Equalities in Constructive Type Theory and Applications in Higher Order Methaphysics

Simone Kilian, ILLC, University of Amsterdam
Supervised by Pablo Rivas Robledo, ILLC, University of Amsterdam

August 29, 2024, updated October 8, 2024

Abstract

Connections between classical and impredicative simple type theory and higher order methaphysics have been studied. We will investigate which kinds of connections there are between intuitionistic and predicative constructive type theory (CTT), i.e. Martin Löf type theory, and higher order methaphysics focusing on identity types and the role they play. After giving a brief introduction of our motivation based on Crosilla [1] and identities in CTT, we are discussing identity judgements and identity propositions under consideration of Klev [5]. Finally we are discussing connections of identities to philosophical debates concerning higher order methaphisics and the role the play in unrestricted quantification mainly based on the discussion in Florio and Jones [2].

1 Introduction

2 Comparison between STT and CTT

We follow Crosilla [1] as our starting point (see talk) to motivate further investigation of the topic. In parts, we incorporate Nördstrom and Petersson [7]. In contrast to STT, CTT implements the *propositions as types* paradigm under the Curry Howard Correspondence. Individuals are the propositions they correspond to.

2.1 Overview over basic ideas of CTT

If we make a judgements in CTT we have to satisfy two different kind of assumptions. We have to answer how an object of that type is defined and what it means for two objects to be identical.

There are four basic kinds of *judgements* which are independent of assumptions.

Firstly, we can make the judgement "A is a type", i.e. $\Gamma \vdash A\ type$ or $\Gamma \vdash A\ prop$, and apply a propositional reading "A is a proposition in context Γ". Note that we shall also make the judgement that Γ is a well-formed context which comes together with rules for its correct formation. Within the simple Curry-Howard-correspondence, it might be more convenient to directly state A or better φ is a type iff φ is a propositional formula.

Secondly, given some type A and and object a together with a semantic explanation of what it means to be an object of that type we can make the judgement "a is an element of type A", i.e. $a : A$ or $\Gamma \vdash a : A$. In the propositional way we can read this as "A is true (in context Γ)".

Definitional equality of types and terms is formed in different ways within Martin Löf type theory. We introduce one notion. Given two types A, B. If an arbitrary object of type A is also an object of type B and if two identical objects of type A are also identical objects of type B. Then types

A, B are equal, i.e. we shall judge "A and B are equal types" and denote $A = B$. As in the preceding judgement type we require that two types must form an equivalence relation, i.e.

$$\frac{A\ type}{A = A}\ (refl) \qquad \frac{A = B}{B = A}\ (sym) \qquad \frac{A = B \quad B = C}{A = C}\ (trans)$$

Given some semantic definition of what it means for two objects to be identical and of what a type is, if $a : A$ and $b : B$, and if a and b both satisfy the semantic explanation for equality of objects of A, then we can make the judgement "a and b are equal elements of type A - denoted with $a = b : A$ - or a and b are definitionally equivalent. Given the judgement $a = b : A$ we shall conclude that $Id(A, a, b)$ holds. The latter is a proposition that asserts that two inhabitants a, b of A are equal. The converse is not necessarily the case. Note that equality between two objects of a type must be an equivalence relation

$$\frac{a : A}{a = a : A}\ (refl) \qquad \frac{a = b : A}{b = a : A}\ (sym) \qquad \frac{a = b : A \quad b = c : A}{a = c : A}\ (trans)$$

Judgement forms of those kinds justify the rules

$$\frac{a : A \quad A = B : prop}{a : B}\ (conv_1) \qquad \frac{a = b : A \quad A = B : prop}{a = b : B}\ (conv_2)$$

These rules are called conversion rules. Such rules are non-standard in other proof systems for first order logic, for example, but can be added as well. The rule $conv_1$ states that if a is a proof (construction) of A and $A = B$, i.e. A and B are definitionally equal, then a is a proof for B as well. The rule $conv_2$ states that if a proof of a is definitionally equal to the proof b of A and A and B are definitionally equal, then a and b are also definitionally equal proofs for B.

To force that quantification over a type becomes meaningful, we need to specify what the inhabitants of a type are. Martin Löf type theory requests a definition of how we can construct typical (canonical) elements of a type and give them an identity criterion. There are four kinds of type introduction rules.

Type formation rules. Those rules state how to form a new type forcing predicative definitions. Consider the following example, where we defer between different readings, for example a propositional reading.

$$\frac{N\ type \qquad}{}\qquad \frac{N\ prop \qquad}{}$$
$$\frac{A\ type \quad B\ type}{A + B\ type} \qquad \frac{A\ prop \quad B\ prop}{A \vee B\ prop}$$

In the following N will denote the type of natural numbers.

Introduction rules. These kinds of rules define what it means to be a canonical element of this type via introducing canonical elements of this type and define when two canonical elements are equal. As an example consider the introduction rules for canonical elements of type N and the rules that define whether two canonical elements of type N are equal.

$$\frac{}{0 : N} \qquad \frac{n : N}{suc(n) : N}$$

define canonical elements of type N

$$\frac{}{0 = 0 : N} \qquad \frac{a = b : N}{suc(a) = suc(b) : N}$$

define when two canonical elemenst of type N are equal

These rules do not state whether two not necessarily canonical elements such as $plus(1, 2)$ and $suc(2)$ are equal. We see that the type N is inductively defined, which makes $N : Set$ (see 2.1.1). Not all sets can be defined inductively, for example type Set cannot. By introduction rules we also understand the introduction rules in a type theory as for example

$$\frac{a : A}{inl(a) : A + B}\ (\vee I) \qquad \frac{b : B}{inl(a) : A + B}$$

or

$$\frac{a : A \quad b : B}{\langle a, b\rangle : A \wedge B}\ (\wedge I)$$

.

Note that a proof in the sense of intuitionism does not mean that there is a derivation. Rather it means that a working mathematician should be able to construct a proof. This is in line with the interpretations within the Curry-Howard Correspondence and for example the BHK interpretation: A construction (proof) of for example $A+B$ consists of an indicator (pointer), e.g. $inl(a)$, together with a construction (proof) of A, i.e. a. We distinguish a proof from a derivation. The derivation in the formal system shows us how to construct a proof of $A + B$ or $A \wedge B$ while a proof as $\langle a, b\rangle$ or $inl(a)$ is more a witness. Consider the second case: if a is a proof (construction) of A and b is a proof of B, then one can construct a proof of $\langle a, b\rangle$ of $A \wedge B$. We might also denote $A \wedge B$ by $A \times B$. Note that the pairing function $\langle \cdot, \cdot \rangle$ is a primitive notion.

Elimination rules. Elimination rules define how to use elements of a certain type and how to define functions on a type introduced by the introduction rule or for example to define projections.

$$\frac{m : A \wedge B}{\pi_1(m) : A}\ (\wedge E_l) \qquad \frac{m : A \wedge B}{\pi_2(m) : B}\ (\wedge E_r)$$

Equality rules (Computation rules). These kinds of rules define how a function defined by the elimination rules acts on canonical elements of the specific type. In particular they explain how statements can be brought in a canonical form. To conclude that $plus(1, 2) = suc(2)$ we need to define what it means that $plus(suc(1), 1) = suc(plus(1, 1))$. Given that we defined addition as plus consider the following example

$$\frac{n : N}{plus(0, n) = n : N} \qquad \frac{m : N \quad n : N}{plus(suc(m), n) = suc(plus(m, n)) : N}$$

Note that the projections from above are not primitive and that they are defined by the pairing function by equality (also: computation) rules:

$$\frac{}{\pi_1(\langle a, b\rangle) = a : A}\ (\wedge eq_1) \qquad \frac{}{\pi_2(\langle a, b\rangle) = b : B}\ (\wedge eq_2)$$

Those rules correspond to detour elimination for $\wedge$ in natural deduction as

$$\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_2 \\ \dfrac{\dfrac{a : A \quad b : B}{\langle a, b\rangle : A \wedge B}\ (\wedge I)}{a : A}\ \wedge E_l \end{array} \quad \Rightarrow \quad \begin{array}{c} \mathcal{D}_1 \\ a : A \end{array}$$

It follows by the rules for judgemental equality introduced above that given $m : A \wedge B$, i.e. m is a proof of $A \wedge B$, there is a proof (construction) a' of A and a proof b' of B s.t. $\langle a', b'\rangle = m : A \wedge B$, implying that by $\pi_1(\langle a', b'\rangle) = a' : A$ and $\pi_1(\langle a', b'\rangle) = \pi_1(m) : A$ that $\pi_1(m) = a' : A$, and analoguously that by $\pi_2(\langle a', b'\rangle) = b' : B$ and $\pi_2(\langle a', b'\rangle) = \pi_2(m) : B$ it follows $\pi_2(m) = b' : B$.

2.1.1 The type *Set*

The type *Set* consists of inductively defined elements. Those elements are inductively defined sets like types of individuals as N. While for such types as N an inductive definition exists it is not possible to give an inductive definition of type *Set*. Hence, we call the type *Set open.*

Firstly, we need to know what it means for *Set* to be a type and secondly, what it means to be an element which is also a type itself. So, for the first we have to explain what a set is and what it means for two sets to be the same. In order to state that A is an object in *Set* we have to explain how to canonical elements - here, elements in constructor form - are formed and when two of them are equal. As usual two sets are equal if they consist of the same elements. If two equal elements are in one set they also have to be equal elements of the other set. This leads us to justify the following rules for $Set-$formation

$$Set\ type$$

and $El-$formation

$$\frac{A : Set}{El(A)\ type}$$

Once we have a set A, we may form a type $El(A)$. Objects of type $El(A)$ are elements of the set A.

We may write A instead of $El(A)$.

Via constants we introduce new mathematical objects like numbers, functions, pairs, but also inductive sets like sets for proof objects. We defer between two different notions.

Primitive constants. The value of a primitive constant is the constant itself: it has only a type and not a definition. Examples are N, $succ$, 0, $\langle\cdot,\cdot\rangle$. Instead of giving it meaning via the rules we give it meaning via the semantics of the theory.
We can introduce N, $succ$, and 0 via $N : Set$, $succ : N \to N$ and $0 : N$.

Defined constants. This type of constants is defined via other objects. If we apply such a *defined constant* to all its arguments we get an expression that computes in one step to its definiens.
We defer defined constants between an *explicitely defined constant* and an *implicitly defined constant*.
An *explicitely defined constant* is a constant c that we use as an abbreviation for some object a

$$c = a : A$$

Examples are

$$1 = succ(0) : N$$
$$I_N = \lambda x : N.x : N \to N$$

An *implicitly defined constant* is a constant that shows what definiens it has when it is applied to its constants, in particular, via pattern matching and recursive definitions. This means that one has to be sure that all well-typed expressions are a definiendum with a unique well-typed definiens.[7] Examples are addition

$$plus : N \to N \to N$$
$$plus(0, y) = y : N$$
$$plus(succ(x), y) = succ(plus(x, y)) : N$$

or the operator for primitive recursion in arithmetic

$$natrec : N \to (N \to N \to N) \to N \to N$$
$$natrec(d, e, 0) = d : N$$
$$natrec(d, e, succ(a)) = e(a, natrec(d, e, a)) : N$$

2.2 Dependent types, predication and differences between STT and CTT

theory	CTT	STT
logic	intuitionistic	classical
types	every proposition is a type, implying infinitely many types of individuals each of them with their own identity criteria	e is the type of indiviudals t the type of propositions
type constructors	$\to, \Sigma, \Pi, Id, +, \times, ..$	$\to$
propositions as .. paradigm	propositions are types: - propositions have inhabitants (i.e. they carry additional information)	a proposition is an element of type t: - a proposition has no elements - types of propositions are of the form $\sigma \to \tau,\ \tau \neq e$ ensuring that we always have an individual

Cruel in the analysis CTT vs. STT might be the notion of predication. E.g. that we might want to express *T'Pol is a Vulcan.* in STT. We might do this using the predicate $Vulcan^{e\to t}$ of functional type and the individual $T'Pol^e$ to express the proposition $Vulcan(T'Pol)^t$.

To understand available options of predication in CTT we give a brief introduction of dependent types in CTT.

2.2.1 Dependent types in CTT - Hypothetical judgements

A hypothetical judgement is a judgement that depends on some context. $x : A \vdash B(x)$ may be read as $B(x)$ is a type if x is in A. B is a family of types or propositions depending (on a term of type) A.

This means that if $a : A$, then $B(a)$ is a type and if $a = b : A$ then $B(a) = B(b)$, i.e. there is a functional behavior of B over A - a propositional function that returns when applied on a of A the proposition $B(a)$.

With this mean we can express relations like $x : A, y : A \vdash older(x, y)$. In general the order in the context might be important. For example given the context $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$ then $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash B(x_1, \ldots, x_n)$. This means in particular that for $a : A_1$, it follows that $x_2 : A_2[a/x_1], \ldots, x_n : A_n[a/x_1] \vdash B[a/x_1]$ *type*.

Similar as in the calculus of constructions (CC) we also have *types that depend on terms (dependent types)* like for example dependent products

$$\frac{\Gamma \vdash A \ type \quad \Gamma, x : A \vdash B(x) \ type}{\Gamma \vdash \Pi_{x:A} B(x) \ type}$$

The elements of such a type are functions that take elements of type A as input and output an element of type $B(a)$. If we consider the introduction rule

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x : A.b(x) : \Pi_{x:A} B(x)}$$

which gives rise to $\forall$ introduction in natural deduction and conversely, the elimination rule corresponds to the $\forall$ elimination rule in natural deduction. A propositional reading implies $\forall x : A \ B(x)$ which yields a constructive interpretation of the universal quantifier: proofs of $(\forall x : A) \ B(x)$ are functions that map an element a of A to a proof of $B(a)$.

Remark 2.1. Note that there are different standard notations. Regarding dependent products as $\Pi_{x:A} B$ or $\Pi x : A.B$, Klev [5] and Nordström et. al [7] use $(x : A)B$ or $(x \in A)B$. Dependent products degenerate if the type B does not depend on A to the implication/function type $A \to B$, i.e. a function space B^A or $A \implies B$ with functions as canonical elements. This is denoted in the other notation as $(A)B$.

Function application $a \ b$ is also denoted by $a(b)$, while abstraction $\lambda x : A.b$ might be denoted by $[x]b$.

In comparison, for the dependent sum we have

$$\frac{\Gamma \vdash A \ type \quad \Gamma, x : A \vdash B(x) \ type}{\Gamma \vdash \Sigma_{x:A} B(x) \ type}$$

and as an introduction rule

$$\frac{\Gamma \vdash a : A \quad \Gamma, b : B[a/x]}{\Gamma \vdash \langle a, b \rangle : \Sigma_{x:A} B(x)}$$

Introduction and elimination rules again correspond to $\exists$ introduction and elimination, since again $(\exists x : A)B(x)$ follows by a propositional reading.

Elements of such types are higher order entities and we quantify over their types. In STT the type of quantification might be more powerful since quantifiers range over the type of all propositions.

2.2.2 Predication

There might be three roles of predication 1. circumscribe, 2. single out a domain of quantification, and 3. express a property of certain elements of the domain.
For the purpose of analyzing the notion of predication, we reconsider the example of *T'Pol is a Vulcan.* in STT, using the predicate $Vulcan^{e \to t}$ of functional type and the individual $T'Pol^e$ to express the proposition $Vulcan(T'Pol)^t$.

We can express this in CTT via the following constructions.

A first approach - to single out a domain of quantification - can be conducted by *type predication*, i.e. we use the predicate Vulcan as a type V of Vulcans. This means that we consider *T'Pol* as an inhabitant of type Vulcan: $T'Pol \ : \ V$. In particular, we distinguish between the category to which T'Pol and the category to which Vulcan belongs.

A second approach - to express a property of certain elements of the domain - might be made possible by the use of a *propositional function* to express the predicate *is a Vulcan* by the judgement $x : P \vdash Vulcan(x)$ *prop*. In this case *T'Pol* is an element of type P, whereas *is a Vulcan* is a family of types, i.e. propositional functions over the type of people P.

Another way to incorporate type predication is by the use of a *function type*. We need to have suitable types for the domain, i.e. the type of people P, and for the codomain This way we basically proceed as in STT: we use a ground type for T'Pol, i.e. some more specific type of individuals, and we use a higher order function for *is a Vulcan*. This function f assigns a value in a suitable type to an element of type P. For this version Crosilla[1] mentions the necessity of an extension of CTT with a new type constructor that we call a universe U_1 which acts as the type of propositions. Once, we defined this we might define the function f from P to U_1.

All approaches have in common that inviduals as *T'Pol* and whatever encodes *is a Vulcan* are incomparable. In particular, we might express metaphysical phenomena that argue for the incommensurability between individuals, propositions, properties, ..

This leads us to the following tabular.

theory	CTT	STT
logic	intuitionistic	classical
types	every proposition is a type, implying infinitely many types of individuals each of them with their own identity criteria	e is the type of indiviudals t the type of propositions
type constructors	$\rightarrow, \Sigma, \Pi, Id, +, \times, ..$	$\rightarrow$
propositions as .. paradigm	propositions are types: - propositions have inhabitants (i.e. they carry additional information)	a proposition is an element of type t: - a proposition has no elements - types of propositions are of the form $\sigma \rightarrow \tau, \ \tau \neq e$ ensuring that we always have an individual
predicativity	predicative (constructive (bottom up) definitions)	impredicative (define propositions in terms of the totality of all propositions)
quantification	.. over a type of propositions means that we quantify over the type of all types types of all types (avoid Girards paradox in CC, i.e. to avoid *prop* : *prop* entails a contradiction) Solution: add new type constructors to avoid paradox i.e. construct a universe U_1, s.t. a code for it $\#(U_1)$ belongs to a higher universe U_2	.. over the type of propositions

3 Propositional and Judgemental Equality in CTT

We follow Crosilla [1] (in parts) and mainly Klev [5].

Crosilla[1] introduced one notion of Martin Löf type theory (MLTT/CTT) which we introduced in the beginning and switches then to the notion of CTT that Klev[5] considers. We will follow the notion Klev introduces.

Following Klev[5] we call the type of all propositions **prop**. Inductively defined types like N and *bool* are considered types of individuals which are objects of type **Set**. The type **Set** is *open*, i.e. there is a definition of what an object of type **Set** is without an instruction on how to construct objects. Hence, we may write **prop** = **Set**, to mean firstly, that every proposition is a type of individuals:
In this case we mean that if the proposition is true, then a proof of it exists. This proof is an object, i.e. a truthmaker in the methaphysical sense.

For the inclusion from right to left Klev[5] states that every individual is a proposition and gives the example of type **N** of natural numbers as a proposition: in MLTT an individual A determines the content that A is inhabited, i.e. a proposition is not a content but something that determines the content.

From the point of view of predicate logic equality is a relation (predicate) that is an equivalence relation. As a consequence whenever $x \doteq y$ is true then by any formula of the language x and y are indiscernible, implying that $\varphi(x)$ is true if and only if $\varphi(y)$ is true.
However, the preceding rule, i.e. $\forall x, y\ (x \doteq y) \to (\varphi(x) \leftrightarrow \varphi(y))$, can only be an axiom schema in first order logic (FOL). Together with the rules for reflexivity, transitivity, and symmetry, we get a notion of Leibniz equality in FOL. Proper Leibniz equality is not firstorderizable.
Wheareas, in higher order logic (HOL) we can formalize Leibniz equality by $\forall x, y(x \doteq y \leftrightarrow (\forall P P(x) \leftrightarrow P(y))$.
Leibniz identity is the notion of *identity of indiscernibles*, i.e. two objects are equal if and only if they satisfy the same properties. In particular, the above notion in predicate logic $\forall x, y(x \doteq y \to (\forall P P(x) \leftrightarrow P(y))$ denotes the *indiscernibility of identicals*. Conversely, given that x and y have the same properties they are equal, i.e. $\forall x, y(\forall P P(x) \leftrightarrow P(y) \to (x \doteq y)$, denotes the *identity of indiscernibles*.

There are different notions of equality in CTT that differ in logical form and strength. Firstly, there is the primary and local notion of *judgemental equality*, like $a = b : A$ and $A = B$. We also call this notion that comes with its own structural rules like defined above *definitional equality* - the relation of being identical by definition also meaning that two terms have the same normal form. This equality is sometimes denoted it with a $\equiv$ instead of $\doteq$. This means that we define in the type introduction rules when canonical and non-canonical elements are equal.

3.1 Propositional Equality

We cannot combine judgements by applying (logical) connectives to them as they apply to propositions, not judgements. In order to express a complex statement including an equality statement CTT introduces the equality statement as a proposition. This equality is called *propositional equality*. This equality is a notion of intensional equality which is not local and can be applied to any type in the type hierarchy in an analogous way.
Given $A\ type, a : A, b : A$, then $Id(A, a, b)$ is a proposition, i.e. a new type created by the type constructor (operator) Id. This proposition asserts that two inhabitants, a, b of A are equal:

$$\frac{\Gamma \vdash A\ Set \quad \Gamma \vdash a : A \quad \Gamma \vdash b : A}{\Gamma \vdash Id(A, a, b) : prop}\ (\mathbf{Id} - form)$$

following Klev.[5] In particular, it is considered the ordinary identity of objects. It might be more familiar to denote $Id(A, a, b)$ by $a =_A b$ or $Id_A(a, b)$.

This kind of identity is in particular of interest for unrestricted quantification.[5][1][2]

Of course, by definition, in particular, by the requirement that $a, b : A$ and that A is a type of individuals, it follows that we can only by applied to arguments of the same types of individuals. Hence, it is not possible to, e.g., regard a *Romulan D'deridex warbird* and a member of a *numerical type* as identical objects of a same *numerical type* if we do not assume a Gödelization of a *D'deridex warbird*. Such statements are not well-formed.
Those kinds of problems fall in the category of cross-type generalisations.

3.1.1 Introduction Rule for Propositional Identity

The insight that $Id(A, a, b)$ is a notion of identity between a and b is not obvious. Indeed, we shall study introduction and elimination rules for this purpose.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash refl(A, a) : Id(A, a, a)}\ (\mathbf{Id} - I)$$

is the introduction rule for the identity type, following Klev's notation[5].

Here, $refl(A, a)$ is a proof (construction in the sense of BHK, and indeed a better specified kind of proof) of $Id(A, a, a)$, meaning in particular identity on a type is indeed a reflexive relation.

In the context of unrestricted quantification we might understand $Id(A, a, a)$ as: for all types of individuals A, any element of A is identical to itself - in particular, we obtained a formulation with two *domain-restricted* universal quantifiers.[5]

Again, after defining the identity introduction rule the constructor $Id(\cdot,\cdot,\cdot)$ is defined, and we are fine to judge $c = refl(A,a) : Id(A,a,a)$. Hence, and conversely to homotopy type theory, we have just one proof (construction) namely $refl(A,a)$ for the identity proposition $Id(A,a,a)$.

When making the judgement $Id(A,a,b)$ then a,b have to refer to the same object. In $refl(A,a) : Id(A,a,a)$ it is clear that a in $Id(A,a,a)$ is the same object.

3.1.2 Elimination Rules for Propositional Identity

The former corresponds to the natural deduction identity term t (also denoted as $=$ and $=tt$). Considering now an elimination rule by Per Martin Löf

$$\frac{Est \quad \varphi(x,x)}{\varphi(s,t)}\ (E-E)$$

where in particular, $\varphi(x,x)$ is obtained by substitution from $\varphi(y,z)[x/y][x/z]$.

Given a derivation of $\varphi(x,x)$ we may conclude that $\varphi(y,z)$ defines a reflexive relation. The rule allows us to infer for given Est and $\varphi(x,x)$ that $\varphi(s,t)$, which is a slightly elegant method to eliminate Est. Hence, the rule implies that E is the smallest reflexive relation on the given domain. In the case of $\varphi(y,z) := \psi(y) \to \psi(z)$ Leibniz's law follows by the arbitrariness of s and t and the reflexivity of x.[5]

In the following and corresponding *elimination rule* of **Id**−elimination we can more explicitely see the variable binding of x:

$$\frac{\begin{array}{c}(x:A)\\ \mathcal{D}\\ p : Id(A,a,b) \quad d(x) : C(x,x)\end{array}}{J(p,x.d(x)) : C(a,b)}\ (\mathbf{Id}-E)$$

Supposing that $x,y,z : A$, the proposition $C(y,z)$ defines a relation on A. Again we denote $C(x,x) := C(y,z)[x/y][x/z]$. From the assumption $x : A$ we assume that we have a derivation of the judgement $d(x) : C(x,x)$, i.e. we have a construction (proof) $d(x)$ of $C(x,x)$. The binding of x and by the presence of the assumptions and the construction of $C(a,b)$ from constructions p and $d(x)$ is explicitely denoted by $x.d(x)$ (using adjusted λ−abstraction notation) in the proof $J(p,x.d(x))$ of $C(a,b)$.[5]

There exist adjusted forms of this rule, compare for example the rule from Nlab [6]

$$\frac{\Gamma, a:A, b:A, p:Id(A,a,b) \vdash C\ type \quad \Gamma, x:A \vdash d(x) : C[x/a, x/b, refl(A,x)/p]}{\Gamma, a:A, b:A, p:Id(A,a,b) \vdash J(d(x),a,b,p) : C}\ (\mathbf{Id}-E)$$

which might be helpful to consider for better understanding on what happens.

3.1.3 Equality Rules for Propositional Identity

Similar to the equality rules for the projections we have the following notion of an equality rule for propositional identity, again denoted recursively and by a similar argumentation metamathematically and by the foundational semantics correct:

$$\frac{}{J(refl(A,a), x.d(x)) \ = \ d(a) \ : \ C(a,a)}\ (\mathbf{Id}-Equ)$$

This rule again corresponds to a detour elimination by , in particular, directly substituting the free occurences of x by t

$$\frac{\begin{array}{c}\mathcal{D}(x)\\ Ett \quad \varphi(x,x)\end{array}}{\varphi(t,t)}\ E-E \quad \Rightarrow \quad \begin{array}{c}\mathcal{D}(x)\\ \varphi(t,t)\end{array}$$

3.1.4 Generalization of Propositional Identity and Universes

A propositional identity type is a type of individuals and hence can be used itself to form an identity. For example, $Id(Id(A,a,b),c,d)$ is an identity type for proofs (constructions) c,d of $Id(A,a,b)$.

The answer to whether a proposition that states that an arbitrary proof of $Id(A,a,a)$ is identical to $refl(A,a)$, i.e.

$$(\forall x : Id(A,a,a))[Id(Id(A,a,a),x,refl(A,a))]$$

, is always inhabited turned out to be difficult. According to Hedberg[3] it is inhabited for a large class of A, while Hofmann and Streicher[4] gave a model as a counterexample, implying that the answer is in general no.

As already discussed above considering Crosilla[1] a universe U has codes of types of individuals as elements while it is itself a type of individuals.
As for this purpose we can also think of decoding what have been encoded. We may have a code $\#(N)$ of N and another code for propositions P,Q and the disjunction $\vee$, i.e. $\#(\vee)(\#(P),\#(Q))$.

Another operation for decoding $t : (U)prop$ would return us the proposition $t(\#(P))$, i.e. P, and $t(\#(Q))$ which decodes to Q respectively.

As decoding itself is an operation we know that decoding must be an operation decoding the encoded elements of U and returning elements of $prop$. It expresses a meta function that operates with respect to that specific universe U, not a normal function type.

This means that it might be extensional in the following sense.

$$\frac{t(x) = u(x) : prop}{t = u : (U)prop}\ x : U$$

However, $Id(prop,P,Q)$ is not well-formed as a proposition is not a type of individuals while $Id(U,P,Q)$ is. Taking the proposition $Id(U,\#(P),\#(Q))$ we can express an identity between $t(\#(P))$ and $t(\#(Q))$.

To introduce the relation **I** to express *encoded equality*, we will first explain what we mean with the relation **E**:
A function f from P to Q takes a proof p of P to a proof $f(p)$ of type Q. To express surjectivity and injectivity of f one might use $Id(P,x,y)$ and $Id(Q,x,y)$. So, if there is a surjective and injective function, i.e. bijective function which preserves structure - an isomorphism, between P and Q we can state that P **E Q** holds.

If now P **I Q** holds on a higher level, then $t(p)\mathbf{E}t(q)$ holds. Without further adjustment we cannot infer in the converse direction. To do so we need what the the univalence axiom of homotopy type theory grants.

Further, P **I Q** holds whenever $Id(U,\#(P),\#(Q))$ is inhabited.

3.2 Judgemental Equality

Judgemental equality $a = b : A$ as a notion of equational logic ensures that $=$ is an equivalence relation. In particular, the rules for identity judgements can be regarded as a system for equational logic. The axioms of such a system would be the definitional equations of such a system, implying that we may consider identity judgments to express definitional identity.[5] In contrast to an untyped language it is clear that in a typed language, a and b have to be of the same type. After deriving the equality one my use a and b interchangably as in an untyped language.

While we can use definitional equality to define computation rules like

$$\frac{n : N}{plus(0,n) = n : N}$$

as we did above. With the propositional equality, one might form a proposition

$$(\forall x : N)\ [Id(N,x,0) \vee (\exists y : N)\ Id(N,x,s(y))]$$

stating that every natural number is identical either to zero or to a successor.
A logical operator $\vee$ cannot be misused using judgements

$$(\forall x : N)\ [x = 0 : N \vee (\exists y : N)\ x = s(y)]$$

- which is not well-formed according to the rules. A judgement like $x = 0 : N$ is an assertion and we cannot use logical operators on assertions.[5]

As earlier discussed within propositional equality $Id(A, a, b)$ requires A to be a type of individuals. Conversely, judgental equality can be created for all types.

Other notions of judgemental identity are, for example, identity between propositions $A = B : prop$, identity between the functions f and g from α to β with $f = g : (\alpha)\beta$, and identity between the propositional functions P and Q with $P = Q : (\alpha)prop$.

Judgemental identity is a *hyperintensional* notion of identity which is less fine than syntactical identity but finer than the relation **I** defined above in 3.1.4.

We refer with *syntactical identity* to, for example, the propositions $Id(N, 3, 3)$ and $Id(N, plus(1, 2), 3)$ which are not syntactical identical, but definitional identical since $plus(1, 2)$ and 3 are definitionally the same, i.e. $plus(1, 2) = 3 : N$.

3.3 Judgemental Identity versus Propositional Identity

Definitional equality can be regarded as a relation on expressions while propositional equality is a notion that expresses that two objects are the same.

If judgemental identity is interpreted as definitional equality then in $a = b : A$ the terms a, b are standing for themself. This means that they are objects or proofs. If they are definitionally identical they are/have the same proof (construction). In contrast to this in $Id(A, a, b)$ the terms a, b are references to the objects a, b.

A suggestion to solve this problem of amiguity in the presentation of terms by Martin Löf is called *Forthcoming*. In particular, we make it more explicit what stands for itself and what is a reference: In the judgement $a = b : A$ we say that a and b are definitionally identical. In $p : Id(A, a, b)$ we say that a and b denote the same object, i.e. refer to the same object.

The idea of *Forthcoming* is that *definitionally identical propositions* have the same proofs

Example 3.1. Klev[5] stipulates that the rule

$$\frac{\Gamma \vdash c : C \quad \Gamma \vdash d : D}{\Gamma \vdash \langle c, d\rangle : C \wedge D}\ (\wedge - I)$$

defines a proposition $A \wedge B$.
By the foundational semantics the statements $A = C : prop$ and $B = D : prop$ are well-formed. Hence, a canonical proof of $A \wedge B$ has the form $\langle c, d\rangle$.

The idea of this example can be generalized.

Considering in this context again identity propositions and the introduction rule

$$\frac{\Gamma \vdash c : C}{\Gamma \vdash refl(C, c) : Id(C, c, c)}\ (Id - I)$$

defines $Id(A, a, b)$. Assuming the judgements $A = C : Set$, $a = c : A$, and $b = c : A$, a canonical proof of $Id(A, a, b)$ has the form $refl(C, c)$.

In natural deduction one may think of a canonical derivation of Etu as

$$\cfrac{\cfrac{\cfrac{\mathcal{D}}{Et't'}\ t' \equiv t}{Ett'}\ t' \equiv u}{Etu}$$

3.3.1 Mathematical Induction

Within the combinatorial reasoning of equational logic we use substitution of equals for equals but one does not have the principle of mathematical induction.

Using definitional equality one cannot derive the commutative law $plus(x, y) = plus(y, x) : N$ as an identity judgement. This follows from the normalization theorem.[5]

Conversely, we can derive $p(x, y) : Id(N, plus(x, y), plus(y, x))$ withing propositional equality. This can be done due to the *elimination* rule for type N, i.e. the rule[6]

$$\frac{\Gamma, x : N \vdash Ctype \quad \Gamma \vdash c_0 : C[0/x] \quad \Gamma, x : N, c : C \vdash c_s : C[s(x)/x] \quad \Gamma \vdash n : N}{\Gamma \vdash ind_N^C(0, c_0, c_s) : C[n/x]} \ (N - E)$$

In this case we use induction over N to obtain the result.

3.3.2 Intensional versus Extensionsional Type Theory

While we can always infer $refl(A, a) : Id(A, a, b)$ from $a = b : A$, the converse is only true if and element of $Id(A, a, b)$ can be constructed. Note that a and b are closed terms, i.e. do not contain free variables, while in the converse direction a, b may contain free variables and hence do not necessary have to be closed which makes the following rule necessary.

If we would like to upgrade the so far introduced CTT to an *extensional theory* we need to add an equality reflection rule where we can derive from the propositional (typal or objectual) equality of terms a definitional equality like

$$\frac{\Gamma, a : A, b : A \vdash p : Id(A, a, b)}{\Gamma, a : A, b : A \vdash a = b : A}$$

In this case since a, b do not have to be closed terms, it is possible to derive the commutative law $plus(x, y) = plus(y, x) : N$[5].

If we do not have such an equality reflection rule, CTT is an *intensional theory*.
In case we do allow such an equality reflection rule the judgements of the form $p : Id(A, a, b)$ and $a = b : A$ become of the same power. Thus, we may no longer understand an identity judgement as an expression of definitional equality.

3.3.3 Identity Functions - Criterial identity

For f to be a function it has to satisfy functionality *if* $a = b$, *then* $f(a) = f(b)$. The identity predicate assumes some notion of identity and hence it is difficult to use the identity predicate to formulate the relevant criterion of identity. The identity predicate itself determines a function.

In CTT we avoid this predicativity problem by using identity judgments when giving the identity criterion for a type. The functionality of the identity function by means of identity judgements is given by

$$\frac{\Gamma \vdash a = a' : A \quad \Gamma \vdash b = b' : A}{\Gamma \vdash Id(A, a, b) = Id(A, a', b') : prop}$$

while indeed, $Id(A)$ can be regarded as a binary propositional function over A taking (a, b) and returning a proposition $Id(A, a, b)$.

4 Unrestriced Quantification

To study different forms of inquiry appearing in logic, methaphysics, and mathematics we are unrestricted quantification (UQ) in general. Taking the linguistic connection to type theory into account it might be useful to study if unrestricted quantification is available in type theories in order to have a framework capable to accomodate our needs. We will follow Crosilla [1] and Florio and Jones [2].

To have an environment for unrestricted quanitfication in order to *quantify over absolutely everything*, following the paradigm *everything is physical*, we study the role of primitive higher order quantification. To be able to avoid paradoxes like, for example, Girard's paradox, we introduce higher order theories. However, once we have a type hierarchy, it is difficult to say what it means for a quantifier to be unrestricted. If we are quantifying over a specific level in the hierarchy, it is by the nature of this quantification not unrestricted. The problem is even more present if we are taking a predicative theory into account. To have a framework that accomodates unrestricted interpretation of quantifiers we need some kind of universal set. Such a universal set may serve as a unrestricted domain of quantification. Higher order theories make available a property or a type that is compatible with an universal property of objects, serving as a domain for unrestricted quantification. However, a type of objects may not be truly universal if it is not applied on other entities that might be present than objects. But then we cannot define a universal property at all. Florio and Jones[2] call this problem the *intuitive problem for unrestricted quantification within type theory (intuitive problem).*

Within set theory the problem is that no set-theoretic domains of quantification contains every object. Within type theory the problem reformulates to no type-theoretic domain of quantification contains every type-theoretic object.

Florio and Jones study the connection between meaningful predicability and unrestricted quantification. They conclude that unrestricted quantification is available in most kind of type theories, by taking Russel's insight about the relation between the structure of predication and quantification into account.

4.0.1 Unrestricted Quantification in Type Theories

We have syntactic restrictions in STT on predication: consider the example $vulcan^{e\to t}$ and $T'Pol^{e}$ again. They ask if different formulations of STT have an impact on the availability of unrestricted quantification. Florio and Jones conclude that unrestricted quantification is available in strict and cumulative type theories but not in a third more permissive theory.

Unrestricted domains contain absolutely everything. We might compare them to universes $U_1, U_2, \ldots$ or the type $*$ and $\Box$ in the calculus of constructions (CC). If we allow that we have unrestricted domains, we rule out any possible counterexample. Further, we run into paradoxes like Girard's paradox.

Definition 4.1 (Russelian Domain)**.** For $\forall x^{e} F^{e\to t}(x^{e})$ a domain is called Russelian if and only if it coincides with $F's$ range of significance (all the entities that can be meaningfully said to be F).

If we cannot say meaningfully that something is F, we can also not say that something is not F. Hence, there cannot be a counterexample to a generalization involving F.

Definition 4.2 (Unresticted Domain)**.** A domain is unrestricted relative to some predication if and only if it is Russelian relative to it.

This means in $\forall x^{e} F^{e\to t}(x^{e})$, for example, that F can only be meaningfully applied to entities of type e. But no property of individuals can be in the range of significance of F and provide a counterexample.

4.1 Unrestricted Quantification in STT

The *intuitive problem* was that the universal property of objects may not be truly universal. In particular, to be truly universal it would not only apply to all objects but also to any other entity recognized by the framework.

Since every individual is self-identical, i.e. $x^0 =_1 x^0$, we might use comprehension to define the property U^1 that holds for all and only the individuals of level 0, i.e. $\forall x^0 U^1(x^0)$.

This means that U might be understood as some unrestricted domain for predications, since it is possessed by all properties of level 1, but U is not possessed by any property of level 1. The formula that U is not possessed by any property of level 1 - $\forall x^1 U^1(x^1)$ - is not well typed.

4.2 Unrestricted Quantification in CTT

The types in CTT are meant to be generalizations of Russel's notion of range of significance as in Definition 4.1.

According to Florio and Jones [2] unrestricted quantification is available also in CTT:

First of all we consider types as domains of quantification or in other words we take domains to be types.
Is there a type D and a predication $(\forall x : D)B(x)$ such that D is Russelian for that predication?

D is Russelian if it coincides with all the entities that can be meaningfully said to be D, i.e. it coincides with $D's$ range of significance.

We have for all $a : A$, $a = a$, which means that for each a in A we can prove that the proposition $Id(A, a, a)$ is true. Hence, $(\forall x : A)Id(A, a, a)$ is true, implying that the type A is Russelian for the predication $(\forall x : A)Id(A, a, a)$. Since $(\forall x : A)Id(A, a, a)$ can be meaningfully applied to all and only the elements of A, there is a domain in CTT that is unrestricted for some predication. Thus, every type is unrestricted for some predication regarding the predication *everything is self-identical*.

Florio and Jones conclude that both CTT and STT satisfy their criteria of unrestricted quantification.

4.3 Unrestricted Quantification in CTT versus STT

Firstly it seams that a type in CTT like for example N_2 (type with 0_2 and 1_2) will not in general play the role of an inclusive domain in philosophical theorizing as the type (property) U in STT.

In CTT it seems to be a promising strategy to introduce a universe U_1, then we can prove for every type A, such that $\#(A)$ is in U_1 that $(\forall x : A)Id(A, x, x)$. Here, U_1 is a Russelian domain for this predication as it coincides with its range of significance. A universe seems to be the kind of inclusive domain in a similar way as e.

Note that we have in CTT with universes more expressive power than in STT since we may quantify over types themselves.

theory	CTT	STT
logic	intuitionistic	classical
types	every proposition is a type, implying infinitely many types of individuals each of them with their own identity criteria	e is the type of indiviudals t the type of propositions
type constructors	$\rightarrow, \Sigma, \Pi, Id, +, \times, ..$	$\rightarrow$
propositions as .. paradigm	propositions are types: - propositions have inhabitants (i.e. they carry additional information)	a proposition is an element of type t: - a proposition has no elements - types of propositions are of the form $\sigma \rightarrow \tau,\ \tau \neq e$ ensuring that we always have an individual
predicativity	predicative (constructive (bottom up) definitions)	impredicative (define propositions in terms of the totality of all propositions)
quantification	.. over a type of propositions means that we quantify over the type of all types types of all types (avoid Girards paradox in CC, i.e. to avoid $prop : prop$ entails a contradiction) Solution: add new type constructors to avoid paradox i.e. construct a universe U_1, s.t. a code for it $\#(U_1)$ belongs to a higher universe U_2	.. over the type of propositions
Are the theories comparable?	Everything in a type like e.g. N_2 is selfidentical	Everything in U is self-identical.

References

[1] Laura Crosilla. "Constructive Type Theory, an Appetizer". In: *Higher-Order Metaphysics*. Ed. by Peter Fritz and Nicholas K. Jones. Oxford University Press, 2024.

[2] Salvatore Florio and Nicholas K. Jones. "Unrestricted Quantification and the Structure of Type Theory". In: *Philosophy and Phenomenological Research* 102.1 (2021), pp. 44–64. DOI: 10.1111/phpr.12621.

[3] Michael Hedberg. "A coherence theorem for Martin-Löf's type theory". In: *Journal of Functional Programming* 8.4 (1998), pp. 413–436. DOI: 10.1017/S0956796898003153.

[4] Martin Hofmann and Thomas Streicher. "83The groupoid interpretation of type theory". In: *Twenty Five Years of Constructive Type Theory*. Oxford University Press, Oct. 1998. ISBN: 9780198501275. DOI: 10.1093/oso/9780198501275.003.0008.

[5] Ansten Klev. "Identity in Martin-Löf type theory". In: *Philosophy Compass* 17.2 (), e12805. DOI: https://doi.org/10.1111/phc3.12805.

[6] nLab authors. *Martin-Löf dependent type theory*. https://ncatlab.org/nlab/show/Martin-L%C3%B6f+dependent+type+theory. Aug. 2024.

[7] B Nördstrom and K Petersson. "Martin-Löf's type theory". In: *Handbook of Logic in Computer Science: Volume 5. Algebraic and Logical Structures*. Oxford University Press, Jan. 2001. ISBN: 9780198537816. DOI: 10.1093/oso/9780198537816.003.0001.