Chapter 7

Doing Natural Language Semantics in an Expressive First-Order Logic with Flexible Typing

CHRIS FOX AND SHALOM LAPPIN

Dept. of Computer Science, University of Essex
and
Dept. of Computer Science, King's College London
E-mail: foxcj@essex.ac.uk and lappin@dcs.kcl.ac.uk

ABSTRACT. We present Property Theory with Curry Typing (PTCT), an intensional first-order logic for natural language semantics. PTCT permits fine-grained specifications of meaning. It also supports polymorphic types and separation types.[1] We develop an intensional number theory within PTCT in order to represent proportional generalized quantifiers like *most*. We use the type system and our treatment of generalized quantifiers in natural language to construct a type-theoretic approach to pronominal anaphora that avoids some of the difficulties that undermine previous type-theoretic analyses of this phenomenon.

7.1 PTCT: Syntax of the basic theory

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) $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$
2. Types $T ::= B \mid \mathsf{Prop} \mid T \Rightarrow S$
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) $\alpha ::= t \in T \mid \bot \mid t =_T s \mid t \cong_T s$

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

[1] Separation types are also known as *sub-types*.

Proceedings of Formal Grammar 2003
G. Jäger, P. Monachesi, G. Penn & S. Wintner (editors).

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.[2]

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 ($^{\mathsf{true}}(t)$).

Later, in section 7.4, we will consider some extensions to the theory.

7.2 A Proof Theory for PTCT

We construct a tableau proof theory for PTCT. 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 ($=_T$): These are the usual α, β, η rules of the untyped λ-calculus, applied to terms of the same type.
- 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.

The following are examples of tableau rules of each kind.

The symbol * indicates that the corresponding proposition has been used, and does not need to be considered again. When a rule requires more than one premise, the premises are separated by commas (,).

4. Rules for Wffs

Conjunction

$$(s \wedge t)^*$$
$$\Big|$$
$$s$$
$$t$$

Negated Conjunction

$$\sim(s \wedge t)^*$$
$$\sim s \qquad \sim t$$

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

Universal Quantification

$\forall x.\varphi$ — *where k occurs on the path (or is the only constant occurring in the path)*
$\varphi[k/x]$

5. Type Inference Rules

General Function Spaces

$t \in (S \Rightarrow T)$ — *where $t' \in S$ is on the path*
$tt' \in T$

Identity within a Type

$t =_T t', \sim(t' \in T)$
$\bot$

Conjunctive Propositions

$t \in \mathsf{Prop}, t' \in \mathsf{Prop}$
$(t \hat{\wedge} t') \in \mathsf{Prop}$

Universal Propositions

$(\lambda x.t) \in (S \Rightarrow \mathsf{Prop})$
$\hat{\forall} x \varepsilon S.t \in \mathsf{Prop}$

6. Truth Rules

Conjunction

$^{\mathsf{true}}(s \hat{\wedge} t)^*$ — *where $s, t \in \mathsf{Prop}$ is on the path*
$^{\mathsf{true}}s \wedge {}^{\mathsf{true}}t$

Universal Quantification

$^{\mathsf{true}}(\hat{\forall} x \epsilon S.t)^*$ — *where $(\lambda x.t) \in (S \Rightarrow \mathsf{Prop})$ is on the path*
$\forall x(x \in S \rightarrow {}^{\mathsf{true}}t)$

7.3 Intensional Identity v. Extensional Equivalence

There are two equivalence 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 (Carnap, 1947; Montague, 1974), without invoking impossible worlds (Muskens, 1995; Barwise, 1997; Gregory, 2002). 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 et al.'s modification of Church's simple theory of types (Fox et al., 2002; Church, 1940)).

7.4 Extending the Type System

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

7. UT: $x \in \Delta \leftrightarrow x = x$

This allows Chierchia's analysis of nominalisation (Chierchia, 1982). Unfortunately this is inconsistent in PTCT if Prop is a type. Consider rr, where $r = \lambda x.\hat{\exists} y \in (\Delta \Rightarrow \mathsf{Prop})[x \mathrel{\hat{=}} y \mathrel{\hat{\wedge}} \hat{\sim} xy]$. However, there are other consistent extensions that can be adopted.

7.4.1 Separation Types

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

8. 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.

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

Wffs 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':

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

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

11. (a) $\lceil a \wedge b\rceil = \lceil a\rceil \mathrel{\hat{\wedge}} \lceil b\rceil$
    (b) $\lceil a \vee b\rceil = \lceil a\rceil \mathrel{\hat{\vee}} \lceil b\rceil$
    (c) $\lceil a \rightarrow b\rceil = \lceil a\rceil \mathrel{\hat{\rightarrow}} \lceil b\rceil$
    (d) $\lceil a \leftrightarrow b\rceil = \lceil a\rceil \mathrel{\hat{\leftrightarrow}} \lceil b\rceil$
    (e) $\lceil a \cong_T b\rceil = \lceil a\rceil \mathrel{\hat{\cong}_T} \lceil b\rceil$
    (f) $\lceil a =_T b\rceil = \lceil a\rceil \mathrel{\hat{=}_T} \lceil b\rceil$
    (g) $\lceil\bot\rceil = \hat{\bot}$
    (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.

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

7.4.2 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.

13. 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 $r = \lambda x.\hat{\exists} y \in (\Delta \Rightarrow \mathsf{Prop})[x \hat{=} y \hat{\wedge} \hat{\sim} xy]$ 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 $s = t$.

7.4.3 Polymorphic Types

We enrich the language of types to include type variables X, and the wffs to include quantification over types $\forall X\varphi, \exists X\varphi$. In the model theory that we present in Section 7.5, types denote terms in the domain of the model, so quantification over types remains first-order. We add $\Pi X.T$ to the language of types, governed by the tableau rule corresponding to the following axiom:

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

Polymorphic types permit us to accommodate the fact that natural language expressions like coordination and certain verbs can 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, and the paradoxes that impredicativity generates, we add a language of Kinds (K) to PTCT:

15. Kinds: $K ::= T \mid \Pi X.K$
16. 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 polymorphism seems to be adequate to express the instances of multiple type assignment that occur in natural languages (van Benthem, 1991).

7.4.4 Final Syntax

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

17. (logical constants) $l \ ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$
(terms) $t \ ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$
(Types) $T \ ::= B \mid \mathsf{Prop} \mid T \Rightarrow S \mid X \mid \{x \in T.\varphi'\} \mid \{x.\varphi'\}$
(Kinds) $K \ ::= T \mid \Pi X.T$
(atomic wff) $\alpha \ ::= t \in K \mid \bot \mid t =_T s \mid t \cong_T s$
(wff) $\varphi \ ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi)$
$\mid (\forall x\varphi) \mid (\exists x\varphi) \mid (\forall X\varphi) \mid (\exists X\varphi) \mid {}^{\mathsf{true}}t$

where φ' is as defined in section 7.4.1.

7.5 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).

7.5.1 Models of the (Extensional) λ-Calculus

18. **Definition** (General Functional Models) A functional model is a structure of the form $\mathcal{D} = \langle D, [D \rightarrow D], \Phi, \Psi \rangle$ where
    (a) D is a non-empty set,
    (b) $[D \rightarrow D]$ is some class of functions from D to D,
    (c) $\Phi : D \rightarrow [D \rightarrow D]$,
    (d) $\Psi : [D \rightarrow D] \rightarrow D$,
    (e) $\Psi(\Phi(d)) = d$ for all $d \in D$

We can interpret the calculus as using the following.

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

where g is an assignment function from variables to elements of D. 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

20. $[\![c]\!]_g = i(c)$

where i assigns elements of D to constants.

21. **Theorem** 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. □

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 sublanguages of PTCT.

22. **Definition** A model of PTCT is $\mathcal{M} = \langle \mathcal{D}, \mathsf{T}, \mathsf{P}, \mathsf{B}, \mathcal{B}, \mathcal{T}, \mathcal{K} \rangle$, where
    (a) $\mathcal{D}$ is a model of the λ-calculus
    (b) $\mathsf{T} : D \rightarrow \{0, 1\}$ models the truth predicate $^{\mathsf{true}}$
    (c) $\mathsf{P} \subset D$ models the class of propositions
    (d) $\mathsf{B} \subset D$ models the class of basic individuals
    (e) $\mathcal{T} \subset \mathcal{K}$ models the class of types
    (f) $\mathcal{B}(\mathsf{B})$ is a set of sets, whose elements partition B into equivalences classes of individuals.
    (g) $\mathcal{K} \subset D$ models the class of kinds

    with sufficient structural constraints on T, P and $\mathcal{T}$ to validate the rules of PTCT.

In the following, we give some examples of the structural constraints.

7.5.2 Kinds and Types

Kinds and types can be interpreted as subsets of D. To ensure that polymorphic types are predicative (i.e. to avoid an implicit circularity in their definition), there is quantification over types, but not over kinds.

To give a first-order account of type quantification, we will consider the interpretation of term *representations* of types. We take types in PTCT to denote terms in $\mathcal{T} \subset D$. If type T in PTCT denotes an individual $S \in \mathcal{T}$, the underlying type (or set of individuals) will be denoted by $^{\cup} S$.

The types in the model, and the interpretation of PTCT's types are specified by rules like those in the following examples.

23. (a) For all $t \in \mathcal{K}$, $^{\cup} t \subset D$
    (b) $[\![X]\!]_{g,\tau} = \tau(X) \in \mathcal{T}$
    (c) If $[\![S]\!]_{g,\tau}, [\![U]\!]_{g,\tau} \in \mathcal{T}$, then $[\![S \Rightarrow U]\!]_{g,\tau} \in \mathcal{T}$
    (d) If $[\![S]\!]_{g,\tau} \in \mathcal{T}$ and $[\![\lceil \varphi' \rceil]\!]_{g,\tau} \in \mathsf{P}$ then $[\![\{x \in S.\varphi'\}]\!]_{g,\tau} \in \mathcal{T}$
    (e) $^{\cup}[\![S \Rightarrow U]\!]_{g,\tau} = \{d \in D : \forall e \in {}^{\cup}[\![S]\!]_{g,\tau}.\Phi(d)e \in {}^{\cup}[\![U]\!]_{g,\tau}\}$
    (f) $^{\cup}[\![\{x \in S.\varphi'\}]\!]_{g,\tau} = \{d \in {}^{\cup}[\![S]\!]_{g,\tau}.\mathcal{M}_{g[d/x],\tau} \models \varphi'\}$

Here, τ is an assignment from type variables to elements of $\mathcal{T}$, and $\lceil \cdot \rceil$ is as defined in section 7.4.1, and Φ is as defined in the model of the λ-calculus.

7.5.3 Propositions

The typing rules for Prop are supported by structural constraints that are exemplified by the following.

24. (a) If $[\![t]\!]_{g,\tau} \in \mathsf{P}$ and $[\![s]\!]_{g,\tau} \in \mathsf{P}$, then $[\![t \hat{\wedge} s]\!]_{g,\tau} \in \mathsf{P}$.
    (b) If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, and $[\![t]\!]_{g[d/x],\tau} \in \mathsf{P}$ for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$, then $[\![\hat{\forall} x \epsilon S.t]\!]_{g,\tau} \in \mathsf{P}$.
    (c) If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, then $[\![t \hat{\cong}_S s]\!]_{g,\tau} \in \mathsf{P}$ iff $[\![s]\!]_{g,\tau}, [\![t]\!]_{g,\tau} \in {}^{\cup}[\![S]\!]_{g,\tau}$.
    (d) If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, then $[\![t \hat{=}_S s]\!]_{g,\tau} \in \mathsf{P}$ iff $[\![s]\!]_{g,\tau}, [\![t]\!]_{g,\tau} \in {}^{\cup}[\![S]\!]_{g,\tau}$.

7.5.4 Truth

The rules for $^{\mathsf{true}}$ are supported by the following structural constraints, amongst others.

25. (a) If $[\![t]\!]_{g,\tau} \in \mathsf{P}$ and $[\![s]\!]_{g,\tau} \in \mathsf{P}$, then $\mathsf{T}([\![t \hat{\wedge} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![t]\!]_{g,\tau}) = 1$ and $\mathsf{T}([\![s]\!]_{g,\tau}) = 1$.
    (b) If $[\![S]\!]_{g,\tau} \in \mathcal{T}$ and $[\![t]\!]_{g[d/x],\tau} \in \mathsf{P}$ for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$, then $\mathsf{T}([\![\hat{\forall} x \epsilon S.t]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![t]\!]_{g[d/x],\tau}) = 1$ for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$.
    (c) i. If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in \mathsf{B}$ then $\mathsf{T}([\![t \hat{\cong}_{\mathsf{B}} s]\!]_{g,\tau}) = 1$ iff there is an A such that $A \in \mathcal{B}(\mathsf{B})$ and $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau}$ are elements of A.
    ii. If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in \mathsf{P}$ then $\mathsf{T}([\![t \hat{\cong}_{\mathsf{Prop}} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![t]\!]_{g,\tau}) = \mathsf{T}([\![s]\!]_{g,\tau})$.
    iii. If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in [\![S \Rightarrow U]\!]_{g,\tau}$, where $[\![S]\!]_{g,\tau}, [\![U]\!]_{g,\tau} \in \mathcal{T}$ then $\mathsf{T}([\![t \hat{\cong}_{(S \Rightarrow U)} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![tx \hat{\cong}_U sx]\!]_{g[d/x],\tau}) = 1$ for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$.
    (d) If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in {}^{\cup}[\![S]\!]_{g,\tau}$, then $\mathsf{T}([\![t \hat{=}_S s]\!]_{g,\tau}) = 1$ iff $[\![t]\!]_{g,\tau} = [\![s]\!]_{g,\tau}$
    (e) If $\mathsf{T}([\![t]\!]_{g,\tau}) = 1$ then $[\![t]\!]_{g,\tau} \in \mathsf{P}$.

Note that we give no rules for the extensional equivalence of elements of comprehension types, separation types or polymorphic types.

7.5.5 Well-Formed Formulae

The language of wff can now be given truth conditions. Some examples follow.

26. $\mathcal{M}_{g,\tau} \models s =_T t$ iff $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in {}^{\cup}[\![T]\!]_{g,\tau}$ and $[\![t]\!]_{g,\tau} = [\![s]\!]_{g,\tau}$
    $\mathcal{M}_{g,\tau} \models s \cong_T t$ iff $\mathsf{T}([\![s]\!]_{g,\tau} \hat{\cong}_T [\![t]\!]_{g,\tau}) = 1$
    $\mathcal{M}_{g,\tau} \models {}^{\mathsf{true}}(t)$ iff $\mathsf{T}([\![t]\!]_{g,\tau}) = 1$
    $\mathcal{M}_{g,\tau} \models t \in T$ iff $[\![t]\!]_{g,\tau} \in {}^{\cup}[\![T]\!]_{g,\tau}$[3]
    $\mathcal{M}_{g,\tau} \models \varphi \wedge \psi$ iff $\mathcal{M}_{g,\tau} \models \varphi$ and $\mathcal{M}_{g,\tau} \models \psi$
    $\mathcal{M}_{g,\tau} \not\models \bot$
    $\mathcal{M}_{g,\tau} \models \forall x \varphi$ iff for all $d \in D$, $\mathcal{M}_{g[d/x],\tau} \models \varphi$
    $\mathcal{M}_{g,\tau} \models \forall X \varphi$ iff for all $S \in \mathcal{T}$, $\mathcal{M}_{g,\tau[S/X]} \models \varphi$

27. **Definition** (Validity) A wff φ of PTCT is *valid* in a model $\mathcal{M}$ iff $\mathcal{M}_{g,\tau} \models \varphi$ for all assignment functions g, τ.

[3] Syntactic constraints on T guarantee that $[\![T]\!]_{g,\tau} \in \mathcal{T}$.

28. **Theorem** (Soundness of PTCT) If $\mathsf{PTCT} \vdash \phi$, then ϕ is valid.
Proof: By induction on the downward correctness of the tableau rules of PTCT. □
29. **Theorem** (Completeness of PTCT) If ϕ is valid, then $\mathsf{PTCT} \vdash \phi$.
Proof: By induction on the upward correctness of the tableau rules of PTCT. □

7.6 An Intensional Number Theory

We add an intensional number theory to PTCT, incorporating the axioms as tableau rules.

30. Terms: we add 0, $succ$, $pred$, add, $mult$, $\hat{most}$ and $|\cdot|_B$
31. Types: we add Num
32. Wffs: we add $zero(t)$, $t \cong_{\mathsf{Num}} t'$, $t <_{\mathsf{Num}} t'$ and $most(p)(q)$
33. Axioms for Num: The usual Peano axioms, adapted to PTCT
34. 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. Of course, when the number theory is added to PTCT, the extended theory is incomplete.

7.7 Representing Proportional Generalized Quantifiers in PTCT

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

35. Cardinality of properties $|p|_B$:
    (a) $p \in (B \Rightarrow \mathsf{Prop}) \wedge \sim\exists x(x \in B \wedge {}^{\mathsf{true}}px) \rightarrow |p|_B \cong_{\mathsf{Num}} 0$
    (b) $p \in (B \Rightarrow \mathsf{Prop}) \wedge b \in B \wedge {}^{\mathsf{true}}pb \rightarrow$
    $|p|_B \cong_{\mathsf{Num}} add(|p - \{x.x \hat{=}_B b\}|_B)(succ(0))$

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

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

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 generalized quantifier within a first-order system.

7.8 A Type-Theoretical Approach to Anaphora

To illustrate the expressiveness of PTCT 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 pronoun 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.

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

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 dependent type obtained from the predicative part of the antecedent clause representation. In the default case, the pronoun variable is bound by a universal quantifier in the language of wffs.

39. Every student arrived.
40. $|\{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\}|_B \cong_{Num}$
    $|\{x \in B.^{\mathsf{true}} student'(x)\}|_B$
41. They sang.
42. $\forall y \in B.(^{\mathsf{true}} sang'(y))(y \in \{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\})$

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

43. John arrived.
44. $^{\mathsf{true}} arrived'(john)$
45. He sang.
46. $\forall y \in B.(^{\mathsf{true}} sang'(y))$
    $(y \in \{x \in B.x = john\})$
47. Some man arrived.
48. $|\{x \in B.^{\mathsf{true}} man'(x) \wedge {}^{\mathsf{true}} arrived'(x) \wedge {}^{\mathsf{true}} \phi(x)\}|_B >_{Num} 0$
49. he sang.

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

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 pairs of variables.

51. Every man who owns a donkey beats it.
52. $|\{x \in B.^{\mathsf{true}}man'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0) \wedge$
$\forall z(^{\mathsf{true}}beat'(x,z))\}|_B \cong_{Num}$
$|\{x \in B.^{\mathsf{true}}man'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0)\}|_B$
where $((x,z) \in \{(y,w) \in B \otimes B.^{\mathsf{true}}own'(y,w) \wedge {}^{\mathsf{true}}donkey'(w)\})$

We have not introduced product types into PTCT. The type constraint on the pair (x,z) is in fact a convenient notational device for representing a curried type constraint which applies in sequence to x and z.

53. $(x \in \{y \in B.\{z \in \{w \in B.^{\mathsf{true}}own'(y,w) \wedge {}^{\mathsf{true}}donkey'(w)\}\})$

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

Ranta (1994) develops an analysis of anaphora within Martin-Löf Type Theory (MLTT). He represents donkey sentences as universal quantification over product types.[4]

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

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

55. (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 and Schubert, 1989).

We can generate these readings 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 a universal quantifier.

56. Every person who had a quarter put it in a parking meter.

[4]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.

57. $|\{x \in B.^{\mathsf{true}} man'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}|_B >_{Num} 0) \wedge$
$\exists z({}^{\mathsf{true}} put\text{-}in\text{-}a\text{-}parking\text{-}meter'(x,z))\}|_B \cong_{Num}$
$|\{x \in B.^{\mathsf{true}} person'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}|_B >_{Num} 0)\}|_B$
where $((x,z) \in \{(y,w) \in B \otimes B.^{\mathsf{true}} had'(y,w) \wedge {}^{\mathsf{true}} quarter'(w)\})$

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

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

58. Most men who own a donkey beat it.

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.

This problem does not arise on our account. *Most* is represented as a cardinality relation (generalized quantifier) in which quantification is over the elements of the set corresponding to the subject restriction rather than over pairs. Therefore the sentence is false in this model.

7.9 Relevance of PTCT to Computational Semantics

We believe that the features of PTCT make it particularly appropriate for implementing theorem proving systems for natural language semantics. It adds appropriate "expressiveness" to a first-order theory, without an undesirable increase in formal power. Historically, higher-order systems have dominated the field in natural language semantics. In general, the set of theorems of such systems are not r.e. For this reason, a first-order system is preferable, if it is sufficiently expressive for the relevant domain. Basic first-order logic by itself does not provide the features that are required for natural language semantics. It is insufficiently expressive for this purpose. PTCT is a first-order system whose expressiveness is designed for natural language semantics. It directly supports fine-grained intensionality and a flexible system of types. We have formulated tableau rules for PTCT, which provide an effective theorem proving procedure (for the logic without the number theory).

7.10 Conclusion

We have constructed a first-order fine-grained intensional logic with flexible Curry typing, PTCT, for the semantic representation of natural languages. PTCT contains typed predicates for intensional identity and extensional equality. Its proof

theory permits us to prove that identity of intension entails identity of extension, but that the converse does not hold.

The theory can be distinguished from Aczel's Frege Structures (Aczel, 1980) and related, weakly typed theories of properties (PT) (Turner, 1988) in two ways. First, there is an explicit notion of polymorphic type within the theory, which is perhaps more appropriate for natural language semantics than the universal type of PT. Second, the type Prop can appear in intensional representations of propositions. This allows us to express the fact that, for example, the universal quantification in statements of the form *John believes everything that Mary believes* ranges only over propositions. In PT, this requirement can only be expressed as an external constraint (Turner, 1997).

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 insure 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.

Unlike alternative hyperintensionalist frameworks that have been proposed, this logic distinguishes among provably equivalent propositions without resorting to impossible worlds to sustain the distinction. The incorporation of Curry typing into the logic allows us to sustain weak polymorphism. Sub-types also permits a us to develop a uniform type-theoretical account of pronominal anaphora with wide empirical coverage.

Bibliography

Aczel, P. (1980). Frege structures and the notions of proposition, truth and set. In Barwise, Keisler, and Keenan, eds., *The Kleene Symposium*, North Holland Studies in Logic, pp. 31–39. North Holland.

Barendregt, H. (1984). *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundation of Mathematics*. North Holland, Amsterdam, second edition.

Barwise, J. (1997). Information and impossibilities. *The Notre Dame Journal of Formal Logic*, **38**:488–515.

Carnap, R. (1947). *Meaning and Necessity*. University of Chicago Press, Chicago.

Chierchia, G. (1982). Nominalisation and Montague grammar: a semantics without types for natural languages. *Linguistics and Philosophy*, **5**:303–354.

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

Fox, C., S. Lappin, and C. Pollard (2002). A higher-order, fine-grained logic for intensional semantics. In G. Alberti, K. Balough, and P. Dekker, eds., *Proceedings of the Seventh Symposium for Logic and Language*, pp. 37–46. Pecs, Hungary.

Gregory, H. (2002). Relevance logic and natural language semantics. In *Proceedings of Formal Grammar 2002*. Trento, Italy.

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

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

Kamp, H. and U. Reyle (1993). *From Discourse to Logic*. Kluwer, Dordrecht.

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

Lappin, S. and N. Francez (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.

Montague, R. (1974). *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven/London. Edited with an introduction by R.H. Thomason.

Muskens, R. (1995). *Meaning and Partiality*. CSLI and FOLLI, Stanford, CA.

Pelletier, J. and L. Schubert (1989). Generically speaking. In G. Chiercihia, B. Partee, and R. Turner, eds., *Propteries, Types, anmd Meaning*, volume 2. Kluwer, Dordrecht.

Ranta, A. (1994). *Type Theoretic Grammar*. Oxford University Press.

Turner, R. (1988). Properties, propositions, and semantic theory. In *Proceedings of Formal Semantics and Computational Linguistics*. Switzerland.

Turner, R. (1997). Types. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, pp. 535–586. Elsevier.

van Benthem, J. (1991). *Language in Action*. Studies in Logic. North-Holland.