Proof Irrelevance in Type-Theoretical Semantics*

Zhaohui Luo

Abstract Type theories have been used as foundational languages for formal semantics. Under the propositions-as-types principle, most modern type systems have explicit proof objects which, however, cause problems in obtaining correct identity criteria in semantics. Therefore, it has been proposed that some principle of *proof irrelevance* should be enforced in order for a type theory to be an adequate semantic language. This paper investigates how proof irrelevance can be enforced, particularly in predicative type systems. In an impredicative type theory such as UTT, proof irrelevance can be imposed directly since the type $Prop$ in such a type theory represents the totality of logical propositions and helps to distinguish propositions from other types. In a predicative type theory, however, such a simple approach would not work; for example, in Martin-Löf's type theory (MLTT), propositions and types are identified and, hence, proof irrelevance would have implied the collapse of all types. We propose that Martin-Löf's type theory should be extended with h-logic, as proposed by Veovodsky and studied in the HoTT project, where proof irrelevance is built-in in the notion of logical proposition. This amounts to MLTT$_h$, a predicative type system that can be adequately employed for formal semantics.

1 Introduction

Formal semantics in modern type theories (MTT-semantics for short) [26, 8] is a framework for natural language semantics, in the tradition of Montague's semantics [33]. The development of MTT-semantics is a part of a wider research endeavour by many researchers who have recognised the potential advantages of rich type

Zhaohui Luo
Royal Holloway, University of London, e-mail: zhaohui.luo@hotmail.co.uk

* Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence (SCI). Springer, 2019.

structures in constructing formal semantics [37, 35, 28, 5, 10, 13, 36, 3, 18]. While Montague's semantics is based on Church's simple type theory [9, 15] (and its models in set theory), MTT-semantics is based on dependent type theories, which we call modern type theories (MTTs), to distinguish them from the simple type theory.

One of the key differences between MTTs and simple type theory is that MTTs have rich type structures with many types, much richer than those in simple type theory. Because of this, in such a rich type system, common nouns can be interpreted as types rather than predicates (see, for example, Ranta's proposal on this in Martin-Löf's type theory [35]). We can call this the CNs-as-types paradigm: it is different from Montague's semantics, which adopts the CNs-as-predicates paradigm. For instance, consider the CN 'book': it is interpreted in Montague's semantics as a predicate of type $e \longrightarrow t$, while in MTT-semantics, it is interpreted as a type $Book$. It has been argued that interpreting CNs as types rather than predicates has several advantages including, for example, better treatment of selectional restriction by typing, compatible use of subtyping in semantic interpretations and satisfactory treatment of some advanced linguistic features such as copredication (see, for example, [24, 26] for further details).[2]

Another feature of MTTs is that their embedded logics are all based on the principle of propositions as types [12, 21] and, in particular, there are proof terms whose types are logical propositions: a logical formula P is true if, and only if, there exists a proof term p of type P. Such formulations with proof terms are rather natural for constructive proof systems: type theories such as Martin-Löf's type theory [30, 31] are such examples which were originally designed for describing constructive mathematics such as that described by Bishop in [4]. However, MTTs can also be applied in applications other than constructive mathematics: their use in MTT-semantics for natural language is such an example. In particular, MTTs provide a semantic framework which is both model-theoretic and proof-theoretic [27] and have many advantages as foundational languages for formal semantics, as compared with the model-theoretic framework of Montague's semantics. MTT-semantics also provides useful mechanisms for successful treatments of various linguistic features such as copredication with subtyping [24, 26] that have been found difficult to be dealt with in the traditional setting.

However, proof terms that inhabit propositions are not completely innocent: when we employ type theories for formal semantics, they cause problems. In particular, their presence makes it difficult, if not impossible, for one to obtain correct identity criteria of CNs. For example, in a dependent type theory, one may use dependent types of pairs[3] to represent CNs modified by intersective adjectives [35]: for instance, a handsome man is a pair of a man and a proof that the man is handsome. Then,

[2] It may be interesting to remark that, recognising that interpreting CNs as types has several advantages, some researchers have suggested that *both* paradigms need be considered including, for instance, Retoré's idea in this respect [36]. A related issue in type-theoretical semantics is how to turn judgemental interpretations into corresponding propositional forms, as studied in [41] which proposes an axiomatic solution for such transformations that can be justified by means of heterogenous equality of type theory.

[3] Technically, Σ-types are used – see §2.2 and Footnote 7 for a further description of Σ-types.

for such representations, one can ask: what is the identity criterion for *handsome man*? An obvious answer should be that it is the same as that for *man*: two handsome men are the same if, and only if, they are the same man. But this is not what the formal interpretation gives us since it also requires that the proofs of the man being handsome be the same. Obviously, this would not be a correct identity criterion for the modified CN *handsome man*. How to solve this problem? It has been proposed that some principle of *proof irrelevance* should be adopted [25]: i.e., any two proofs of the same logical proposition should be the same. In the above example, it implies that any two proof terms of a man being handsome should be the same. In §2, we shall introduce the issue of identity criteria for CNs and illustrate that proof irrelevance provides a nice solution to this problem.

A type theory can either be predicative or impredicative. Examples of the former include Martin-Löf's type theory [34] as implemented in the proof assistant Agda [1] and those for the latter include the type theory UTT [23] as implemented in Lego/Plastic [29, 6] and pCIC as implemented in Coq [11]. A notable difference is that, in impredicative type theories, there is usually a type $Prop$ of all logical propositions, while in a predicative type theory, we usually do not have such a type. This difference is significant when we consider a proof irrelevance principle. For impredicative type theories, imposing proof irrelevance is pretty straightforward: we simple stipulate that, for any proposition of type $Prop$, any two proof terms of type P are the same [25] (more details can be found in §2.3). However, how to consider proof irrelevance in a predicative type theory is not a simple matter anymore: for example, in Martin-Löf's type theory, people usually follow Martin-Löf to identify types with propositions. In this case, it would be quite absurd to impose proof irrelevance for all propositions because that would have meant to identify objects of all types! It is unclear whether and how proof irrelevance can be enforced in a predicative type theory, a point of view expressed in the following quotation, taken from [25]:

> It is also worth noting that, although proof irrelevance can be considered for impredicative type theories directly as above, it is unclear how this can be done for predicative type theories. For instance, in Martin-Löf's type theory [30, 31], propositions are identified with types. Because of such an identification, one cannot use the above rule to identify proofs, for it would identify the objects of a type as well. Put in another way, proof irrelevance is incompatible with the identification of propositions and types. In order to introduce proof irrelevance, one has to distinguish logical propositions and types (see, for example, [23]).

The above-mentioned identification of types and propositions gives rise to a logic based on the principle of propositions as types – the usual logic in Martin-Löf's type theory – let's call it the PaT logic[4]. In other words, one cannot enforce a principle of proof irrelevance in Martin-Löf's type theory (MLTT) with PaT logic. Therefore, to use MLTT with PaT logic for formal semantics suffers of the problem we have described above: one cannot (or at least, would be very difficult to) obtain correct identity criteria in semantic interpretations of CNs, among other things.[5]

[4] PaT stands for 'propositions as types'.

[5] See the second paragraph of the Conclusion section for another potential issue in this respect, but this is out of the scope of the current paper.

The main contribution of the current paper is to solve this problem of how to consider proof irrelevance for Martin-Löf's type theory. Recently, based on Martin-Löf's type theory, researchers have developed Homotopy Type Theory (HoTT) [40] for the study of foundation and formalisation of mathematics. One of the developments in the HoTT project is its logic (sometimes called h-logic), developed by Voevodsky, based on the idea that a logical proposition, called a mere proposition, is a type that is either a singleton or empty. In other words, proof irrelevance is built-in in h-logic and this, among other things, has given rise to a logic with a type of all (small) propositions. Our proposed solution is that MLTT should be extended with h-logic and the resulting language, MLTT$_h$, can then be used adequately as a foundational language for MTT-semantics. MLTT$_h$ is a proper extension of Martin-Löf's type theory, although it is a subsystem of type system for HoTT as described in [40]. In §3, we shall first discuss the above problem briefly and then describe h-logic and MLTT$_h$ and illustrate how to use MLTT$_h$ in formal semantics.

2 Identity Criteria and Proof Irrelevance

2.1 Identity Criteria of Common Nouns

As first observed by Geach [16] and later discussed by many others, common nouns are associated with their *criteria of identity*. Intuitively, a CN represents a concept that does not only have a criterion of application, to be employed to determine whether the concept applies to an object, but a criterion of identity, to be employed to determine whether two objects of the concept are the same. It has been argued that CNs are distinctive in this since other lexical terms like verbs and adjectives do not have such criteria of identity (cf, Baker's arguments in [2]).

The notion of criteria of identity can be traced back to Frege [14] when he considered abstract mathematical objects such as numbers or lines. Geach has noticed that such criteria of identity exist for every common noun and are the basis for counting [16, 19, 2].[6] The idea can be illustrated by considering the following examples (1) and (2), where the CNs 'passenger' and 'person' have different criteria of identity, from which it is easy to see that (1) does not imply (2).

(1) EasyJet has transported 1 million passengers in 2010.
(2) EasyJet has transported 1 million persons in 2010.

Several formalisations of the CN 'passenger' are discussed in [25], some of which give intended (correct) identity criteria while the others do not. In particular, when we use logical propositions in formalisations, some principle of proof irrelevance should be enforced.

[6] In general, one may say that an interpretation of a CN should actually be a setoid – a type together with an identity criterion, although in most cases, the situation is more straightforward and can be simplified in that one does not have to mention the identity criterion anymore [7].

2.2 Proof Terms and Proof Irrelevance

Based on the Curry-Howard principle of propositions as types [12, 21], modern type theories contain *proof terms* of propositions. Let's consider a simple example: the logical proposition A in (3) can be proved by the term p in (4); in other words, (5) is a correct judgement stating that p is a proof term of A (formally, one says that the judgement (5) can be derived):

(3) $A = \forall P : Nat \to Prop\ \forall x : Nat.\ P(x) \supset P(x)$
(4) $p = \lambda P : Nat \to Prop\ \lambda x : Nat\ \lambda y : P(x).\ y$
(5) $p : A$

A logical formula is true if, and only if, there exists a proof of the formula: in the above example, A is true since there is p which is a proof of A. Note that a logical proposition may have more than one proof; put in another way, proof terms are not necessarily unique: for some propositions, there may be proofs of the same proposition that are not the same.

Proof terms cause problems when we consider identity criteria of CNs. Their presence makes it difficult, if not impossible, for one to obtain correct identity criteria of CNs. For example, in a dependent type theory, one may use Σ-types of pairs[7] to represent CNs modified by intersective adjectives [35], as the following example shows:

(6) $[\![handsome\ man]\!] = \Sigma(Man, handsome)$

where Man is the type of men and $handsome$ is a predicate over domain Man. If we ask what the identity criterion for *handsome man* is, we would have a problem, since the above interpretation (6) would not give us the intended identity criterion [25]. Obviously, the correct (or intended) identity criterion for *handsome man* should be the same as that for *man*: two handsome men are the same if, and only if, they are the same man. But this is not what the formal interpretation (6) gives us: an object of type $\Sigma(Man, handsome)$ is a pair (m, h) where m is of type Man and h of type $handsome(m)$ and, for two handsome men (m, h) and (m', h') to be the same, we require that it is not only the case that m and m' are the same, but also that h and h' are the same! In other words, two handsome men being the same would have required that the proof terms that prove that they are handsome are the same as well. If there are more than one proof, say h and h', that a man m is handsome, then m

[7] A Σ-type is an inductive type of dependent pairs. Here is an informal description of the basic laws governing Σ-types (see, for example, [31] for the formal rules and further explanations).

- If A is a type and B is an A-indexed family of types, then $\Sigma(A, B)$, or sometimes written as $\Sigma x{:}A.B(x)$, is a type.
- $\Sigma(A, B)$ consists of pairs (a, b) such that a is of type A and b is of type $B(a)$.
- Σ-types are associated projection operations π_1 and π_2 so that $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$, for every (a, b) of type $\Sigma(A, B)$.

When $B(x)$ is a constant type (i.e., always the same type no matter what x is), the Σ-type degenerates into the product type $A \times B$ of non-dependent pairs.

as handsome man with proof h would be different from m with proof h'. Obviously, this would not be a correct identity criterion for *handsome man*.

Some examples to illustrate the above problem are given by Tanaka in a talk at Ohio [39], in the context of studying Sundholm's approach to constructive generalised quantifiers [38]. Consider the following sentence:

(7) Most persons who travelled by plane are happy.

where the semantics of *most* requires correct counting while the Σ-type interpretation of the phrase '*person who travelled by plane*' does not give correct counting since it involves proof terms. To explain, we have:

(8) $[\![$*person who travelled by plane*$]\!] = \Sigma x{:}Person\Sigma y{:}Plane.\ travel(x, y)$,

where $travel(x, y)$ is a proposition expressing that x travelled by plane y. One of the reasons that (8) may give incorrect counting results is that there could be more than one proof of type $travel(x, y)$ and, as a consequence, an interpretation of (7) based on (8) would not be correct.

The interpretation (8) has another problem: its use of (the second) Σ as existential quantifier is also problematic in counting, as pointed out by Tanaka in her talk [39]. A more adequate interpretation would be (9), where the usual existential quantifier $\exists$ is used:

(9) $[\![$*person who travelled by plane*$]\!] = \Sigma x{:}Person\exists y{:}Plane.\ travel(x, y)$.

Note that, while $\Sigma(A, P)$ is a type, $\exists(A, P)$ is a logical proposition. Although the usual existential quantifier $\exists$ exists in impredicative type theories such as UTT, it does not exist in Martin-Löf's type theory.

In order to solve the above problem, it has been proposed that some principle of *proof irrelevance* should be adopted [25]: i.e., any two proofs of the same logical proposition should be the same. In the above examples, it implies that, for any man m, any two proof terms of $handsome(m)$ should be the same (proof irrelevance for proposition $handsome(m)$) and that, for any person x and any plane y, any two proof terms of $travel(x, y)$ should be the same (proof irrelevance for proposition $travel(x, y)$). The interpretations (6) and (9) give correct counting results with proof irrelevance.

Note that, in type theory, proof irrelevance for all logical propositions requires that there be a clear distinction between the types standing for logical propositions (such as that in (3)) and the other types (such as a type of humans and a type of numbers). In an impredicative type theory, such a distinction can easily be made, while in a predicative type theory, it is not clear how to do it. In the following subsection §2.3, we discuss how this can be done for impredicative type theories and, in §3, we shall investigate how it could be done for predicative type theories.

2.3 Proof Irrelevance in Impredicative Type Theories

In impredicative type theories such as UTT [23] and pCIC [11], there is a type $Prop$ of all logical propositions. As a type, $Prop$ represents the totality of all logical propositions. For instance, a formula of the form $\exists X : Prop.\ ...$ says that 'there exists logical proposition X such that ...'. As another example, the proposition in (3) quantifies over all predicates P with domain Nat. With such a distinctive totality $Prop$ of logical propositions, it becomes straightforward to express a principle of proof irrelevance that states that any two proofs of the same logical proposition be the same. As proposed in [25], in the impredicative type theory UTT, this can be captured by the following rule:

$$(*) \qquad \frac{\Gamma \vdash P : Prop \quad \Gamma \vdash p : P \quad \Gamma \vdash q : P}{\Gamma \vdash p = q : P}$$

Intuitively, it say that, if P is a logical proposition (i.e., P is of type $Prop$) and if p and q are two proof terms of P, then p and q are the same.[8]

Consider the semantic interpretation of *handsome man* in (6). With proof irrelevance, two handsome men are the same if, and only if, they are the same man because, for any man m, any two proofs of proposition $handsome(m)$ are the same, according to rule $(*)$ that expresses proof irrelevance.

As noted above, in order to state the principle of proof irrelevance, there must be a clear distinction between logical propositions and other types so that proof irrelevance can be imposed for the former (and not for the latter). This is the case for impredicative type theories, as the $(*)$ rule illustrates. However, a rule like $(*)$ would not be available for predicative type theories such as Martin-Löf's type theory with PaT logic, as to be explained in the following section.[9]

2.4 Most: Counting and Anaphoric Reference

Anaphora representation was an early application of dependent type theory, first studied by Sundholm [37], as an alternative to dynamic semantics [22, 20, 17]. Consider the simple donkey sentence (10), whose interpretation in Martin-Löf's

[8] Here, the equality = is the definitional equality in type theory. In §3, when we consider h-logic, the equality will be the propositional equality. We shall not emphasise the differences of these two equalities in the current paper, partly because it would not affect our understandings of the main issues.

[9] A reviewer has asked the question whether 'one should use impredicative type theories rather than predicative ones for studying logics of natural language' (and maybe others would ask similar questions.) Some people would have drawn this conclusion and, in particular, those (including the author himself) who do not think that impredicativity is problematic would have agreed so. However, some others may think otherwise, believing that impredicativity is problematic in foundations of mathematics (or even in general); if so, predicative type theories would then have their merits and should be considered.

type theory would be (11), where Σ acts as the existential quantifier and F and D are the types that interpret *farmer* and *donkey*, respectively. Note that the use of Σ as existential quantifier is the key to the solution here: if the second Σ in (11) is changed into a traditional (weak) existential quantifier, the interpreting sentence would not be well-formed because the term $\pi_1(\pi_2(z))$ would be ill-typed.

(10) If a farmer owns a donkey, he beats it.
(11) $\Pi z : (\Sigma x{:}F\Sigma y{:}D.\ own(x,y)).\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

However, as explained above in §2.2 with (7) and (8), using Σ as existential quantifier in this simple way causes problems in counting. This can be made clear by the following example (12)[10] and its interpretation (13) in Martin-Löf's type theory, where *most* is interpreted by means of the quantifier $Most$ defined in [38].

(12) Most farmers who own a donkey beat it.
(13) $Most\ z : (\Sigma x{:}F\Sigma y{:}D.\ own(x,y)).\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

The semantic interpretation (13) fails to respect correct counting which becomes important for the truth of (12): because of the second Σ, the proofs of $own(x,y)$ contribute to counting in a wrong and unintended way. This was already realised by Sundholm himself, who proposed some *ad hoc* method to deal with this (see, for example, [39]).

Unlike MLTT, in some type theories there exist both the strong Σ and a weak existential quantifier. For example, in UTT, we have both Σ and $\exists$, the latter being the traditional existential quantifier in its embedded higher-order logic. This has opened a new possibility of using *both* in a semantic interpretation. For instance, (12) can be interpreted in UTT as (14), where, with proof irrelevance, counting is correctly dealt with in (14), and so is anaphoric reference as well.

(14) $Most\ z : [\Sigma x{:}F\exists y{:}D.\ own(x,y)].$
$\qquad \forall y' : [\Sigma y{:}D.own(\pi_1(z),y)].\ beat(\pi_1(z), \pi_1(y'))$

The above interpretation (14) is a strong one – most donkey-owning farmers beat all donkeys he owns. A weaker interpretation would mean that most donkey-owning farmers beat some donkeys he owns, which would be interpreted as (15), obtained from (14) by changing $\forall$ into $\exists$, which deals with counting correctly as well.

(15) $Most\ z : [\Sigma x{:}F\exists y{:}D.\ own(x,y)].$
$\qquad \exists y' : [\Sigma y{:}D.own(\pi_1(z),y)].\ beat(\pi_1(z), \pi_1(y'))$

Remark 1 Here, we have considered how to use both strong and weak sum types in dealing with counting and anaphora in type theories such as UTT. In this respect, instead of doing this, one might consider extending a type theory to become a 'dynamic type theory', in the same way as extending FOL to becomes dynamic predicate logic [17]. The author believes, however, that this is too much a price to pay: like dynamic predicate logic, such a dynamic type theory completely loses its standard logical properties and would become a rather strange logical system.

[10] Thanks to Justyna Grudzińska for a discussion about this example.

For instance, dynamic predicate logic is very much a non-standard logical system: among other things, it is non-monotonic and the notion of dynamic entailment fails to be reflexive or transitive. The author does not think that an underlying logic for NL semantics should be too far from a usual system that is well understood.

3 Formal Semantics in Martin-Löf's Type Theory with H-logic

In this section, we first discuss the problem of imposing proof irrelevance in Martin-Löf's type theory (MLTT) with PaT logic and then detail our proposal of extending MLTT with h-logic, as studied and developed by Voevodsky in the HoTT project [40], and employing the extension MLTT$_h$ as a foundational language for formal semantics.

Homotopy Type Theory (HoTT) is a new research field in the study of foundations of mathematics, among other related things. It was first initiated by Vladimir Voevodsky who, with others, has organised a special year about this at the Institute of Advanced Study in Princeton that has resulted in the HoTT book [40], among other things. HoTT extends MLTT with two things: the univalence axiom and higher inductive types. In particular, as a part of a larger development, Voevodsky has studied a notion of proposition as a type whose objects are all the same, which is later on coined in the HoTT project as *mere propositions*.

What we shall propose and study is a subsystem of the HoTT type theory: the system will be called MLTT$_h$ – it only extends MLTT with the logic of mere propositions, called h-logic. We shall briefly show, by giving simple examples, that MLTT$_h$ may adequately be used for MTT-semantics.

3.1 MLTT with PaT Logic: a Problem

Although the basic idea of MLTT with PaT logic is based on the propositions-as-types principle, Martin-Löf has gone a step further: not only every proposition is a type, but vice versa: every type is also a proposition. In other words, with the standard PaT logic of MLTT, propositions and types are *identified*. As explained in the introduction, this identification has caused a problem in incorporating proof irrelevance. If proof irrelevance were imposed for every proposition in MLTT, then unfortunately every type, which is also a proposition (in Martin-Löf's sense), would collapse as well: if a and b are two objects of type A, we'd have that a and b are the same because type A is a proposition. This is obviously absurd and unacceptable..

As we saw in §2.3, in an impredicative type theory such as UTT, the distinction between propositions and types is clear: in such a type theory, there is the type $Prop$ of all logical propositions: a type is a logical proposition if, and only if, it is an object of type $Prop$. Therefore, a principle of proof irrelevance can be stated and imposed in a straightforward way by a rule like $(*)$ in §2.3. Such a rule cannot be formulated, and hence unavailable, in MLTT with PaT logic. As pointed out in [25], Martin-Löf's type theory with PaT logic is inadequate for MTT-semantics, because it is impossible for one to impose a principle of proof irrelevance and, as explained, proof irrelevance would be needed to obtain correct identity criteria for CNs.

3.2 H-logic in HoTT

In HoTT, a logical proposition is a type whose objects are all propositionally equal to each other. To distinguish them from other types, which in MLTT are also called propositions, they are sometimes called *mere propositions*.

Definition 1 (mere proposition [40]) A type A is a *mere proposition* if for all $x, y : A$ we have that x and y are equal.

Formally, let U be the smallest universe in MLTT and $A : U$. Then A is a mere proposition in h-logic if the following is true:

$$\mathsf{isProp}(A) = \Pi x, y{:}A.\ Id_A(x, y),$$

where Id is the propositional equality (called Id-type) in MLTT. We can define the type of mere propositions in U to be the following Σ-type:

$$Prop_U = \Sigma X{:}U.\ \mathsf{isProp}(X).$$

In the following, we shall omit U and write PROP for $Prop_U$. Note that PROP is different from $Prop$ in an impredicative type theory like UTT: $Prop$ contains all logical propositions in the type theory while PROP does not – it only contains the mere propositions in the predicative universe U; sometimes, we say that PROP is the type of *small* mere propositions. Another thing to note is that an object of PROP is not just a type – it is a pair (A,p) such that A is a type in U and p is a proof of $\mathsf{isProp}(A)$ (i.e., A is a mere proposition).

The traditional logical operators can be defined and some of these definitions (e.g., those for disjunction and existential quantifier) use the following truncation operation that turns a type into a mere proposition.

- *Propositional Truncation.* Let A be a type. Then, there is a higher inductive type $||A||$ specified by the following rules, where the elimination operator κ_A satisfies the definitional equality $\kappa_A(f, |a|) = f(a)$:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash |a| : ||A||} \qquad \frac{\Gamma\ valid}{\Gamma \vdash \mathsf{isProp}(||A||)\ true} \qquad \frac{\Gamma \vdash \mathsf{isProp}(B) \quad \Gamma \vdash f : A \to B}{\Gamma \vdash \kappa_A(f) : ||A|| \to B}$$

Note that $||A||$ is a higher inductive type and, in particular, in turning a type A into a mere proposition $||A||$, one imposes that there is a proof of $\mathsf{isProp}(||A||)$, i.e.,

$\|A\|$ is a mere proposition – in other words, every two proofs of $\|A\|$ are equal (propositionally).[11]

The traditional logical operators can now be defined for h-logic as follows, where we denote them by means of an extra dot-sign: for example, $\dot{\wedge}$ for the conjunction connective in h-logic.

- $true = \mathbf{1}$ (the unit type).
- $false = \emptyset$ (the empty type).
- $P \mathbin{\dot{\wedge}} Q = P \times Q$.
- $P \mathbin{\dot{\supset}} Q = P \rightarrow Q$.
- $\dot{\neg}\, P = P \rightarrow \emptyset$.
- $\dot{\forall} x{:}A.P(x) = \Pi x{:}A.P(x)$.
- $P \mathbin{\dot{\vee}} Q = \|P + Q\|$.
- $\dot{\exists} x{:}A.P(x) = \|\Sigma x{:}A.P(x)\|$.

Please note that the typing operators on the right hand side are those used in MLTT to define the corresponding logical operators.

The reader may have noticed that the truncation operation is only used in the last two cases (disjunction and existential quantification), but not for defining the other logical connectives: for example, the logical conjunction $P \mathbin{\dot{\wedge}} Q$ is directly defined as the product type $P \times Q$, rather than $\|P \times Q\|$. The reason is that $\times$, the product typing operator, *preserves* the property of being mere propositions: if P and Q are mere propositions, so is $P \times Q$. This property of preservation also holds for the operators such as implication and universal quantification. However, it does not hold for disjunction or existential quantification: for example, even when P and Q are mere propositions, $P + Q$ is not a mere proposition and, therefore, the truncation operator has to be used to turn $P + Q$ into a mere proposition $\|P + Q\|$.

3.3 MLTT$_h$ and Its Adequacy for Formal Semantics

MLTT$_h$ extends Martin-Löf's type theory (MLTT) (see Part III of [34] for its formal description) with the h-logic in HoTT [40], as described above. MLTT$_h$ does not include other extensions of MLTT in the HoTT project: in particular, it does not have the univalence axiom or any other higher inductive types except those in h-logic. Since it is a subsystem of the HoTT type theory, MLTT$_h$ has nice meta-theoretic properties including logical consistency.

We claim that MTT-semantics can be done adequately in MLTT$_h$. Since in MLTT$_h$ there is the totality Prop of (small) mere propositions, we can approximate the notion of predicate as follows: a predicate over type A is a function of type $A \rightarrow$ Prop. Therefore, we can interpret linguistic entities such as verb phrases, modifications by intersective adjectives, etc. as we have done before based on UTT. For example, the modified CN (16) can be interpreted as (17), where $hs : Man \rightarrow$ Prop is a predicate

[11] For people who are familiar with type theory, this implies that canonicity fails to hold for the resulting type theory.

in MLTT$_h$ expressing the property of being handsome:

(16) handsome man
(17) $\Sigma m{:}Man, \pi_1(hs(m)))$

More precisely, for any man $m : Man$, $hs(m)$ is a pair (A, p) where A is a type representing that m is handsome and p is a proof term showing that A is a mere proposition. That is why we have to use the operator π_1 of first projection to get the first component of $hs(m)$ to form the Σ-type.

Proof irrelevance is built-in in the notion of mere proposition. In h-logic as described above, every two proofs of a mere proposition are equal (by definition, for the propositional equality Id). In particular, this is imposed for $\|A\|$ when a type A, which may not be a mere proposition, is turned into a mere proposition $\|A\|$. For instance, considering the semantic interpretation (17) of (16), we have that two handsome men are the same if, and only if, they are the same man, because any two proof terms of the mere proposition $\pi_1(hs(m))$ are the same. Therefore, the problem described in §2.2 and §3.1 is solved satisfactorily in MLTT$_h$.[12]

3.4 Most in MLTT$_h$

MLTT$_h$ also contains a weak version of the existential quantifier – the operator $\dot{\exists}$ as defined above. Therefore, the sentence (12), repeated below as (18), can be given the semantic interpretation (19) in MLTT$_h$, where $\dot{\exists}$ is used as the weak existential quantifier.

(18) Most farmers who own a donkey beat it.
(19) $Most\ z : [\Sigma x{:}F\dot{\exists} y{:}D.\ own(x, y)].$
$\quad\quad \forall y' : [\Sigma y{:}D.own(\pi_1(z), y)].\ beat(\pi_1(z), \pi_1(y'))$

Note that the above interpretation (19) in MLTT$_h$ is similar to (14) in UTT, with the only difference that $\exists$ in (14) is changed into $\dot{\exists}$ in (19).

4 Conclusion

In this paper, we have discussed that, in order to obtain adequate identity criteria for CNs, a principle of proof irrelevance should be adopted in a type-theoretical semantics. In particular, we showed that, unlike impredicative type theories like UTT, this presents a problem for predicative type theories such as MLTT. The paper then proceeds to show how one may extend MLTT by means of h-logic, the logic

[12] Of course, we recognise that MLTT$_h$ is a *proper* extension of MLTT, although this seems to be the best one could do (but further research may be needed to see whether it is possible to do otherwise).

of mere propositions studied in the HoTT project, to obtain MLTT_h, which is then claimed to be an adequate foundational language for MTT-semantics.

Usually, we have included Martin-Löf's type theory as one of the MTTs employable for MTT-semantics (just like impredicative type theories such as UTT). This paper may be regarded as a justification for this practice. Of course, for this, MLTT need be extended with h-logic, rather than using its original standard PaT logic.[13] It should be emphasised that further study is needed to justify our claim that MLTT_h can adequately deal with all the semantic matters as studied based on UTT, although intuitively we do not see any serious problems. To mention a potential issue: in a predicative type theory, formally there is no totality of all propositions (and hence no totality of predicates) – one can only have *relative* totalities of propositions or predicates using predicative universes (cf., PROP in §3.2). This is not ideal but it is to be seen whether it causes any serious problems.

The existence of both strong and weak sums (Σ and $\exists$) in type theories like UTT and MLTT_h has brought a new light in semantic treatments for sentences involving unbound anaphora [32], as some examples about *most* have illustrated in this paper. However, a general study in this respect is still needed to see how far one can go.

Acknowledgements Partially supported by EU COST Action EUTYPES (CA15123, Research Network on Types).

References

1. Agda. The Agda proof assistant (developed at Chalmers, Sweden). `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php`, 2008.
2. M. Baker. *Lexical categories: Verbs, nouns and adjectives*, volume 102. Cambridge University Press, 2003.
3. D. Bekki. Representing anaphora with dependent types. *LACL 2014, LNCS 8535*, 2014.
4. E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, 1967.
5. P. Boldini. Formalizing context in intuitionistic type theory. *Fundamenta Informaticae*, 42(2):1–23, 2000.
6. P. Callaghan and Z. Luo. An implementation of LF with coercive subtyping and universes. *Journal of Automated Reasoning*, 27(1):3–27, 2001.
7. S. Chatzikyriakidis and Z. Luo. Identity criteria of common nouns and dot-types for copredication. *Oslo Studies in Language*, 10(2), 2018.
8. S. Chatzikyriakidis and Z. Luo. *Formal Semantics in Modern Type Theories*. Wiley & ISTE Science Publishing Ltd., 2019. (to appear).
9. A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(1), 1940.
10. R. Cooper. Records and record types in semantic theory. *J. Logic and Compututation*, 15(2), 2005.
11. The Coq Development Team. *The Coq Proof Assistant Reference Manual (Version 8.0), INRIA*, 2004.
12. H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland Publishing Company, 1958.

[13] Although the current work has not been published until now, its idea, i.e., using HoTT's h-logic instead of the PaT logic, has been in the author's mind for a long time. This has partly contributed to the decision of including Martin-Löf's type theory as one of the MTTs for MTT-semantics.

13. R. Dapoigny and P. Barlatier. Modeling contexts with dependent types. *Fundamenta Informaticae*, 21, 2009.
14. G. Frege. *Grundlagen der Arithmetik*. Basil Blackwell, 1884. (Translation by J. Austin in 1950: The Foundations of Arithmetic).
15. D. Gallin. Intensional and higher-order modal logic: with applications to Montague semantics. 1975.
16. P. Geach. *Reference and Generality*. Cornell University Press, 1962.
17. J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1), 1991.
18. J. Grudzińska and M. Zawadowski. Generalized quantifiers on dependent types: A system for anaphora. *S. Chatzikyriakidis and Z. Luo (eds.). Modern Perspectives in Type-Theoretical Semantics*, 2017.
19. A. Gupta. *The Logic of Common Nouns*. Yale University Press, 1980.
20. I. Heim. File change semantics and the familiarity theory of definiteness. *In R. Bäuerle et al (eds.) Meaning, Use and Interpretation of Language*, 1983.
21. W. A. Howard. The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic*. Academic Press, 1980.
22. H. Kamp. A theory of truth and semantic representation. *In J. Groenendijk et al (eds.) Formal Methods in the Study of Language*, 1981.
23. Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.
24. Z. Luo. Type-theoretical semantics with coercive subtyping. *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, 2010.
25. Z. Luo. Common nouns as types. In D. Bechet and A. Dikovsky, editors, *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*, 2012.
26. Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.
27. Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? *Invited talk at Logical Aspects of Computational Linguistics 2014 (LACL 2014), Toulouse. LNCS 8535*, pages 177–188, 2014.
28. Z. Luo and P. Callaghan. Coercive subtyping and lexical semantics (extended abstract). *LACL'98 (extended abstracts), available in request to the first author or as* `http://www.cs.rhul.ac.uk/home/zhaohui/LACL98.abstract.ps`, 1998.
29. Z. Luo and R. Pollack. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Dept of Computer Science, Univ of Edinburgh, 1992.
30. P. Martin-Löf. An intuitionistic theory of types: predicative part. In H.Rose and J.C.Shepherdson, editors, *Logic Colloquium'73*, 1975.
31. P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
32. F. Moltmann. Unbound anaphoric pronouns: E-type, dynamic and atructured-propositions approaches. *Synthese*, 153, 1983.
33. R. Montague. *Formal Philosophy*. Yale University Press, 1974. Collected papers edited by R. Thomason.
34. B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990.
35. A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
36. C. Retoré. The Montagovian generative lexicon $\lambda\mathsf{Ty}_n$: a type theoretical framework for natural language semantics. In R. Matthes and A. Schubert, editors, *Proc of TYPES2013*, 2013.
37. G. Sundholm. Proof theory and meaning. *D. Gabbay and F. Guenthner (eds.). Handbook of Philosophical Logic, Vol III*, 1986.
38. G. Sundholm. Constructive generalized quantifiers. *Synthese*, 79(1):1–12, 1989.
39. R. Tanaka. Generalized quantifiers in dependent type semantics. Talk given at Ohio State University, 2015.
40. The Univalent Foundations Program. Homotopy type theory: Univalent foundations of mathematics. Technical report, Institute for Advanced Study, 2013.
41. T. Xue, Z. Luo, and S. Chatzikyriakidis. Propositional forms of judgemental interpretations. *Proc of Workshop on Natural Language and Computer Science. Oxford*, 2018.