First-Order, Curry-Typed Logic for Natural Language Semantics*

Chris Fox[1], Shalom Lappin[2]**, and Carl Pollard[3]

[1] Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, U.K.
foxcj@essex.ac.uk
[2] Department of Computer Science, King's College London
The Strand, London WC2R 2LS, U.K.
lappin@dcs.kcl.ac.uk
[3] Department of Linguistics, Ohio State University
222 Oxley Hall, Columbus, OH 43210, U.S.A.
pollard@ling.ohio-state.edu

Abstract. The paper presents Property Theory with Curry Typing (PTCT) where the language of terms and well-formed formulæ are joined by a language of types. In addition to supporting fine-grained intensionality, the basic theory is essentially first-order, so that implementations using the theory can apply standard first-order theorem proving techniques. The paper sketches a system of tableau rules that implement the theory. Some extensions to the type theory are discussed, including type polymorphism, which provides a useful analysis of conjunctive terms. Such terms can be given a single polymorphic type that expresses the fact that they can conjoin phrases of any one type, yielding an expression of the same type.

1 Introduction

Since Carnap [4], higher-order theories that characterise intensions as functions from possible worlds to extensions, as in Montague [20] have dominated formal semantics. Such theories yield a semantics which is not sufficiently fine grained—logically equivalent expressions are co-intensional and so intersubstitutable in all contexts, including the complements of propositional attitude predicates—and the higher-order nature of such theories does not aid implementation in automatic theorem provers.

An alternative view, which we refer to as *hyperintensionalism*, posits propositions as independent intensional entities, and takes truth to be a derived property. In the past twenty years a variety of hyperintensionalist theories have been proposed, including Thomason [25], situation semantics [3, 2, 24], Landman [18], property theory [5, 26, 27], Muskens [21], and Lappin and Pollard [19]. With

* Originally presented at NLULP02. This version contains minor technical corrections.

** The second author's research is funded by grant number AN2687/APN 9387 from the Arts and Humanities Research Board of the United Kingdom.

the exception of Turner's property theory [27], which axiomatises Aczel's Frege Structures [1], these theories have focused on developing a model theory in which logical equivalence does not entail synonymy.

Property Theory takes propositions and properties as primitive entities whose identity criteria are not dependent upon truth conditions. Whilst Turner's theory appears adequate to cover Montague's fragment of natural language, and extensions have been proposed to cover other phenomena [11, 12], one shortcoming is that the theory has no explicit notion of type. This restricts the way in which certain propositions can be expressed within the theory [28], and makes it hard to formulate correspondence results with more conventional typed intensional logics.

2 Typed Theories

Typed logics, based upon Church style typing [6] typically have a basic set of (intensional) entities e, propositions P, and general functional types $T \Longrightarrow T'$. Such logics require that terms (or expressions) belong to exactly one type (monomorphism), and that there is no universal type. These requirements are imposed, in part, to avoid paradoxes in the underlying set-theoretic model.

The solution in strongly typed theories is to ban self application by banning polymorphism (membership of more than one type). More precisely: types are stratified/well founded, start with a base type and recursively define additional types as functions over the types that we have so far (this also leads to a ban on a universal type). All wff have a type, and abstraction and quantification have typed variables (typing may be implicit or explicit).

An additional justification for accepting a typed system is that natural language appears to have categories that correspond with types, although they are perhaps more flexible in character.

Another more radical approach is presented by Property Theory. Unlike the theories based upon conventional higher-order logic, Turner's Property Theory (PT) is based upon the untyped λ-calculus [7] extended with terms that are intended to represent logical constants. A first-order meta-theory then allows us to evaluate the truth conditions of terms that felicitously represent propositions.

It would be useful to show correspondence relations between PT and higher-order logics. This could aid the transfer of the analysis of semantic phenomena between theories, and would allow proposed semantic representations in the higher-order frameworks to be explored in a first-order system. However, the fact that higher-order logics are typed, and PT is essentially untyped poses a problem for this project. It is possible to mimic types with a sortal system [27], but correspondence results would then have to assume meaning postulates linking the behaviour of the types with the sorts of PT.

Another approach emerges if we observe that PT is a metatheory over the untyped λ-calculus; there are other metatheories over the untyped λ-calculus, such as (constructive) programming logics. These logics include an explicit language of types. Rather than incorporating types into the language of the λ-calculus,

as in [6], such theories take the terms of untyped λ-calculus as given, and then adopt rules and axioms to assign types to terms [8, 9]. This way of adding types to λ-calculus leads to a more expressive system [17]. The types will not lead to a higher-order system, provided that they do not include abstraction and universal quantification (over types).

The syntax of a basic version of such a theory might be defined as follows:

$$\begin{array}{ll}
\text{(terms)} & t ::= x \mid c \mid \lambda x(t) \mid (t)t \\
\text{(Types)} & T ::= B \mid T \Longrightarrow S \\
\text{(atomic wff)} & \alpha ::= (t = s) \mid \bot \mid t \in T \\
\text{(wff)} & \varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\forall x \varphi) \mid (\exists x \varphi)
\end{array}$$

where x stands for variables, c stands for constants, and B stands for basic types. Axioms and inference rules can then be given in terms of sequents of wffs.

To provide a first-order vehicle for natural language semantics, it is necessary for the language of terms of such a theory to be embellished with some logical constants, that will be used to "represent" propositions, properties and relations.

Turner's PT [27] can be thought of as a version of such a theory where there are only two types, corresponding with *True* and *Proposition*.[1] Here, a conceptually similar approach is adopted, except that (i) there is a more comprehensive system of types, (ii) *True* does not correspond to a type, and (iii) quantification represented within the language of terms is typed. This helps to maintain some correspondence with logics that adopt Church typing, and makes it easier to explore extensions to the type theory without leading directly to a paradox.

Enriching the language of terms in this way is similar to conventional higher-order logic, but rather than give rules and axioms to attribute the expected logical behaviour directly to the (higher-order) propositional terms, the rules and axioms are instead expressed in terms of the first-order language of wffs. The logic that results can be described as "Property Theory with Curry Typing" (PTCT).

3 The Language of PTCT

Adopting the above proposal leads to the following language:

$$\begin{array}{ll}
\text{(logical constants)} & l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=} \mid \hat{\cong}_T \mid \epsilon \\
\text{(terms)} & t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t \\
\text{(Types)} & T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S \\
\text{(atomic wff)} & \alpha ::= (t = s) \mid \bot \mid t \in T \mid t \cong_T s \\
\text{(wff)} & \varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi) \\
 & \qquad \mid (\forall x \varphi) \mid (\exists x \varphi) \mid {}^{\mathsf{true}}t
\end{array}$$

[1] In the original formulation of Turner's PT, there are only two languages, that of terms and that of wffs, as opposed to the three of PTCT (terms, types and wffs). *True* and *Proposition* are represented as predicates within the language of wff, rather than as types.

We have made the assertion of truth a wff, rather than a type assignment in order to avoid complications when considering extension to the theory. There are two notions of equality: $\hat{=}$ is intended to correspond to intensional identity, and $\hat{\cong}$ corresponds to extensional equivalence. This follows the proposal of Fox and Lappin [13].[2] Note that equivalence in PTCT does not entail intensional identity. Note that in this theory $\cong$ is relative to a type; in general it will be possible for a term to have more than one type, and we do not necessarily wish to maintain that two terms can only be extensionally equivalent if they are equivalent in all the types they have.[3] Identity itself, $=$, is independent of the typing of the terms (if two terms are identical, they will necessarily have the same types and be identical regardless of which types they have).

Not every term will have a type, but every term that has a type in this basic theory will correspond with an expression in conventional higher-order logic.

The resultant theory may be more powerful than is required just for natural language semantics, but it makes it convenient to show correspondences between conventional higher-order logic and a first-order language. The Curry typing is useful when we consider polymorphic variants. Once we have an explicit language of types independent of any set-theoretic interpretation, it is more convenient to explore more unusual types and type constructions.

3.1 Logic of Wff

We can give the language of wff the standard behaviour of first-order logic with the following natural-deduction style rules.

Introduction and Elimination of $\wedge$

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \wedge i \qquad \frac{\varphi \wedge \psi}{\varphi} \wedge e \qquad \frac{\varphi \wedge \psi}{\psi} \wedge e$$

Introduction and Elimination of $\rightarrow$

$$\frac{\begin{array}{c}[\varphi] \\ \vdots \\ \psi\end{array}}{\varphi \rightarrow \psi} \rightarrow i \qquad \frac{\varphi \quad \varphi \rightarrow \psi}{\psi} \rightarrow e$$

[2] Gilmore [16] constructs an intensional simple theory of types (ITT) in which an intensional ($=$) and an extensional ($=_e$) identity predicate are defined. His proposal differs from that of Fox and Lappin [13], and Fox, Lappin and Pollard and [14] in that (i) the extensional identity predicate is not type general, and is only defined for propositions and predicates, and (ii) for Fox, Lappin and Pollard, identity and equivalence are primitive, whereas Gilmore defines them in terms of substitution and bi-implication.

[3] It is always possible to consider a stronger version of the theory where $\cong$ is untyped.

Introduction and Elimination of $\vee$

$$\frac{\varphi}{\varphi \vee \psi}\ \vee i \qquad \frac{\psi}{\varphi \vee \psi}\ \vee i \qquad \frac{\varphi \vee \psi \quad \begin{array}{c}[\varphi]\\ \vdots\\ \eta\end{array} \quad \begin{array}{c}[\psi]\\ \vdots\\ \eta\end{array}}{\eta}\ \vee e$$

Absurdity

$$\frac{\bot}{\varphi}\ absurd$$

Introduction and Elimination of $\sim$ Negation can be defined as follows:

Negation $\sim\varphi =_{\mathsf{def}} \varphi \rightarrow \bot$

The usual axioms for constructive negation can then be stated.

$$\frac{\begin{array}{c}[\varphi]\\ \vdots\\ \bot\end{array}}{\sim\varphi}\ \sim i \qquad \frac{\varphi \quad \sim\varphi}{\bot}\ \sim e$$

As will be seen below, a trivial addition will yield a classical system.

Introduction and Elimination of $\forall$

$$\frac{\varphi}{\forall x\varphi}\ \forall i \qquad \frac{\forall x\varphi}{\varphi[t/x]}\ \forall e$$

Side condition on $\forall i$ x is not free in any undischarged assumptions in which φ holds.

Introduction and Elimination of $\exists$

$$\frac{\varphi[t/x]}{\exists x\varphi}\ \exists i \qquad \frac{\exists x\varphi \quad \begin{array}{c}[\varphi]\\ \vdots\\ \eta\end{array}}{\eta}\ \exists e$$

Side condition on $\exists i$ t must be free for x in φ.
Side condition on $\exists e$ x must not occur free in η or any assumptions other than φ on which the upper occurrence of η depends.

Classical Wff Adding any of the following will lead to a classical system:

$$\frac{\begin{array}{c}[\sim\varphi]\\ \vdots\\ \bot\end{array}}{\varphi}\ neg \qquad \frac{}{\sim\varphi \vee \varphi}\ lem \qquad \frac{\sim\sim\varphi}{\varphi}\ dn$$

4 Rules for Identity and Equivalence

Either the axioms:

ρ $x = x$
σ $x = y \to y = x$
τ $(x = y \wedge y = z) \to x = z$

or the rules:

$$\frac{}{x = x}\ \rho \quad \frac{x = y}{y = x}\ \sigma \quad \frac{x = y \quad y = z}{x = z}\ \tau$$

can be added to make $=$ into an equivalence relation.

Similar rules can be added for $\cong_T$, except that in this case there are additional typing restrictions:

$$\frac{x \in T}{x \cong_T x}\ \rho_{\cong} \quad \frac{x, y \in T \quad x \cong_T y}{y \cong_T x}\ \sigma_{\cong} \quad \frac{x, y, z \in T \quad x \cong_T y \quad y \cong_T z}{x \cong_T z}\ \tau_{\cong}$$

4.1 Rules for λ-terms

The λ-terms care governed by the conventional axioms:

α $\lambda x.t = \lambda y.t[y/x]$ Provided y is not free in t
β $(\lambda x.t)y = t[y/x]$
μ $x = y \to ux = uy$
ν $u = v \to ux = vx$
ξ $\forall x(t = s) \to \lambda y.t = \lambda y.s$ where x and y denote the same variable

The last three may also be given as the following rules:

$$\frac{s = s'}{ts = ts'}\ \mu \quad \frac{s = s'}{st = s't}\ \nu \quad \frac{t = s}{\lambda x.t = \lambda x.s}\ \xi$$

The system $\rho, \sigma, \tau, \alpha, \beta, \mu, \nu, \xi$ constitutes the intensional, untyped λ calculus (LC).

We can add either of the following axioms to give the extensional version of the untyped calculus:

Ext $\forall x(tx = sx) \to (t = s)$
η $\lambda x.(tx) = t$

Note that LC+Ext is equivalent to LC+η. Such extensionality of the the underlying λ-calculus would not undermine the intensionality of the system as these axioms govern the behaviour of intensional identity, and are independent of the extensional notion of equivalence in the language of wff.[4]

[4] We are grateful to Paul Gilmore for pointing this out to us.

4.2 Type Assignment Rules

Function type introduction and elimination can be stated as follows:

$$\frac{t \in (S \Longrightarrow T) \quad t' \in S}{tt' \in T} \Longrightarrow e \qquad \frac{\begin{array}{c}[x \in S] \\ \vdots \\ t \in T\end{array}}{\lambda x(t) \in (S \Longrightarrow T)} \Longrightarrow i$$

These rules are derivable from the axiom of general function spaces

GFS $f \in (S \Longrightarrow T) \leftrightarrow \forall x(x \in S \rightarrow fx \in T)$

Equivalence Within a Type We can also have the following:

$$\frac{t \in T \quad t' = t}{t' \in T} =_T$$

4.3 Prop Typing Rules

We need to be able to determine which terms represent propositions.

Absurdity

$$\frac{}{\hat{\bot} \in \mathsf{Prop}}$$

Binary Connectives

$$\frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop}}{t \mathbin{\hat{\wedge}} t' \in \mathsf{Prop}} \qquad \frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop}}{t \rightarrow t' \in \mathsf{Prop}} \qquad \frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop}}{t \mathbin{\hat{\vee}} t' \in \mathsf{Prop}}$$

Quantifiers

$$\frac{(\lambda x.t) \in (S \Longrightarrow \mathsf{Prop})}{(\hat{\forall} x \epsilon S.t) \in \mathsf{Prop}} \qquad \frac{(\lambda x.t) \in (S \Longrightarrow \mathsf{Prop})}{(\hat{\exists} x \epsilon S.t) \in \mathsf{Prop}}$$

Identity and Equivalence

$$\frac{t \in T \quad t' \in T}{t \mathbin{\hat{=}} t' \in \mathsf{Prop}} \qquad \frac{t \in T \quad t' \in T}{t \mathbin{\hat{\cong}}_T t' \in \mathsf{Prop}}$$

4.4 Rules for Truth $^{\mathsf{true}}$

The following rules give the truth conditions of those terms that represent propositions. The truth conditions themselves are stated as expressions in the language of wff.

Propositions are in Prop

$$\frac{^{\mathsf{true}}t}{t \in \mathsf{Prop}}$$

Absurdity

$$\frac{^{\mathsf{true}}\hat{\bot}}{\bot}$$

Binary Connectives

$$\frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad ^{\mathsf{true}}(t \mathbin{\hat{\wedge}} t')}{^{\mathsf{true}}t \wedge {}^{\mathsf{true}}t'} \qquad \frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad ^{\mathsf{true}}(t \mathbin{\hat{\to}} t')}{^{\mathsf{true}}t \to {}^{\mathsf{true}}t'}$$

$$\frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad ^{\mathsf{true}}(t \mathbin{\hat{\vee}} t')}{^{\mathsf{true}}t \vee {}^{\mathsf{true}}t'} \qquad \frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad ^{\mathsf{true}}(t \mathbin{\hat{\leftrightarrow}} t')}{^{\mathsf{true}}t \leftrightarrow {}^{\mathsf{true}}t'}$$

Quantifiers

$$\frac{\lambda x.t \in (S \Longrightarrow \mathsf{Prop}) \quad ^{\mathsf{true}}(\hat{\forall} x \epsilon S.t)}{\forall x(x \in S \to {}^{\mathsf{true}}(t))} \qquad \frac{\lambda x.t \in (S \Longrightarrow \mathsf{Prop}) \quad ^{\mathsf{true}}(\hat{\exists} x \epsilon S.t)}{\exists x(x \in S \wedge {}^{\mathsf{true}}(t))}$$

Identity

$$\frac{t \in T \quad t' \in T \quad ^{\mathsf{true}}(t \mathbin{\hat{=}} t')}{t = t'}$$

Equivalence

$$\frac{t \in T \quad t' \in T \quad ^{\mathsf{true}}(t \mathbin{\hat{\cong}}_T t')}{t \cong_T t'}$$

$$\frac{t \in (S \Longrightarrow T) \quad t' \in (S \Longrightarrow T) \quad t \cong_{(S \Longrightarrow T)} t' \quad s \in S}{ts \cong_T t's}$$

$$\frac{t \in (S \Longrightarrow T) \quad t' \in (S \Longrightarrow T) \quad \forall x(x \in S \to tx \cong_T t'x)}{(t \cong_{(S \Longrightarrow T)} t')}$$

$$\frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad t \cong_{\mathsf{Prop}} t' \quad ^{\mathsf{true}}t}{^{\mathsf{true}}t'}$$

We can derive:

$$\frac{t \in (S \Longrightarrow \mathsf{Prop}) \quad t' \in (S \Longrightarrow \mathsf{Prop}) \quad {}^{\mathsf{true}}(t \mathrel{\hat{\cong}}_{(S \Longrightarrow \mathsf{Prop})} t') \quad s \in S}{{}^{\mathsf{true}}ts \leftrightarrow {}^{\mathsf{true}}t's}$$

$$\frac{t \in \mathsf{Prop} \quad t' \in \mathsf{Prop} \quad {}^{\mathsf{true}}(t \mathrel{\hat{\cong}}_{\mathsf{Prop}} t')}{{}^{\mathsf{true}}t \leftrightarrow {}^{\mathsf{true}}t'}$$

5 Two Extensions

There are many ways in which this system can be extended and strengthened. Here we will sketch two possibilities that are relevant for the analysis of type-general expressions in natural language semantics. Such expressions are exemplified by "is fun," which accepts individuals, gerunds and infinitives, as illustrated in the following sentences:

John is fun.
Tennis is fun.
Playing tennis is fun.
To play tennis is fun.

One proposal, suggested by Chiercha and Turner [5], is to incorporate a universal type. A second possibility is to allow the expression of polymorphic types within the logic.[5]

Other extensions to the system are also possible, such as incorporating dependent sum and product types, which can be used to model natural language discourse [22, 23, 10, 15, 12]. We will not elaborate on these possibilities here.

5.1 A Universal Type

We can add the Universal Type Δ to the language of types, which is then governed by the following axiom:

UT $x \in \Delta \leftrightarrow x = x$

This might appear inconsistent, as it would be in standard ZF set theory, but we can see that it is not, by noting that $x \in x$ is not in the language.

The expression "is fun" can now be given the type $\Delta \Longrightarrow \mathsf{Prop}$. Similarly, conjunctive terms, such as "and" and "or" can be given the type $\Delta \Longrightarrow \Delta \Longrightarrow \Delta$, allowing any two terms to be combined to give. This follows the approach of Chiercha and Turner [5]. Unfortunately, this does not guarantee that the individual conjuncts and the final conjoined expression are of the same type.

[5] The system given so far is *implicitly* polymorphic, but as it stands, there is no way of expressing polymorphic types directly.

5.2 Polymorphic Types

To add polymorphic types, first we have to enrich the language of types to include type variables X, and our language of wff to include quantification over types (using these type variables) $\forall X\varphi, \exists X\varphi$.

We also need to add comprehension types of the form $\{x : \varphi'\}$, where φ' are first order, and do not contain bound type variables. This excludes expressions of the form $\{x : \exists X(x \in X)\}$ (the set of terms that have a type).

The proof rules for quantification over types are as expected.

Now we can express universal (implicit) polymorphic types, by adding $\Pi X.T$ to the language of types, which is governed by the following axiom:

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

Such types represent infinite intersections of types. As an example, $\Pi X(X \Longrightarrow X)$ is the type of function in every general function space (such as $\lambda x.x$). The expression "is fun" can now be given the type $\Pi X.(X \Longrightarrow P)$.

To allow conjunctions of terms and nominalised expressions (such as gerunds and infinitives) to appear as arguments to type general verbs, natural language conjunction "and" can be given the type $\Pi X(X \Longrightarrow X \Longrightarrow X)$. This guarantees that the conjuncts and the final conjunctive expression are all of the same type, although there is no restriction on what that type might be.

These two approaches to dealing with so-called type-shifting phenomena—exemplified by nominalisation and conjunction—are not exclusive: if we adopt additional extensions, we can obtain a theory in which there is both a universal type and polymorphic types. Other analyses become available if we increase the power and expressiveness of the PTCT's language of types to include, for example, sub-types, union types, and existential types.

6 Comparison with Higher Order Logic

PTCT can be better understood by contrasting it with the language of Higher Order Logic (HOL) [6, 28], whose syntax can be formulated along the following lines (this is essentially the syntax of Montague [20], but where propositions are basic rather than functions from indices to truth):

$$\begin{array}{ll} \text{(Terms)} & t \ ::= x^T \mid \varphi^T \mid \lambda x^T(t) \mid (t)t \mid \varphi \rightarrow \psi \mid \forall x^T \varphi \\ \text{(Types)} & T ::= P \mid e \mid T \Longrightarrow T \\ \text{(Judgements)} & \alpha ::= (t =_T s) \mid t \in T \end{array}$$

The type P corresponds to propositions. If φ, ψ are terms of type P, and T is a type then $\varphi \rightarrow \psi$ and $\forall x^T \varphi$ are propositions. Rules of inference then govern acceptable type judgments and appropriate behaviour of quantification and implication in the language of terms.

Acceptable sequents have a consequent in the language of Judgements, or the language of propositional terms, where B is a finite set of propositions and

judgments. For completeness we give the basic classical HOL theory, omitting the rules and axioms that govern the typed λ-calculus:

$$\frac{B, \varphi \vdash \psi}{B \vdash \varphi \to \psi} \to \text{intro} \quad \frac{B \vdash \varphi \quad B \vdash \varphi \to \psi}{B \vdash \psi} \to \text{elim}$$

$$\frac{B \vdash \varphi[x^T]}{B \vdash \forall x^T.\varphi} \forall \text{ intro} \quad \frac{B \vdash \forall x^T.\varphi}{B \vdash \varphi[t^T/x^T]} \forall \text{ elim}$$

The usual side conditions apply.

LEM $\forall \varphi^P.\varphi \vee \sim\varphi$

The other connectives can be introduced definitionally.

In contrast to HOL:

1. PTCT adopts Curry-style typing of the untyped λ-calculus [8, 9], as opposed to Church-style typing;
2. the logical behaviour of PTCT is not imposed directly on the (propositional) terms, instead all the appropriate behaviour is expressed in a first-order language of wff, where the truth theory for the language of terms is specified.

7 Other Approaches

There are other ways in which a language with Curry-style typing and a first-order language of wffs could be used to represent natural language semantics which may merit further exploration. For example:

1. With a suitably rich polymorphic type theory, predicates could be taken to correspond to types. Predication then corresponds to a type judgment, which is an atomic wff. For natural language semantics, where predicates and propositions may appear as arguments, the language of terms would have to include types and internalised type judgments, giving rise to a higher-order theory.
2. The language of types could be enriched to include a type corresponding to *True*. Truth then becomes a type judgment. Care would have to be taken with such a formulation, to avoid the semantic paradoxes.

Turner [28] presents a Higher-Order Logic with Curry typing (HOLCU). PTCT differs from this in that the language of wffs is first-order.

8 Consistency of PTCT

Here we sketch an approach to demonstrating the consistency of PTCT with respect to HOL.[6] For simplicity, we assume that the connectives of the wff of PTCT are to be defined in terms of $\forall$ and $\to$.

[6] The proposal follows the presentation of the proof of soundness for HOLCU as given by Turner [28].

First we define a mapping from HOL expressions into the wff of PTCT using the function *trans*, where:

$$trans\ (\forall x^T.\varphi) \equiv \forall x(x \in T \rightarrow trans\ (\varphi))$$
$$trans\ (\varphi \rightarrow \psi) \equiv trans\ (\varphi) \rightarrow trans\ (\psi)$$

We then seek to show that this mapping is sound (every valid sequent in HOL is mapped to a valid sequent in PTCT), and complete (for each valid sequent in PTCT, a valid sequent in HOL can be found which maps to it), by way of induction over the derivations of HOL and PTCT respectively. If the mapping is complete, then PTCT is consistent.

The theory PTCT incorporates additional primitives, in particular $\hat{\cong}$, which has no analogue in HOL. For this reason, consistency would then have to be shown with respect to a logic which incorporates this notion of extensional identity, such as the system described by Fox, Lappin and Pollard [13, 14].

9 Tableau Rules

To indicate how PTCT can be implemented as a computational theory, we sketch the corresponding tableau system. 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 (,).

A proof that these rules precisely captures PTCT is left as an exercise.

9.1 Rules for Wffs

Top

$\sim\top$
|
$\bot$

Conjunction

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

Negated Conjunction

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

Disjunction

$(s \vee t)^*$
$s \quad t$

Negated Disjunction

$\sim(s \vee t)^*$
|
$\sim s$
$\sim t$

Implication

$(s \rightarrow t)^*$
$\sim s \quad t$

Negated Implication

$\sim(s \rightarrow t)^*$
|
s
$\sim t$

Bi-Implication

$$\begin{array}{c} (s \leftrightarrow t)^* \\ \diagup \quad \diagdown \\ s \quad\quad \sim s \\ t \quad\quad \sim t \end{array}$$

Negated Bi-Implication

$$\begin{array}{c} \sim(s \leftrightarrow t)^* \\ \diagup \quad \diagdown \\ \sim s \quad\quad \sim t \\ t \quad\quad s \end{array}$$

Boolean Negation

$$\begin{array}{l} \sim\sim s^* \\ \quad | \\ \quad s \end{array}$$

Negated Quantification

$$\begin{array}{l} \sim Qx\varphi^* \text{ where } Q' \text{ is the dual of } Q \\ \quad | \\ Q'x \sim \varphi \end{array}$$

Universal Quantification

$$\begin{array}{ll} \forall x.\varphi & \text{where } k \text{ occurs on the path} \\ \quad | & \text{(or is the only constant} \\ \varphi[k/x] & \text{occuring in the path)} \end{array}$$

Existential Quantification

$$\begin{array}{ll} \exists x.\varphi^* & \text{where } k \text{ does not occur on the} \\ \quad | & \text{path} \\ \varphi[k/x] & \end{array}$$

Identity

$$\begin{array}{l} t = s, \varphi \\ \quad | \\ \varphi[t/s] \end{array}$$

Non-Identity

$$\begin{array}{l} \sim(t = t) \\ \quad | \\ \quad \bot \end{array}$$

α-reduction

$$\begin{array}{ll} \ldots \lambda x.t \ldots & \text{where } y \text{ is a bound} \\ \quad | & \text{variables that appear} \\ \ldots \lambda y.t[y/x] \ldots & \text{in the path and is} \\ & \text{not free in } t \end{array}$$

β-reduction

$$\begin{array}{l} \ldots (\lambda u.t)(a) \ldots \\ \quad\quad | \\ \quad \ldots t[a/u] \ldots \end{array}$$

η-reduction

$$\begin{array}{l} \ldots (\lambda u.t)(u) \ldots \\ \quad\quad | \\ \quad \ldots t \ldots \end{array}$$

Equivalence (1)

$$\begin{array}{ll} \sim(t \cong_T t) & \text{where } t \in T \text{ is on the path} \\ \quad | & \\ \quad \bot & \end{array}$$

Equivalence (2)

$$\begin{array}{ll} (t \cong_T s), \sim(s \cong_T t) & \text{where } t, s \in T \text{ is} \\ \quad\quad | & \text{on the path} \\ \quad\quad \bot & \end{array}$$

Equivalence (3)

$$\begin{array}{ll} (s \cong_T u), (u \cong_T t) & \text{where } s, u, t \in T \\ \quad | & \text{is on the path} \\ (s \cong_T t) & \end{array}$$

Equivalence (4)

$$\begin{array}{ll} (s \cong_{\mathsf{Prop}} t)^* & \text{where } s, t \in \mathsf{Prop} \text{ is on} \\ \diagup \quad \diagdown & \text{the path} \\ s \quad\quad \sim s & \\ t \quad\quad \sim t & \end{array}$$

Co-extensionality (1)

$$\begin{array}{ll} (s \cong_{(S \Longrightarrow T)} t) & \text{where } u \in S \text{ and} \\ \quad | & s, t \in (S \Longrightarrow T) \text{ are} \\ (s(u) \cong_T t(u)) & \text{on the path} \end{array}$$

Co-extensionality (2)

$$\begin{array}{ll} \sim(s \cong_{(S \Longrightarrow T)} t)^* & \text{where } u \text{ is not on} \\ \quad | & \text{the path} \\ \quad u \in S & \\ \sim(s(u) \cong_T t(u)) & \end{array}$$

9.2 Type Inference Rules

General Function Spaces

$t \in (S \Longrightarrow T)$ *where* $t' \in S$ *is on the path*
$\quad | \quad$
$\quad tt' \in T$

Equivalence within a Type

$t = t'$ *where* $t \in T$ *is on the path*
$\quad | \quad$
$t' \in T$

Conjuctive Propositions

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

Disjuctive Propositions

$t \in \mathsf{Prop}, t' \in \mathsf{Prop}$
$\quad | \quad$
$\quad (t \hat{\vee} t') \in \mathsf{Prop}$

Implicative Propositions

$t \in \mathsf{Prop}, t' \in \mathsf{Prop}$
$\quad | \quad$
$\quad (t \rightarrow t') \in \mathsf{Prop}$

Bi-implicative Propositions

$t \in \mathsf{Prop}, t' \in \mathsf{Prop}$
$\quad | \quad$
$\quad (t \hat{\leftrightarrow} t') \in \mathsf{Prop}$

Universal Propositions

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

Existential Propositions

$(\lambda x.t) \in (S \Longrightarrow \mathsf{Prop})$
$\quad | \quad$
$\quad \hat{\exists} x \varepsilon S.t \in \mathsf{Prop}$

True Propositions

$\quad {}^{\mathsf{true}}s$
$\quad | \quad$
$s \in \mathsf{Prop}$

Equivalence

$\quad t \in T, t' \in T$
$\quad | \quad$
$(t \hat{=}_T t') \in \mathsf{Prop}$

9.3 Truth Rules

Absurdity

$({}^{\mathsf{true}}\bot)^*$
$\quad | \quad$
$\quad \bot$

Conjunction

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

Disjunction

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

Implication

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

Bi-Implication

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

Universal Quantification

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

Existential Quantification

$\quad {}^{\mathsf{true}}(\hat{\exists} x \epsilon S.t)^*$ *where* $(\lambda x.t) \in (S \Longrightarrow \mathsf{Prop})$ *is on the path*
$\quad | \quad$
$\exists x(x \in S \wedge {}^{\mathsf{true}}t)$

10 Future Work

There are extensions to PTCT that are of relevance to natural language semantics. We could incorporate separation types and dependent types into the

language of types in order to the model discourse phenomena [22, 23, 10, 15, 12], and add (intensional) Peano arithmetic to deal with the full range of generalised quantifiers [13].

11 Conclusions

The paper presents a highly intensional theory that can emulate some key aspects of typed logics within a first-order framework. In this system, extensional equivalence does not entail intensional identity. It is sufficiently expressive to lend itself to extensions that incorporate universal and polymorphic types, which can be used to model some natural language nominalisation phenomena, and the type-general nature of conjunction.

References

[1] 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.

[2] J Barwise and J. Etchemendy. Information, infons, and inference. In R. Cooper, K. Mukai, and J. Perry, editors, *Situation Theory and Its Applications*, volume 1, pages 33–78. CSLI, Stanford, CA, 1990.

[3] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press (Bradford Books), Cambridge, MA, 1983.

[4] R. Carnap. *Meaning and Necessity*. University of Chicago Press, Chicago, 1947.

[5] G. Chierchia and R. Turner. Semantics and property theory. *Linguistics and Philosophy*, 11:261–302, 1988.

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

[7] A. Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941.

[8] H.B Curry and R. Feys. *Combinatory Logic*, volume 1 of *Studies in Logic*. North Holland, 1958.

[9] H.B. Curry, R. Hindley, and J. Seldin. *Combinatory Logic*, volume 2 of *Studies in Logic*. North Holland, 1958.

[10] R. Dávila-Pérez. *Semantics and parsing in intuitionistic categorial grammar*. PhD thesis, University of Essex, 1995.

[11] Nick Davies. Towards a first-order theory of reasoning agents. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-90)*, 1990.

[12] C. Fox. *The Ontology of Language*. CSLI Lecture Notes. CSLI, Stanford, 2000.

[13] C. Fox and S. Lappin. A framework for the hyperintensional semantics of natural language with two implementations. In P. de Groote, G. Morrill, and C. Retore, editors, *Logical Aspects of Computational Linguistics*, Springer Lecture Notes in Artificial Intelligence, pages 175–192. Springer-Verlag, Berlin and New York, 2001.

[14] C. Fox, S. Lappin, and C. Pollard. A higher-order intensional logic for fine-grained semantic representation. To appear in the proceedings of the Intensional Logic Workshop, ESSLLI, 2002.

[15] C.J. Fox. Discourse representation, type theory and property theory. In H. Bunt, R. Muskens, and G. Rentier, editors, *Proceedings of the International Workshop on Computational Semantics*, pages 71–80, ITK, Tilburg, 1994.

[16] P.C. Gilmore. An intensional type theory: Motivation and cut-elimination. *Journal of Symbolic Logic*, 66:383–400, 2001.

[17] R. Hindley and J. Seldin. *Introduction to Combinators and the Lambda Calculus*. London Mathematical Society Student Texts 1. Cambridge University Press, 1986.

[18] F. Landman. Pegs and alecs. In *Towards a Theory of Information. The Status of Partial Objects in Semantics*, Groningen-Amsterdam Studies in Semantics, pages 97–136. Foris, Dordrecht, 1986.

[19] S. Lappin and C. Pollard. A hyperintensional theory of natural language interpretation without indices or situations. ms., King's College, London and Ohio State University, 1999.

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

[21] R. Muskens. *Meaning and Partiality*. CSLI and FOLLI, Stanford, CA, 1995.

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

[23] A. Ranta. *Type Theoretic Grammar*. Oxford University Press, 1994.

[24] J. Seligman and L. Moss. Situation theory. In J. van Bentham and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsvier, North Holland, Amsterdam, 1997.

[25] R. Thomason. A modeltheory for propositional attitudes. *Linguistics and Philosophy*, 4:47–70, 1980.

[26] R. Turner. A theory of properties. *Journal of Symbolic Logic*, 52(2):455–472, June 1987.

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

[28] R. Turner. Types. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 535–586. Elsevier, 1997.