Discourse Representation, Type Theory and Property Theory

Chris Fox
Department of Computer Science
University of Essex
Colchester CO4 3SQ, UK
Phone: +44 (0)206 82 5958
e-mail: foxcj@essex.ac.uk

August 1994

Abstract

Since Aristotle, it has been accepted that the appropriate interpretation of sentences is as propositions, and that general terms should be interpreted as properties. Recent proposals for natural language semantics have used constructive type theories such as Martin-Löf's Type Theory MLTT [Martin-Löf, 1982, Martin-Löf, 1984] which treat anaphora and 'Donkey' sentences using dependent types [Sundholm, 1989, Ranta, 1991, Davila-Perez, 1994]. However, MLTT conflates the notions of proposition and property. This work shows how, within property theory, dependent-type expressions representing natural language discourse can be mapped systematically into first-order expressions with a classical notion of propositionhood, distinct from that of properties.

1. Introduction

It is well known that compositional semantics with classical logic has problems producing the desired representation of some simple examples involving pronouns (anaphoric expressions). Some of the problems are exemplified by intersentential anaphora and 'Donkey' sentences:

A man walked in. He whistled.
If a farmer owns a donkey, he beats it.

Following the standard Montagovian treatment, the following expressions are derived:

$$\exists x(\text{man}'x \,\&\, \text{walked-in}'x) \,\&\, \text{whistled}'x$$
$$\exists x(\text{farmer}'x \,\&\, \exists y(\text{donkey}'y \,\&\, \text{owns}'yx) \rightarrow (\text{beats}'yx))$$

These are erroneous. In the first example, the final x is not bound correctly. In the second example, not only is the y not bound correctly in the final predication, but the x should really be read as universally quantified.

This is not a problem concerning whether we can write down an expression with the appropriate truth conditions (clearly we can), but whether we can derive such an expression *systematically*.

Examples such as these are usually taken as motivation for dropping classical logics for natural language semantics, in favour of non-classical theories with discourse markers (Discourse Representation Theory [Kamp, 1981]) or with non-standard semantics for the logical quantifiers and connectives (Dynamic Predicate Logic (DPL) [Groenendijk and Stokhof, 1991]), or by adopting an intuitionistic logic such as Martin-Löf's Type Theory [Martin-Löf, 1982, Martin-Löf, 1984] [Sundholm, 1989, Ranta, 1991, Davila-Perez, 1994].

Classical logic is usually the first victim. However, many existing proposals are flawed in comparison with classical first-order logic in that they do not have a straightforward interpretation, or a tractable proof theory.

These examples can be represented in a first-order framework. Following Ranta and Davila-Perez we can treat these examples in MLTT. This, in turn, can be incorporated into Turner's Property Theory (PT), a first-order logic which axiomatises of Aczel's Frege structures [Turner, 1992, Aczel, 1980] (see also [Smith, 1984]). Sentences are then represented with non-classical propositions, where propositions are properties which hold of their proofs. However, this paper will show that, in the truth conditions, we do not have to abandon a classical notion of propositionhood for sentences.

Using MLTT implemented in PT may have some advantages over strictly adhering to MLTT: PT is more intensional than vanilla MLTT; also, it offers more flexibility in that it provides a meta-language in which operations can be defined without encoding them in λ-calculus.

The paper proceeds by introducing a dependent type analysis of anaphora, with a constructive notion of propositionhood, and shows how the representations which result can systematic be given a first-order interpretation, with a classical notion of propositionhood.

2. Dependent Types

On reflection, we can see that there is a similarity between discourse referents and variables in procedural computer programs. We might say that, in these examples, the indefinite noun phrases are acting like "let" assignments to variables, and that the pronouns can be disambiguated as variable names. As with local variables, there are restrictions on their accessibility.

These intuitions have lead to theories such as Dynamic Predicate Logic (DPL). In DPL, indefinites are represented as existential quantifiers, as in Montague's theory, but the semantics are altered so that the quantifiers may bind outside their scope. This is achieved in much the same way as variables are treated in the semantics of procedural programming languages (e.g. [Hoare, 1969]). In the model for classical predicate logic, there is an assignment function (usually g) which maps variables to constants. Outside the scope of a quantifier, the function normally maps variables to arbitrary constants. In DPL, the semantics of the existential quantifier modify the assignment function, so that subsequent free occurrences of the quantified variable are mapped to the same constant which satisfies the quantified expression.

The conditional (material implication) also has its semantics modified, so that the consequent must hold for all values which satisfy an existentially quantified antecedent.

This results in a theory with a classical appearance. However, it is a non-classical as the semantics are so drastically changed.

Following this similarity with issues in programming, we can see that there is a rival perspective, related to program specification and verification with functional languages. Sundholm and Ranta have shown how Donkey sentences can be represented in MLTT, which can be seen as a specification language for programs written in λ-calculus.

In MLTT, there are types and elements (or witnesses) of types. If T is a type, then:

$$w \varepsilon T$$

says that w is an element of that type.

Types can be seen as specifications, and elements of the types are then programs that meet the specification. Alternatively, types can be said to correspond with propositions, and elements of a type with proofs of that proposition. This is a form of intuitionistic logic. A proposition is true if we can produce a proof of it. If the class of specified proofs is non-empty (or inhabited), then the proposition is true.

Various sorts of complex types can be defined in MLTT. Those of interest for Donkey sentences and intersentential anaphora are the *dependent types*. These allow us to treat the witness for a type as a *context* for subsequent types. Taking type operator Σ as a relevant example:

$$\Sigma x \varepsilon f . g$$

is a type formed from the types f, g. Viewed as a proposition, it is true if we can find an h such that:

$$h \varepsilon \Sigma x \varepsilon f . g$$

Witnesses for this type are pairs, so $h = \langle a, b \rangle$, where:

$$a \varepsilon f$$
$$b \varepsilon g[a/x]$$

Here we see the context dependence created by the Σ type. The operator Σ can be viewed as a form of existential quantification (or conjunction). Selection operators $\mathsf{fst}, \mathsf{snd}$ pick the first and second elements of pairs. Such selectors can be used to pick out the antecedent of a pronoun.

If x here is a witness to a nominal expression, then it is available in subsequent parts of the discourse.

Skipping the details of the translation procedure, the first example discourse would be represented:

$$\Sigma y \varepsilon (\Sigma x \varepsilon \text{man}' \text{walked-in}' x)(\text{whistled}'(\text{he}_0))$$

Sentence concatenation, relative clauses and indefinites are represented using Σ types.

If the anaphora he_0 is resolved, by replacing it with $\mathsf{fst}(y)$, then this MLTT proposition will be true if there it has a witness of the form:

$$\langle\langle m, \varphi\rangle, \psi\rangle$$

where m is a man, and φ is a proof that that man walked in, and ψ is a proof that that man whistled. All the nominal objects referred to by definite descriptors are available in subsequent parts of the discourse. The proof ψ is of $(\text{whistled}'(\text{he}_0))$ where occurrences of y are replaced by $\langle m, \varphi\rangle$. When $\mathsf{fst}(y)$ is substituted for the pronoun, this is equivalent to $\text{whistled}'(m)$.

With the second example, we need to make use of another dependent types operator Π.

$$\Pi x \varepsilon f.g$$

holds of functions which take all proofs, or witnesses w of f to a proof of $g[w/x]$. This is similar to universal quantification (or implication). It can be used to represent the universal determiner "all", and "if ... then ..." constructions. The second example can be represented as:

$$\Pi x \varepsilon (\Sigma y \varepsilon \text{man}'.(\Sigma z \varepsilon \text{donkey}' \text{own}' zy).(\\ \text{beat}'(\text{it}_0)(\text{he}_0))$$

The sentence is true if we can find a function that maps objects of the form:

$$\langle f, \langle d, \varphi\rangle\rangle$$

where f is a farmer, d is a donkey, and φ is a proof that f owns d, into a proof of:

$$\text{beats}'(\text{it}_0, \text{he}_0)$$

The anaphora are resolved if we replace them with selectors as follows:

$$\text{beat}'(\mathsf{fst}(\mathsf{snd}(x))(\mathsf{fst}(x))$$

where x becomes instantiated with $\langle f, \langle d, \varphi\rangle\rangle$:

$$\text{beat}'(\mathsf{fst}(\mathsf{snd}(\langle f, \langle d, o\rangle\rangle))(\mathsf{fst}(\langle f, \langle d, o\rangle\rangle))$$

which is equivalent to:

$$\text{beat}' df$$

For a compositional analysis, this is not the whole story. One particular problem concerns relative clauses. In a discourse containing the following:

[A man]$_i$ who ate cheese died.
He$_i$ was young.

where the pronoun "he" is coindexed with the man mentioned in the first sentence, the structure of the first subject noun phrase is:

a (man who ate cheese)

In the semantics, the man denoted by the noun is embedded in a witness of the form:

$$\langle \text{man}, \langle \text{cheese}, \varphi\rangle\rangle$$

The pronoun "He$_i$" refers to the man, not this structure. The semantic rules must be amended to recurse through the structure and recover the man himself. Davila-Perez achieves using a function app [Davila-Perez, 1994], which has the following behaviour:

$$\mathsf{app} yxv = \begin{cases} \mathsf{app} ya(\mathsf{fst} v) & \text{if } x = \Sigma ab \\ yv & \text{otherise} \end{cases}$$

Although this function appears in the representation of all determiners, its role is only apparent with relative clauses, where it effectively picks out the part of the proof, or context which is a witness to the main noun (rather than the parts of the proof which verify that the rest of the relative clause which applies to it). As an example, the sentence:

Every man who owns a book reads it.

can be represented with:

$$\Pi z\varepsilon(\Sigma x\varepsilon \mathrm{m}'.(\Sigma y\varepsilon \mathrm{b}'.(\mathrm{o}'yx))).$$
$$(\mathsf{app}([\mathrm{r}'(\mathrm{it}_0)][\Sigma x\varepsilon \mathrm{m}'.\Sigma y\varepsilon \mathrm{b}'.(\mathrm{o}'yx)))][z])$$

where $\mathrm{it}_0 = \mathsf{fst}(\mathsf{snd}z)$. Using the definition of app, this becomes equivalent to:

$$\Pi z\varepsilon(\Sigma x\varepsilon \mathrm{m}'.(\Sigma y\varepsilon \mathrm{b}'.(\mathrm{o}'yx))).$$
$$(\mathrm{r}'(\mathsf{fst}(\mathsf{snd}z))(\mathsf{fst}(\mathsf{fst}(z))))$$

This specifies a function which can take a man, a book and a proof of ownership, to a proof that the book is read by its owner.

Davila defines app using the Universe of Small Sets, although its effect can be implemented using a device similar to Cooper Storage, during parsing. For more details of natural language semantics in the MLTT, the reader is referred to [Davila-Perez, 1994].

This paper will focus on giving a systematic, classical first-order interpretation of type expressions containing just the operators Σ, Π. Constructive interpretations of negation, sentential conjunction, implication, relative clauses, universal and existential quantification can be defined in terms of these two operators.

I do not intend to address the general problem of conjunctions here, or disjunction. It can be noted that disjunction may be represented using disjoint union. The sentence:

A bicycle or tricycle disappeared.

would be represented as something like:

$$\Sigma x\varepsilon(\mathrm{bicycle}' \oplus \mathrm{tricycle}').\mathrm{disappeared}'x$$

The antecedent is satisfied by a pair, the second element of which is a witness to the noun, and the first is a flag which indicates which disjunct it is a witness to. Clearly, anaphoric reference to the object itself is possible. It is not clear whether the flag can play a special role in the semantics.

The use of MLTT seems like an elegant solution to the problem anaphoric reference in discourse. It is an existing theory, with a well defined behaviour that achieves the required results without giving unusual interpretations to the logical quantifiers and connectives. However, it is a non-classical theory; MLTT conflates the notions of proposition and property. In MLTT, "some bikes" is just as much a proposition as "there are some bikes".

Two further points against this theory are that: there is no means of distinguishing false propositions; and there is (as yet) no extensional-intensional contrast in the basic theory. Both are required in natural language semantics.[1]

To remedy both of these defects, we can implement these ideas in property theory. Indeed, this gives some additional flexibility, and provides a classical notion of propositions and the standard connectives. As will be shown, this allows a classical interpretation of natural language truth conditions—where sentences are represented by propositions that are distinct from properties—in combination with a dependent-type treatment of anaphora.[2]

3. Property Theory

Ray Turner's axiomatisation of Aczel's Frege Structures [Turner, 1990, Turner, 1992, Aczel, 1980], PT can be split into two components, or levels. The first is a language of terms, which consists of the untyped λ-calculus, embellished with logical constants. A restricted class of these terms will correspond to *propositions*. When combined appropriately using the logical constants, other propositions result. As an example, given the propositions t, s, the 'conjunction' of these, $t \wedge s$, is also a proposition, where $\wedge$ is a logical constant.

Some of the propositions will, further, be *true* propositions. When combining propositions with the logical constants, the truth of the resultant proposition will depend upon the truth of the constituent propositions. Considering the previous example, if t, s are both propositions, then $t \wedge s$

[1]This drawback might be overcome if we add a Universe of propositions to the terms of MLTT, but this seems to be to admit the primitive nature of propositions, and so produce a realist theory, counter to the intended spirit of MLTT.

[2]Although there is insufficient space to give details here, property theory with pairs and selectors (as used in MLTT semantics) can be used to model abstraction and application in 'λ-DRT' and Aczel-Lunnon set abstraction in Situation Theory [Aczel and Lunnon, 1991].

will be a true proposition if and only if t and s are true propositions.

There may be terms that form propositions when applied to another term. These terms are the properties. The act of predication is modelled by λ-application.

The propositionhood and truth of terms is formalised in a meta-language of well formed formulae (wff). This is a first-order logic with just two primitive predicates: P, for "is a proposition"; and T, for "is a true proposition". Axioms concerning T are restricted so that only terms that are propositions are considered.

This is a highly intensional theory as the notion of equality is that of the λ-calculus: propositions are not to be equated just because they are always true together; similarly, properties are not to be equated just because they hold of the same terms (i.e. form true propositions with the same terms). Propositions in the language of terms may have the same truth conditions when T is applied, but this does not force them to be the same proposition, so we might have:

$$\mathrm{T}(s) \leftrightarrow \mathrm{T}(t)$$

but that does not mean that the terms are equal:

$$s = t$$

Similarly, in the language of wff, properties may hold of the same terms, yet they may be distinct. The λ-equality of terms is thus weaker than the notion of logical equivalence obtained when considering truth conditions in the meta-language.

The Formal Theory

The following presents a formalisation of the languages of terms and wff, together with the axioms that provide the closure conditions for P and T.

The Language of terms

Basic Vocabulary:

Individual variables:	$x, y, z, \ldots$
Individual constants:	$c, d, e, \ldots$
Logical constants:	$\vee, \wedge, \neg, \Rightarrow, \Xi, \Theta$

Inductive Definition of Terms:

(i) Every variable or constant is a term.
(ii) If t is a term and x is a variable then $\lambda x.t$ is a term.
(iii) If t and t' are terms then $t(t')$ is a term.

The Language of Wff

Inductive Definition of Wff:

(i) If t and s are terms then $s = t, \mathrm{P}(t), \mathrm{T}(t)$ are atomic wff.
(ii) If φ and φ' are wff then $\varphi \,\&\, \varphi'$, $\varphi \,\mathrm{v}\, \varphi'$, $\varphi \rightarrow \varphi'$, $\sim\varphi$ are wff.
(iii) If φ is a wff and x a variable then $\exists x\varphi$ and $\forall x\varphi$ are wff.

The theory is governed by the following axioms:

Axioms of The $\lambda\beta$-Calculus

$$\begin{aligned} \lambda x.t &= \lambda y.t[y/x] \text{ } y \text{ not free in } t \\ (\lambda x.t)t' &= t[t'/x] \end{aligned}$$

This defines the equivalence of terms.

The closure conditions for proposition-hood are given by the following axioms:

Axioms of Propositions

(i) $\mathrm{P}(t) \,\&\, \mathrm{P}(s) \rightarrow \mathrm{P}(t \wedge s)$
(ii) $\mathrm{P}(t) \,\&\, \mathrm{P}(s) \rightarrow \mathrm{P}(t \vee s)$
(iii) $\mathrm{P}(t) \,\&\, (\mathrm{T}(t) \rightarrow \mathrm{P}(s)) \rightarrow \mathrm{P}(t \Rightarrow s)$
(iv) $\mathrm{P}(t) \rightarrow \mathrm{P}(\neg t)$
(v) $\forall x \mathrm{P}(t) \rightarrow \mathrm{P}(\Theta\lambda x.t)$
(vi) $\forall x \mathrm{P}(t) \rightarrow \mathrm{P}(\Xi\lambda x.t)$
(vii) $\mathrm{P}(s \approx t)$

Truth conditions can be given for those terms that are propositions:

Axioms of Truth

(i) $\mathrm{P}(t)\,\&\,\mathrm{P}(s) \rightarrow (\mathrm{T}(t \wedge s) \leftrightarrow \mathrm{T}(t)\,\&\,\mathrm{T}(s))$
(ii) $\mathrm{P}(t)\,\&\,\mathrm{P}(s) \rightarrow (\mathrm{T}(t \vee s) \leftrightarrow \mathrm{T}(t)\,\mathrm{v}\,\mathrm{T}(s))$
(iii) $\mathrm{P}(t)\,\&\,(\mathrm{T}(t) \rightarrow \mathrm{P}(s)) \rightarrow$
$(\mathrm{T}(t \Rightarrow s) \leftrightarrow \mathrm{T}(t) \rightarrow \mathrm{T}(s))$
(iv) $\mathrm{P}(t) \rightarrow (\mathrm{T}(\neg t) \leftrightarrow\ \sim\mathrm{T}(t))$
(v) $\forall x\mathrm{P}(t) \rightarrow (\mathrm{T}(\Theta\lambda x.t) \leftrightarrow \forall x\mathrm{T}(t))$
(vi) $\forall x\mathrm{P}(t) \rightarrow (\mathrm{T}(\Xi\lambda x.t) \leftrightarrow \exists x\mathrm{T}(t))$
(vii) $\mathrm{T}(t \approx s) \leftrightarrow t = s$
(viii) $\mathrm{T}(t) \rightarrow \mathrm{P}(t)$

The last axiom states that only propositions may have truth conditions.

Note that the quantified propositions $\Theta\lambda x.t$, $\Xi\lambda x.t$ can be written as $\Theta x(t)$, $\Xi x(t)$, where the λ-abstraction is implicit.

This basic theory is very weak. The general approach for analysing semantic phenomena is to amend the theory either definitionally, as is done when adding dependent type constructors, or by strengthening it with more axioms and primitive notions, such as for events and plurals. This is, of course, in addition to obtaining appropriate representations for natural language phrases.

4. Types in Property Theory

Various types can be defined in the theory.

4.1. Basic Types

The notions of n-place relations can be defined recursively:

(i) $Rel_0(t) \leftrightarrow \mathrm{P}(t)$
(ii) $Rel_n(\lambda x.t) \leftrightarrow Rel_{n-1}(t)$

We can write $Rel_1(t)$ as $\mathrm{Pty}(t)$ and and $\lambda x.t$ as $\{x : t\}$. In keeping with this set-like notation, we can write $\mathrm{T}(tx)$ as $x\varepsilon t$, especially if t is a property.

4.2. Type Operators

We can also give definitions for intersection $\cap$, union $\cup$, difference $-$, cartesian product $\otimes$, disjoint union $\oplus$, and function space $\mapsto$ operators [Turner, 1992]. Only intersection and disjoint union will be illustrated here:

$$\begin{aligned} \cap &=_{def} \lambda f.\lambda g.\{x : fx \wedge gx\} \\ \oplus &=_{def} \lambda f.\lambda g.\{z : \\ &\qquad (\mathsf{fst}(z) \approx 0 \wedge f(\mathsf{snd}(z))) \\ &\qquad \vee (\mathsf{fst}(z) \approx 1 \wedge g(\mathsf{snd}(z)))\} \end{aligned}$$

which trivially support the theorems:

$$\begin{aligned} z\varepsilon(t \cap s) &\leftrightarrow z\varepsilon t\,\&\,z\varepsilon s \\ z\varepsilon(t \oplus s) &\leftrightarrow (\mathsf{fst}(z) = 0\,\&\,\mathsf{snd}(z)\varepsilon t) \\ &\qquad \vee (\mathsf{fst}(z) = 1\,\&\,\mathsf{snd}(z)\varepsilon s) \end{aligned}$$

Pairs $\langle , \rangle$ and $\mathsf{fst}, \mathsf{snd}$ have their usual definitions:

$$\begin{aligned} \mathsf{fst} &=_{def} \lambda p.p\lambda xy.x \\ \mathsf{snd} &=_{def} \lambda p.p\lambda xy.y \\ \langle x, y\rangle &=_{def} \lambda z.z(x)(y) \end{aligned}$$

so that:

$$\begin{aligned} \mathsf{fst}(\langle x, y\rangle) &=_\beta x \\ \mathsf{snd}(\langle x, y\rangle) &=_\beta y \end{aligned}$$

4.3. Dependent Types

The dependent type operators Σ, Π of MLTT can be defined with:

$$\begin{aligned} \Pi &=_{def} \lambda f.\lambda g.\{h : \Theta x(fx \Rightarrow gx(hx))\} \\ \Sigma &=_{def} \lambda f.\lambda g.\{h : f(\mathsf{fst}(h)) \wedge g(\mathsf{fst}(h))(\mathsf{snd}(h))\} \end{aligned}$$

These definitions support the following theorems:

If $\mathrm{Pty}(f)$ and $\forall x(x\varepsilon f \rightarrow \mathrm{Pty}(gx))$ then:

$$\begin{aligned} &\mathrm{Pty}(\Pi fg) \\ &\mathrm{Pty}(\Sigma fg) \end{aligned}$$

and:

$$\begin{aligned} h\varepsilon\Pi fg &\leftrightarrow \forall x(x\varepsilon f \rightarrow hx\varepsilon gx) \\ h\varepsilon\Sigma fg &\leftrightarrow \mathsf{fst}(h)\varepsilon f\,\&\,\mathsf{snd}(h)\varepsilon g(\mathsf{fst}(h)) \end{aligned}$$

To paraphrase these definitions, $h\varepsilon\Pi fg$ means that h is a function which takes an element (or 'proof'/'witness') of f and gives a 'proof' of g applied to that element of f. The expression $h\varepsilon\Sigma fg$ means that h is a pair, where the first component of the pair is an element of f, and the second is an element of g applied to that element of f.

With both of these types the evaluation of g is dependent upon the chosen element, or 'proof', of f. In some sense then, the meaning of g depends upon the context created by f. This gives us the means to give an MLTT based treatment of anaphora and discourse in PT.

Where before we had $\Pi x\varepsilon f.g$, we can write $\Pi f(\lambda x.g)$, although the former notation will be used below when there is no confusion in the intended meaning of ε. Similarly $\{x : fx \wedge gx\}$ will be written $\{x\varepsilon f.gx\}$.

Besides being a classical theory, PT has the advantages over MLTT. It is more intensional (false propositions are not necessarily equated), and has a simple extensional/intensional distinction (truth conditions expressed in the language of well-formed formulae constitute the extensional expressions, and intensional expression are represented in the language of terms).

Also, we can, if necessary, step outside the constraints imposed by MLTT. As an example, we can define the effect of the function app, discussed above in §2, in the language of well-formed formulae:

$$\begin{array}{l} \mathrm{P}(\mathsf{app}(y)(\Sigma ab)(v)h) \rightarrow \\ \quad (h\varepsilon\mathsf{app}(y)(\Sigma ab)(v) \leftrightarrow h\varepsilon\mathsf{app}(y)(a)(\mathsf{fst}v)) \end{array}$$

$$\begin{array}{l} \mathrm{P}(\mathsf{app}(y)(x)(v)h) \,\&\, {\sim}\exists ab(x = \Sigma ab) \rightarrow \\ \quad (h\varepsilon\mathsf{app}(y)(x)(v) \leftrightarrow h\varepsilon yv) \end{array}$$

5. Witness Suppression

With the machinery given above, we can interpret MLTT in PT. This has been done in more detail by Jan Smith [Smith, 1984]. However, this still interprets sentences as properties. To give an interpretation which treats sentences as propositions in the classical sense requires more work. In effect, the need for witnesses to propositions which do not contain nominals must be suppressed.

PT gives us type expressions as well as the classical logical connectives and quantifiers. This means that we could give a classical representation for the example sentence such as:

$$\exists x\varepsilon\{y\varepsilon\text{man}'.\text{walked-in}'y\}.\text{whistled}'(\text{he}_0)$$

where $\text{he}_0 = x$. The representation quantifies over those objects of type "man who walks".

It is not clear that we can derive such representations compositionally, at least not without assuming that all determiners are many ways ambiguous, as they may act as both quantifiers and type operators. One approach to obtain the desired representations is to compositionally obtain a representation in MLTT (implemented in PT), and then to suppress witnesses, turning the outermost type operators into classical quantifiers.

First, we can take a term t to be a *dependent type*, $\mathcal{DT}(t)$, if it is of the form $\Sigma x\varepsilon s.r$ or $\Pi x\varepsilon s.r$.

Using this notion, the outermost type operators in a discourse can be given a classical interpretation by the function $\mathcal{C}$:

$$\begin{array}{rcl} \mathcal{C}(\Sigma x\varepsilon f.g) & = & \exists x\varepsilon\mathcal{T}(f).\mathcal{C}(g) \\ & & \text{where } g\varepsilon\mathcal{DT} \\ \mathcal{C}(\Sigma x\varepsilon f.g) & = & \exists x\varepsilon\mathcal{T}(f).\mathcal{C}'(g) \\ & & \text{where } g\not\varepsilon\mathcal{DT} \\ \mathcal{C}(\Pi x\varepsilon f.g) & = & \forall x\varepsilon\mathcal{T}(f).\mathcal{C}(g) \\ & & \text{where } g\varepsilon\mathcal{DT} \\ \mathcal{C}(\Pi x\varepsilon f.g) & = & \forall x\varepsilon\mathcal{T}(f).\mathcal{C}'(g) \\ & & \text{where } g\not\varepsilon\mathcal{DT} \end{array}$$

The function $\mathcal{C}'$ takes MLTT propositions (which require a witness) and produces classical propositions. In the PT interpretation, this means that the argument for the witness is removed. The classical proposition should be true whenever the corresponding MLTT proposition is inhabited:

$$\begin{array}{l} \text{Pty}(g) \rightarrow \mathrm{P}(\mathcal{C}'g) \\ \text{Pty}(g) \,\&\, \exists h(h\varepsilon g) \rightarrow \mathrm{T}(\mathcal{C}'g) \end{array}$$

Assuming just nominal anaphora, we only want to keep the witnesses to classical properties representing nouns, and to suppress witnesses that arise out of the constructive interpretation of propositions.

We can define $\mathcal{T}$ for the operators Σ, Π as follows:

$$\begin{array}{rcl} \mathcal{T}(\Sigma x\varepsilon f.g) & = & \{x\varepsilon\mathcal{T}(f).\mathcal{C}'(g)\} \\ & & \text{where } g\not\varepsilon\mathcal{DT} \\ & = & \Sigma x\varepsilon\mathcal{T}(f).\mathcal{T}(g) \\ & & \text{otherwise} \\ \mathcal{T}(\Pi x\varepsilon f.g) & = & \\ & & \Pi x\varepsilon\mathcal{T}(f).\{z\varepsilon\mathcal{T}(f).\mathcal{C}'(g) \wedge x \approx z\} \\ & & \text{where } g\not\varepsilon\mathcal{DT} \\ & = & \Pi x\varepsilon\mathcal{T}(f).\mathcal{T}(g) \\ & & \text{otherwise} \\ \mathcal{T}(t) & = & t \text{ otherwise.} \end{array}$$

For the Π type containing no further nominals, this results in an identity function which is useful for plural pronouns. There is insufficient space to describe the application of this here.

The other basic type operator that should be considered is that of disjoint union $\oplus$. Defining the translation of disjoint union into a classical interpretation is left as an exercise (constructive negation, cartesian product and function space operators can be defined in terms of Π, Σ).

Essentially, if a term in $\mathcal{DT}$ contains another 'consequent' term in $\mathcal{DT}$, then it contains the representation of a nominal, and so should keep sufficient witness information to allow anaphoric reference to that embedded nominal. Otherwise, additional structure should be removed, so that witnesses to non-nominal terms are suppressed.

In the case of the Σ type, this means that we just require a type, or specification for the nominal 'antecedent'. For the Π type, in effect we produce a specification for the identity function with a side condition. If the relevant clause is true, then the side condition will hold, and the identity function will be specified, otherwise no function will satisfy the specification.

Plugging in our toy example, "A man walked in. He whistled.", representing "man" by m′, "walked in" by w-in′ and "whistled" by w′, we have:

$$\begin{aligned}&\mathcal{C}(\Sigma y\varepsilon(\Sigma x\varepsilon \text{m}'.\text{w-in}'x).(\text{w}'(\mathsf{fst}(y))))\\ &\quad= \exists y\varepsilon\mathcal{T}(\Sigma x\varepsilon \text{m}'.\text{w-in}'x).\mathcal{C}'(\text{w}'(\mathsf{fst}(y)))\\ &\quad= \exists y\varepsilon\{x\varepsilon\mathcal{T}(\text{m}').\mathcal{C}'(\text{w-in}'x)\}.(\text{w}'(\mathsf{fst}(y)))\\ &\quad= \exists y\varepsilon\{x\varepsilon \text{m}'.\text{w-in}'x\}.(\text{w}'(\mathsf{fst}(y)))\end{aligned}$$

Clearly this is mistaken: we have the wrong the selector function. Replacing the pronoun with $\mathsf{fst}(y)$ in the MLTT representation yields the wrong result and may be incorrectly typed when the representation is mapped into a classical proposition. The classical proposition requires that the pronoun be replaced by y. There are several possible solutions to this problem:

(i) Cause the substitution of a pair for y, with an arbitrary second element.
(ii) Rewrite the selector function so that with $\mathcal{C}(\Sigma y\varepsilon f.g)$ the term $\mathsf{fst}(y)$ in g becomes y.
(iii) Only replace pronouns by selector functions appropriate for the classical interpretation.
(iv) Employ a primitive form for the selector functions, and do not substitute the actual selector functions until the truth conditions are determined.

Only the first two options shall be considered here. Concerning the first option, the function $\mathcal{C}$ must be redefined as follows, for Σ:

$$\mathcal{C}(\Sigma x\varepsilon f.g) = \exists y\varepsilon\mathcal{T}(f).\mathcal{T}(g[x/\langle y, \emptyset\rangle])$$

where y is not free in g. Using this will yield the representation:

$$\exists y\varepsilon\{x\varepsilon \text{man}'.\text{walked-in}'x\}.(\text{whistled}'(\mathsf{fst}(\langle y, \emptyset\rangle)))$$

which by reduction is:

$$\exists y\varepsilon\{x\varepsilon \text{man}'.\text{walked-in}'x\}.(\text{whistled}'(y))$$

Concerning the Donkey sentences:

If a farmer owns a donkey, he beats it.

DRT treats this with conditional sub-DRSs. DPL modifiers the semantics of material implication. In MLTT, this is translated into:

$$\begin{aligned}&\Pi x\varepsilon(\Sigma y\varepsilon \text{farmer}'.(\Sigma z\varepsilon \text{donkey}'.\text{owns}'zy)).\\ &\quad\text{beats}'(\text{it}_0, \text{he}_0)\end{aligned}$$

where $\text{it}_0 = \mathsf{fst}(\mathsf{snd}(x))$ and $\text{he}_0 = \mathsf{fst}(x)$.

To remove the extra non-nominal witness information (the proof of ownership, and of beating), we apply $\mathcal{C}$ to the representation, and obtain the classical proposition:

$$\begin{aligned}&\forall x\varepsilon\Sigma y\varepsilon \text{farmer}'.\{z\varepsilon \text{donkey}'.\text{owns}'zy\}.\\ &\quad\text{beats}'(\text{it}_0, \text{he}_0)\end{aligned}$$

Here, the representation is effectively quantifying over pairs:

$$\langle f, d\rangle$$

such that f is a farmer, and d is a donkey owned by f.

Again, missing from this account is an indication of how the selection functions in the anaphoric expressions are work when they assume the presence of the deleted witness information. The selector functions for the example as represented in MLTT are $\text{it}_0 = \mathsf{snd}(x)$ and $\text{he}_0 = \mathsf{fst}(x)$. Following the same option take above, the translation defined by, for example:

$$\mathcal{T}(\Sigma x\varepsilon a.b) = \{x\varepsilon\mathcal{T}(a).\mathcal{C}'(b)\}$$

must be amended to:

$$\mathcal{T}(\Sigma x \varepsilon a.b) = \{y \varepsilon \mathcal{T}(a).\mathcal{C}'(b[x/\langle y, \emptyset\rangle])\}$$

where y is not free in c.

Concerning the second option (which proposes that the selector function be rewritten), one way of proceeding is to find which parts of the binary tree of nested pairs have been pruned, and then amend the selector functions that are affected by this pruning.

The following function yields a binary tree with binary valued leaves which are intended to reflect the patterns of deletion after non-nominal witness suppression:

$$\begin{array}{lcl} \mathsf{cut}(\Sigma x \varepsilon f.g) & = & \langle \mathsf{cut}(f), \mathsf{cut}(g)\rangle \\ & & \text{where } g \in \mathcal{DT} \\ \mathsf{cut}(\Sigma x \varepsilon f.g) & = & \langle \mathsf{cut}(f), \mathsf{true}\rangle \\ & & \text{where } g \notin \mathcal{DT} \\ \mathsf{cut}(\Pi x \varepsilon f.g) & = & \mathsf{false} \\ \mathsf{cut}(t) & = & \mathsf{false} \text{ otherwise} \end{array}$$

Any function which rewrites selector functions will depend upon the value of cut for the expression in which the selector appears.

The function R_g which rewrites the selector-functions, where g is a term representing a dependent type, can be defined as follows:

$$\begin{array}{ll} \multicolumn{2}{l}{\mathsf{R}_g[s_1(s_2(s_3 \ldots (s_n z) \ldots))]} \\ = & \mathsf{R}_g[s_2(s_3 \ldots (s_n z) \ldots)] \\ & \text{when } s_1 = \mathsf{fst} \\ & \text{and } \mathsf{snd}(s_2(s_3 \ldots (s_n \mathsf{cut}(g)) \ldots)) = \mathsf{true} \\ = & s_1(\mathsf{R}_g[s_2(s_3 \ldots (s_n z) \ldots)]) \\ & \text{otherwise} \end{array}$$

where s_m $(1 \leq m \leq n)$ is one of fst, snd.

The intension is that g is the type expression in which the selector function appears.

Now we are in a position to give the witness suppression functions, amended to rewrite any selector functions as appropriate. The definition of $\mathcal{C}$ should be altered along the following lines:

$$\begin{array}{l} \mathcal{C}(\Sigma x \varepsilon f.g) = \exists x \varepsilon \mathcal{T}(f).\mathcal{C}(g') \\ \text{where} \\ g' = g[\mathsf{fst}(s_2(\ldots(s_n x)\ldots))/\mathsf{R}_g(s_2(\ldots(s_n x)\ldots))] \end{array}$$

The clauses which define $\mathcal{T}$ need to be amended using the pattern:

$$\begin{array}{l} \mathcal{T}(\Pi x \varepsilon f.g) = \Pi x \varepsilon \mathcal{T}(f).\mathcal{T}(g') \\ \text{where} \\ g' = g[s_1(\ldots(s_n x)\ldots)/\mathsf{R}_g(s_1(\ldots(s_n x)\ldots))] \end{array}$$

Note that this will not work for arbitrary type expressions: we cannot guarantee a canonical form for the selector functions. The functions assume that the selector functions only apply to the bound variable and not to some variable equated with it. It must be assumed that only the operators Σ, Π appear in the representation.

To achieve the correct results with relative clauses, the function app must be amended with the addition of one clause for non-dependent sum types:

$$\mathsf{app}yxv = \begin{cases} \mathsf{app}ya(\mathsf{fst}v) & \text{if } x = \Sigma ab \\ y\mathsf{fst}v & \text{if } x = \{y\varepsilon a.b\} \\ yv & \text{otherise} \end{cases}$$

In the interests of clarity, I have defined the functions $\mathcal{C}, \mathcal{T}, \mathcal{C}', \mathsf{cut}, \mathsf{R}_g$ outside PT, although they can have there effect implemented directly in the theory by additional axioms of truth.

Due to lack of space, it is left as an exercise to prove that the representation of a sentence which is typed as a property will be mapped into a proposition by $\mathcal{C}$.

6. Conclusions

It has been shown how a dependent type analysis of anaphoric resolution can be given a systematic interpretation using a classical notion of propositionhood, distinct from the notion of 'property'. The ideas were expressed in a first-order property theory. Several advantages follow from this. Unlike higher order interpretations used in Montague's IL and dynamic versions of it, classical first-order theories have a semi-decidable proof theory. This is surely useful if we wish to do something with our representations, other than admire them.

Property theory also has fine-grained intensionality, as propositions are taken to be primitive in PT. It is more intensional than both possible worlds based approaches (where propositions are equated if they are necessarily true together) and basic MLTT (where propositions are defined by their proofs, and hence equal if false[3]). In addition, the weak typing of PT permits a fairly simple treatments of nominalisation.

One question which suggests itself is whether the classical semantics sketched above can be obtained compositionally. A problem is presented by the context dependence of the interpretation of the determiners. For example, the indefinite "a" has three possible translations depending upon its context: it can be a classical existential quantifier (when at the outermost level of the discourse); a set forming operator (when there are no embedded dependent types); or a dependent type operator (otherwise). This makes it difficult to maintain a strictly compositional analysis, unless the determiner is taken to be ambiguous.

In the treatment above, the non-compositionality arises in the process of obtaining a classical interpretation. Arguably, other treatments of anaphora in discourse often employ some non-compositional context sensitivity because of the nature of the phenomena.

The paper gives a classical interpretation of the Donkey sentences via MLTT. This has the side effect of showing how MLTT expressions themselves can be mapped to classical propositions. However, for the purposes of NL semantics, the initial representation need not be MLTT, it could be some representation which just specifies the predicate argument structure of the sentences. Further work is needed to improve the coverage.

Acknowledgements: This paper is based on a suggestion made by Ray Turner. The text was prepared using facilities provided by the Department of Computer Science at Essex University.

References

[Aczel and Lunnon, 1991] Peter Aczel and Rachel Lunnon. Universes and parameters. In Jon Barwise, Mark Gawron, Gordon Plotkin, and Syun Tutiya, editors, *Situation Theory and its Applications II*, CSLI Lecture Notes. 1991.

[Aczel, 1980] P. Aczel. Frege structures and the notions of proposition, truth and set. In Barwise, Keisler, and Keenan, editors, *The Kleene Symposium*, North Holland Studies in Logic, pages 31–39. North Holland, 1980.

[Davila-Perez, 1994] Rogelio Davila-Perez. Constructive type theory and natural language, May 1994. Computer Science Memorandum 206, University of Essex.

[Groenendijk and Stokhof, 1990] J. Groenendijk and M. Stokhof. Dynamic Montague grammar. ITLI Prepublication Series LP–90–02, 1990.

[Groenendijk and Stokhof, 1991] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[Hoare, 1969] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):567–580, 583, 1969.

[Kamp, 1981] H. Kamp. Theory of truth and semantic representation. In J.A.K. Groenendijk, T.M.V. Janssen, and M.B.J. Stokhof, editors, *Formal Methods in the Study of Language*, Mathematical Centre Tracts 135, pages 277–322. Amsterdam, 1981.

[Martin-Löf, 1982] P. Martin-Löf. Constructive mathematics and computer programming. In Cohen, Los, Pfeiffer, and Podewski, editors, *Logic, Methodology and Philosophy of Science VI*, pages 153–179. North Holland, 1982.

[Martin-Löf, 1984] P. Martin-Löf. *Studies in Proof Theory (Lecture Notes).* Bibliopolis, Napoli, 1984.

[Ranta, 1991] Aarne Ranta. Intuitionistic categorial grammar. *Linguistics and Philosophy*, 14:203–239, 1991.

[Smith, 1984] Jan M. Smith. An interpretation of Martin-Löf's Type Theory in a type-free theory of propositions. *Journal of Symbolic Logic*, 49, 1984.

[Sundholm, 1989] G. Sundholm. Constructive generalised quantifiers. *Synthese*, 79:1–12, 1989.

[Turner, 1990] R. Turner. *Truth and Modality for Knowledge Representation.* Pitman, 1990.

[Turner, 1992] Raymond Turner. Properties, propositions and semantic theory. In Michael Rosner and Roderick Johnson, editors, *Computational Linguistics and Formal Semantics*, Studies in Natural Language Processing, pages 159–180. Cambridge University Press, Cambridge, 1992.

[3]See Footnote 1