A Model-Theoretic Reconstruction of Type-Theoretic Semantics for Anaphora

Matthew Gotham
http://orcid.org/0000-0002-6549-1248

University of Oslo, Oslo, Norway
matthew.gotham@ifikk.uio.no

Abstract. I present an analysis of the interpretation of anaphora that takes concepts from type-theoretic semantics, in particular the use of the Σ and Π dependent type constructors, and incorporates them into a model-theoretic framework. The analysis makes use of (parametrically) polymorphic lexical entries. The key ideas are that, in the simplest case, eventualities can play the role that proof objects do in type-theoretic semantics; that more complex, compositionally-defined, structures can play that role in other cases; and that pronouns can be modelled by context-dependent functions from proof objects of the preceding discourse (in this sense) to entities.

Keywords: anaphora · donkey sentences · polymorphism

1 Introduction

Type-theoretic semantics (TTS) is a variety of proof-theoretic semantics according to which the meaning of a sentence is a type in some underlying type theory, such that objects of that type are proofs of the proposition expressed by the sentence; we may say, then, that the sentence is true if and only if there is some object of that type [18].

Type theories chosen for TTS tend to be based on the intuitionistic type theory (ITT) of Martin-Löf [15], which contributes the particularly important (for TTS analyses) dependent type constructors Σ and Π, defined as in (1).

(1) a. If A is a type and, on the assumption that x is of type A, B is a type, then $(\Sigma x : A)(B)$ and $(\Pi x : A)(B)$ are types.
b. Objects of type $(\Sigma x : A)(B)$ are ordered pairs $\langle a, b\rangle$ such that a is of type A and b is of type $B[a/x]$.
c. Objects of type $(\Pi x : A)(B)$ are functions f with domain A such that for any object a of type A, $f(a)$ is of type $B[a/x]$.

In TTS, Σ and Π types can be used to give the meanings of existentially and universally quantified sentences, respectively.[1] For example, on the assumptions

[1] Σ can also be used to give the meaning of conjunction, and Π implication; this is reflected in the lexical entries given in Figure 1. Limitations of space prevent any further consideration of these connections here.

that DONKEY is the type of donkeys and that (for any x) $\textsc{bray}(x)$ is the type of proofs that x is braying, (2-a)–(2-b) have the interpretations given by the types shown in (3-a)–(3-b) respectively.

(2) a. A donkey is braying.
b. Every donkey is braying.

(3) a. $(\Sigma x : \textsc{donkey})(\textsc{bray}(x))$
b. $(\Pi x : \textsc{donkey})(\textsc{bray}(x))$

In this analysis, (2-a) is true iff there is some object of the type shown in (3-a), i.e. an ordered pair consisting of a donkey and a proof that that donkey is braying. Likewise, (2-b) is true iff there is some object of the type shown in (3-b), i.e. a function mapping every donkey to a proof that that donkey is braying.

TTS gives us the resources to formalize the famous 'donkey sentence' (4) from [6], in a way that respects the syntax of the English sentence.

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

On the most natural interpretation of (4), the interpretation of *it* co-varies with that of *a donkey*. The well-known problem that this example poses for model-theoretic semantic (MTS) theories in the tradition of [16] is that this co-variation cannot straightforwardly be accounted for. In the natural naïve formalization of (4), shown in (5), the variable y in the consequent of the conditional is outside the scope of the existential quantifier.[2] Its interpretation would therefore not covary with donkeys according to standard model theory.[3]

(5) $\forall x.(\mathsf{farmer}(x) \wedge \exists y.\mathsf{donkey}(y) \wedge \mathsf{own}(x,y)) \rightarrow \mathsf{beat}(x,y)$

There is a truth-conditionally adequate formalization of (4), shown in (6), but this leaves an explanatory gap as to why the (normally) existential *a donkey* should end up being translated as a universal quantifier.

(6) $\forall x.\forall y.(\mathsf{farmer}(x) \wedge (\mathsf{donkey}(y) \wedge \mathsf{own}(x,y))) \rightarrow \mathsf{beat}(x,y)$

By contrast, as first pointed out in [21], in TTS the meaning of (4) can be expressed by the type shown in (7), where Π has been given as the meaning of *every* and Σ has been given as the meaning of *a*, as expected. (This translation also makes use of the projections p and q, where for any ordered pair $\langle a,b\rangle$, $p(\langle a,b\rangle) = a$ and $q(\langle a,b\rangle) = b$.)

(7) $(\Pi z : (\Sigma x : \textsc{farmer})((\Sigma y : \textsc{donkey})(\textsc{own}(x,y))))(\textsc{beat}(p(z),p(q(z))))$

The type shown in (7) is the type of functions f such that:

[2] Here and throughout the paper, a dot following a variable binder will often be used instead of parentheses to indicate unbounded scope to the right.

[3] One option, therefore, is to change the model theory so that (5) would be interpreted in the desired way. This, explicitly, is the approach taken in Dynamic Predicate Logic (DPL) [8]. Discourse Representation Theory (DRT) [13] and File Change Semantics (FCS) [10] take a similar approach.

– the domain of f is the set of ordered pairs $\langle a, b\rangle$ such that:
  - a is a farmer, and
  - b is an ordered pair $\langle c, d\rangle$ such that
    - c is a donkey, and
    - d is a proof that a owns c, and
– f maps every $\langle a, \langle c, d\rangle\rangle$ in its domain to a proof that a beats c.

If and only if there is such a function, then there is an object of the type given in (7), and therefore, according to TTS, (4) is true on the intended interpretation.

The insight that ITT can be fruitfully applied to natural language semantics has been expanded into a detailed system in [18], and the analysis of anaphora, including especially cases like (4), has been developed and improved recently in [2]. In this paper, I will give an implementation of the ideas underlying the TTS analysis of anaphora, largely based on [2], in a model-theoretic framework.[4] This implementation has a two-fold motivation. Firstly, it enables us to examine the extent to which the TTS account of anaphora genuinely depends on an enriched type theory. Secondly, it provides insight into the relationship between the TTS account of anaphora and some 'dynamic' accounts in the MTS literature, given the similarity between the system that we end up with and some of them.

The paper is structured as follows. In Section 2 a translation of the account of [2] into higher-order logic will be built up in stages. In Section 3 I will expand the analysis to generalized quantifiers and plurals. Section 4 concludes.

2 Translating TTS for Anaphora into MTS

2.1 A First Pass

As a first step in our implementation we can consider again the interpretations given by TTS to (2-a)–(2-b) shown in (3-a)–(3-b) respectively. For the case of the Σ type constructor, and hence the existential claim, we have the gloss shown in (8).

(8) *A donkey is braying* is true iff there is an ordered pair consisting of a donkey and a proof that that donkey is braying.

What would constitute a proof that a donkey is braying? In the light of the helpful discussion in [18], §2.26, I will take the proof object to be an event, and in general the proof objects denoted by VPs to be eventualities. We will need in our underlying type theory a basic type v for eventualities, then, in addition to the basic types e for entities and t for truth values. For reasons to be discussed in Section 2.2, I will also assume that the unit type 1 is among our basic types. As (8) shows that we want to express ordered pairs, we will also

[4] By this, I mean that meanings will be given as expressions of a logical language, which are taken to be dispensable in favour of *their* interpretations in a model (as in [16]), which is where the 'real' semantics is. Expressions of the language of type theory are not understood this way in TTS; see [14] and [18], §2.27.

need the type constructor $\times$ for binary product types in addition to the standard type constructor $\rightarrow$ for functional types, along with associated term constructors $(.,.)$ for pairing and $[.]_0$ and $[.]_1$ for left and right projections, respectively.

Given these considerations, (2-a) can be translated into higher-order logic as shown in (9). Note that I am assuming a 'Davidsonian' approach to events according to which predicates denote relations between individuals and events directly, rather than mediated by theta roles.

(9) $\exists a^{e \times v}.\mathsf{donkey}([a]_0) \wedge \mathsf{bray}(a)$

$\equiv$ $\exists x^{e}.\exists e^{v}.\mathsf{donkey}(x) \wedge \mathsf{bray}(x, e)$

However, recall that the non-empty type condition reflected in (8) ('there is...') comes from TTS—that is to say, from the natural language semantic application of ITT and not from the definitions of the type constructors themselves; it is not, for example, reflected in (3-a). It therefore seems more appropriate to say that the compositionally-constructed interpretation of (2-a) is as shown in (10), and that existential closure of the abstracted variable a comes about from a discourse process. As we will see in Section 2.2, this will also allow indefinites to bind pronouns outside of what is normally thought to be their scope.

(10) $\lambda a^{e \times v}.\mathsf{donkey}([a]_0) \wedge \mathsf{bray}(a)$

These considerations lead to the provisional lexical entry for a, expressed in TTS by Σ, shown in (11).[5]

(11) $\lambda P^{e \rightarrow t}.\lambda V^{e \rightarrow v \rightarrow t}.\lambda a^{e \times v}.P([a]_0) \wedge V([a]_0)([a]_1)$

Now let us consider the Π type constructor, and the interpretation given to (2-b) in (3-b) as glossed in (12).

(12) *Every donkey is braying* is true iff there is a function mapping every donkey to a proof that that donkey is braying.

Here, we meet a complication that is not present in the discussion of Σ. Given the definition of the Π type constructor, the domain of the function alluded to in (12) should just be the set of donkeys. But this is not straightforward to accomplish in higher-order logic without having a type of donkeys—which is precisely a feature of TTS that we want to eliminate. The technique that I will adopt at this point is simply to say that the function is defined on the whole domain of entities, but that its interpretation is constrained in the right way whenever it applies to a donkey. Therefore, (3-b) can be translated as shown in (13), and hence the provisional lexical entry for *every*, expressed in TTS by Π, can be given as shown in (14).

(13) $\lambda f^{e \rightarrow v}.\forall x^{e}.\mathsf{donkey}(x) \rightarrow \mathsf{bray}(x, f(x))$

[5] In the type annotations, here and throughout the rest of the paper, brackets are omitted where possible, on the understanding that both $\times$ and $\rightarrow$ associate to the right and that $\times$ binds more tightly than $\rightarrow$.

(14) $\lambda P^{e\to t}.\lambda V^{e\to v\to t}.\lambda f^{e\to v}.\forall x^{e}.P(x)\to V(x)(f(x))$

We meet another complication when we consider the interpretation of the relative pronoun *who*. As shown in (7), the TTS analysis formalizes this using Σ, like the indefinite article—this, in fact, is part of what makes the extended binding scope possible. So we might expect the lexical entry to be as shown in (15).

(15) $\lambda V^{e\to v\to t}.\lambda P^{e\to t}.\lambda a^{e\times v}.P([a]_0)\wedge V([a]_0)([a]_1)$

However, on the basis of the provisional lexical entry given for a in (11) and other natural assumptions, the VP *owns a donkey* would be translated as shown in (16), and the types of (15) and (16) don't fit together.

(16) $\lambda x^{e}.\lambda a^{e\times v}.\mathsf{donkey}([a]_0)\wedge\mathsf{own}(x,a)$

This example shows up the need for some limited polymorphism in our lexical entries. (16) is not quite of the right type to be an argument for (15) because it contains extra information about an indefinite that was an argument to the embedded verb, and therefore is of type $e\to e\times v\to t$ rather than $e\to v\to t$ as expected by (15).

A similar issue is brought to light if we use (15) to derive the interpretation of a modified noun not containing any indefinites, for example *donkey who brays*, as shown in (17).

(17) $\lambda a^{e\times v}.\mathsf{donkey}([a]_0)\wedge\mathsf{bray}(a)$

Unsurprisingly—given that (15) is just a permutation of (11)—(17) is identical to the interpretation derived for *a donkey brays* in (10). It is of type $e\times v\to t$, not $e\to t$, and therefore not the right type to be an argument to (11).

The final limitation of the account so far is that there is no obvious way to incorporate pronouns. In the system set out in the next section, I will follow [2] and address this limitation by introducing a notion of context and a mechanism for updating it.

2.2 Full Implementation

In Figure 1 I have given an initial list of lexical entries for a fragment to be used in this paper.

Some remarks are in order. Firstly, how should we understand the lowercase Greek letters in the type annotations? I prefer to think of them as metavariables over types, such that what we have in Figure 1 are schemata over lexical entries. An alternative is to think of them as genuine type variables that should really be abstracted over as in System F [7], such that e.g. the translation for *and* would be as shown in (18), with the universally-quantified type indicated.

(18) $\Lambda\alpha.\Lambda\beta.\Lambda\gamma.\lambda p.\lambda q.\lambda i.\lambda a.p(i)([a]_0)\wedge q(i,[a]_0)([a]_1):$
$\forall\alpha.\forall\beta.\forall\gamma.(\alpha\to\beta\to t)\to(\alpha\times\beta\to\gamma\to t)\to\alpha\to\beta\times\gamma\to t$

What I want to stress, though, is that in either case we do not need the whole power of System F, since we only need (and want) the type variables to range

over unquantified types. [5] has provided a set-theoretic model theory for this kind of polymorphism, whereas there are no set-theoretic models for the full System F [19].

$$
\begin{aligned}
\mathit{and} &\mapsto \lambda p^{\alpha\to\beta\to t}.\lambda q^{\alpha\times\beta\to\gamma\to t}.\lambda i^{\alpha}.\lambda a^{\beta\times\gamma}.p(i)([a]_0)\wedge q(i,[a]_0)([a]_1)\\
\mathit{if} &\mapsto \lambda p^{\alpha\to\beta\to t}.\lambda q^{\alpha\times\beta\to\gamma\to t}.\lambda i^{\alpha}.\lambda f^{\beta\to\gamma}.\forall x^{\beta}.p(i)(x)\to q(i,x)(f(x))\\
\mathit{not} &\mapsto \lambda Q^{e\to\alpha\to\beta\to t}.\lambda x^{e}.\lambda i^{\alpha}.\lambda f^{\beta\to\beta}.\forall b^{\beta}.Q(x)(i)(b)\to f(b)\neq f(b)\\
a &\mapsto \lambda P.\lambda V.\lambda i^{\beta}.\lambda a^{(e\times\alpha)\times\gamma}.P([a]_0)(i)\wedge V([[a]_0]_0)(i,[a]_0)([a]_1)\\
\mathit{every} &\mapsto \lambda P.\lambda V.\lambda i^{\beta}.\lambda f^{(e\times\alpha)\to\gamma}.\forall a^{e\times\alpha}.P(a)(i)\to V([a]_0)(i,a)(f(a))\\
\mathit{who} &\mapsto \lambda V.\lambda P.\lambda a^{e\times\alpha\times\gamma}.\lambda i^{\beta}.P([a]_0,[[a]_1]_0)(i)\wedge V([a]_0)(i,([a]_0,[[a]_1]_0))([[a]_1]_1)\\
&\qquad \text{where } P: e\times\alpha\to\beta\to t \text{ and } V: e\to\beta\times e\times\alpha\to\gamma\to t\\
\mathit{donkey} &\mapsto \lambda a^{e\times 1}.\lambda i^{\alpha}.\mathsf{donkey}([a]_0)\\
\mathit{brays} &\mapsto \lambda x^{e}.\lambda i^{\alpha}.\lambda e^{v}.\mathsf{bray}(x,e)\\
\mathit{owns} &\mapsto \lambda D^{(e\to\alpha\to v\to t)\to\beta\to\gamma\to t}.\lambda x^{e}.D\big(\lambda y^{e}.\lambda a^{\alpha}.\lambda e^{v}.\mathsf{own}(x,y,e)\big)\\
\mathit{regrets} &\mapsto \lambda D^{(v\to\alpha\to v\to t)\to\beta\to\gamma\to t}.\lambda x^{e}.D\big(\lambda d^{v}.\lambda a^{\alpha}.\lambda e^{v}.\mathsf{regret}(x,d,e)\big)\\
\mathit{Giles} &\mapsto \lambda P^{e\to\alpha\to\beta\to t}.\lambda i^{\alpha}.\lambda a^{e\times\beta}.P([a]_0)(i)([a]_1)\wedge[a]_0=\mathsf{giles}\\
\mathit{he} &\mapsto \lambda V^{e\to\alpha\to\beta\to t}.\lambda i^{\alpha}.V(g^{\alpha\to e}(i))(i)\\
\mathit{it} &\mapsto \lambda V^{\alpha\to\beta\to\gamma\to t}.\lambda i^{\beta}.V(g^{\beta\to\alpha}(i))(i)\\
&\qquad \text{where } g \text{ stands for an arbitrarily-chosen free variable}
\end{aligned}
$$

Fig. 1. Some (schematic) lexical entries

Secondly, note that all the lexical entries incorporate an extra (polymorphic) argument position for input context, and that context is updated and passed on in appropriate ways. For example, the lexical entry for *and* requires that the first conjunct, and the conjunction as a whole, be dependent on input context i of (some) type α. The second conjunct is then dependent on i extended with the contribution of the first conjunct, of type β. The effect of this will be seen in the treatment of examples to be considered. N.B. for the sake of transparency, the lexical entry given for conjunction is 'leftward-looking', i.e. the first argument is the first conjunct.

Thirdly, the lexical entry for the common noun has a 'dummy' position filled by abstraction over the unit type. This is so as to achieve uniformity for modified and unmodified common nouns: both *donkey* and *donkey who brays* will be of

type $e\times\alpha\rightarrow\beta\rightarrow t$, for some α and β, and hence the issue of type mismatch raised in Section 2.1 with respect to these examples does not arise.[6]

Fourthly and finally, the lexical entries given for the pronouns *he* and *it* contain a free variable. The idea is that this free variable is resolved in context, subject to constraints that will be discussed. This aspect of the analysis is certainly not crucial: [11] has shown how, with the appropriate syntax, apparently free variables can actually be lambda-bound and retained in interpretation until such point as they are discharged. In the interest of making as few assumptions about syntax as possible, however, I retain a free variable analysis.

2.3 Examples

Now let us consider some examples, beginning with the example of cross-sentential binding shown in (19), where *it* is most readily interpreted as roughly synonymous with *the donkey that brays*.

(19) A donkey brays. Giles owns it.

We assume that the input context for (19) contains no information, so it is $*:1$.[7] In what follows I will use the notation $[word]^{var:=type}$, meaning the translation of *word* on the assumption that the type metavariable *var* is resolved to *type*.

The interpretation of the first sentence of (19) proceeds as follows:

$$a\ donkey \mapsto [a]^{\alpha,\beta:=1;\gamma:=v}\left([donkey]^{\alpha:=1}\right)$$
$$\Rightarrow_\beta \lambda V^{e\rightarrow 1\times e\times 1\rightarrow v\rightarrow t}.\lambda i^1.\lambda a^{(e\times 1)\times v}.\mathsf{donkey}([[a]_0]_0)\wedge V([[a]_0]_0)(i,[a]_0)([a]_1)$$
$$a\ donkey\ brays \mapsto [a\ donkey]\left([brays]^{\alpha:=1\times e\times 1}\right)$$
$$\Rightarrow_\beta \lambda i^1.\lambda a^{(e\times 1)\times v}.\mathsf{donkey}([[a]_0]_0)\wedge \mathsf{bray}([[a]_0]_0,[a]_1)$$

The interpretation is not dependent on the input context: λi is a vacuous abstraction. The second sentence is then interpreted as follows:

$$owns\ it \mapsto [owns]^{\alpha,\beta:=1\times(e\times 1)\times v;\gamma:=v}\left([it]^{\alpha:=e;\beta:=1\times(e\times 1)\times v;\gamma:=v}\right)$$
$$\Rightarrow_\beta \lambda x^e.\lambda i^{1\times(e\times 1)\times v}.\lambda e^v.\mathsf{own}(x,g^{1\times(e\times 1)\times v\rightarrow e}(i),e)$$
$$Giles\ owns\ it \mapsto [Giles]^{\alpha:=1\times(e\times 1)\times v;\beta:=v}([owns\ it])$$
$$\Rightarrow_\beta \lambda i^{1\times(e\times 1)\times v}.\lambda a^{e\times v}.\mathsf{own}([a]_0,g^{1\times(e\times 1)\times v\rightarrow e}(i),[a]_1)\wedge[a]_0=\mathsf{giles}$$

[6] The same issue prompted [2] to switch from treating common nouns as type-denoting to predicate-denoting.

[7] In the rest of the paper this will be referred to as 'the null context', and will generally be assumed.

This time the interpretation is dependent on the input context because of the pronoun: λi is not a vacuous abstraction. Putting the sentences together, we have:

$$
\begin{aligned}
(19) \mapsto\ & [\mathit{and}]^{\alpha:=1;\beta:=(e\times 1)\times v;\gamma:=e\times v}\,([a\ \mathit{donkey}\ \mathit{brays}])\,([\mathit{Giles}\ \mathit{owns}\ \mathit{it}]) \\
\Rightarrow_\beta\ & \lambda i^1.\lambda a^{((e\times 1)\times v)\times e\times v}.\big(\mathsf{donkey}([[[a]_0]_0]_0) \wedge \mathsf{bray}([[[a]_0]_0]_0, [[a]_0]_1)\big) \\
& \qquad \wedge \big(\mathsf{own}\big([[a]_1]_0, g^{1\times(e\times 1)\times v\to e}(i, [a]_0), [[a]_1]_1\big) \\
& \qquad\quad \wedge [[a]_1]_0 = \mathsf{giles}\big)
\end{aligned}
$$

Now the interpretation is potentially dependent on the input context, because i is within the argument to the free variable g. However, g is also fed $[a]_0$, which means that *it* in (19) can refer back to a referent introduced in the first clause.

g is a free variable and is contextually resolved. However, we can impose constraints on what a natural resolution would be. What we want to say is that if g is a function the domain of which is a tuple (of tuples ...), then a natural resolution of g is a function that selects an element of (an element of ...) that tuple. This requirement can be given the recursive definition shown in (20).

(20) For any types α, β and γ:
- a. $\lambda b^\alpha.b$ is a natural resolution function (NRF).
- b. $\lambda b^{\alpha\times\beta}.[b]_0$ is an NRF.
- c. $\lambda b^{\alpha\times\beta}.[b]_1$ is an NRF.
- d. For any terms $F : \beta\to\gamma$ and $G : \alpha\to\beta$, $\lambda b^\alpha.F(G(b))$ is an NRF if F and G are NRFs.

So in particular, $\lambda b^{1\times(e\times 1)\times v}.[[[b]_1]_0]_0$ is a natural resolution function. With this resolution, (19) would be interpreted as shown in (21).

$$
\begin{aligned}
(21)\quad & \lambda i^1.\lambda a^{((e\times 1)\times v)\times e\times v}.\big(\mathsf{donkey}([[[a]_0]_0]_0) \wedge \mathsf{bray}([[[a]_0]_0]_0, [[a]_0]_1)\big) \\
& \qquad \wedge \big(\mathsf{own}\big([[a]_1]_0, \lambda b([[[b]_1]_0]_0)(i, [a]_0), [[a]_1]_1\big) \\
& \qquad\quad \wedge [[a]_1]_0 = \mathsf{giles}\big) \\
\Rightarrow_\beta\quad & \lambda i^1.\lambda a^{((e\times 1)\times v)\times e\times v}.\big(\mathsf{donkey}([[[a]_0]_0]_0) \wedge \mathsf{bray}([[[a]_0]_0]_0, [[a]_0]_1)\big) \\
& \qquad \wedge \big(\mathsf{own}\big([[a]_1]_0, [[[a]_0]_0]_0, [[a]_1]_1\big) \wedge [[a]_1]_0 = \mathsf{giles}\big)
\end{aligned}
$$

This is the interpretation of the two-sentence discourse shown in (19). It is a relation, as in common in dynamic semantic systems, between an input and an output. It is also common in dynamic semantic systems to give a derived truth definition for this relational meaning. What that amounts to in this case is to take the input to be the null context $* : 1$ (as discussed above), and then existentially close the result;[8] as noted in Section 2.1, existential closure achieves the effect of the non-empty type condition from TTS. If we do that, then we

[8] This corresponds closely to the truth definition for DRT proposed in [12], p. 149.

derive (22) from (21).

$$(22)\quad \exists a^{((e\times 1)\times v)\times e\times v}.\big(\mathsf{donkey}([[[a]_0]_0]_0)\wedge \mathsf{bray}([[[a]_0]_0]_0,[[a]_0]_1)\big)$$
$$\wedge\ \big(\mathsf{own}\big([[a]_1]_0,[[[a]_0]_0]_0,[[a]_1]_1\big)\wedge [[a]_1]_0=\mathsf{giles}\big)$$
$$\equiv\quad \exists x^e.\exists e^v.\exists y^e.\exists {e_1}^v.\big(\mathsf{donkey}(x)\wedge \mathsf{bray}(x,e)\big)\wedge\big(\mathsf{own}(y,x,e_1)\wedge y=\mathsf{giles}\big)$$

(22) accurately represents the intended interpretation of (19).

The system provides a semantic account for why the interpretation of *it* cannot covary with donkeys in (23) or (the most natural interpretation of)[9] (24).

(23) Every donkey brays. Giles owns it.

(24) Giles does not own a donkey. It brays.

(23) is interpreted as follows:

$$[and]^{\alpha:=1;\beta:=\tau;}_{\gamma:=e\times v}\ \big([every]^{\alpha,\beta:=1}\ \big([donkey]^{\alpha:=1}\big)\ \big([brays]^{\alpha:=1\times e\times 1}\big)\big)$$
$$\big([Giles]^{\alpha:=1\times\tau;\beta:=v}\ \big([owns]^{\alpha,\beta:=1\times\tau;\gamma:=v}\ \big([it]^{\alpha:=e;\beta:=1\times\tau;\gamma:=v}\big)\big)\big)$$
$$\Rightarrow_\beta \lambda i^1.\lambda a^{\tau\times e\times v}.\forall x^{e\times 1}\big(\mathsf{donkey}([x]_0)\to \mathsf{bray}([x]_0,[a]_0(x))\big)$$
$$\wedge\ \big(\mathsf{own}([[a]_1]_0,g^{1\times\tau\to e}(i,[a]_0),[[a]_1]_1)\wedge[[a]_0]_0=\mathsf{giles}\big)$$

Where $\tau := e\times 1\to v$. Given the type of the free variable g, there is no way to get a bound reading for the pronoun.

The explanation in the case of (24) is essentially the same. It is (most naturally) interpreted as follows:

$$[and]^{\alpha:=1;\beta:=e\times(\tau\to\tau);\gamma:=v}$$
$$\Big([Giles]^{\alpha:=1;}_{\beta:=\tau\to\tau}\ \Big([not]^{\alpha:=1;}_{\beta:=\tau}\ \Big([own]^{\alpha:=1\times e\times 1;}_{\beta:=1;\gamma:=\tau}\ \Big([a]^{\alpha,\beta:=1;}_{\gamma:=v}\ \big([donkey]^{\alpha:=1}\big)\Big)\Big)\Big)\Big)$$
$$\Big([it]^{\alpha:=e;\beta:=1\times e\times(\tau\to\tau);\gamma:=v}\ \Big([brays]^{\alpha:=1\times e\times(\tau\to\tau)}\Big)\Big)$$
$$\Rightarrow_\beta \lambda i^1.\lambda a^{(e\times(\tau\to\tau))\times v}.\Big(\forall b^\tau\big(\big(\mathsf{donkey}([[b]_0]_0)\wedge\mathsf{own}([[a]_0]_0,[[b]_0]_0,[b]_1)\big)$$
$$\to [[a]_0]_1(b)\neq[[a]_0]_1(b)\big)$$
$$\wedge\ [[a]_0]_0=\mathsf{giles}\Big)\wedge\mathsf{bray}(g^{(1\times e\times(\tau\to\tau))\to e}(i,[a]_0),[a]_1)$$

Where $\tau := (e\times 1)\times v$. Once again, given the type of the free variable g, there is no way to get an unattested bound reading for the pronoun.

As in TTS, the treatment of negation here[10] is inspired by the equivalence, in classical and intuitionistic logic, of $\neg\phi$ and $\phi\to\bot$. A proof object of *Giles*

[9] Some speakers may allow an interpretation of (24) on which *a donkey* takes wider scope than negation. In that case, the pronoun could anaphorically refer back to the donkey.

[10] Figure 1 defines VP negation, which is derived from sentential negation in the obvious way. The VP formulation is more transparent in terms of compositional semantics, and also makes Giles available for anaphoric reference.

does not own a donkey is taken to be a pair consisting of Giles and a function mapping every state of Giles owning a donkey to an absurd (non-self-identical) object. Therefore, for there to be a proof object of *Giles does not own a donkey*, there cannot be a proof object of *Giles owns a donkey*.

Now we can look at a couple of genuine donkey sentences.

(25) Every farmer who owns a donkey feeds it.

$$\begin{aligned}&\mapsto [\mathit{every}]^{\substack{\alpha:=1\times\tau;\\ \beta:=1;\gamma:=v}} \Big([\mathit{who}]^{\substack{\alpha,\beta:=1;\\ \gamma:=\tau}} \Big([\mathit{owns}]^{\substack{\alpha:=\sigma\times e\times 1;\\ \beta:=\sigma;\gamma:=\tau}} \left([a]^{\alpha,\beta:=\sigma}\left([\mathit{donkey}]^{\alpha:=\sigma}\right)\right)\Big) \\ &\qquad\qquad \left([\mathit{farmer}]^{\alpha:=\sigma;\beta:=1;\gamma:=v}\right)\Big) \\ &\qquad\quad \left([\mathit{feeds}]^{\alpha,\beta:=1\times e\times 1\times\tau;\gamma:=v}\left([\mathit{it}]^{\alpha:=e;\beta:=1\times e\times 1\times\tau;\gamma:=v}\right)\right) \\ &\Rightarrow_\beta \lambda i^1.\lambda f^{e\times 1\times\tau\to v}.\forall a^{e\times 1\times\tau}.\big(\mathsf{farmer}([a]_0) \wedge \big(\mathsf{donkey}([[[[a]_1]_1]_0]_0) \\ &\qquad\qquad \wedge\, \mathsf{own}([a]_0, [[[[a]_1]_1]_0]_0, [[[a]_1]_1]_1)\big)\big) \\ &\qquad\quad \to \mathsf{feed}([a]_0, g^{1\times\tau\to e}(i,a), f(a))\end{aligned}$$

Where $\sigma := 1\times e\times 1$ and $\tau := (e\times 1)\times v$, as for the rest of the discussion of (25).

This time, the resolution function that we want is $\lambda b.[[[[[b]_1]_1]_1]_0]_0$.[11] If g is resolved in this way then we derive the interpretation shown in (26).

(26)
$$\begin{aligned}&\lambda i^1.\lambda f^{(e\times 1)\times\tau\to v}.\forall a^{(e\times 1)\times\tau}.\big(\mathsf{farmer}([a]_0) \wedge \big(\mathsf{donkey}([[[[a]_1]_1]_0]_0) \\ &\qquad \wedge\, \mathsf{own}([a]_0, [[[[a]_1]_1]_0]_0, [[[a]_1]_1]_1)\big)\big) \\ &\qquad \to \mathsf{feed}([a]_0, \lambda b([[[[[b]_1]_1]_1]_0]_0)(i,a), f(a)) \\ \Rightarrow_\beta\ &\lambda i^1.\lambda f^{(e\times 1)\times\tau\to v}.\forall a^{(e\times 1)\times\tau}.\big(\mathsf{farmer}([a]_0) \wedge \big(\mathsf{donkey}([[[[a]_1]_1]_0]_0) \\ &\qquad \wedge\, \mathsf{own}([a]_0, [[[[a]_1]_1]_0]_0, [[[a]_1]_1]_1)\big)\big) \\ &\qquad \to \mathsf{feed}([a]_0, [[[[a]_1]_1]_0]_0, f(a))\end{aligned}$$

If we apply (26) to the null context and then apply existential closure, we get (27).

(27)
$$\begin{aligned}&\exists f^{(e\times 1)\times\tau\to v}.\forall a^{(e\times 1)\times\tau}.\big(\mathsf{farmer}([a]_0) \wedge \big(\mathsf{donkey}([[[[a]_1]_1]_0]_0) \\ &\qquad \wedge\, \mathsf{own}([a]_0, [a]_{1,1,0,0}, [[[a]_1]_1]_1)\big)\big) \\ &\qquad \to \mathsf{feed}([a]_0, [[[[a]_1]_1]_0]_0, f(a)) \\ \equiv\ &\forall x^e.\forall y^e.\forall e^v.(\mathsf{farmer}(x) \wedge (\mathsf{donkey}(y) \wedge \mathsf{own}(x,y,e))) \to \exists {e_1}^v.\mathsf{feed}(x,y,e_1)\end{aligned}$$

The interpretation just derived is equivalent to that derived for (28), as shown below.

[11] As stated, $\lambda b.[[b]_1]_0$ is also a possible resolution function, which would have *it* varying with farmers rather than donkeys, which is obviously not a possible reading of (25). This reading could be ruled out by tweaking the lexical entry for *every*, but only at the cost of ruling out interpretations that we *do* want when we have an embedded clause. The mechanism for ruling out violations of 'Principle B' must come from somewhere else.

(28) If a farmer owns a donkey, he feeds it.

$$\mapsto [\mathit{if}]^{\substack{\alpha:=1;\\ \beta:=\tau;\\ \gamma:=v}} \Big([a]^{\substack{\alpha,\beta:=1;\\ \gamma:=(e\times 1)\times v}} \big([\mathit{farmer}]^{\alpha:=1}\big) \Big([\mathit{owns}]^{\substack{\alpha:=\sigma\times e\times 1;\\ \beta:=\sigma;\\ \gamma:=(e\times 1)\times v}} \Big([a]^{\substack{\alpha:=1;\\ \beta:=\sigma;\\ \gamma:=v}} \big([\mathit{donkey}]^{\alpha:=\sigma}\big)\Big)\Big)\Big)$$

$$\big([\mathit{he}]^{\alpha:=1\times\tau} \big([\mathit{feeds}]^{\alpha,\beta:=1\times\tau;\gamma:=v} \big([\mathit{it}]^{\alpha:=e;\beta:=1\times\tau;\gamma:=v}\big)\big)\big)$$

$$\Rightarrow_\beta \lambda i^1.\lambda f^{\tau\to v}.\forall x^\tau.\big(\mathsf{farmer}([[x]_0]_0) \wedge (\mathsf{donkey}([[[x]_1]_0]_0) \wedge \mathsf{own}([[x]_0]_0, [[[x]_1]_0]_0, [[x]_1]_1))\big) \to \mathsf{feed}(g^{1\times\tau\to e}(i,x), h^{1\times\tau\to e}(i,x), f(x))$$

Where $\sigma := 1\times e\times 1$ and $\tau := (e\times 1)\times(e\times 1)\times v$, as for the rest of the discussion of (28).

The resolution functions that give us the desired outcome are $g := \lambda b.[[[b]_1]_0]_0$ and $h := \lambda b.[[[[b]_1]_1]_0]_0$. With these in place, and with (28) then applied to the null context and existentially closed, we derive the interpretation shown in (29).

(29) $$\exists f^{\tau\to v}.\forall x^\tau.\big(\mathsf{farmer}([[x]_0]_0) \wedge (\mathsf{donkey}([[[x]_1]_0]_0) \wedge \mathsf{own}([[x]_0]_0, [[[x]_1]_0]_0, [[x]_1]_1))\big) \to \mathsf{feed}([[x]_0]_0, [[[x]_1]_0]_0, f(x))$$

$$\equiv \quad \forall x^e.\forall y^e.\forall e^v.\big(\mathsf{farmer}(x) \wedge (\mathsf{donkey}(y) \wedge \mathsf{own}(x,y,e))\big) \to \exists {e_1}^v.\mathsf{feed}(x,y,e_1)$$

The interpretations derived for (25) and (28) are thus equivalent, and in both cases it is the 'strong' interpretation that is derived; namely, that every farmer feeds every donkey that he owns. Weak readings will be discussed in the next section.

Finally, note how the use of eventualities as the equivalent of proof objects in this account, plus the polymorphism in the translation of *it*, allow for the most salient interpretation of (30) to be accounted for.

(30) If a farmer beats a donkey, he regrets it.

If we proceed as for (28) but with $[\mathit{regrets}]([\mathit{it}]^{\alpha:=v})$ instead of $[\mathit{feeds}]([\mathit{it}]^{\alpha:=e})$ and with appropriate resolution of the free variables introduced by the pronouns, the interpretation derived is as shown in (31) (where $\tau := (e\times 1)\times(e\times 1)\times v$).

(31) $$\exists f^{\tau\to v}.\forall x^\tau.\big(\mathsf{farmer}([[x]_0]_0) \wedge (\mathsf{donkey}([[[x]_1]_0]_0) \wedge \mathsf{beat}([[x]_0]_0, [[[x]_1]_0]_0, [[x]_1]_1))\big) \to \mathsf{regret}([[x]_0]_0, [[x]_1]_1, f(x))$$

$$\equiv \quad \forall x^e.\forall y^e.\forall e^v.\big(\mathsf{farmer}(x) \wedge (\mathsf{donkey}(y) \wedge \mathsf{beat}(x,y,e))\big) \to \exists d^v.\mathsf{regret}(x,e,d)$$

3 Plurals, Generalized Quantifiers and Weak Readings

The lexical entries given in Figure 1 do not cover plurals or determiners that resist analysis in first-order terms, like *most*. Some lexical entries illustrating the general approach to be taken in extending to these cases are shown in Figure 2.

$$\textit{two} \mapsto \lambda P.\lambda V.\lambda i^{\beta}.\lambda X.\, |\lambda x^{e}.\exists a^{\alpha}.\exists d^{\gamma}.X((x,a),d)| = 2 \\ \wedge \forall b^{e\times\alpha}.\forall c^{\gamma}.X(b,c) \rightarrow \big(P(b)(i) \wedge V([b]_0)(i,b)(c)\big)$$

$$\textit{fewer than two} \mapsto \\ \lambda P.\lambda V.\lambda i^{\beta}.\lambda X.\, |\lambda x^{e}.\exists a^{\alpha}.\exists d^{\gamma}.X((x,a),d)| < 2 \\ \wedge \forall b^{e\times\alpha}\big(\forall c^{\gamma}.X(b,c) \rightarrow \big(P(b)(i) \wedge V([b]_0)(i,b)(c)\big)\big) \\ \wedge\ \neg\exists Y.\big(\forall m^{(e\times\alpha)\times\gamma}.X(m) \rightarrow Y(m) \\ \wedge \neg\forall m^{(e\times\alpha)\times\gamma}.Y(m) \rightarrow X(m)\big) \\ \wedge \forall b^{e\times\alpha}.\forall c^{\gamma}.Y(b,c) \rightarrow \big(P(b)(i) \wedge V([b]_0)(i,b)(c)\big)$$

$$\textit{most}_{\textit{weak}} \mapsto \lambda P.\lambda V.\lambda i^{\beta}.\lambda X.\, |\lambda x^{e}.\exists a^{\alpha}.\exists d^{\gamma}.X((x,a),d)| > \frac{|\lambda x^{e}.\exists a^{\alpha}.P(x,a)(i)|}{2} \\ \wedge \forall b^{e\times\alpha}.\forall c^{\gamma}.X(b,c) \rightarrow \big(P(b)(i) \wedge V([b]_0)(i,b)(c)\big)$$

$$\textit{most}_{\textit{strong}} \mapsto \lambda P.\lambda V.\lambda i^{\beta}.\lambda X.\, |\lambda x^{e}.\exists a^{\alpha}.\exists d^{\gamma}.X((x,a),d)| > \frac{|\lambda x^{e}.\exists a^{\alpha}.P(x,a)(i)|}{2} \\ \wedge\ \forall y^{e}\big(\forall m^{\alpha}.\forall n^{\beta}.\big(P(y,m)(n) \wedge \exists o^{\alpha}.\exists r^{\gamma}.X((y,o),r)\big) \\ \rightarrow \exists s^{\gamma}.X((y,m),s)\big) \\ \wedge \forall b^{e\times\alpha}.\forall c^{\gamma}.X(b,c) \rightarrow \big(P(b)(i) \wedge V([b]_0)(i,b)(c)\big)$$

where $P : e{\times}\alpha{\rightarrow}\beta{\rightarrow}t, V : e{\rightarrow}\beta{\times}e{\times}\alpha{\rightarrow}\gamma{\rightarrow}t$ and $X, Y : (e{\times}\alpha){\times}\gamma{\rightarrow}t$

$$\textit{them} \mapsto \lambda V^{\alpha\rightarrow\beta\rightarrow\gamma\rightarrow t}.\lambda i^{\beta}.\lambda c^{\gamma}.\forall a^{\alpha}.G^{\beta\rightarrow\alpha\rightarrow t}(i) \rightarrow V(a)(i)(c)$$

where G stands for an arbitrarily-chosen free variable

Fig. 2. More (schematic) lexical entries

The general strategy for these cases will be to set up 'witness sets' in the sense of [1], or, more precisely, higher-order functions from which witness sets can be recovered. For example, the interpretation of *two donkeys VP* will be (a function from input contexts to) a higher-order function from which sets of two donkeys can be recovered. Of course, the nature of witness sets for quantifiers that aren't monotone-increasing means that the lexical entries for those will have to be more complex so as to enforce some version of what [20] calls the 'maximal participant condition'; this is evidenced by the lexical entry shown for *fewer than two*, which requires that the witness set not be a proper subset of any other witness set also in the extension of the VP.

A further point worth noting is that, throughout this section, the denotation for nouns and verbs with plural agreement is taken to be the same as that for those with singular agreement. Relatedly, only distributive readings of plurals are derived. Considering the distributive/collective distinction at the same time as everything else would take us too far afield.

Let us consider another simple example, in (32).

(32) Two donkeys bray. Giles owns them.

$$
\begin{aligned}
&\textit{two donkeys bray} \mapsto [\textit{two}]^{\alpha,\beta:=1;\gamma:=v}\left([\textit{donkey}]^{\alpha:=1}\right)\left([\textit{bray}]^{\alpha:=1}\right)\\
&\Rightarrow_\beta \lambda i^1.\lambda X^{(e\times 1)\times v\to t}.\left|\lambda x^e.\exists a^1.\exists d^v.X((x,a),d)\right| = 2\\
&\qquad\qquad \wedge \forall b^{e\times 1}.\forall c^v.X(b,c) \to \left(\mathsf{donkey}([b]_0) \wedge \mathsf{bray}([b]_0,c)\right)\\
&\textit{Giles owns them} \mapsto\\
&[\textit{Giles}]^{\alpha:=1\times\tau;\beta:=v}\left([\textit{owns}]^{\alpha,\beta:=1\times\tau;\gamma:=v}\left([\textit{them}]^{\alpha:=e;\beta:=1\times\tau;\gamma:=v}\right)\right)\\
&\Rightarrow_\beta \lambda i^1\times\tau.\lambda a^{e\times v}.\forall y^e\left(G^{1\times\tau\to e\to t}(i)(y) \to \left(\mathsf{own}([a]_0,y,[a]_1)\right) \wedge \mathsf{giles} = [a]_0\right)\\
&\therefore (32) \mapsto [\textit{and}]^{\alpha:=1;\beta:=\tau;\gamma:=e\times v}\left([\textit{two donkeys bray}]\right)\left([\textit{Giles owns them}]\right)\\
&\Rightarrow_\beta \lambda i^1.\lambda a^{\tau\times e\times v}.\left|\lambda x^e.\exists y^1.\exists d^v.[a]_0((x,y),d)\right| = 2\\
&\qquad\qquad \wedge \forall b^{e\times 1}\left(\forall c^v.[a]_0(b,c) \to \left(\mathsf{donkey}([b]_0) \wedge \mathsf{bray}([b]_0,c)\right)\right)\\
&\qquad\qquad \wedge \forall y^e.G^{1\times\tau\to e\to t}(i,[a]_0)(y) \to \left(\mathsf{own}([[a]_1]_0,y,[[a]_1]_1)\right.\\
&\qquad\qquad\qquad\qquad \left.\wedge\, [[a]_1]_0 = \mathsf{giles}\right)
\end{aligned}
$$

Where $\tau := (e\times 1)\times v\to t$.

Since we are now dealing with plural (set) entities, the definition of natural resolution functions given in (20) is inadequate. We need to extend the definition so that we can extract a set of entities from a set of tuples. Clauses d. and e. in (33), an extension of (20), achieve this.

(33) For any types α, β and γ:

a. $\lambda b^\alpha.b$ is a natural resolution function (NRF).
b. $\lambda b^{\alpha\times\beta}.[b]_0$ is an NRF.
c. $\lambda b^{\alpha\times\beta}.[b]_1$ is an NRF.
d. $\lambda b^{\alpha\times\beta\to t}.\lambda Y^\alpha.\exists Z^\beta.b(Y,Z)$ is an NRF.
e. $\lambda b^{\alpha\times\beta\to t}.\lambda Y^\alpha.\exists Z^\beta.b(Z,Y)$ is an NRF.
f. For any terms $F : \beta\to\gamma$ and $G : \alpha\to\beta$, $\lambda b^\alpha.F(G(b))$ is an NRF if F and G are NRFs.

(33) means that $\lambda b^{1\times((e\times 1)\times v\to t)}.\lambda x^e.\exists n^1.\exists e^v.[b]_1((x,n),e)$ is a natural resolution function. With G in (33) instantiated to this function, the interpretation

proceeds as follows.

$$
\begin{aligned}
(34)\quad & \lambda i^1.\lambda a^{\tau\times e\times v}.\,\big|\lambda x^e.\exists y^1.\exists d^v.[a]_0((x,y),d)\big| = 2 \\
& \qquad \wedge \forall b^{e\times 1}\big(\forall c^v.[a]_0(b,c) \rightarrow \big(\mathsf{donkey}([b]_0) \wedge \mathsf{bray}([b]_0,c)\big)\big) \\
& \qquad \wedge \forall y^e.\exists n^1\big(\exists e^v.[(i,[a]_0)]_1((y,n),e)\big) \\
& \qquad\qquad \rightarrow \big(\mathsf{own}([[a]_1]_0,y,[[a]_1]_1) \wedge [[a]_1]_0 = \mathsf{giles}\big) \\
\Rightarrow_\beta\quad & \lambda i^1.\lambda a^{\tau\times e\times v}.\,\big|\lambda x^e.\exists y^1.\exists d^v.[a]_0((x,y),d)\big| = 2 \\
& \qquad \wedge \forall b^{e\times 1}\big(\forall c^v.[a]_0(b,c) \rightarrow \big(\mathsf{donkey}([b]_0) \wedge \mathsf{bray}([b]_0,c)\big)\big) \\
& \qquad \wedge \forall y^e.\exists n^1\big(\exists e^v.[a]_0((y,n),e)\big) \\
& \qquad\qquad \rightarrow \big(\mathsf{own}([[a]_1]_0,y,[[a]_1]_1) \wedge [[a]_1]_0 = \mathsf{giles}\big)
\end{aligned}
$$

Where $\tau := (e\times 1)\times v\rightarrow t$.

Applied to the empty context and then existentially closed, the interpretation comes out as shown in (35).

$$
\begin{aligned}
(35)\quad & \exists a^{((e\times 1)\times v\rightarrow t)\times e\times v}.\,\big|\lambda x^e.\exists y^1.\exists d^v.[a]_0((x,y),d)\big| = 2 \\
& \qquad \wedge \forall b^{e\times 1}\big(\forall c^v.[a]_0(b,c) \rightarrow \big(\mathsf{donkey}([b]_0) \wedge \mathsf{bray}([b]_0,c)\big)\big) \\
& \qquad \wedge \forall y^e.\exists n^1\big(\exists e^v.[a]_0((y,n),e)\big) \\
& \qquad\qquad \rightarrow \big(\mathsf{own}([[a]_1]_0,y,[[a]_1]_1) \wedge [[a]_1]_0 = \mathsf{giles}\big) \\
\equiv\quad & \exists R^{e\times v\rightarrow t}.\exists z^e.\exists e^v.\,|\lambda x^e.\exists d^v.R(x,d)| = 2 \\
& \qquad \wedge \forall v^e\big(\forall c^v.R(v,c) \rightarrow (\mathsf{donkey}(v) \wedge \mathsf{bray}(v,c))\big) \\
& \qquad \wedge \forall y^e.\exists b^v(R(y,b)) \rightarrow (\mathsf{own}(z,y,e) \wedge z = \mathsf{giles})
\end{aligned}
$$

We are now in a position to look at a proportional donkey sentence, (36), under both strong and weak readings.

(36) Most farmers who own a donkey feed it.

With appropriate type instantiations, the weak reading of the sentence is derived as shown below.

$$
\begin{aligned}
& \lambda i^1.\lambda X^{(e\times\tau)\times v\rightarrow t}.\,|\lambda x^e.\exists a^\tau.\exists d^v.X((x,a),d)| > \\
& \qquad \frac{\left|\begin{array}{l}\lambda x^e.\exists a^\tau.\mathsf{farmer}(x) \wedge \big(\mathsf{donkey}([[[a]_1]_0]_0) \\ \qquad\qquad\qquad\qquad \wedge \mathsf{own}(x,[[[a]_1]_0]_0,[[a]_1]_1)\big)\end{array}\right|}{2} \\
& \qquad \wedge\ \forall b^{e\times\tau}.\forall c^v.X(b,c) \rightarrow \big(\big(\mathsf{farmer}([b]_0) \wedge \big(\mathsf{donkey}([[[[b]_1]_1]_0]_0) \\
& \qquad\qquad\qquad\qquad \wedge \mathsf{own}([b]_0,[[[[b]_1]_1]_0]_0,[[[b]_1]_1]_1)\big)\big) \\
& \qquad\qquad\qquad\qquad \wedge \mathsf{feed}([b]_0, g^{1\times e\times\tau\rightarrow e}(i,b), c)\big)
\end{aligned}
$$

Where $\tau := 1\times(e\times 1)\times v$, as for the rest of the discussion of (36).

In this case the resolution function that we want is $\lambda b^{1\times e\times\tau}.[[[[[b]_1]_1]_1]_0]_0$. With this resolution, followed by application to the empty context and existential closure, we end up with the interpretation shown in (37).

(37) $\exists X^{(e\times\tau)\times v\to t}.\,|\lambda x^e.\exists a^\tau.\exists d^v.X((x,a),d)| >$

$$\frac{\left|\begin{array}{l}\lambda x^e.\exists a^\tau.\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}([[[a]_1]_0]_0)\\ \qquad\qquad\qquad\qquad\wedge\,\mathsf{own}(x,[[[a]_1]_0]_0,[[a]_1]_1)\big)\end{array}\right|}{2}$$

$$\wedge\,\forall b^{e\times\tau}.\forall c^v.X(b,c)\to\big(\big(\mathsf{farmer}([b]_0)\wedge\big(\mathsf{donkey}([[[[b]_1]_1]_0]_0)$$
$$\wedge\,\mathsf{own}([b]_0,[[[[b]_1]_1]_0]_0,[[[b]_1]_1]_1)\big)\big)$$
$$\wedge\,\mathsf{feed}([b]_0,[[[[b]_1]_1]_0]_0,c)\big)$$

$\equiv$ $\exists Y^{e\times e\times v\times v\to t}.\,|\lambda x^e.\exists y^e.\exists d^v.\exists e^v.Y(x,y,d,e)| >$

$$\frac{\big|\lambda x^e.\exists y^e.\exists e^v.\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}(y)\wedge\mathsf{own}(x,y,e)\big)\big|}{2}$$

$$\wedge\,\forall x^e.\forall y^e.\forall d^v.\forall e^v.Y(x,y,d,e)$$
$$\to\big(\big(\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}(y)$$
$$\wedge\,\mathsf{own}(x,y,d)\big)\big)\wedge\mathsf{feed}(x,y,e)\big)$$

Suppressing mention of eventualities for the sake of simplicity, (37) expresses the existence of a set Y of farmer-donkey pairs such that the number of farmers in Y is greater than half the number of farmers who own a donkey; and for every farmer-donkey pair in Y, the farmer owns the donkey and feeds the donkey. It does not require that every farmer in Y feed every donkey that he owns. For that, we need the lexical entry given in Figure 2 for the strong version of *most*. With this in place, and the same resolution for the pronoun, we end up with the interpretation shown in (38).

(38) $\exists X^{(e\times\tau)\times v\to t}.\,|\lambda x^e.\exists a^\tau.\exists d^v.X((x,a),d)| >$

$$\frac{\left|\begin{array}{l}\lambda x^e.\exists a^\tau.\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}([[[a]_1]_0]_0)\\ \qquad\qquad\qquad\qquad\wedge\,\mathsf{own}(x,[[[a]_1]_0]_0,[[a]_1]_1)\big)\end{array}\right|}{2}$$

$$\wedge\,\forall y^e\big(\forall m^\tau.\forall n^1.\big(\mathsf{farmer}(y)\wedge\mathsf{donkey}([[[m]_1]_0]_0)$$
$$\wedge\,\mathsf{own}(y,[[[m]_1]_0]_0,[[m]_1]_1)$$
$$\wedge\,\exists o^\tau.\exists r^v.X((y,o),r)\big)\to\exists s^v.X((y,m),s)\big)$$
$$\wedge\,\forall b^{e\times\tau}.\forall c^v.X(b,c)\to\big(\big(\mathsf{farmer}([b]_0)\wedge\big(\mathsf{donkey}([[[[b]_1]_1]_0]_0)$$
$$\wedge\,\mathsf{own}([b]_0,[[[[b]_1]_1]_0]_0,[[[b]_1]_1]_1)\big)\big)$$
$$\wedge\,\mathsf{feed}([b]_0,[[[[b]_1]_1]_0]_0,c)\big)$$

$$
\begin{aligned}
\equiv \quad & \exists Y^{e\times e\times v\times v\to t}.\, |\lambda x^e.\exists y^e.\exists d^v.\exists e^v.Y(x,y,d,e)| > \\
& \qquad \frac{\left|\lambda x^e.\exists y^e.\exists e^v.\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}(y)\wedge\mathsf{own}(x,y,e)\big)\right|}{2} \\
& \qquad \wedge \forall y^e\big(\forall z^e.\forall d^v.\big(\mathsf{farmer}(y)\wedge\mathsf{donkey}(z)\wedge\mathsf{own}(y,z,d) \\
& \qquad\qquad \wedge \exists v^e.\exists c^v.\exists r^v.Y(y,v,c,r)\big) \\
& \qquad\qquad \to \exists s^v.Y(y,z,d,s)\big) \\
& \qquad \wedge \forall x^e.\forall y^e.\forall d^v.\forall e^v.Y(x,y,d,e) \\
& \qquad\qquad \to \big(\big(\mathsf{farmer}(x)\wedge\big(\mathsf{donkey}(y) \\
& \qquad\qquad\quad \wedge \mathsf{own}(x,y,d)\big)\big)\wedge\mathsf{feed}(x,y,e)\big)
\end{aligned}
$$

In addition to what is expressed in (37), (38) requires that for every farmer-donkey pair in Y, if the farmer owns any other donkey then that farmer-donkey pair is in Y as well. This captures the strong reading.

4 Discussion and Conclusion

I have presented a framework for capturing many anaphoric relationships which is inspired by the concepts behind TTS analyses of these phenomena, but without actually using an enriched type theory like ITT. Nevertheless, the type theory used is not exactly the simple theory of types either, as the account crucially relies on type polymorphism, either at the object-level or at the meta-level. This requirement, however, appears to be at least partly independent of the TTS/MTS distinction, since type polymorphism is also required in TTS analyses once the account is extended to include generalized quantifiers, as in [22].

On the MTS side, there is an obvious similarity between the account presented in this paper and accounts that make use of lists or stacks to keep track of discourse referents, for example [3], [4], [9] and [17]. That in itself suggests a connection between MTS and TTS approaches to anaphora, particularly between lists/stacks of discourse referents and (Martin-Löf) proof objects, that should be further explored.

References

1. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. Linguistics and Philosophy 4(2), 159–219 (1981)
2. Bekki, D.: Representing anaphora with dependent types. In: Asher, N., Soloviev, S. (eds.) Logical Aspects of Computational Linguistics, pp. 14–29. No. 8535 in Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2014)
3. Dekker, P.: Predicate logic with anaphora. In: Harvey, M., Samelmann, L. (eds.) Proceedings of Semantics and Linguistic Theory. vol. 4, pp. 79–95 (1994)
4. van Eijck, J.: Incremental dynamics. Journal of Logic, Language and Information 10, 319–351 (2001)

5. Emms, M.: Polymorphic quantifiers. In: Stokhof, M., Torenvliet, L. (eds.) Proceedings of the Seventh Amsterdam Colloquium. Institute for Language, Logic and Information, Amsterdam (1989)
6. Geach, P.T.: Reference and Generality. Contemporary Philosophy, Cornell University Press, Ithaca, NY (1962)
7. Girard, J.Y.: The system F of variable types, fifteen years later. Theoretical Computer Science 45, 159–192 (1986)
8. Groenendijk, J., Stokhof, M.: Dynamic predicate logic. Linguistics and Philosophy 14(1), 39–100 (1991)
9. de Groote, P.: Towards a Montagovian account of dynamics. In: Gibson, M., Howell, J. (eds.) Proceedings of Semantics and Linguistic Theory. vol. 16, pp. 1–16 (2006)
10. Heim, I.: The Semantics of Definite and Indefinite Nouns Phrases. Ph.D. thesis, University of Massachussetts, Amherst (1982)
11. Jacobson, P.: Towards a variable-free semantics. Linguistics and Philosophy 22(2), 117–184 (1999)
12. Kamp, H., van Genabith, J., Reyle, U.: Discourse Representation Theory. In: Gabbay, D.M., Guenther, F. (eds.) Handbook of Philosophical Logic, vol. 15, pp. 125–394. Springer, Dordrecht, 2 edn.
13. Kamp, H., Reyle, U.: From Discourse to Logic. No. 42 in Studies in Linguistics and Philosophy, Kluwer, Dordrecht
14. Luo, Z.: Formal semantics in modern type theories: Is it model-theoretic, proof-theoretic, or both? In: Asher, N., Soloviev, S. (eds.) Logical Aspects of Computational Linguistics. pp. 177–188. No. 8535 in Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2014)
15. Martin-Löf, P.: An intuitionistic theory of types: Predicative part. In: Rose, H., Shepherdson, J. (eds.) Logic Colloquium '73, pp. 73–118. No. 80 in Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam (1975)
16. Montague, R.: The proper treatment of quantification in ordinary English. In: Suppes, P., Moravcsik, J., Hintikka, J. (eds.) Approaches to Natural Language, pp. 221–242. D. Reidel, Dordrecht (1973)
17. Nouwen, R.: On dependent pronouns and dynamic semantics. Journal of Philosophical Logic 36(2), 123–154
18. Ranta, A.: Type-Theoretical Grammar. No. 1 in Indices, Oxford (1994)
19. Reynolds, J.C.: Polymorphism is not set-theoretic. In: Kahn, G., MacQueen, D.B., Plotkin, G. (eds.) Semantics of Data Types, pp. 145–156. No. 173 in Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (1984)
20. Steedman, M.: Taking Scope. MIT Press, Cambridge, MA (2012)
21. Sundholm, G.: Proof theory and meaning. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, vol. 3, pp. 471–506. D. Reidel, Dordrecht (1986)
22. Tanaka, R., Nakano, Y., Bekki, D.: Constructive generalized quantifiers revisited. In: Nakano, Y., Satoh, K., Bekki, D. (eds.) New Frontiers in Artificial Intelligence, pp. 115–124. No. 8417 in Lecture Notes in Computer Science, Springer, Cham (2014)