On Type-Theoretical Semantics of Donkey Anaphora

Zhaohui Luo[⋆]
Royal Holloway, University of London
zhaohui.luo@hotmail.co.uk

Abstract. Donkey sentences are among the challenging examples that present a difficult problem in compositional logical semantics. Their semantic treatment is one of the early applications of dependent type theory to linguistic semantics, where the type constructor Σ is used to play a double role, both as the existential quantifier and as the subset constructor. However, it is known that this method is inadequate because it fails to deal with counting properly – this is also called the cardinality problem. In this paper, we analyse this problem to explicate that it originates from the use of Σ to play the double role and propose to consider the semantics of donkey sentences in a type theory with both Σ and a traditional existential quantifier. It is shown that, with both operators, donkey sentences can be given adequate semantic interpretations which, in particular, take care of counting properly.

1 Introduction

Donkey sentences, as first studied by Geach [14] and exemplified in (1), where an anaphoric expression refers to an existentially quantified entity, are among the challenging examples that present a difficult problem in compositional logical semantics.

(1) Every farmer who owns a donkey beats it.

Their studies (and that of trans-sentential anaphora) have led to the development of dynamic semantics such as Discourse Representation Theory (DRT) [22, 23, 17] and Dynamic Predicate Logic (DPL) [16], which are widely accepted by linguistic semanticists as a framework with proper means to deal with anaphora and, however, require us to consider substantial changes of the underlying logical systems for formal semantics.[1] (See [3], among others, for a recent summary of the dynamic approach to donkey anaphora.)

In the mid-80s, as one of the early applications of dependent type theory in logical semantics, researchers such as Mönnich [32] and Sundholm [39] have proposed to use Martin-Löf's type theory [30] to deal with donkey anaphora, where

⋆ This work is partially supported by the EU research network EUTypes (COST Action CA15123).

[1] For example, DPL is a rather non-standard logical system: among other things, it is non-monotonic and the notion of dynamic entailment fails to be reflexive or transitive [16, 15].

Σ-types are employed to play a double role, representing the existential quantifier as well as the subset constructor. Σ-types $\Sigma x{:}A.P(x)$ are also called *strong sums*, as opposed to the traditional existentially quantified formulas $\exists x{:}A.P(x)$ which are called *weak sums*. They are called so because, from an object of the strong sum, one can obtain its witness by means of a projection operation, while this is not possible for the weak sum. It is because of the availability of witness projection that an anaphoric reference can be obtained from an object of a Σ-type, while this is not possible for an existential quantification in the traditional case (and hence the problem in the first place). However, it is known that this method of using Σ-types to deal with donkey anaphora suffers from a problem of counting [40, 41], also called the cardinality problem, and fails to provide us an adequate solution (see §2 for more details).

In this paper, we contend that the cardinality problem of the above type-theoretical approach has come from a double role played by Σ, as an existential quantifier, on the one hand, and as a structural mechanism to represent collections (subsets) of objects, on the other. These two roles should be separated and played by different type constructors. But in traditional logics (for example, first-order logic or simple type theory) or in Martin-Löf's type theory, only either $\exists$ or Σ exists, not both, and therefore there is no way to consider such a separation of labour. We show that, in a type theory with both strong and weak sums, donkey sentences can be given adequate semantics in which counting is taken into proper account.

Our proposal is also linked to the research on different readings of donkey sentences and, in particular, the strong and weak readings as studied by Chierchia and others [7, 8, 24]. Also, donkey anaphora are closely related to (and, for some researchers, they are examples of) the so-called E-type anaphora, as first studied by Evans [12, 13], which may be interpreted by means of descriptions (see, for example, [34] for a recent discussion). It is not surprising that Σ-types are essentially useful in semantic interpretations of donkey sentences since they have close links to descriptions [30, 5, 31] and we shall give some brief discussions about this.

Combining strong and weak sums in type theory is a subtle matter that needs us to tread carefully, for otherwise we may easily slip into inconsistency or other problems. Although there are already some results in this respect [25], they are not widely known, partly because of their technical nature. We shall discuss this issue briefly, explaining both possibilities and potential problems.

In the following section §2, we shall explain the concepts of strong and weak sums, define the notion of cardinality for finite types, and illustrate the counting/cardinality problem of the Σ-type interpretation of donkey sentences. §3 describes our proposed solution with both strong and weak sums: in §3.1, we first briefly describe the type structure of type theory UTT which has both Σ and $\exists$, followed by §3.2 to explain how donkey sentences may be interpreted in UTT, and then by §3.3 which considers E-type anaphora and the use of Σ-types for descriptions. In §4, we shall briefly explain the possibilities and potential problems in having both strong and weak sums in a type theory, followed by some concluding remarks, including that about considering dynamics in type theory.

2 Strong and Weak Sums in Type Theory

In this section, we explicate the concepts of weak sums (for example, traditional existential quantifiers), to illustrate the original problem of using existentially quantified variables to interpret donkey anaphora, and strong sums (Σ-types), to explain the counting problem when using Σ-types to play a double role in interpreting donkey sentences, as proposed by Mönnich [32] and Sundholm [39].

Weak sums (existential quantifiers). Under the Curry-Howard propositions-as-types principle [11, 21], traditional existentially quantified formulas are examples of weak sum types of the form $\exists x.P(x)$. In first-order logic, depending on whether it is intuitionistic or classical, the existential quantifier can be introduced directly or defined by means of the universal quantifier together with negation, respectively. In higher-order logic (or simple type theory) as used in Montague's semantics, where there is an impredicative type $\mathbf{t}$ of all formulas, it can be either directly introduced or defined by means of the universal quantifier as in (2), where x ranges over entities and X over formulas of type $\mathbf{t}$.[2]

(2) $\exists x.P(x) = \forall X.\ (\forall x.(P(x) \Rightarrow X)) \Rightarrow X.$

It is known that, given a proof of $\exists x.P(x)$, although one knows that there is an entity such that P holds, in the logical calculus one cannot find out which entity it is. It is because of this that an anaphoric reference to an existentially quantified entity becomes problematic. For example, in a traditional compositional semantics, the donkey sentence (1) would obtain (3) as its interpretation, where the free variable y in $beat(x, y)$ is out of the scope of the existential quantifier $\exists$ and is different from the bound variable y bounded by $\exists$.

(3) (#) $\forall x.\ [farmer(x)\ \&\ \exists y.(donkey(y)\ \&\ own(x, y))] \Rightarrow beat(x, y)$

This illustrates the original problem in interpreting donkey sentences, as mentioned at the beginning of Introduction.

Strong sums (Σ-types). Σ is a dependent type constructor. If A is a type and B is a family of types that depend on objects of type A, then $\Sigma x{:}A.B(x)$ is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. Σ-types are associated with the projection operators π_1 and π_2 so that, for (a, b) of type $\Sigma x{:}A.B(x)$, $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Formally, Σ-types are governed by the inference rules in Appendix A.

[2] The fact that other logical operators can be defined in higher-order logical systems by means of universal quantifier was discovered in the 60s by Prawitz [35] (and several others, independently) and, this is the same in an impredicative type theory – see definitions in (8-9) and Appendix C.

Besides being useful mechanisms to organise structures in various applications, Σ-types may also play other roles. For example, in Martin-Löf's type theory [30], Σ also plays the role of existential quantifier in its logic.[3] This is the basis for Mönnich [32] and Sundholm [39] to propose using Σ-types to interpret donkey sentences.[4] For instance, the donkey sentence (1) can be interpreted as (4), in which F_Σ, as defined in (5), is the type intended to represent the collection of donkey-owning farmers, where F and D are the types that interpret farmer and donkey, respectively.

(4) $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

(5) $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x, y)$

Σ-types are strong in the sense that from a proof of $\Sigma x{:}A.P(x)$ one can preform the first projection operation to obtain the witness of this 'existentially' quantified formula and it is because of this, if Σ is used as existential quantifier, one can project out its witness from a proof term of the Σ-type, even outside the Σ-type concerned (the terms $\pi_1(z)$ and $\pi_1(\pi_2(z))$ in (4) are such examples).

The type F_Σ in (5) contains two occurrences of Σ and they play two different roles: the first acts as a structural mechanism to represent the collection of the farmers who own donkeys and the second as the existential quantifier to say that there exists a donkey owned by the farmer concerned. As we shall see below, using Σ to play this double role is problematic. In particular, F_Σ is in fact representing a collection whose cardinality (the number of its objects) is different from that of the collection of donkey-owning farmers and, therefore, the semantic interpretation (4) of (1) is inadequate [40, 41].

Counting and cardinality of finite types. When a type A is finite in the sense that it has finitely many objects, it is possible to define its cardinality $|A|$ as the number of its objects. Formally, a type is finite if, for some n, it is isomorphic to $Fin(n)$, the type with exactly n objects – see Appendix B. For example, the cardinality of a finite Σ-type is the number of pairs in the type.

The problem of counting (or the cardinality problem) can be illustrated by considering the sentence in (6), where the quantifier Every in (1) is replaced by Most. Its formal semantics by means of Σ-types in Martin-Löf's type theory, as proposed by Mönnich [32] and Sundholm [39], is given in (7), which can be seen as obtained by replacing $\forall$ by the quantifier $Most_S$. Here, $Most_S$ is defined by Sundholm in [40] (S in $Most_S$ for Sundholm) so that, for a finite type A, $Most_S\ x{:}A.P(x)$ is true if, and only if, more than half of the objects in A satisfy P.

(6) Most farmers who own a donkey beat it.

[3] This is concerned with intuitionistic philosophy – a strongly minded intuitionist may believe that the witness of a proven existentially quantified formula can be obtained internally in a logical calculus. We omit further discussions here.

[4] In formal semantics based on modern type theories, CNs such as 'farmer' and 'donkey' are interpreted as types (rather than predicates). This was first proposed by Mönnich [32] and Sundholm [39] and further elaborated in [36, 26].

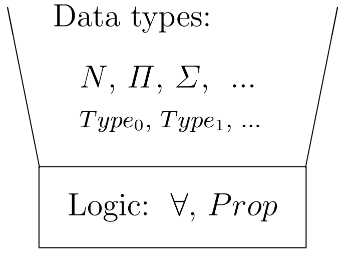

Fig. 1. The type structure in UTT.

(7) $Most_S\ z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$, where F_Σ is defined in (5).

Let us now consider the cardinality of F_Σ, as defined in (5). Because of the second Σ in F_Σ, $|F_\Sigma|$ is not that of the collection of donkey-owning farmers; instead, to calculate $|F_\Sigma|$, we'd have to count every triple (x, y, p) of farmers x, donkeys y and proofs p that x owns y. For example, if there are ten farmers, one of whom owns twenty donkeys and beats all of them, and the other nine own one donkey each and do not beat their donkeys. Then, $|F_\Sigma| \geq 29$ (it is an inequality because, if farmer x owns donkey y, there may be more than one proof that x owns y), but the number of farmers who do not beat their donkeys is 9. Therefore, the above semantics (7) of (6) would be true in such a case, which is obviously incorrect.[5]

3 Donkey Anaphora: Type-Theoretical Semantics with Both Strong and Weak Sums

In this section, we shall first introduce the dependent type theory UTT (Unifying Theory of dependent Types) [25], which has both strong and weak sums, and then show how donkey sentences like (1) and (6) can be interpreted in UTT, giving adequate treatments for different readings and taking care of counting in a proper way as well.

3.1 UTT: an impredicative type theory

The type structure of UTT consists of two parts: the world of data types and that of logical propositions (see Fig. 1). It contains various types such as dependent product types (Π-types), strong sum types (Σ-types), the type N of natural numbers, the predicative universes $Type_i$ ($i \in \omega$), and many others.

[5] This is similar to the 'proportion problem' when one uses DRT to interpret such donkey sentences, where one counts farmer-donkey pairs rather than the donkey-owning farmers. See [24] and [3], among others, for discussions.

UTT also contains an impredicative type universe *Prop* of logical propositions which provide means to describe logical properties of objects of any type (see Appendix C). Formally, UTT can be considered as the combination of Martin-Löf's (intensional) type theory [29, 33] with Coquand-Huet's Calculus of Constructions [10]. In computer science, type theories such as UTT have been implemented in theorem proving systems (called proof assistants) for formalisation of mathematics and verification of programs, and recently, they have been used for formal reasoning based on linguistic semantics (see, for example, [6]).[6]

Note that UTT contains both strong sums $\Sigma x{:}A.B(x)$ (Σ-types) as 'data types' and weak sums $\exists x{:}A.P(x)$ (existentially quantified types) as logical propositions, and this is essential when considering semantic interpretations of donkey sentences in §3.2 below.

Logic and proof irrelevance. In UTT, a type is a logical proposition if it is of type *Prop*. The type universe *Prop* is impredicative and, therefore, the other logical operators can be defined by means of the operator $\forall$ for universal quantification (cf., Footnote 2). For example, the conjunction operator and the existential quantifier $\exists$ can be defined as in (8) and (9), respectively, and the definitions of the other operators can be found in Appendix C.

(8) $P \ \wedge \ Q = \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X$

(9) $\exists x : A.P(x) = \forall X{:}Prop.(\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X$

The principle of *proof irrelevance* says that any two proofs of the same logical proposition should be the same. For instance, it implies that, for farmer x and donkey y, any two proof terms of the proposition $own(x, y)$ should be the same. It has been shown that, when employing a type theory for natural language semantics, proof irrelevance should be enforced [26, 27]. Note that, because in UTT there is a clear distinction between logical propositions and other types (the former being those of type *Prop*), it is straightforward to introduce proof irrelevance by means of the following rule [43, 26]:

$$\frac{P : Prop \quad p : P \quad q : P}{p = q : P}$$

Intuitively, it says that, if P is a logical proposition and if p and q are proof terms of P, then p and q are the same. In particular, according to the above rule, every proposition of type *Prop* is either an empty type or a singleton type. In terms of cardinality, we have $|P| \leq 1$ for every $P : Prop$ and, therefore, if A is finite and $Q : A \rightarrow Prop$ is a predicate over A, then we have

(10) $|\Sigma x{:}A.Q(x)| \leq |A|$.

[6] There are several proof assistants based on type theories including Agda [1] based on Martin-Löf's type theory, Coq [9] implementing the type theory pCIC, and Lego/Plastic [28, 4] implementing UTT. It may be worth remarking that Coq's type system pCIC is very similar to UTT – this is especially the case after Coq's universe Set became predicative in 2004 (it was impredicative in earlier versions).

3.2 Semantic interpretations of donkey anaphora in UTT

When a type theory has both strong and weak sums (Σ-types and $\exists$-propositions as in UTT), together with proof irrelevance, there is a new way to semantically interpret donkey sentences, which takes care of counting adequately. We'll use the example (6), which is repeated as (11) below, to explain.

(11) Most farmers who own a donkey beat it.

In §2, we have shown that, because in Martin-Löf's type theory Σ is used to play a double role, the semantic interpretation (7) of (11) is inadequate because it gets counting wrong. In that definition, we have used quantifier $Most_S$ defined in Martin-Löf's type theory and, here, we can define a semantic interpretation of the quantifier 'most' in UTT in a similar fashion as in [40] but with a crucial difference: instead of Σ, we shall use $\exists$ as defined in (9) as the existential quantifier and, intuitively, for a finite A, $Most\ x{:}A.P(x)$ also means that more than half of the objects in A satisfy P. Note that, $Most_S\ x{:}A.P(x)$ is a non-propositional type, but $Most\ x{:}A.P(x)$ is a logical proposition of type $Prop$ (see Appendix D for the formal definition).

Having defined $Most$ in UTT, we can now interpret the donkey sentence (11) as (12), in which $F_\exists$ is defined in (13):

(12) $Most\ z : F_\exists.\ \forall y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

(13) $F_\exists = \Sigma x{:}F.\ \exists y{:}D.own(x, y)$

Note that $|\exists y{:}D.own(x, y)| \leq 1$, that is, if $\exists y{:}D.own(x, y)$ is true, the cardinality of the proposition is 1. Therefore, the type $F_\exists$ correctly represents the collection of donkey-owning farmers, as intended, and the above semantics (12) is adequate and, in particular, it deals with counting correctly.

Researchers have studied different readings (in particular, strong and weak readings) of donkey anaphora, as studied by Chierchia [7, 8] and others. For instance, the strong and weak readings of (11) are (14) and (15), respectively:

(14) Most farmers who own a donkey beat *the donkeys they own.*

(15) Most farmers who own a donkey beat *some donkeys they own.*

The above interpretation (12) of (11) is a strong one, interpreting (14) directly: most donkey-owning farmers beat *all* donkeys they own. A weaker interpretation of its weak reading (15) would be (16), obtained from (12) by changing $\forall$ into $\exists$:

(16) $Most\ z : F_\exists.\ \exists y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

People have also considered more sophisticated examples with donkey anaphora. For example, (17) is one of them, taken from Brasoveanu's thesis [2, 3], in which the readings for the donkey anaphora are different ('a TV' having a strong reading and 'a credit card' a weak one). Its type-theoretical semantics with both strong and weak sums is given in (18).

(17) Every person who buys a TV and has a credit card uses it to pay for it.

(18) $\forall z : \Sigma x{:}Person.\ \exists y_1{:}TV.\ buy(x, y_1) \wedge \exists y_2{:}Card.\ own(x, y_2)$
$\quad \forall y : \Sigma y_1{:}TV.\ buy(\pi_1(z), y_1)$
$\quad \exists y' : \Sigma y_2{:}Card.\ own(\pi_1(z), y_2).$
$\quad\quad pay(\pi_1(z), \pi_1(y), \pi_1(y'))$

One may change the quantifier Every in (17) into Most (and make other minor changes in the sentence to make it grammatically correct by changing, for example, 'person/buys/has/uses' into 'persons/buy/have/use') and, in that case, we can use the quantifier *Most* defined in UTT (see Appendix D) to interpret the sentence and the resulting interpretations take care of counting correctly as well.

3.3 E-type anaphora

Here, we discuss, albeit rather briefly, the so-called E-type anaphora to which donkey anaphora are closely related (and, for some researchers, donkey anaphora are examples of E-type anaphora).[7] E-type anaphora are first studied by Evans [12, 13], and further discussed by many, including [18] among others. They can be interpreted by means of descriptions [37, 38] (see, for example, [34] for a recent discussion). An example, due to Evans, is (19). Note that the pronoun 'they' in (19) is not bound by 'Few' for otherwise the semantic interpretation of the sentence would be incorrect. A common conceptual answer, proposed by Evans, is that these pronouns are *descriptive* in that they can be paraphrased by means of descriptions as exemplified in (20) that paraphrases (19).

(19) Few congressmen admire Kennedy, and they are very junior.

(20) Few congressmen admire Kennedy, and *the congressmen that do admire Kennedy* are very junior.

As pointed out by Martin-Löf [30], strong sum types (Σ-types) are related to descriptions, because he regards them as logical propositions as well. If one considers $\Sigma x{:}A.B(x)$ as the existentially quantified formula and because it is strong, therefore its first projection operator π_1 gives us an internal means of obtaining the witness from a proof of the existentially quantified formula. As explained in §2, this is stronger than the traditional existential operator $\exists$ for which such a projection operator does not exist, and it is exactly because of this that Σ offers a form of description, as pointed out by Martin-Löf and further studied by Carlström [5] and Mineshima [31]. For example, the E-type example (19) may be interpreted as (21), either in Martin-Löf's type theory or in UTT, where we assume that the quantifier *Few* has been defined and C is the type that interprets 'congressman':

(21) $Few\ x{:}C.admire(x, K) \wedge \forall z{:}[\Sigma x{:}C.admire(x, K)].junior(\pi_1(z))$

[7] Here, I use the term 'E-type' for a kind of anaphora, rather than an approach to solving anaphora ('the E-type approach' as people often put it).

However, it should be made clear that, since Σ-types are not the same as traditional existentially quantified formulas, it is unclear how far one may go in analysing E-type anaphora by means of Σ-types. Actually, it would not go very far since, as analysed above, using Σ as existential quantifier does cause problems such as counting, which would show up in context of E-type anaphora as well.

4 Combining Strong and Weak Sums in Type Theories

It is worth mentioning that combining both strong and weak sums at the same time in a type theory is a subtle matter and, if not careful, it is easy to get into problems. We summarise some of the known results in this section so that the readers can become aware of them. However, because of the technical nature of the results (and their proofs), we will only be brief and sketch some of them briefly and informally.[8]

Adding strong sums to impredicative type theories. First, let's consider how to add strong sums (Σ-types) to an impredicative type theory, where one already has the weak sum ($\exists$-propositions). Note that, as briefly described in §3.1, although it has both Σ-types (as data types) and $\exists$-propositions (as logical formulas), UTT does not have 'Σ-propositions' because the so-called 'large Σ-propositions' would lead to inconsistency and the so-called 'small Σ-propositions' would make the weak sum types become strong. Consider, for example, to add large Σ-propositions into the impredicative universe $Prop$ by means of the following rule (together with those for introduction and eliminations that we omit):

$$(*) \qquad \frac{A\ type \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

It turns out that such Σ-propositions cannot be consistently added – if they were added using the above rule $(*)$ (and related ones), the resulting type theory would be inconsistent in the sense that even the false proposition would become provable [19, 25].

One may want to add Σ-propositions (so-called small Σ-propositions) by a rule like the following, this time restricting A to be a proposition of type $Prop$:

$$\frac{A : Prop \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

Although the resulting type theory may be consistent[9], there is another problem: the addition of such small strong sum as propositions in $Prop$ would make the

[8] These results, except that about MLTT$_h$, are discussed by the author in [25] (§2.3.2 of [25], in particular), from which the interested reader may obtain more information.
[9] This consistency is a folklore – most researchers, including the author, believe that it is the case, although the author has not seen a proof of it.

weak sum proposition $\exists x{:}A.P(x)$ become strong (rather unexpectedly!) in the sense that there is now an internal function in the type theory that, from a proof of $\exists x{:}A.P(x)$, returns an object $a : A$ such that $P(a)$ holds. That would mean that the traditional existential quantifier is not weak anymore – such a side effect is of course problematic and, in particular, would make the interpretation method we proposed above in §3.2 fail to deal with counting correctly.

Therefore, neither of the above large or small Σ-propositions is a viable possibility and, put in another way, the approach taken in UTT seems to be the only viable approach to adding Σ to an impredicative type theory.

Adding weak sums to predicative type theories. Now, let's consider how to add weak sums to a predicative type theory such as Martin-Löf's type theory MLTT [29, 33],[10] where we already have strong sum types (Σ-types).

We first remark that defining a form of 'weak sum types' in a similar way as the definition in (9), but with the difference of replacing the impredicative universe *Prop* by a predicative universe U_i in MLTT, would not work. This would amount to the definition in (22), where $A : U_i$ and $B : A \to U_i$:

(22) $\exists_i x{:}A.B(x) = \Pi X{:}U_i.\ (\Pi x{:}A.(B(x) \to X)) \to X$

The above definition (22) does not work because it fails to deliver a weak sum type – $\exists_i x{:}A.B(x)$ thus defined is actually strong (!) and, in fact, it is equivalent to the strong sum type $\Sigma x{:}A.B(x)$ (see §2.3.2 of [25] for a proof).

Therefore, it seems that the only possible way to add a weak sum type to MLTT is to consider MLTT$_h$, as proposed by the author in [27], where MLTT is extended with the h-logic for the HoTT type theory [20]. In MLTT$_h$, in particular, the existentially quantified formula is defined to be the truncation of the corresponding Σ-type, as in (23). (We omit the details here and interested readers may consult [27] and the references about HoTT [20] there.)

(23) $\dot{\exists} x{:}A.B(x) = \|\Sigma x{:}A.B(x)\|$

The existential quantifier $\dot{\exists}$ is a weak sum and, therefore, can be used to define the semantic interpretations of donkey sentences as proposed in §3.2. For example, employing MLTT$_h$ (instead of UTT) for our formal semantics, the donkey sentence (11) can be interpreted as (24), where $Most_h$ is defined as in Appendix D, but in MLTT$_h$, with $\dot{\exists}$ to replace $\exists$ in the definition.

(24) $Most_h\ z : F_{\dot{\exists}}.\ \forall y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

(25) $F_{\dot{\exists}} = \Sigma x{:}F.\ \dot{\exists} y{:}D.own(x, y)$

5 Concluding Remarks

In this paper, we have studied how to consider donkey anaphora in type theory, pointing out that the cardinality problem in the proposed type-theoretic semantics in Martin-Löf's type theory is due to the fact that strong sums (Σ-types)

[10] We use MLTT to denote Martin-Löf's *intensional* type theory as described in [29] or Part III of [33], not his extensional type theory [30].

are used to play a double role in the representation. We then show that, in a type theory (like UTT) with both strong and weak sums, donkey sentences can be given adequate semantics which, in particular, does not suffer from the cardinality problem. The paper then briefly discussed the issue of how to incorporate both strong and weak sums in type theory: it explains that, for impredicative type theories, it seems that UTT provides the only viable approach and, for predicative type theories, a proposal of studying MLTT_h provides a promising way forward.

It is worth pointing out that our analysis and proposal are given in a completely proof-theoretic fashion. This is rather different from the model-theoretic approaches that have been considered in the literature (see, for example, [3]). This work may be taken as a part of the more general endeavour of studying a type-theoretic approach to dynamics in semantics. Two remarks are in order here. The first is about a possible suggestion to extend a type theory into a 'dynamic type theory', just like extending the first-order logic to become dynamic predicate logic [16]. We do not think that this is a right way forward partly because, even if such a 'dynamic type theory' is possible (a big if), for type theory to lose its standard properties to become a non-standard logical system is too much a price to pay (cf., Footnote 1). Secondly, existing work on proof theory of dynamic semantic systems like DPL (see, for example, [42]) has shown the difficulties in doing proof theory for such non-standard systems. In logical semantics, it would be preferable, if we can, to stay with more standard logical systems.

Acknowledgement. I am very grateful to Justyna Grudzińska for a discussion about (6), the example involving 'most', which has motivated me to start the research reported in this paper. My thanks also go to the anonymous reviewers for their useful comments that have helped to improve the paper.

References

1. Agda: The Agda proof assistant (v2). URL: `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php` (2008)
2. Brasoveanu, A.: Structured Nominal and Modal Reference. Ph.D. thesis, The State University of New Jersey (2007)
3. Brasoveanu, A., Dotlacil, J.: Donkey anaphora: farmers and bishops. In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics (2021)
4. Callaghan, P., Luo, Z.: An implementation of LF with coercive subtyping and universes. Journal of Automated Reasoning 27(1), 3–27 (2001)
5. Carlström, J.: Interpreting descriptions in intensional type theory. The Journal of Symbolic Logic 70(2) (2005)
6. Chatzikyriakidis, S., Luo, Z.: Proof assistants for natural language semantics. In: International Conference on Logical Aspects of Computational Linguistics. pp. 85–98. Springer (2016)

7. Chierchia, G.: Anaphora and dynamic logic. ITLI Publication Series for Logic, Semantics and Philosophy of Language, LP90-07 (1990)
8. Chierchia, G.: Anaphora and dynamic binding. Linguistics and Philosophy 15 (1992)
9. Coq: The Coq Proof Assistant Reference Manual (Version 8.3), INRIA (2010)
10. Coquand, T., Huet, G.: The calculus of constructions. Information and Computation 76(2-3), 95–120 (1988)
11. Curry, H., Feys, R.: Combinatory Logic, vol. 1. North Holland Publishing Company (1958)
12. Evans, G.: Pronouns, quantifiers and relative clauses. Canadian Journal of Philosophy 7 (1977)
13. Evans, G.: Pronouns. Linguistic Inquiry 11(2) (1980)
14. Geach, P.: Reference and Generality: An Examination of Some Medieval and Modern Theories. Cornell University Press (1962)
15. Gillies, A.: (Re-)reading 'dynamic predicate logic' (electronic manuscript)
16. Groenendijk, J., Stokhof, M.: Dynamic predicate logic. Linguistics and Philosophy pp. 39–100 (1991)
17. Heim, I.: The Semantics of Definite and Indefinite Noun Phrases. Ph.D. thesis, University of Massachusetts (1982)
18. Heim, I., Kratzer, A.: Semantics in Generative Grammar. Blackwell (1998)
19. Hook, J., Howe, D.: Impredicative strong existential equivalent to Type:Type. Technical Report TR86-760, Cornell University (1986)
20. HoTT: Homotopy type theory: Univalent foundations of mathematics, the univalent foundations program. Tech. rep., Institute for Advanced Study (2013)
21. Howard, W.: The formulae-as-types notion of construction. In: Hindley, J., Seldin, J. (eds.) To H. B. Curry: Essays on Combinatory Logic. pp. 479–490. Academic Press (1980), (Notes written and distributed in 1969.)
22. Kamp, H.: A theory of truth and semantic representation. In J. Groenendijk et al (eds.) Formal Methods in the Study of Language pp. 189–222 (1981)
23. Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer (1993)
24. Kanazawa, M.: Weak vs. strong readings of donkey sentences and monotonicity inference in a dynamic setting. Linguistics and Philosophy 17(2) (1994)
25. Luo, Z.: Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press (1994)
26. Luo, Z.: Common nouns as types. In: Bechet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351 (2012)
27. Luo, Z.: Proof irrelevance in type-theoretical semantics. Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence (SCI) pp. 1–15 (2019), springer.
28. Luo, Z., Pollack, R.: LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Dept of Computer Science, Univ of Edinburgh (1992)
29. Martin-Löf, P.: An intuitionistic theory of types: predicative part. In: H.Rose, J.C.Shepherdson (eds.) Logic Colloquium'73 (1975)
30. Martin-Löf, P.: Intuitionistic Type Theory. Bibliopolis (1984)
31. Mineshima, K.: Aspects of Inference in Natural Language. Ph.D. thesis, Keio University (2013)
32. Mönnich, U.: Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen. Habilitation. University of Tübingen (1985)
33. Nordström, B., Petersson, K., Smith, J.: Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press (1990)

34. Nouwen, R.: E-type pronouns: congressmen, sheep and paychecks. In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics (2021)
35. Prawitz, D.: Natural Deduction, a Proof-Theoretic Study. Lmqvist and Wiksell (1965)
36. Ranta, A.: Type-Theoretical Grammar. Oxford University Press (1994)
37. Russell, B.: On denoting. Mind 14(56) (1905)
38. Russell, B.: Introduction to Mathematical Philosophy. George Allen and Unwin (1919)
39. Sundholm, G.: Proof theory and meaning. In: Handbook of philosophical logic, pp. 471–506. Springer (1986)
40. Sundholm, G.: Constructive generalized quantifiers. Synthese 79(1), 1–12 (1989)
41. Tanaka, R.: Generalized quantifiers in dependent type semantics. Talk given at Ohio State University (2015)
42. Veltman, F.: Proof systems for Dynamic Predicate Logic (2000)
43. Werner, B.: On the strength of proof-irrelevant type theories. Logical Methods in Computer Science 4(3) (2008)

A Rules for Σ-types

$$\frac{A\ type \quad B\ type\ [x{:}A]}{\Sigma x{:}A.B\ type}$$

$$\frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{(a,b) : \Sigma x{:}A.B}$$

$$\frac{p : \Sigma x{:}A.B}{\pi_1(p) : A} \qquad \frac{p : \Sigma x{:}A.B}{\pi_2(p) : [\pi_1(p)/x]B}$$

$$\frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{\pi_1(a,b) = a : A} \qquad \frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{\pi_2(a,b) = b : [a/x]B}$$

B Cardinality of Finite Types

We give the formal definition of finite types. It will use the auxiliary type $Fin(n)$ for which we define first.

The type $Fin(n)$, indexed by $n : N$ with N being the type of natural numbers, consists of exactly n objects and can be specified by means of the following introduction rules (we omit their elimination and computation rules):

$$\frac{n : N}{zero(n) : Fin(n+1)}$$

$$\frac{n : N \quad i : Fin(n)}{succ(n,i) : Fin(n+1)}$$

The cardinality of a finite type A, notation $|A|$, is defined to be n if, and only if, A is isomorphic to $Fin(n)$, that is, in the type theory concerned, there is a bijective function between A and $Fin(n)$. In particular, $|Fin(n)| = n$, since the identity function over $Fin(n)$ is bijective.

C Logic in UTT

The logic in UTT[11] consists of the impredicative universe $Prop$, specified by the following rules:

$$\frac{}{Prop\ type} \qquad \frac{P : Prop}{P\ type}$$

and the operator $\forall$ for universal quantification, specified by

$$\frac{A\ type \quad P : Prop\ [x{:}A]}{\forall x{:}A.P : Prop}$$

$$\frac{x{:}A \vdash b : P \quad P : Prop\ [x{:}A]}{\lambda x{:}A.b : \forall x{:}A.P}$$

$$\frac{f : \forall x{:}A.P \quad a : A}{f(a) : [a/x]P}$$

$$\frac{P : Prop\ [x{:}A] \quad b : P\ [x{:}A] \quad a : A}{(\lambda x{:}A.b)(a) = [a/x]b : [a/x]P}$$

In UTT, other logical operators can be defined by means of $\forall$ and here are some definitions (see, for example, §5.1 of [25]):

$$\begin{aligned}
P \Rightarrow Q &= \forall x : P.\ Q \\
\mathbf{true} &= \forall X : Prop.\ X \Rightarrow X \\
\mathbf{false} &= \forall X : Prop.\ X \\
P\ \wedge\ Q &= \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X \\
P\ \vee\ Q &= \forall X : Prop.\ (P \Rightarrow X) \Rightarrow (Q \Rightarrow X) \Rightarrow X \\
\neg P &= P \Rightarrow \mathbf{false} \\
\exists x : A.P(x) &= \forall X : Prop.\ (\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X \\
(a =_A b) &= \forall P : A \to Prop.\ P(a) \Rightarrow P(b)
\end{aligned}$$

D *Most* in UTT

Let A be a finite type with $|A| = n_A$, $P : A \to Prop$ a predicate over A, and $Fin(n)$ the types with n objects defined in Appendix B. Then, in UTT, the logical proposition $Most\ x{:}A.P(x)$ of type $Prop$ is defined as follows, where $inj(f)$ is a proposition expressing that f is an injective function:

$$\begin{aligned}
Most\ x{:}A.P(x) = \exists k : N.\ (k \geq \lfloor n_A/2 \rfloor + 1) \\
\wedge\ \exists f{:}Fin(k) \to A.\ inj(f) \wedge \forall x{:}Fin(k).P(f(x))
\end{aligned}$$

[11] One can find its definition in §9.2.1 of [25], where it is specified in terms of the logical framework LF.