\mathbb{N}$ is a type;
- $0 : \mathbb{N}$;
- if $n : \mathbb{N}$, then $\mathrm{succ}(n) : \mathbb{N}$.

Universes

Another feature of Martin-Löf type theory is an infinite hierarchy of Russell-style universes

$$\mathcal{U}_0 : \mathcal{U}_1 : \cdots : \mathcal{U}_i : \mathcal{U}_{i+1} : \cdots$$

such that $A : \mathcal{U}_i$ implies $A : \mathcal{U}_{i+1}$, deputed to formalise the notion of type

Definition 3

A term a is a type if it is a term of a universe, i.e., if there is an index $i \in \mathbb{N}$ such that $a : \mathcal{U}_i$.

The index i of $\mathcal{U}_i$ is omitted when irrelevant.

Martin-Löf type theory

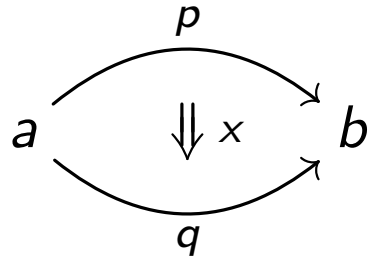
Given $a, b : A$, the propositional equality type $a =_A b$ is defined by introducing the term $\mathrm{refl}_a : a =_A a$, stating that equality is reflexive.

The idea behind propositional equality is that if there is a term of type $a =_A b$ then a and b are equal.
In general, if a and b are closed terms then $a =_A b$ is empty when a is syntactically different from b up to conversions, but nothing prevents that an equality type contains terms other than refl.

Propositional equality

The introduction of $=_A$ arised a problem: what is its structure? How does $=_A$ behave as a set? The solution was to leave out sets, focusing on other mathematical theories.

Streicher considered *category theory*, interpreting equality types as ∞-*groupoids*. Thus, a term $p : a =_A b$ becomes a *morphism* between the objects a and b, $x : p =_{a=_A b} q$ is a morphism between p and q and so on.



Voevodsky used *homotopy theory*, seeing $p : a =_A b$ as a *path* between a and b.

Induction

An important feature of Martin-Löf type theory is the notion of *induction*, which generalises the one on natural numbers: to prove a statement on all naturals, it suffices to prove it for 0 and succ(n), provided that it holds for n.

The idea is that to prove a statement about all the terms of a certain type, it is enough to prove it for the canonical terms.

For example, in the case of $+$, to prove that a type C depending on $A+B$ is inhabited it is enough to show that we can construct a term of C from

- $i_1^B(a) : A+B$ and
- $i_2^A(b) : A+B$.

Propositional equality

This allows, for example, to prove that propositional equality is symmetric

Proposition 4

For every type A and every $a, b : A$ there is a function $(a =_A b) \to (b =_A a)$ mapping $p : a =_A b$ to $p^{-1} : b =_A a$ such that $\mathrm{refl}_a^{-1} \equiv \mathrm{refl}_a$ for each $a : A$.

Proof.

Given $a, b : A$ and $p : a =_A b$ we want to construct a term $p^{-1} : b =_A a$. By induction, it is enough to do this in the trivial case in which b is a and p is refl_a; hence, p^{-1} has to be of type $a =_A a$. Thus, we can just define refl_a^{-1} as refl_a, and the general case follows by induction. □

Propositional equality

Propositional equality is also transitive, and those operations are well behaved

Proposition 5

For every type A and every $a, b, c : A$ there is a function $(a =_A b) \to (b =_A c) \to (a =_A c)$ mapping $p : (a =_A b)$ and $q : (b =_A c)$ to $p \cdot q : (a =_A c)$ such that $\mathrm{refl}_a \cdot \mathrm{refl}_a \equiv \mathrm{refl}_a$ for each $a : A$.

Proposition 6

Let A be a type, $a, b, c, d : A$, $p : (a =_A b)$, $q : (b =_A c)$ and $r : (c =_A d)$. Then

- $p = p \cdot \mathrm{refl}_b$ *and* $p = \mathrm{refl}_a \cdot p$;
- $p^{-1} \cdot p = \mathrm{refl}_b$ *and* $p \cdot p^{-1} = \mathrm{refl}_a$;
- $(p^{-1})^{-1} = p$;
- $p \cdot (q \cdot r) = (p \cdot q) \cdot r$.

Propositional equality

Functions $f : A \to B$ behave functorially on paths, i.e., they respect equality, or preserve paths; from a topological point of view, we say that they are "continuous".

Lemma 7

If $f : A \to B$ and $x, y : A$, then there is an operation

$$\mathrm{ap}_f : (x =_A y) \to (f(x) =_B f(y))$$

such that, for each $x : A$, $\mathrm{ap}_f(\mathrm{refl}_x) \equiv \mathrm{refl}_{f(x)}$. Sometimes, abusing notation, instead of $\mathrm{ap}_f(p)$ we will write $f(p)$.

Proof.

By induction, it suffices to assume $p \equiv \mathrm{refl}_x$. In this case, can be defined $\mathrm{ap}_f(p) :\equiv \mathrm{refl}_{f(x)} : f(x) = f(x)$. □

Propositional equality

Since $\Pi_{x:A}B(x)$ is a generalization of $A \to B$, we are interested in proving a dependent version of Lemma 7. But, given $p : x = y$, $f(x) : B(x)$ and $f(y) : B(y)$ may be terms of different types, thus we can not construct the type $f(x) = f(y)$. The solution is to use the path p as a way to relate $B(x)$ and $B(y)$, as shown in the following

Lemma 8

Given a family of types $B : A \to \mathcal{U}_i$ and a path $p : x =_A y$, there is a function $p_* : B(x) \to B(y)$.

Proof.

By path induction it suffice to assume $p :\equiv \mathrm{refl}_x$. But, under this hypothesis, it is enough to take $(\mathrm{refl}_x)_* : B(x) \to B(x)$ as the identity function. □

The function p_* is called transport function.

Propositional equality

This result allows us to prove a generalized version of Lemma 7.

Lemma 9

Let $f : \Pi_{x:A} P(x)$. *Then there is a map*

$$\mathsf{apd}_f : \Pi_{p:x=y}(p_*(f(x)) =_{P(y)} f(y)) \ .$$

Proof.

By induction, it is enough to assume that p is refl_x. In this case, the desired equation is $(\mathsf{refl}_x)_*(f(x)) = f(x)$, which holds judgementally. □

Introduction to Homotopy Type Theory

Part 2

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.y. 2020/2021

Syllabus

Homotopy type theory

- equivalence
- equality for type formers
- function extensionality
- univalence
- higher inductive types

Homotopies

Two important features of Homotopy type theory are the axioms of univalence and function extensionality.

Before introducing them we will focus on the interaction between propositional equality and the type formers. To reach this purpose, we need to introduce the notions of *homotopy* and *equivalence*.

Definition 10

Given $P : A \to \mathcal{U}_i$ and $f, g : \Pi_{x:A} P(x)$, a *homotopy* from f to g is a term of type $(f \sim g) :\equiv \Pi_{x:A}(f(x) = g(x))$.

Equivalence

Informally, given $f : A \to B$, we call $\mathrm{isequiv}(f)$ a type denoting that f is an equivalence. This notion is subtle, and will be studied in details later.

The most intuitive notion of equivalence, mimicking isomorphism in categories, may be

$$\mathrm{qinv}(f) :\equiv \Sigma_{g:B\to A}\left((f \circ g \sim \mathrm{id}_B) \times (g \circ f \sim \mathrm{id}_A)\right) \quad ;$$

when $\mathrm{qinv}(f)$ is inhabited, we say that g is a *quasi-inverse* of f.

Equivalences

The notion of quasi-inverse is unsatisfactory; thus, by now we consider a more general notion of equivalence, i.e.,

$$\mathsf{isequiv}(f) :\equiv (\Sigma_{g:B\to A}(f\circ g \sim \mathrm{id}_B)) \times (\Sigma_{h:B\to A}(h\circ f \sim \mathrm{id}_A)) \quad .$$

Of course, if f has a quasi-inverse then it is an equivalence, i.e., $\mathsf{qinv}(f)$ implies $\mathsf{isequiv}(f)$.

Definition 11

Given $A, B : \mathscr{U}$, denote as $A \simeq B$ the type $\Sigma_{f:A\to B}\mathsf{isequiv}(f)$.

Equality for type formers

For many types A, the equality $x =_A y$ can be characterized, up to equivalence, in terms of the data used to construct x and y.

We will see that it holds for product, dependent pairs and unit type, and generalise the property to dependent product and universes through the function extensionality and univalence axioms.
Equality for coproducts and natural numbers can be characterized in a different way, using a coding function.

Equality for type formers

If $x, y : A \times B$ and $p : x =_{A \times B} y$, by Lemma 7 can be obtained the paths $\mathsf{pr}_1(p) :\equiv \mathsf{ap}_{pr_1}(p) : \mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)$ and $\mathsf{pr}_2(p) :\equiv \mathsf{ap}_{pr_2}(p) : \mathsf{pr}_2(x) =_A \mathsf{pr}_2(y)$. Thus, there is a function

$$(x =_{A \times B} y) \to (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)) \ . \qquad (1)$$

Theorem 12

For any $x, y : A \times B$, the function (1) is an equivalence.

Equality for type formers

Proof. (i)

In this case, we can prove a stronger version of isequiv, i.e., that (1) has a quasi-inverse

$$(\mathrm{pr}_1(x) =_A \mathrm{pr}_1(y)) \times (\mathrm{pr}_2(x) =_B \mathrm{pr}_2(y)) \to (x =_{A\times B} y) \quad . \qquad (2)$$

By the induction rule for product, we can assume that $x \equiv (a, b)$ and $y \equiv (a', b')$ for some $a, a' : A$ and $b, b' : B$, i.e., that both x and y are pairs. Thus, we need to construct a function $(a =_A a') \times (b =_B b') \to ((a, b) =_{A\times B} (a', b'))$. By induction on the product we may assume given $p : a =_A a'$ and $q : b =_B b'$, and by two path inductions we may assume $a \equiv a'$, $b \equiv b'$, $p \equiv \mathrm{refl}_a$ and $q \equiv \mathrm{refl}_b$; thus, $(a, b) \equiv (a', b')$ and we can take the required function as $\mathrm{refl}_{(a,b)}$. It remains to prove that it is a quasi-inverse of (1). ↪

Equality for type formers

↪ Proof. (ii)

Let $r : x =_{A \times B} y$. By path induction on r assume that $x \equiv y$ and $r \equiv \mathsf{refl}_x$. Thus, by definition of $\mathsf{ap}_{\mathsf{pr}_1}$ and $\mathsf{ap}_{\mathsf{pr}_2}$, $r \equiv \mathsf{refl}_x$ is mapped by (1) to the pair $\left(\mathsf{refl}_{\mathsf{pr}_1(x)}, \mathsf{refl}_{\mathsf{pr}_2(x)}\right)$. By induction on x, we may assume $x \equiv (a, b)$, thus $\left(\mathsf{refl}_{\mathsf{pr}_1(x)}, \mathsf{refl}_{\mathsf{pr}_2(x)}\right) \equiv (\mathsf{refl}_a, \mathsf{refl}_b)$, which is mapped by (2) to $\mathsf{refl}_{(a,b)} \equiv r$.
Conversely, let $s : (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y))$. By induction on x and y we assume that $x \equiv (a, b)$ and $y \equiv (a', b')$; then, by induction on $s : (a =_A a') \times (b =_B b')$, assume $s \equiv (p, q)$ with $p : a = a'$ and $q : b = b'$. Finally, by induction on p and q, assume $p \equiv \mathsf{refl}_a$ and $q \equiv \mathsf{refl}_b$. In this case, (2) maps $s \equiv (\mathsf{refl}_a, \mathsf{refl}_b)$ to $\mathsf{refl}_{(a,b)}$, which is sent back by (1) to $(\mathsf{refl}_a, \mathsf{refl}_b) \equiv s$. □

Equality for type formers

We have shown in particular that (1) has an inverse (2), which we denote as

$$\mathsf{pair}^{=} : (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)) \to (x =_{A \times B} y) \ .$$

Notice that, given $p : \mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)$ and $q : \mathsf{pr}_2(x) =_A \mathsf{pr}_2(y)$,

$$\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{pair}^{=}(p, q)) = p$$
$$\mathsf{ap}_{\mathsf{pr}_2}(\mathsf{pair}^{=}(p, q)) = q$$

and, given $r : x =_{A \times B} y$, $r = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(r), \mathsf{ap}_{\mathsf{pr}_2}(r))$.

A special case of Theorem 12 yields to $z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$ with $z : A \times B$, i.e., the propositional uniqueness principle for products.

Equality for type formers

Generalising to dependent pairs, if $p : w =_{\Sigma_{x:A} P(x)} w'$, then $\mathrm{pr}_1(p) : \mathrm{pr}_1(w) =_A \mathrm{pr}_1(w')$, but in general $\mathrm{pr}_2(w)$ and $\mathrm{pr}_2(w')$ are not terms of the same type. Hence, a similar result can be obtained using transport

Theorem 13

Let $P : A \to \mathcal{U}$, and $w, w' : \Sigma_{x:A} P(x)$. Then there is an equivalence

$$(w = w') \simeq \Sigma_{(p:\mathrm{pr}_1(w) = \mathrm{pr}_1(w'))} p_*(\mathrm{pr}_2(w)) = \mathrm{pr}_2(w') \quad .$$

As before, a propositional uniqueness principle for dependent pair types can be obtained as a particular case: if $z : \Sigma_{x:A} P(x)$, then $z = (\mathrm{pr}_1(z), \mathrm{pr}_2(z))$.

Equality for type formers

An analogous result holds for the unit type, justifying the fact that $\mathbf{1}$ can be considered a nullary product.

Theorem 14

If $x, y : \mathbf{1}$, $(x =_{\mathbf{1}} y) \simeq \mathbf{1}$. *As a consequence,* $x, y : \mathbf{1}$ *implies* $x =_{\mathbf{1}} y$.

Proof.

We can trivially define a function $(x = y) \to \mathbf{1}$ sending all the terms of $x = y$ to $*$. Conversely, if $x, y : \mathbf{1}$, by induction we can assume that $x \equiv * \equiv y$; thus $\mathrm{refl}_* : x = y$ yields a constant function $\mathbf{1} \to (x = y)$. It remains to show that the two functions are quasi-inverses.
Let $u : \mathbf{1}$; by induction we assume $u \equiv *$, which is also the result of the composite $\mathbf{1} \to (x = y) \to \mathbf{1}$. In the other direction, let $p : x = y$; by path induction, we can suppose $x \equiv y$ and $p \equiv \mathrm{refl}_x$, and by induction on $\mathbf{1}$ we assume $x \equiv *$. Thus, the composite $(x = y) \to \mathbf{1} \to (x = y)$ maps p to $\mathrm{refl}_x \equiv p$. □

Function extensionality

After those results we would naturally expect that, if A is a type, $B : A \to \mathcal{U}$, and $f, g : \Pi_{x:A} B(x)$, then

$$(f = g) \simeq \left(\Pi_{x:A} \left(f(x) =_{B(x)} g(x)\right)\right) \quad .$$

Indeed, we can only prove by path induction that there is a function

$$\mathsf{happly} : (f = g) \to \left(\Pi_{x:A} \left(f(x) =_{B(x)} g(x)\right)\right) \quad . \tag{3}$$

For the converse, we need to assume

Axiom 15 (Function extensionality)

If A is a type, $B : A \to \mathcal{U}$ and $f, g : \Pi_{x:A} B(x)$, then the function (3) is an equivalence.

Univalence

A similar generalisation can be done for universes, relating the equality type $A =_{\mathcal{U}} B$ to the homotopy type $A \simeq B$.

Lemma 16

For each $A, B : \mathcal{U}$ there is a function

$$\mathrm{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B) \ . \tag{4}$$

Proof.

Seeing the identity function $\mathrm{id}_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}$ as a type family indexed by $\mathcal{U}$, we can associate to a path $p : A =_{\mathcal{U}} B$ a transport function $p_* : A \to B$. We need to prove that p_* is an equivalence.
By induction we assume that $p \equiv \mathrm{refl}_A$, which implies $p_* \equiv \mathrm{id}_A$, and $\mathrm{id}_A : A \to A$ is an equivalence with quasi-inverse id_A itself. Thus p_* is an equivalence, and we can define $\mathrm{idtoeqv}(p)$ as p_*. □

Univalence

The converse can not be proved by Martin-Löf type theory enriched with the function extensionality Axiom 15. Thus, we need

Axiom 17 (Univalence)

For any $A, B : \mathcal{U}$, the function (4) is an equivalence.

When a universe satisfies the univalence axiom, it is said to be *univalent*.

A proper choice of the notion of equivalence is fundamental here; indeed, taking $\mathrm{isequiv}(f) :\equiv \mathrm{qinv}(f)$ in the definition of $A \simeq B$ allows to construct an inhabitant of $\mathbf{0}$, i.e., it leads to inconsistency.

Equaliy for type formers

Coproducts and natural numbers have a slightly different characterization of equality.

For coproducts, we would expect that $\left(i_1^B(a_1) = i_1^B(a_2)\right) \simeq (a_1 = a_2)$, $\left(i_2^A(b_1) = i_2^A(b_2)\right) \simeq (b_1 = b_2)$ and $\left(i_1^B(a) = i_2^A(b)\right) \simeq \mathbf{0}$. Indeed,

Theorem 18

Let $a_0 : A$, *and define* $\mathrm{code} : A + B \to \mathcal{U}$ *by*

$$\mathrm{code}\left(i_1^B(a)\right) :\equiv (a_0 = a) \ ,$$
$$\mathrm{code}\left(i_2^A(b)\right) :\equiv \mathbf{0} \ .$$

Then, for every $x : A + B$, $\left(i_1^B(a_0) = x\right) \simeq \mathrm{code}(x)$.

An analogous result can be obtained fixing $b_0 : B$ instead of $a_0 : A$.

Equaliy for type formers

Something similar holds for natural numbers:

Theorem 19

Define $\mathsf{code} : \mathbb{N} \to \mathbb{N} \to \mathscr{U}$ *as*

$$\begin{aligned}
&\mathsf{code}(0,0) :\equiv \mathbf{1} \ , \\
&\mathsf{code}(\mathsf{succ}(m),0) :\equiv \mathbf{0} \ , \\
&\mathsf{code}(0,\mathsf{succ}(n)) :\equiv \mathbf{0} \ , \\
&\mathsf{code}(\mathsf{succ}(m),\mathsf{succ}(n)) :\equiv \mathsf{code}(m,n) \ .
\end{aligned}$$

Then, for every $m, n : \mathbb{N}$, $(m = n) \simeq \mathsf{code}(m,n)$.

Higher inductive types

Homotopy type theory, or univalent type theory, consists in Martin-Löf type theory enriched with the axioms of univalence and function extensionality, and the higher inductive types.

A *higher inductive type* A is a type of which are specified not only the canonical terms $a : A$, but also the non-trivial terms of the corresponding equality type $=_A$, and of the equality type $=_{=_A}$, and so on.

Those new structures allow to intuitively represent a lot of mathematical objects.

Higher inductive types - the circle

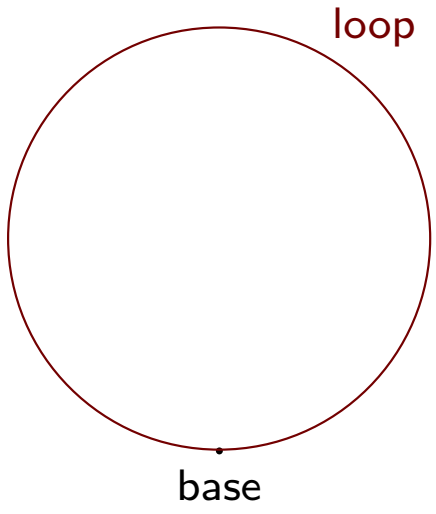
An example of higher inductive type is the circle $\mathbb{S}^1$.

It is constructed as the type such that

- $\mathsf{base} : \mathbb{S}^1$;
- $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$.

Those terms can be seen as a point, base, and a path from base to base, called loop, distinguished from $\mathsf{refl}_{\mathsf{base}}$.

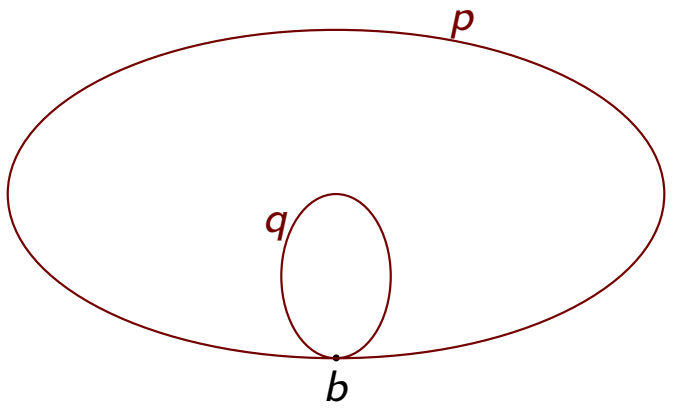
As we have seen loop^{-1}, loop^3 etc can be obtained by inverting and composing loop.



Higher inductive types - the torus

Similarly, the torus T^2 is generated by

- $b : T^2$;
- $p : b =_{T^2} b$;
- $q : b =_{T^2} b$;
- $t : p \cdot q =_{(b =_{T^2} b)} q \cdot p$.



Hence there is a point b, two paths from b to b and a rule stating that their composition is commutative.

Higher inductive types - pushouts

Also categorical structures can be represented using the higher inductive types.
Let $f : C \to A$ and $g : C \to B$. Their pushout $A \sqcup^C B$ can be defined as

- $\mathrm{inl} : A \to A \sqcup^C B$;
- $\mathrm{inr} : B \to A \sqcup^C B$;
- for each $c : C$, $\mathrm{glue}(c) : (\mathrm{inl}(f(c)) = \mathrm{inr}(g(c)))$.

Hence, $A \sqcup^C B$ is the disjoint union of A and B with the additional requirement that $f(c)$ and $g(c)$ are equal for every $c : C$.

Introduction to Homotopy Type Theory

Part 3

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.y. 2020/2021

Syllabus

Classical logic

- sets
- mere propositions
- truncation
- axiom of choice

Sets

To formalise Mathematics inside type theory, it is useful to introduce an internal notion of set. The idea is that a type is a set when it has no higher homotopical information: all the paths between the same two terms are equal.

Definition 20

A type A is a *set* if for all $x, y : A$ and $p, q : x = y$ holds $p = q$. Thus,

$$\mathsf{isSet}(A) :\equiv \Pi_{(x,y:A)} \Pi_{(p,q:x=y)} (p = q) \ .$$

With this definition, sets behave as in category theory: their elements are abstract points whose structure is ruled by functions and relations.

Sets

Example 21

The type $\mathbf{1}$ is a set. Indeed, by Theorem 14, if $x, y : \mathbf{1}$, $(x =_{\mathbf{1}} y) \simeq \mathbf{1}$. Hence, since any two elements of $\mathbf{1}$ are equal, this holds also for the elements of $x =_{\mathbf{1}} y$.

Example 22

The type $\mathbf{0}$ trivially is a set.

Example 23

The type $\mathbb{N}$ is a set. Indeed, by Theorem 19 all equality types $x =_{\mathbb{N}} y$ are equivalent to $\mathbf{0}$ or $\mathbf{1}$, and any two inhabitants of $\mathbf{0}$ and $\mathbf{1}$ are equal.

Sets

Example 24

If A and B are sets, then $A \times B$ is. Let $x, y : A \times B$, and $p, q : x = y$. Then by Theorem 12 we have $p = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(p), \mathsf{ap}_{\mathsf{pr}_2}(p))$ and $q = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(q), \mathsf{ap}_{\mathsf{pr}_2}(q))$. But, since A is a set, $\mathsf{ap}_{\mathsf{pr}_1}(p) = \mathsf{ap}_{\mathsf{pr}_1}(q)$, and since B is a set $\mathsf{ap}_{\mathsf{pr}_2}(p) = \mathsf{ap}_{\mathsf{pr}_2}(q)$. Hence $p = q$.

Example 25

It can be shown that:

- if $B : A \to \mathscr{U}$ is such that $B(x)$ is a set for each $x : A$, then $\Pi_{x:A} B(x)$ is a set;
- if A is a set and $B : A \to \mathscr{U}$ such that $B(x)$ is a set for each $x : A$, then $\Sigma_{x:A} B(x)$ is a set.

Sets

As a counterexample, we prove that $\mathcal{U}_i$ is not a set. To do so, we need to introduce the type $\mathbf{2} :\equiv \mathbf{1} + \mathbf{1}$ with two elements, $0_\mathbf{2} : \mathbf{2}$ and $1_\mathbf{2} : \mathbf{2}$, such that $0_\mathbf{2} \neq 1_\mathbf{2}$.

Proposition 26

The universes $\mathcal{U}_i$ are not sets.

Proof.

To prove this result, we need to show a type $A : \mathcal{U}_i$ and a path $p : A =_{\mathcal{U}_i} A$ which is not equal to refl_A. Take $A \equiv \mathbf{2}$, and define $f : A \to A$ as $f(0_\mathbf{2}) :\equiv 1_\mathbf{2}$ and $f(1_\mathbf{2}) :\equiv 0_\mathbf{2}$. Then $f(f(x)) = x$ for all $x : \mathbf{2}$ by case analysis, so f is an equivalence. Hence, by univalence, f originates a path $p : A = A$. But p must be different from refl_A. Indeed, if they were equal, then by univalence f would be the identity function of A, which would imply $0_\mathbf{2} = 1_\mathbf{2}$. □

Classical logic

Through the propositions as types correspondence, a statement as "there exists $x : A$ such that $B(x)$" is interpreted as a Σ type $\Sigma_{x:A} B(x)$, i.e., to prove this statement is needed an *evidence* of the element x such that $B(x)$ is provable. The same holds, as we have seen, for disjunction.
Hence, types can contain more information than simply truth or falsity: a term of a type is an algorithm allowing to construct an evidence of the truth of the proposition represented by the type.

This is not what happens in classical logic, where it is possible to prove the existence of an object without being able to construct it. This is done, for example, using the law of double negation.

Classical logic

It is provable that the law of double negation and the law of excluded middle are incompatible with univalence. Define $\neg A :\equiv (A \to \mathbf{0})$.

Theorem 27

It is not the case that for all $A : \mathscr{U}_i$ we have $\neg(\neg A) \to A$.

Theorem 28

It is not the case that for all $A : \mathscr{U}_i$ we have $A + (\neg A)$.

Despite the technicalities, those results are proved taking $A \equiv \mathbf{2}$, as already done for Proposition 26. Notice that the type $\mathbf{2}$ has two distinct terms.

Mere propositions

To obtain classical logic, we can restrict the attention to the types which do not contain more information than a truth value. This allows to exclude types like $\mathbf{2}$.

Definition 29

We say that a type P is a *mere proposition* if $x =_P y$ for all $x, y : P$. In our formalism,

$$\mathsf{isProp}(P) :\equiv \Pi_{x,y:P}(x =_P y) \ .$$

Mere propositions

Proposition 30

If P and Q are mere propositions such that there are $f : P \to Q$ and $g : Q \to P$, then $P \simeq Q$.

Proof.

Since P is a mere proposition, for any $x : P$ we have $g(f(x)) = x$. Similarly, for any $y : Q$ it holds that $f(g(y)) = y$. Thus, f and g are quasi-inverses. □

Proposition 31

Let P be a mere proposition such that $x_0 : P$. Then $P \simeq \mathbf{1}$.

Proof.

Let $f : P \to \mathbf{1}$ be such that $f(x) :\equiv *$, and $g : \mathbf{1} \to P$ such that $g(u) :\equiv x_0$. Then, the claim follows from the previous proposition and the fact that $\mathbf{1}$ is a mere proposition, as seen before. □

Mere propositions

Thus, the laws of excluded middle and double negation can be redefined starting from mere propositions:

$$\Pi_{A:\mathcal{U}}(\mathsf{isProp}(A) \to (A + \neg A))$$

$$\Pi_{A:\mathcal{U}}(\mathsf{isProp}(A) \to (\neg\neg A \to A))$$

With this reformulation, the above laws are independent from univalent type theory. Thus, they can safely be assumed as axioms.

Mere propositions

Example 32

If A and B are mere propositions, then $A \times B$ is a mere proposition. Indeed, let $x, y : A \times B$. Then $x = (\mathrm{pr}_1(x), \mathrm{pr}_2(x))$ and $y = (\mathrm{pr}_1(y), \mathrm{pr}_2(y))$. Since A and B are mere propositions, $\mathrm{pr}_1(x) = \mathrm{pr}_1(y)$ and $\mathrm{pr}_2(x) = \mathrm{pr}_2(y)$. Hence, $x = y$.

Example 33

If $B : A \to \mathscr{U}$ is such that $B(x)$ is a mere proposition for every $x : A$, then $\Pi_{x:A} B(x)$ is a mere proposition. Indeed, given $f, g : \Pi_{x:A} B(x)$, since $B(x)$ is a mere proposition we have that $f(x) = g(x)$. Hence, by function extensionality, $f = g$.
In particular, if B is a mere proposition, so is $A \to B$. Which implies that, since $\mathbf{0}$ is a mere proposition, so is $\neg A \equiv (A \to \mathbf{0})$.

Mere propositions

Not all type formers preserve mere propositions. Indeed, $A + B$ is not in general a mere proposition, even if A and B are. See, for example, $\mathbf{2} = \mathbf{1} + \mathbf{1}$. Indeed, $A + B$ is a constructive sort of "or", whose terms indicate which is the true disjunct.
The same holds for $\Sigma_{x:A} B(x)$. This is a constructive interpretation of the existential quantifier, which remembers the witness x which makes $B(x)$ true.

It is possible to truncate those type constructors, to obtain more classical sorts of or and exists. This is done using the higher inductive types.

Truncation

A generic type A can be "truncated" to be a mere proposition. Thus, given a type A, define a type $||A||$, which is called its *truncation*, such that

- for any $a : A$ there is $|a| : ||A||$;
- for any $x, y : ||A||$, $x =_{||A||} y$.

Hence, if A is inhabited, so is $||A||$; moreover, $||A||$ is a mere proposition.

It is also required that, if B is a mere proposition and $f : A \to B$, then there is $g : ||A|| \to B$ such that $g(|a|) \equiv f(a)$ for all $a : A$.

This means that a truncated type $||A||$ does not contain more information than the fact that A is inhabited, i.e., that it is true.

Truncation

Thus, the propositions as types correspondence between mere propositions and classical logic becomes

mere propositions (types)	(classical) propositions
$\mathbf{1}$	$\top$
$\mathbf{0}$	$\bot$
$A \times B$	$A \wedge B$
$\|\|A + B\|\|$	$A \vee B$
$A \to B$	$A \supset B$
$\Pi_{x:A} B(x)$	$\forall x \in A. B(x)$
$\|\|\Sigma_{x:A} B(x)\|\|$	$\exists x \in A. B(x)$

Axiom of choice

The statement "if for all $x : X$ there exists an $a : A(x)$ such that $P(x, a)$, then there exists a function $g : \Pi_{x:A} A(x)$ such that for all $x : X$ we have $P(x, g(x))$" is the axiom of choice.

In the canonical propositions-as-types interpretation, in which $\exists x \in A.B(x)$ corresponds to $\Sigma_{x:A} B(x)$, it is always true; moreover, its antecedent is equivalent to the conclusion.

Theorem 34

Let $X : \mathcal{U}$, $A : X \to \mathcal{U}$ and $P : \Pi_{x:X} A(x) \to \mathcal{U}$. Then there is a function

$$\left(\Pi_{x:X} \Sigma_{a:A(x)} P(x, a)\right) \to \left(\Sigma_{g:\Pi_{x:X} A(x)} \Pi_{x:X} P(x, g(x))\right) \tag{5}$$

mapping f into $(\lambda x.\mathrm{pr}_1(f(x)), \lambda x.\mathrm{pr}_2(f(x)))$. This function is, indeed, an equivalence.

Axiom of choice

Proof.

Using the induction principle for Σ-types to reduce to the case of a pair, we define a quasi-inverse by sending (g, h) to $\lambda x.(g(x), h(x))$. Given $f : \Pi_{x:X} \Sigma_{a:A(x)} P(x, a)$, composing we obtain the function

$$\lambda x.(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) \ . \tag{6}$$

But $(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) = f(x)$ and, by function extensionality, (6) is equal to f.
On the other hand, given (g, h), the composition yelds $(\lambda x.g(x), \lambda x.h(x))$, which is judgementally equal to (g, h). □

This is not what we would expect, because in Mathematics the axiom of choice is, indeed, an axiom. Notice that the function g is already specified from the beginning, and so there are no real choices to be made.

Axiom of choice

We can reformulate the axiom of choice in type theory using truncation. Consider $X : \mathcal{U}$, $A : X \to \mathcal{U}$, $P : \Pi_{x:X} A(x) \to \mathcal{U}$ such that X is a set, $A(x)$ is a set for all $x : X$ and $P(x, a)$ is a mere proposition for all $x : X$ and $a : A(x)$. Then the axiom of choice asserts that

$$\left(\Pi_{x:X} \left\| \Sigma_{a:A(x)} P(x, a) \right\|\right) \to \left\| \Sigma_{g:\Pi_{x:X} A(x)} \Pi_{x:X} P(x, g(x)) \right\| \quad .$$

It can be rephrased again as "if for all $x : X$ there exists an $a : A(x)$ such that $P(x, a)$, then there exists a function $g : \Pi_{x:A} A(x)$ such that for all $x : X$ we have $P(x, g(x))$", but here existence is interpreted in the classical sense.
Notice that the second truncation means that the function g is not determined or specified in any known way.

This formulation of the axiom of choice is independent from univalent type theory, so it can be consistently assumed as an axiom.

Introduction to Homotopy Type Theory

Part 4

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.y. 2020/2021

Syllabus

Notion of equivalence

- contractibility
- definitions of equivalence

Contractibility

All the terms of a mere proposition P are equal, but there can be no terms in P; this does not happen with contractible types

Definition 35

A type A is said to be *contractible* if there is $a : A$, called the *center of contraction*, such that $a =_A x$ for every $x : A$. In type theory,

$$\mathsf{isContr}(A) :\equiv \Sigma_{a:A} \Pi_{x:A} (a =_A x) \ .$$

Contractibility

The following lemmas characterize the notion of contractible type. Although we are not going to prove them, they will be useful to prove some other results we are interested in.

Lemma 36

Given a type A, the following are logically equivalent:

1. *A is contractible;*
2. *A is a mere proposition, and there is a point $a : A$;*
3. *A is equivalent to $\mathbf{1}$.*

Contractibility

Lemma 37

Let A be a type, and $a : A$. Then the type $\Sigma_{x:A}(a = x)$ is contractible.

Lemma 38

If $P : A \to \mathcal{U}_i$,

- *if $P(x)$ is contractible for every $x : A$, then $\Sigma_{x:A} P(x)$ is equivalent to A;*
- *if A is contractible with center a, then $\Sigma_{x:A} P(x)$ is equivalent to $P(a)$.*

Contractibility

The notion of contraction, following the Category Theory style, can be generalized to functions $f : A \to B$ as follows

Definition 39

Given a map $f : A \to B$ and a point $y : B$, the *fiber* of f over y is

$$\mathrm{fib}_f(y) :\equiv \Sigma_{x:A}(f(x) = y) \quad .$$

Definition 40

A function $f : A \to B$ is said to be *contractible* if the fiber $\mathrm{fib}_f(y)$ is contractible for each $y : B$, i.e.,

$$\mathrm{isContr}(f) :\equiv \Pi_{y:B}\mathrm{isContr}(\mathrm{fib}_f(y)) \quad .$$

Equivalence

Now we are ready to discuss the definition of equivalence. As anticipated, the more intuitive notion is the one of qinv. Thus, we want the type isequiv(f) to satisfy the following

1. qinv(f) $\to$ isequiv(f);
2. isequiv(f) $\to$ qinv(f);
3. isequiv(f) is a mere proposition.

Of course qinv(f) satisfies (1) and (2), but it may not be a mere proposition.

Equivalence

Lemma 41

Let $f : A \to B$ *such that* $\mathrm{qinv}(f)$ *is inhabited. Then*

$$\mathrm{qinv}(f) \simeq (\Pi_{x:A}(x = x)) \quad .$$

Proof. (i)

Since $\mathrm{qinv}(f)$ is inhabited by hypothesis, there is $e : \mathrm{isequiv}(f)$ and so $(f, e) : A \simeq B$. By univalence $\mathrm{idtoeqv} : (A = B) \to (A \simeq B)$ is an equivalence, thus (f, e) can be written as $\mathrm{idtoeqv}(p)$ for some $p : A = B$. By path induction assume $p \equiv \mathrm{refl}_a$, which implies $f \equiv \mathrm{id}_A$. Thus, we only need to prove $\mathrm{qinv}(\mathrm{id}_A) \simeq (\Pi_{x:A}(x = x))$. ↪

Equivalence

↪ Proof. (ii)

By definition $\mathrm{qinv}(\mathrm{id}_A) \equiv \Sigma_{g:A\to A}((g \sim \mathrm{id}_A) \times (g \sim \mathrm{id}_A))$, which is equivalent to $\Sigma_{g:A\to A}((g = \mathrm{id}_A) \times (g = \mathrm{id}_A))$ by function extensionality, and can be shown to be equivalent to $\Sigma_{h:\Sigma_{g:A\to A}(g=\mathrm{id}_A)}(\mathrm{pr}_1(h) = \mathrm{id}_A)$. By Lemma 37 $\Sigma_{(g:A\to A)}(g = \mathrm{id}_A)$ is contractible with center $(\mathrm{id}_A, \mathrm{refl}_{\mathrm{id}_A})$, and this type is equivalent to $\mathrm{id}_A = \mathrm{id}_A$ by Lemma 38. Finally, by function extensionality, $\mathrm{id}_A = \mathrm{id}_A$ is equivalent to $\Pi_{x:A} x = x$. □

In the HoTT book it is claimed that the above Theorem can be proved also avoiding univalence.

Equivalence

Lemma 42

Let $P : A \to \mathcal{U}$ be such that $P(x)$ is a mere proposition for every $x : A$, and $u, v : \Sigma_{(x:A)} P(x)$. If $\mathrm{pr}_1(u) = \mathrm{pr}_1(v)$, then $u = v$.

Lemma 43

Let A be a type and $a : A$, $q : a = a$ such that

(i) *the type $a =_A a$ is a set;*

(ii) *for all $x : A$ holds $\|a = x\|$;*

(iii) *for all $p : a = a$ holds $p \cdot q = q \cdot p$.*

Then there is $f : \Pi_{x:A}(x = x)$ such that $f(a) = q$.

Equivalence

Theorem 44

There are two types A, B and $f : A \to B$ such that $\mathrm{qinv}(f)$ *is not a mere proposition.*

Proof. (i)

By Lemma 41, it is enough to find a type A such that $\Pi_{x:A}(x = x)$ is not a mere proposition. Then, since $\mathrm{qinv}(\mathrm{id}_A)$ is inhabited, it is not a mere proposition. ↪

Equivalence

↪ Proof. (ii)

Define $A :\equiv \Sigma_{X:\mathcal{U}} \| \mathbf{2} = X \|$; we want to find $g : \Pi_{x:A}(x = x)$ which is not equal to $\lambda x.\mathrm{refl}_x$. Let $a :\equiv (\mathbf{2}, |\mathrm{refl}_{\mathbf{2}}|) : A$, and notice that, by univalence and Lemma 42, $(a = a) \simeq (\mathbf{2} = \mathbf{2}) \simeq (\mathbf{2} \simeq \mathbf{2})$. Let $q : a = a$ be the path corresponding to $e : \mathbf{2} \simeq \mathbf{2}$ such that $e(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}$ and $e(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}}$. Since $(a = a) \simeq (\mathbf{2} \simeq \mathbf{2})$, which is a set, (i) of Lemma 43 holds. It can also be shown that (ii) and (iii) hold, thus there is $g : \Pi_{x:A}(x = x)$ such that $g(a) = q$. But e is not equal to id_a, thus q is not equal to refl_a, and g is not equal to $\lambda x.\mathrm{refl}_x$. Hence, $\Pi_{x:A}(x = x)$ is not a mere proposition. □

Equivalence

Thus, we need a different definition for isequiv(f). Indeed, as stated in Theorem 47, we can find three notions of equality which satisfy the conditions (1), (2) and (3), and which are equivalent.

Definition 45

We say that a function $f : A \to B$ is a *half adjoint equivalence* if there are a function $g : B \to A$ and two homotopies $\eta : g \circ f \sim \mathrm{id}_A$ and $\varepsilon : f \circ g \sim \mathrm{id}_B$ such that there is a homotopy $\tau : \Pi_{x:A} f(\eta x) = \varepsilon(fx)$. Thus, we define

$$\mathrm{ishae}(f) :\equiv \Sigma_{(g:B\to A)} \Sigma_{(\eta:g\circ f\sim \mathrm{id}_A)} \Sigma_{(\varepsilon:f\circ g\sim \mathrm{id}_B)} \Pi_{x:A} \left(f(\eta x) = \varepsilon(fx)\right) \quad .$$

Equivalence

Definition 46

For each $f : A \to B$, define the types of *left inverses* and *right inverses* to f

$$\mathsf{linv}(f) :\equiv \Sigma_{g:B\to A}(g \circ f \sim \mathsf{id}_A) \qquad \mathsf{rinv}(f) :\equiv \Sigma_{g:B\to A}(f \circ g \sim \mathsf{id}_B) \ ;$$

we say that f is *bi-invertible* if it has both a left inverse and a right inverse, i.e.,

$$\mathsf{biinv}(f) :\equiv \mathsf{linv}(f) \times \mathsf{rinv}(f) \ .$$

Bi-invertibility is the notion of equivalence we used previously.

Equivalence

Theorem 47

Given $f : A \to B$, *the types* $\mathsf{ishae}(f)$, $\mathsf{biinv}(f)$ *and* $\mathsf{isContr}(f)$ *satisfy the conditions (1), (2) and (3) above. Moreover* $\mathsf{ishae}(f) \simeq \mathsf{biinv}(f) \simeq \mathsf{isContr}(f)$, *thus they can be indifferently used to define* $\mathsf{isequiv}(f)$.

Proof.

We show that $\mathsf{qinv}(f) \to \mathsf{biinv}(f)$ and $\mathsf{biinv}(f) \to \mathsf{qinv}(f)$. The rest of the proof can be found in the HoTT book; we stress that, to show that $\mathsf{biinv}(f)$, $\mathsf{ishae}(f)$ and $\mathsf{isContr}(f)$ are mere propositions, is used the function extensionality axiom.
If $(g, \alpha, \beta) : \mathsf{qinv}(f)$, it is trivially obtained $(g, \alpha, g, \beta) : \mathsf{biinv}(f)$.
Conversely, if $(g, \alpha, h, \beta) : \mathsf{biinv}(f)$, then $(g, \alpha, \beta') : \mathsf{qinv}(f)$ with $\gamma(x) :\equiv \beta(g(x)) \cdot h(\alpha(x))$ and $\beta' : g \circ f \sim \mathrm{id}_A$ defined by $\beta'(x) :\equiv \gamma(f(x)) \cdot \beta(x)$. □

Equivalence

When introducing the univalence axiom, it is fundamental to require that isequiv(f) is a mere proposition. Indeed, it is provable that

Theorem 48

For $A, B : \mathcal{U}$ define

$$\text{idtoqinv}_{A,B} : (A = B) \to \Sigma_{f:A\to B}\text{qinv}(f)$$

by path induction in the obvious way. Let qinv*-univalence denote the modified form of the univalence axiom which asserts that for all $A, B : \mathcal{U}$ the function* $\text{idtoqinv}_{A,B}$ *has a quasi-inverse.*
Then, qinv*-univalence is inconsistent.*