A Type-Theoretic Approach to Anaphora and Ellipsis Resolution[*]

Chris Fox and **Shalom Lappin**
Department of Computer Science
University of Essex — King's College, London
Wivenhoe Park — The Strand
Colchester CO4 3SQ — London WC2R 2LS
United Kingdom
foxcj@essex.ac.uk lappin@dcs.kcl.ac.uk

Abstract

We present an approach to anaphora and ellipsis resolution in which pronouns and elided structures are interpreted by the dynamic identification in discourse of type constraints on their semantic representations. The content of these conditions is recovered in context from an antecedent expression. The constraints define separation types (sub-types) in Property Theory with Curry Typing (PTCT), an expressive first-order logic with Curry typing that we have proposed as a formal framework for natural language semantics.

1 Introduction

We present a type-theoretic account of pronominal anaphora and ellipsis resolution within the framework of Property Theory with Curry Typing (PTCT), a first-order logic with comparatively rich expressive power achieved through the addition of Curry typing. PTCT is a fine-grained intensional logic that permits distinct propositions (and other intensional entities) to be provably equivalent. It supports functional, separation (sub), and comprehension types. It also allows a weak form of polymorphism, which seems adequate to capture type-general expressions in natural language. The proof and model theories of PTCT are classically Boolean with respect to negation, disjunction, and quantification. Quantification over functional and type variables is restricted to remain within the domain of a first-order system.

We take the resolution of pronominal anaphora to be a dynamic process of identifying the value of a type parameter with an appropriate part of the representation of an antecedent. The parameter corresponds to the specification of a sub-type condition on a quantified individual variable in PTCT, where the content of this condition is recovered from part of the antecedent expression. We propose a unified treatment for pronouns that accommodates bound variable, E-type, and donkey anaphora.

We represent ellipsis as the identification of a value of a separation type parameter for expressions in the ellipsis site. The separation type provides a predicate for the bare element(s) in the ellipsis structure. The value of the parameter is constructed by abstraction on an antecedent expression. When variables corresponding to pronouns are contained in the separation type retrieved from the antecedent, typing conditions on these variables are imported into the interpretation of the elided term. Different specifications of these conditions may be possible, where each specification produces a distinct reading. Using alternative resolutions of typing constraints on the pronoun variables in the ellipsis site permits us to explain the distinction between strict vs. sloppy readings of pronouns under ellipsis. It also allows us to handle antecedent contained ellipsis without invoking syntactic mechanisms like quantifier phrase movement or semantic operations like storage. Our treatment of ellipsis is similar to the higher-order unification (HOU) analysis proposed in Dalrymple, Shieber, and Pereira (1991) and Shieber, Pereira, and Dalrymple (1996). However, there are a number of important differences between the two approaches which we will take up in Section 7.

In Section 2 we give a brief summary of the main features of PTCT, particularly the type definitions. More detailed descriptions are provided in Fox, Lappin, and Pollard (2002a), Fox, Lappin, and Pollard (2002b), Fox and Lappin (2003a), and Fox and Lappin (2003b). In Section 3 we add an intensional number theory, and in Section 4 we characterize generalized quantifiers (GQs) corresponding to quantified NPs within PTCT. Section 5 gives our treatment of pronominal anaphora,

[*]The research of the second author has been supported by grant number AN/2687/APN from the Arts and Humanities Research Board of the UK, and grant number RES-000-23-0065 from the Economic and Social Research Council of the UK.

and we present our account of ellipsis in Section 6. In Section 7 we compare our account of pronominal anaphora to Ranta (1994)'s analysis of donkey anaphora within Martin-Löf Type Theory (MLTT) and our treatment of ellipsis to the HOU approach. Finally, in Section 8 we present some conclusions and consider directions for future work.

2 PTCT

The core language of PTCT consists of the following sub-languages:

(1) Terms $t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$
logical constants $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\perp} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$

(2) Types $T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S$

(3) Wff $\varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi) \mid (\forall x \varphi) \mid (\exists x \varphi) \mid {}^{\mathsf{true}}t$
atomic wff $\alpha ::= (t =_T s) \mid \perp \mid t \in T \mid t \cong_T s$

The language of terms is the untyped λ-calculus, enriched with logical constants. It is used to *represent* the interpretations of natural language expressions. It has no internal logic. With an appropriate proof theory, the simple language of types together with the language of terms can be combined to produce a Curry-typed λ-calculus. The first-order language of wffs is used to formulate type judgements for terms, and truth conditions for those terms judged to be in Prop.[1]

It is important to distinguish between the notion of a proposition itself (in the language of wff), and that of a term that *represents* a proposition (in the language of terms). ${}^{\mathsf{true}}(t)$ will be a true wff whenever the proposition represented by the term t is true, and a false wff whenever the proposition represented by t is false. The representation of a proposition t ($\in \mathsf{Prop}$) is distinct from its truth conditions (${}^{\mathsf{true}}(t)$).

We construct a tableau proof theory for PTCT.[2] Its rules can be broken down into the following kinds.

- The basic connectives of the wff: These have the standard classical first-order behaviour.
- Identity of terms ($=$): These are the usual rules of the untyped λ-calculus with α, β and η reduction.
- Typing of λ-terms: These are essentially the rules of the Curry-typed calculus, augmented with rules governing those terms that represent propositions (Prop).
- Truth conditions for Propositions: Additional rules for the language of wffs that govern the truth conditions of terms in Prop (which represent propositions).
- Equivalence ($\cong_T$): The theory has an internal notion of extensional equivalence which is given the expected behaviour.

There are two equality notions in PTCT. $t \cong_T s$ states that the terms t, s are extensionally equivalent in type T. Extensional equivalence is represented in the language of terms by $t \hat{\cong}_T s$. $t =_T s$ states that two terms are intensionally identical. The rules for intensional identity are essentially those of the $\lambda\alpha\beta\eta$-calculus. It is represented in the language of terms by $t \hat{=}_T s$. It is necessary to type the intensional identity predicate in order to avoid paradoxes when we introduce comprehension types.

The rules governing equivalence and identity are such that we are able to derive $t =_T s \rightarrow t \cong_T s$ for all types inhabited by t (s), but not $t \cong_T s \rightarrow t =_T s$. As a result, PTCT can sustain fine-grained intensional distinctions among provably equivalent propositions. Therefore, we avoid the reduction of logically equivalent expressions to the same intension, a reduction which holds in classical intensional semantics, without invoking impossible worlds. Moreover, we do so within a first-order system that uses a flexible Curry typing system rather than a higher-order logic with Church typing (as in Fox, Lappin, and Pollard's (2002c) modification of Church's (1940) Simple Theory of Types).

One possible extension that we could consider is to add a universal type Δ to the types, and rules corresponding to the following axiom.

(4) UT: $x \in \Delta \leftrightarrow x = x$

Unfortunately this is inconsistent in PTCT if Prop is a type. Consider rr, where $r = \lambda x.\hat{\exists} y \in$

[1] Negation is defined by $\sim p =_{\mathsf{def}} p \rightarrow \perp$. Although we could formulate a constructive theory, in the following we assume rules that yield a classical Boolean version of the theory.

[2] For an introduction to tableau proof procedures for first-order logic with identity see Jeffrey (1982). Fitting (1996) presents an implemented tableau theorem prover for first-order logic with identity, and he discusses its complexity properties.

$(\Delta \Longrightarrow \mathsf{Prop})[x \hat{=} y \hat{\wedge} \hat{\sim} xy]$. However, there are other consistent extensions that can be adopted.

2.1 Separation Types

We add $\{x \in T : \varphi'\}$ to the types, and a tableau rule that implements the following axiom.

(5) SP: $z \in \{x \in T : \varphi'\} \leftrightarrow (z \in T \wedge \varphi'[z/x])$

Note that there is an issue here concerning the nature of φ. To ensure the theory is first-order, this type needs to be term representable, so φ' must be term representable. To this end, we can define a term representable fragment of the language of wffs. First, we introduce syntactic sugar for typed quantification in the wffs.

(6) (a) $\forall_T x \varphi =_{\mathsf{def}} \forall x(x \in T \rightarrow \varphi)$
(b) $\exists_T x \varphi =_{\mathsf{def}} \exists x(x \in T \wedge \varphi)$

Wff's with these typed quantifiers, and no free-floating type judgements will then have direct intensional analogues—that is, term representations—which will always be propositions. We can define representable wffs by φ':

(7) $\varphi' ::= \alpha' \mid (\varphi' \wedge \psi') \mid (\varphi' \vee \psi') \mid (\varphi' \rightarrow \psi') \mid (\varphi' \leftrightarrow \psi') \mid (\forall_T x \varphi') \mid (\exists_T x \varphi) \mid {}^{\mathsf{true}}t$
atomic representable wffs
$\alpha' ::= (t =_T s) \mid \perp \mid t \cong_T s$

The term representations of representable wffs $\lceil \alpha' \rceil$ are given by the following.

(8) (a) $\lceil a \wedge b \rceil = \lceil a \rceil \hat{\wedge} \lceil b \rceil$
(b) $\lceil a \vee b \rceil = \lceil a \rceil \hat{\vee} \lceil b \rceil$
(c) $\lceil a \rightarrow b \rceil = \lceil a \rceil \hat{\rightarrow} \lceil b \rceil$
(d) $\lceil a \leftrightarrow b \rceil = \lceil a \rceil \hat{\leftrightarrow} \lceil b \rceil$
(e) $\lceil a \cong_T b \rceil = \lceil a \rceil \hat{\cong}_T \lceil b \rceil$
(f) $\lceil a =_T b \rceil = \lceil a \rceil \hat{=}_T \lceil b \rceil$
(g) $\lceil \perp \rceil = \hat{\perp}$
(h) $\lceil {}^{\mathsf{true}}t \rceil = t$
(i) $\lceil \forall_T x.a \rceil = \hat{\forall} x \epsilon T \lceil a \rceil$
(j) $\lceil \exists_T x.a \rceil = \hat{\exists} x \epsilon T \lceil a \rceil$

Now we can express separation types as $\{x \in S.\varphi'\}$, which can be taken to be sugar for $\{x \epsilon S.\lceil \varphi' \rceil\}$.

The following theorem is an immediate consequence of the recursive definition of representable wffs and their term representations.

Theorem 1 (Representability) $\lceil \varphi' \rceil \in \mathsf{Prop}$ *for all representable wffs* φ', *and furthermore* ${}^{\mathsf{true}}\lceil \varphi' \rceil \leftrightarrow \varphi'$.

2.2 Comprehension Types

Usually comprehension can be derived from SP and UT. We are forgoing UT to avoid paradoxes, so we have to define comprehension independently. The same arguments apply as for SP concerning representability. We add the type $\{x : \varphi'\}$ and a tableau rule corresponding to the following axiom.

(9) COMP: $z \in \{x : \varphi\} \leftrightarrow \varphi[z/x]$

Given that COMP = SP + UT, where UT is the Universal Type $\Delta = \{x : x \hat{=} x\}$, we would derive a paradox if $=$ was not typed. This is because in PTCT Prop is a type. So rr, where $r = \lambda x.\hat{\exists} y \in (\Delta \Longrightarrow \mathsf{Prop})[x \hat{=} y \hat{\wedge} \hat{\sim} xy]$ produces a paradoxical propositional. Our use of a typed intensional identity predicate filters out the paradox because it must be possible to prove that the two expressions for which $=_T$ is asserted are of type T independently of the identity assertion. $s =_T t$ iff $s, t \in T$ and $s = t$.

2.3 Polymorphic Types

We enrich the language of types to include type variables X, and the wffs to include quantification over types $\forall X \varphi, \exists X \varphi$.

We add $\Pi X.T$ to the language of types, governed by the tableau rule corresponding to the following axiom:

(10) PM: $f \in \Pi X.T \leftrightarrow \forall X(f \in T)$

Polymorphic types permit us to accommodate the fact that natural language expressions such as coordination and certain verbs can apply as functions to arguments of different types.

Note that PM is impredicative (the type quantification ranges over the types that are being defined). To avoid this, we add a language of Kinds (K) to PTCT.

(11) Kinds: $K ::= T \mid \Pi X.K$

(12) PM′: $f \in \Pi X.K \leftrightarrow \forall X(f \in K)$ where X ranges only over types.

This constraint limits quantification over types to type variables that take non-Kind types as values. Therefore, we rule out iterated type-polymorphism in which functional polymorphic types apply to polymorphic arguments. In fact, this weak version of polymporphism seems to be adequate to express the instances of multiple type assignment that occur in natural languages (van Benthem, 1991).

2.4 Final Syntax

Adopting the extensions discussed above, which do not allow the derivation of a paradox, leads to the following language.

(13) Terms $t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$
(logical constants) $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$

(14) Types $T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S \mid X \mid \{x \in T.\varphi'\} \mid \{x.\varphi'\}$

(15) Kinds $K ::= T \mid \Pi X.T$

(16) Wff $\varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi)$
$\mid (\forall x\varphi) \mid (\exists x\varphi) \mid (\forall X\varphi) \mid (\exists X\varphi) \mid {}^{\mathsf{true}}t$
(atomic wff) $\alpha ::= (t =_T s) \mid \bot \mid t \in K \mid t \cong_T s$

where φ' is as defined in section 2.1.

2.5 A Model Theory for PTCT

In order to give a model for PTCT, we first need a model of the untyped λ-calculus. This will form the model for PTCT's language of terms. Here we present Meyer's model (Meyer, 1982).

Definition 1 (General Functional Models) *A functional model is a structure of the form $\mathcal{D} = \langle D, [D \rightarrow D], \Phi, \Psi\rangle$ where*

(1) D is a non-empty set,

(2) $[D \rightarrow D]$ is some class of functions from D to D,

(3) $\Phi : D \rightarrow [D \rightarrow D]$,

(4) $\Psi : [D \rightarrow D] \rightarrow D$,

(5) $\Psi(\Phi(d)) = d$ for all $d \in D$

We can interpret the calculus as using the following,where g is an assignment function from variables to elements of D.

(17)
$$[\![x]\!]_g = g(x)$$
$$[\![\lambda x.t]\!]_g = \Psi(\lambda d.[\![t]\!]_{g[d/x]})$$
$$[\![ts]\!]_g = \Phi([\![t]\!]_g)[\![s]\!]_g$$

This interpretation exploits the fact that Φ maps every element of D into a corresponding function from D to D, and Ψ maps functions from D to D into elements of D.

Note we require that functions of the form $\lambda d.[\![t]\!]_{g[d/x]}$ are in the class $[D \rightarrow D]$ to ensure that the interpretation is well defined.

In the case where we permit constant terms, then we can add the clause

(18) $[\![c]\!]_g = i(c)$

where i assigns elements of D to constants.

Theorem 2 *If $t = s$ in the extensional untyped λ-calculus (with ξ and η), then $[\![t]\!]_g = [\![s]\!]_g$ for each assignment g.*

Proof: By induction on the derivations. $\square$

A model $\mathcal{M}$ of PTCT is constructed on the basis of a simple extensional model of the untyped λ-calculus (Meyer, 1982; Barendregt, 1984; Turner, 1997), with additional structure added to capture the type rules and the relation between the sub-languages of PTCT.

On the basis of the full proof and model theories, we prove the soundness and completeness of PTCT.[3]

3 An Intensional Number Theory

We add an intensional number theory to PTCT, incorporating rules corresponding to the axioms in (23).

(19) Terms: $0 \mid succ \mid pred \mid add \mid mult \mid \hat{most} \mid |\cdot|_B$

(20) Types: Num

(21) Wffs: $zero(t) \mid t \cong_{\mathsf{Num}} t' \mid t <_{\mathsf{Num}} t' \mid most(p)(q)$

(22) Axioms for Num: The usual Peano axioms, adapted to PTCT

(23) Axioms for $<_{\mathsf{Num}}$:

(a) $y \in \mathsf{Num} \rightarrow 0 <_{\mathsf{Num}} succ(y)$
(b) $x \in \mathsf{Num} \rightarrow x \not<_{\mathsf{Num}} 0$
(c) $x \in \mathsf{Num} \wedge y \in \mathsf{Num} \rightarrow (succ(x) <_{\mathsf{Num}} succ(y) \leftrightarrow x <_{\mathsf{Num}} y)$

The model theory can be extended in a straightforward way to support these new rules of the proof theory.

[3]The full proof and model theories for PTCT, and the proofs for soundness and completeness are presented by Fox and Lappin (2003b).

When we incorporate the intensional number theory into PTCT we lose completeness of the proof theory. However, it is important to recognize that incompleteness sets in only when tableau rules that encode number-theoretic inferences are applied. The basic logic and type system of PTCT without these rules and their corresponding definitions in the model theory remains complete.

4 Representing Proportional Generalized Quantifiers in PTCT

By defining the cardinality of properties, we can express the truth conditions of proportional quantifiers in PTCT.

The cardinality of properties $|p|_B$ is formulated as follows.

(24) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge \sim\exists x(x \in B \wedge {}^{\mathsf{true}}px) \rightarrow$
$|p|_B \cong_{\mathsf{Num}} 0$

(25) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge b \in B \wedge {}^{\mathsf{true}}pb \rightarrow$
$|p|_B \cong_{\mathsf{Num}}$
$add(|\lambda x(px \mathbin{\hat{\wedge}} \hat{\sim} x \mathbin{\hat{=}}_B b)|_B)(succ(0))$

The cardinality of types can be defined in a similar way.

We represent $most(p)(q)$ as follows:

(26) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge q \in (B \Longrightarrow \mathsf{Prop}) \rightarrow$
$most(p)(q) \leftrightarrow$
$|\{x \in B.{}^{\mathsf{true}}px \wedge \sim {}^{\mathsf{true}}qx\}|_B$
$<_{\mathsf{Num}} |\{x \in B.{}^{\mathsf{true}}px \wedge {}^{\mathsf{true}}qx\}|_B$

Given that PTCT is a first-order theory in which all quantification is limited to first-order variables, this characterization of *most* effectively encodes a higher-order GQ within a first-order system.

5 A Type-Theoretical Approach to Anaphora

We combine our treatment of generalised quantifiers with our characterisation of separation types to provide a unified type-theoretic account of anaphora. We assume that all quantified NPs are represented as cardinality relations on the model of our treatment of *most*. Pronouns are represented as appropriately typed free variables. If the free variable is within the scope of a set forming operator that specifies a sub-type and it meets the same typing constraints as the variable bound by the operator, the variable can be interpreted as bound by the operator through substitution under α identity. This interpretation yields the bound reading of the pronoun.

(27) Every man loves his mother.

(28) $|\{x \in B.{}^{\mathsf{true}}man'(x)$
$\wedge\ {}^{\mathsf{true}}love'(x, mother\text{-}of'(y))\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.{}^{\mathsf{true}}man'(x)\}|_B$
$\rightarrow$
$|\{x \in B.{}^{\mathsf{true}}man'(x)$
$\wedge\ {}^{\mathsf{true}}love'(x, mother\text{-}of'(x))\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.{}^{\mathsf{true}}man'(x)\}|_B$

Representations of this kind are generated by compositional semantic operations as described by (for example) Lappin (1989), and Lappin and Francez (1994).

When the pronoun is interpreted as dependent upon an NP which does not bind it, we represent the pronoun variable as constrained by a separation type parameter whose value is supplied by context. Generally, the value of this type parameter is determined in two parts. The initial type membership condition is imported directly from the antecedent corresponding to the GQ. The wff part of the separation type corresponds to the relation between the restriction and the predication in the antecedent clause. In the default case, the pronoun variable is bound by a universal quantifier in the language of wffs.

(29) Every student arrived.

(30) $|\{x \in B.{}^{\mathsf{true}}student'(x) \wedge {}^{\mathsf{true}}arrived'(x)\}|_B$
$\cong_{Num} |\{x \in B.{}^{\mathsf{true}}student'(x)\}|_B$

(31) They sang.

(32) $\forall y \in A.({}^{\mathsf{true}}sang'(y))$
where $A = \{x \in B.{}^{\mathsf{true}}student'(x)$
$\wedge\ {}^{\mathsf{true}}arrived'(x)\}$

In the case of proper names and existentially quantified NP antecedents we obtain the following.

(33) John arrived.

(34) ${}^{\mathsf{true}}arrived'(john)$

(35) He sang.

(36) $\forall y \in A.({}^{\mathsf{true}}sang'(y))$
where $A = \{x \in B.{}^{\mathsf{true}}x \mathbin{\hat{=}}_B john$
$\wedge\ {}^{\mathsf{true}}arrived'(x)\}$

(37) Some man arrived.

(38) $|\{x \in B.{}^{\mathsf{true}}man'(x) \wedge {}^{\mathsf{true}}arrived'(x) \wedge {}^{\mathsf{true}}\phi(x)\}|_B >_{Num} 0$

(39) He sang.

(40) $\forall y \in A.({}^{\mathsf{true}}sang'(y))$
where $A = \{x \in B.{}^{\mathsf{true}}man'(x) \wedge {}^{\mathsf{true}}arrived'(x) \wedge {}^{\mathsf{true}}\phi(x)\}$

ϕ is a predicate that is specified in context and uniquely identifies a man who arrived in that context.

We handle donkey anaphora in PTCT through a type constraint on the variable corresponding to the pronoun.

(41) Every man who owns a donkey beats it.

(42) $|\{x \in B.{}^{\mathsf{true}}man'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0) \wedge \forall z \in A({}^{\mathsf{true}}beat'(x,z))\}|_B$
$\cong_{Num}$
$|\{x \in B.{}^{\mathsf{true}}man'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0)\}|_B$
where
$A = \{y \in B.{}^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}$

The representation asserts that every man who owns at least one donkey beats all of the donkeys that he owns.

Our type-theoretic account of donkey anaphora is similar in spirit to the E-type analysis proposed by Lappin and Francez (1994). There is, however, an important difference. Lappin and Francez (1994) interpret an E-type pronoun as a choice function from the elements of an intersective set specified by the clause containing the antecedent NP to a range of values determined by this NP. Both the domain and range of the function are described informally in terms of the semantic representation of the antecedent clause. On the type-theoretic approach proposed here the interpretation of the E-type pronoun is specified explicitly through type constraints on variables in the semantic representation language. Therefore our account provides a more precise and properly formalized treatment of pronominal anaphora.

We can generate existential readings of donkey sentences (Pelletier & Schubert, 1989) by treating the principle that the free variable representing a pronoun is bound by a universal quantifier as defeasible. We can then substitute an existential for the universal quantifier.

(43) Every person who had a quarter put it in a parking meter.

(44) $|\{x \in B.{}^{\mathsf{true}}person'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}had'(x,y) \wedge {}^{\mathsf{true}}quarter'(y)\}|_B >_{Num} 0) \wedge \exists z \in A({}^{\mathsf{true}}put\text{-}in\text{-}meter'(x,z))\}|_B$
$\cong_{Num}$
$|\{x \in B.{}^{\mathsf{true}}person'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}had'(x,y) \wedge {}^{\mathsf{true}}quarter'(y)\}|_B >_{Num} 0)\}|_B$
where $A = \{y \in B.{}^{\mathsf{true}}had'(x,y) \wedge {}^{\mathsf{true}}quarter'(y)\}$

This representation asserts that every person who had a quarter put at least one quarter that he/she had in a parking meter.

The default presence of a universal quantifier in the PTCT representation of a donkey pronoun is an instance of a pragmatic maximality condition of the kind that Lappin and Francez (1994) invoke to explain the preferred readings of sentences like (41). As they observe, lexical semantic and pragmatic factors can override a maximality constraint in cases like (43). We represent the suspension of the maximality requirement by substituting an existential for the universal quantifier binding the variable corresponding to the pronoun in these sentences.

6 Ellipsis

Let S be a parameter that is instantiated by separation types. We can represent a clause containing an elided VP like (45) as (46).

(45) John sings, and Mary does too.

(46) ${}^{\mathsf{true}}sings'(john) \wedge mary \in S$

Assuming that *mary* and *john* are both of type B, we can abstract on *john* to obtain the separation type $\{x \in B.{}^{\mathsf{true}}sings'(x)\}$ from the antecedent in order to resolve S. This yields the desired interpretation of the elided clause in (47).[4]

[4]The presentation adopted here requires a slight change to PTCT as it is formulated elsewhere (Fox et al., 2002a,

(47) $^{\mathsf{true}}sings'(john) \wedge {}^{\mathsf{true}}sings'(mary)$

We have not introduced product types into PTCT, but we are assuming that all predicate types are curried functions. For simplicity of notation we represent transitive and ditransitive verbs as multi-argument functional expressions, but we continue to assume that they are interpreted as a sequence of curried functions. Similarly we assume that separation types corresponding to curried function predicate can be built up through curried function application.

Our treatment of VP ellipsis extends directly to gapping (48).[5]

(48) Mary reviewed Principia and Max Ulysses.

(49) $^{\mathsf{true}}reviewed'(john, principia)$
$\wedge\ (max, ulysses) \in S$

(50) $S = \{(x, y) \in B \otimes B.{}^{\mathsf{true}}reviewed'(x, y)\}$

(51) $^{\mathsf{true}}reviewed'(john, principia) \wedge$
$^{\mathsf{true}}reviewed'(max, ulysses)$

It also applies to pseudogapping (52).[6]

(52) Max introduced Rosa to Sam before
Bill did Mary to John.

(53) $^{\mathsf{true}}introduced'(max, rosa, sam)$
before
$(bill', mary, john) \in S$

(54) $S = \{(x, y, z) \in B \otimes B \otimes B.$
$^{\mathsf{true}}introduced'(x, y, z)\}$

(55) $^{\mathsf{true}}introduced'(max, rosa, sam)$
before
$^{\mathsf{true}}introduced'(bill, mary, john)$

If we combine our treatment of ellipsis with our account of pronominal anaphora, the representation of the distinction between strict and sloppy readings of pronouns under ellipsis is straightforward.

2002b; Fox & Lappin, 2003a) in order to allow free-floating type judgements within the language of terms. This extension should not be problematic if, for example, we limit such assertions of type membership to terms that have a normal form. The free floating type judgements can also be given a property-theoretic presentation, where membership is represented by functional application.

[5]Here we use product types. Product types can be added to PTCT, or the examples can be represented using an equivalent curried form.

[6]Here we assume there is some suitable treatment of the temporal ordering of the circumstances described by the propositions p and q in the expression p before q without further elaboration.

(56) John loves his mother, and Bill does too.

(57) $\forall x \in A({}^{\mathsf{true}}loves'(john, mother\text{-}of'(x))$
$\wedge\ bill \in S$

Let S be defined as follows.

$$S = \{y \in B.\forall x \in A({}^{\mathsf{true}}loves'(y, mother\text{-}of'(x))\}$$

If the type parameter A on the variable x is specified as $\{w \in B.{}^{\mathsf{true}}w \hat{=}_B\ john\}$ in the antecedent clause prior to the resolution of S, then a strict reading of the pronoun results. If A is determined after the value of S is identified, then it can be taken as $\{w \in B.{}^{\mathsf{true}}w \hat{=}_B\ bill\}$, which provides the sloppy reading.

Finally, consider the antecedent contained ellipsis (ACE) structure in (58).

(58) Mary read every book that John did.

Restrictive relative clauses modify head nouns of NPs. Therefore, it is reasonable to impose the condition that the conjunct corresponding to a restrictive relative clause in the propositional part of the sub-type expression of a GQ contain an occurrence of the variable bound by the set operator of the sub-type. This is, in effect, a non-vacuousness constraint on relative clause modification. It requires that a relative clause be interpreted as a modifier that contributes a restriction to the head noun.

Given this constraint the representation of (58) is (59).

(59) $|\{x \in B.{}^{\mathsf{true}}book'(x) \wedge$
$(john, x) \in S) \wedge$
$^{\mathsf{true}}read'(mary, x)\}|_B$
$\cong_{\mathsf{Num}}$
$|\{x \in B.{}^{\mathsf{true}}book'(x) \wedge$
$(john, x) \in S\}|_B$

Taking the conjunct of (59) that corresponds to the matrix clause as the antecedent and abstracting over both its arguments we obtain the separation type specified in (60). This yields (61) as the interpretation of (58).[7]

(60) $S = \{(y, w).{}^{\mathsf{true}}read'(y, w)\}$

(61) $|\{x \in B.{}^{\mathsf{true}}book'(x) \wedge {}^{\mathsf{true}}read'(john, x)$
$\wedge\ {}^{\mathsf{true}}read'(mary, x)\}|_B$
$\cong_{\mathsf{Num}}$
$|\{x \in B.{}^{\mathsf{true}}book'(x) \wedge {}^{\mathsf{true}}read'(john, x)\}|_B$

[7]For simplicity we suppress typing on the variables y and w here.

(61) asserts that every book that John read Mary read, which is the intended reading of (58).

We have generated this interpretation without using a syntactic operation of quantifier raising, as in the analysis of Fiengo and May (1994) or a semantic procedure of storage, as in the HOU treatment of Dalrymple et al. (1991). We also do not require a syntactic trace (Lappin, 1996) or a SLASH feature (Lappin, 1999) in the ellipsis site. The presence of the variable bound by the set operator of the sub-type as the second argument of the function which assigns a value to the elided PTCT expression is motivated by a general condition on the representation of restrictive relative clauses as non-vacuous conjuncts in a GQ.

7 Comparison with Other Type-Theoretical Approaches

Ranta develops an analysis of anaphora within Martin-Löf Type Theory (MLTT) (Ranta, 1994). He represents donkey sentences as universal quantification over product types.[8]

(62) $\Pi z : ((\Sigma x : man)(\Sigma y : donkey)(x\ owns\ y))$
$(p(z)\ beats\ p(q(z))$

In this example, z is a variable over product pairs, and p and q are left and right projections, respectively, on the product pair, where

(63) (a) $p(z) : man$
(b) $q(z) : (\Sigma y : donkey)(p(z)\ owns\ y)$
(c) $p(q(z)) : donkey$
(d) $q(q(z)) : (p(z)\ owns\ p(q(z)))$.

Ranta's account does not generate the existential reading of donkey sentences (Pelletier & Schubert, 1989).

As Ranta acknowledges, his universal quantification-over-pairs analysis follows DRT (Kamp & Reyle, 1993) in inheriting the proportionality problem in a sentence like the following (Heim, 1990; Kadmon, 1990).

(64) Most men who own a donkey beat it.

On the universal quantification-over-pairs account of donkey anaphora, contrary to the desired interpretation, the sentence is true in a model in which ten men own donkeys, nine men own a single donkey each and do not beat it, while the tenth man owns ten donkeys and beats them all. On the preferred reading the sentence is false in this model.

Ranta cites Sundholm (1989)'s solution to the proportionality problem. This suggestion involves positing a second quantifier *most* on product pairs $(\Sigma x : A)(B(x))$ that is interpreted as applying an A injection to the pairs, where this injection identifies only the first element of each pair as within the domain of quantification. Defining an additional mode of quantification as a distinct reading of *most* in order to generate the correct interpretation of (64) would seem to be an ad hoc approach to the difficulty. There is no evidence for taking *most* as ambiguous between two quantificational readings beyond the need to avoid the inability of the quantification-over-pairs analysis to yield the correct results for this case. Assuming two modes of quantification adds considerable complication to the type-theoretic approach to anaphora without independent motivation.

The proportion problem does not arise on our account. *Most* is represented as a cardinality relation (GQ) in which quantification is over the elements of the set corresponding to the subject restriction rather than over pairs. Therefore the sentence is evaluated as false in the model that creates difficulties for DRT and for Ranta, without the need to adopt additional devices.

On the HOU approach to ellipsis proposed in Dalrymple et al. (1991) and Shieber et al. (1996) the elided predicate is represented as a higher-order variable, which is unified with a lambda term obtained from the antecedent clause through abstraction over the arguments that correspond to the bare arguments of the ellipsis site. This term is then applied to the bare arguments to produce an interpretation of the elided structure.

Our PTCT-based analysis of ellipsis is similar in approach to the HOU view. In both cases correspondences are set up between a sequence of phrases in an ellipsis site and an antecedent clause, and a predicate term is abstracted from the antecedent for application to elements in the elided clause. However, while HOU solves an equation with a higher-order variable to obtain a lambda expression, our account uses a parameter that is resolved to a separation type expression. In our model theory, type variables take terms as

[8]Note that here expressions of the form $\Pi x : (T)(S)$ denote a dependent product type. This is not to be confused with PTCT's polymorphic types, whose form $(\Pi X.T)$ is superficially similar.

values. Therefore, even if a separation type parameter is construed as a variable of PTCT, we remain within the first-order resources of PTCT.

In addition, HOU requires storage to extract a quantified NP from its antecedent-contained position in the semantic representation of an ACE structure. By contrast, we are able to interpret these NPs *in situ* by virtue of the presence of a bound variable in the part of a sub-type in a GQ representation that corresponds to the relative clause of the ACE.

8 Conclusions and Future Work

We have developed type-theoretic treatments of pronominal anaphora and ellipsis within the framework of PTCT, a first-order fine-grained intensional logic with flexible Curry typing. Our account of anaphora has wider empirical coverage than Ranta's (1994) MLTT analysis. Our account of ellipsis avoids the higher-order variables of HOU, and we do not require an operation of storage to handle ACE structures.

The application of PTCT to anaphora and ellipsis illustrates its considerable expressive resources. The primary advantage of PTCT is the fact that it provides the expressiveness of a higher-order system with rich typing while remaining a first-order logic with limited formal power.

We have provided a model theory for PTCT using extensional models for the untyped λ-calculus enriched with interpretations of Curry types. The restrictions that we impose on comprehension types, quantification over types, and the relation between the three sublanguages of PTCT ensure that it remains a first-order system in which its enriched expressive power comes largely through quantification over terms and the representation of types as terms within the language.

In future work we will be investigating the possibility of incorporating product types into PTCT without taking it out of the class of first-order systems and also determine the most appropriate way of incorporating the representation of free-floating type judgements in the language of terms. Product types will permit us to simplify our representations of k-ary predicates and the sub-types defined in terms of them. They will also permit us to capture dependent types, which we require to deal with certain kinds of anaphora, such as donkey pronouns in conditional sentences.

We will consider a property-theoretic variant of the treatment of anaphora and ellipsis which exploits properties and application rather than the types and type-membership used in the type-theoretic treatment presented in this paper. We will then examine extensions that establish correspondences between types and properties. One aim of this would be to show an equivalence between the type-theoretic and property-theoretic approaches to anaphora and ellipsis.

We will explore PTCT as a semantic representation language in implemented systems of natural language interpretation. As part of this research we will be constructing a theorem prover that uses PTCT's tableau proof theory. We will also investigate the implementation of our proposed approaches to anaphora and ellipsis resolution.

References

Barendregt, H. (1984). *The lambda calculus: Its syntax and semantics* (Vol. 103, Second ed.). Amsterdam: North Holland.

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, *5*, 56–68.

Dalrymple, M., Shieber, S., & Pereira, F. (1991). Ellipsis and higher-order unification. *Linguistics and Philosophy*, *14*, 399–452.

Fiengo, R., & May, R. (1994). *Indices and identity.* Cambridge, MA: MIT.

Fitting, M. (1996). *First-order logic and automated theorem proving.* Berlin: Springer.

Fox, C., & Lappin, S. (2003a). Doing natural language semantics in an expressive first-order logic with flexible typing. In G. P. G. Jaeger, P. Monachesi & S. Wintner (Eds.), *Proceedings of formal grammar 2003* (pp. 89–102). Vienna: Technical University of Vienna.

Fox, C., & Lappin, S. (2003b). *An expressive first-order logic with flexible typing for natural language semantics.* University of Essex and King's College, London: unpublished ms.

Fox, C., Lappin, S., & Pollard, C. (2002a). First-order curry-typed logic for natural language semantics. In S. Winter (Ed.), *Proceedings of the 7th international workshop on natural language understanding and logic programming* (pp. 175–192). Copenhagen: University of Copenhagen.

Fox, C., Lappin, S., & Pollard, C. (2002b). Intensional first-order logic with types. In G. Alberti, K. Balough, & P. Dekker (Eds.), *Proceedings of the seventh symposium for logic and language* (pp. 47–56). Pecs, Hungary: University of Pecs.

Fox, C., Lappin, S., & Pollard, C. (2002c). A higher-order fine-grained logic for intensional semantics. In G. Alberti, K. Balough, & P. Dekker (Eds.), *Proceedings of the seventh symposium for logic and language* (pp. 37–46). Pecs, Hungary: University of Pecs.

Heim, I. (1990). E-type pronouns and donkey anaphora,. *Linguistics and Philosophy*, *13*, 137–177.

Jeffrey, R. (1982). *Formal logic: Its scope and limits.* New York: McGraw-Hill.

Kadmon, N. (1990). Uniqueness. *Linguistics and Philosophy*, *13*, 237–324.

Kamp, H., & Reyle, U. (1993). *From discourse to logic.* Dordrecht: Kluwer.

Lappin, S. (1989). Donkey pronouns unbound. *Theoretical Linguistics*, *15*, 263–286.

Lappin, S. (1996). The interpretation of ellipsis. In S. Lappin (Ed.), *Handbook of contemporary semantic theory* (pp. 145–175). Oxford: Blackwell.

Lappin, S. (1999). An hpsg account of antecedent contained ellipsis. In S. Lappin & E. Benmamoun (Eds.), *Fragments: Studies in ellipsis and gapping* (pp. 68–97). New York: Oxford University Press.

Lappin, S., & Francez, N. (1994). E-type pronouns, i-sums, and donkey anaphora. *Linguistics and Philosophy*, *17*, 391–428.

Meyer, A. (1982). What is a model of the lambda calculus? *Information and Control*, *52*, 87–122.

Pelletier, J., & Schubert, L. (1989). Generically speaking. In G. Chiercihia, B. Partee, & R. Turner (Eds.), *Propteries, types, anmd meaning* (Vol. 2). Dordrecht: Kluwer.

Ranta, A. (1994). *Type theoretic grammar.* Oxford University Press.

Shieber, S., Pereira, F., & Dalrymple, M. (1996). Interactions of scope and ellipsis. *Linguistics and Philosophy*, *19*, 527–552.

Sundholm, G. (1989). Constructive generalised quantifiers. *Synthese*, *79*, 1–12.

Turner, R. (1997). Types. In J. van Benthem & A. ter Meulen (Eds.), *Handbook of logic and language* (pp. 535–586). Elsevier.

van Benthem, J. (1991). *Language in action.* Amsterdam: Amsterdam.