Curry-Typed Semantics in Typed Predicate Logic

CHRIS FOX

Abstract: Various questions arise in semantic analysis concerning the nature of types. These questions include whether we need types in a semantic theory, and if so, whether some version of simple type theory (STT, Church, 1940) is adequate or whether a richer more flexible theory is required to capture our semantic intuitions. Propositions and propositional attitudes can be represented in an essentially untyped first-order language, provided a sufficiently rich language of terms is adopted. In the absence of rigid typing, care needs to be taken to avoid the paradoxes, for example by constraining what kinds of expressions are to be interpreted as propositions (Turner, 1992). But the notion of type is ontologically appealing. In some respects, STT seems overly restrictive for natural language semantics. For this reason it is appropriate to consider a system of types that is more flexible than STT, such as a Curry-style typing (Curry & Feys, 1958). Care then has to be taken to avoid the logical paradoxes. Here we show how such an account, based on the Property Theory with Curry Types (PTCT, Fox & Lappin, 2005), can be formalised within Typed Predicate Logic (TPL, Turner, 2009). This presentation provides a clear distinction between the classes of types that are being used to (i) avoid paradoxes (ii) allow predicative polymorphic types. TPL itself provides a means of expressing PTCT in a uniform language.

Keywords: Curry types, semantics, logic, intensionality

1 Introduction

In previous work the author has explored *Property Theory*, and issues that arise in intensionality, plurals, mass terms, discourse (Fox, 2000); *Property Theory with Curry Typing* (PTCT), and the representation of polymorphism, quantifier scoping, ellipsis (Fox & Lappin, 2005); "judgemental" proof-theoretic approaches to the semantics of imperatives (Fox, 2012a), obligations and permissions (Fox, 2012b), questions (Fox, 2013); as well as various related methodological issues (Fox, 2014; Fox & Turner, 2012).

The aim of this paper is to revisit earlier work on intensionality, but using a judgemental approach within a unifying framework. For natural language

semantics we need fine-grained intensionality in order to capture the fact that we can have distinct beliefs about propositions with the same truth conditional, as well as having distinct questions with the same answerhood conditions, and distinct commands with the same satisfaction conditions. This requires something more fine-grained than Montague-style possible worlds semantics (Fox, 2000; Fox & Lappin, 2005). It is also appropriate to have flexible typing, to characterise the behaviour of polymorphic operators (and also to allow novel approaches to ellipsis and anaphora). This can be achieved by adopting a Curry-style approach to types (Curry & Feys, 1958). If possible, there should be some sensitivity to ontological questions, in particular we may wish to avoid an ontological collapse that results if everything is reduced to set-theoretic model-theory (Fox, 2014; Fox & Turner, 2012).

A central part of the approach adopted here is to regard logic as first-class citizen, and not merely a stepping-stone to a set-theoretic model. This is motivated by a desire to avoid ontological collapse, and treat ontological intuitions "faithfully". Avoiding ontological reduction helps to avoid unintended consequences.

1.1 An Existing Account

A pre-existing account for a logic with fine-grained intensionality and flexible typing is *Property Theory with Curry Typing* (PCTC Fox & Lappin, 2005). This consists of a "federation" of languages, in particular,

1. a language of wffs,
2. the untyped λ-calculus,
3. A language of types.

Wffs are used to express behaviour within a proof-theory. The λ-calculus provides the terms of the wffs. Intensional expressions are then *represented* by such terms.

Arguably, there are some æsthetic issues with the original presentation of PTCT. The multi-level syntax may appear rather clumsy. And the original tableaux-style presentation is hard to extend. There are also some other concerns, such as how the a theory that focuses on propositions can be combined with analyses of non-propositional utterances and related phenomena, including imperatives, deontic expressions, questions and answers.

One way of resolving these issues is to adopt a cleaner, judgement-based proof-theoretic approach, were all the basic notions of theory are expressed in terms of a unified language of judgements.

1.2 Typed Predicate Logic

An alternative, unifying framework in which to formulate PTCT, and other theories, is provided by Typed Predicate Logic (TPL, Turner, 2008, 2009). TPL provides us with a minimal but expressive meta-theory. In some sense, it can be used in the same way that possible worlds is often used: it provides a generic framework, but without possible worlds problems (such as limited intensionality, and issues around ultimately reducing everything to sets). TPL allows us to express both syntactic and logical behaviour.

The basic judgements of TPL are as follows

1. T Type — T is a type
2. $t : T$ — t belongs to type T
3. p Prop — p is a proposition
4. p True (or just "p") — p is true

Such judgements appear with sequents of the form

$$\Gamma \vdash \Phi$$

where Φ is any judgement, and Γ a context. In TPL, both the syntactic and logical behaviour of a logic or language is expressed using sequent rules of the following form.

$$\frac{\Gamma_1 \vdash \Phi_1 \quad \Gamma_2 \vdash \Phi_2 \quad \ldots \quad \Gamma_n \vdash \Phi_n}{\Gamma \vdash \Phi}$$

Sometime we elide the contextual part of the judgement "$\Gamma_i \vdash$" when context Γ_i is the same for all judgements in a sequent rule.

We can exemplify the use of TPL by formulating some rules for propositions. Grammatical formation rules for conjunction can be expressed as follows,

$$\frac{\Gamma \vdash p \text{ Prop} \quad \Gamma \vdash q \text{ Prop}}{\Gamma \vdash p \wedge q \text{ Prop}} \wedge F$$

so if p and q are both propositions, then so is $p \wedge q$.

Logical rules for conjunction then characterise its classical behaviour.[1]

$$\frac{\Gamma \vdash p \ \text{True} \quad \Gamma \vdash q \ \text{True}}{\Gamma \vdash p \wedge q \ \text{True}} \wedge + \qquad \frac{\Gamma \vdash p \wedge q \ \text{True}}{\Gamma \vdash p \ \text{True}} \wedge -$$

For universal quantification (typed), we can have the following formation rule

$$\frac{\Gamma, x : T \vdash p \ \text{Prop}}{\Gamma \vdash \forall x \epsilon T \cdot p \ \text{Prop}} \forall F$$

and the following logical rules.

$$\frac{\Gamma, x : T \vdash t \ \text{True}}{\Gamma \vdash \forall x \epsilon T \cdot t \ \text{True}} \forall + \qquad \frac{\Gamma \vdash \forall x \epsilon T \cdot t \ \text{True} \quad \Gamma \vdash s : T}{\Gamma \vdash t[s/x] \ \text{True}} \forall -$$

There are a number of ways in which we could show the consistency of a logic formulating in this way. For example, we might seek to show how the theory can be viewed as a conservative extension of another (sound) theory. Or we could build a model theory. In that case, the model theory is to be viewed as secondary to the logical theory; it is merely a tool to demonstrate properties of the logic, and not necessarily the intended framework for semantic analysis.

Here we shall see how PTCT can be reformulated within the framework of TPL. The proof-theoretic approach of TPL means that expressions can be thought of as being intensional "by default". Care will still need to be taken to avoid logical paradoxes.

2 PTCT in TPL

Theories are formulated in TPL by adding sequent rules. For PTCT we need a type system that allows for:

1. Curry-typed λ-calculus (for compositional semantics with flexible typing);
2. object-language representable types
3. type quantification (for polymorphic types, which avoids the need for "type-raising");

[1] Note that for a proper treatment, we would also need to give structural rules of *assumption*, *thinning* and *substitution* to allow the formation and logical rules to capture all aspects of the expected behaviour of the logical operators.

The nature of types, and object-level judgements, need to be constrained to avoid logical paradoxes, and to avoid impredicativity. To this end, PTCT has a simple stratification of types to avoid paradoxes and impredicativity while remaining weak.[2] The key kinds of types in PTCT can be classed as follows.

1. B — the basic type of individuals
2. Λ — "untyped" λ-calculus terms
3. τ — object level types
4. σ — quantifiable types

The type Λ is not an object level type. That is, it cannot appear *within* the language that we are formulated. The type τ is essentially a "type of types" but limited to those types that can appear within the expressions of the target language. As we shall see, the syntax will be further restricted so that there are no free-floating type judgements. The σ types are those over which we can quantify in the target language, as with polymorphic types. To avoid impredicativity, such types will exclude those that themselves involve quantification over types. The significance of some of these constraints will be considered again below (Section 3).

2.1 Structural Rules

We first need to give housing keeping rules for "well-formedness" judgements. These are the usual rules for *assumption*, *thinning* and *substitution*, but specific to the case of formation rules in this TPL framework.[3]

$$\frac{\Gamma \vdash T \ \text{Type}}{\Gamma, x : T \vdash x : T}\ \textit{Ass} \qquad \frac{\Gamma \vdash T \ \text{Type} \quad \Gamma \vdash \Phi}{\Gamma, x : T \vdash \Phi}\ \textit{Thin}$$

$$\frac{\Gamma, x : T \vdash \Phi[x] \quad \Gamma \vdash t : T}{\Gamma \vdash \Phi[x/t]}\ \textit{Sub}$$

[2] We take it that weakness is a virtue particularly when it can be shown that relevant patterns of behaviour can be captured elegantly without resorting to more powerful formal machinery.

[3] We formulate these structural rules in a way where the context is explicitly narrowed down to just that part that is relevant to syntactic judgements, assuming that the syntax and logic are independent, but have chosen not to do so in order to simplify the exposition.

Here Φ stands for any type judgement. We also need versions of these structural rules for the logic behaviour, as follows.

$$\frac{\Gamma \vdash t \ \text{Prop}}{\Gamma, t \vdash t}\ \mathit{Ass}' \qquad \frac{\Gamma \vdash t \ \text{Prop} \quad \Gamma \vdash \Phi}{\Gamma, t \vdash \Phi}\ \mathit{Thin}'$$

$$\frac{t : \Lambda}{t =_\Lambda t}\ \mathit{id}_1 \qquad \frac{t =_\Lambda s \quad \phi[t]}{\phi[s]}\ \mathit{id}_2$$

Rules of this nature are not distinctive of PTCT, except that we have narrowed identity to just Λ terms.

2.2 λ-calculus

We can give formation/closure rules for λ-calculus/Λ, and identity rules to capture $\alpha\beta$-equivalence in TPL. These are of relevance to the structural rules for substitution. First we have the rules for the syntactic category of Λ.

$$\frac{s : \Lambda \quad t : \Lambda}{st : \Lambda}\ \Lambda_{\text{app}} \qquad \frac{\Gamma, x\Lambda \vdash t : \Lambda}{\Gamma \vdash\colon \lambda x.t : \Lambda}\ \Lambda_{\text{abs}}$$

Function types for Λ/λ are governed by the following.

$$\frac{\Gamma, x : S \vdash t : T}{\Gamma \vdash \lambda x.t : S \Rightarrow T}\ \lambda_\Rightarrow{+} \qquad \frac{s : S \Rightarrow T \quad t : S}{st : T}\ \lambda_\Rightarrow{-}$$

And we have the usual α, β rules.[4]

$$\frac{s : \Lambda \quad t : \Lambda \quad r : \Lambda \quad t =_\Lambda r}{s =_\Lambda s[t/r]}\ \Lambda\alpha$$

$$\frac{\Gamma, x : \Lambda \vdash t : \Lambda \quad \Gamma \vdash s : \Lambda}{\Gamma \vdash (\lambda x.t)s =_\Lambda t[s/x]}\ \Lambda\beta$$

2.3 Propositions

TPL has a notion of proposition. But this is in the meta-theory. We need to give formation rules for the object level PTCT propositions (π) and the logical rules of truth for PTCT propositions.[5] Rather than give all the details of

[4] We could also add η.

[5] Note that the structural rules play a role in avoiding considering the truth of non-propositional expressions. There is insufficient space to demonstrate this, and it is left as an exercise for the interested reader.

all the PTCT connectives and quantifiers, we illustrate the nature of the rules using conjunction ($\wedge$) and universal quantification ($\forall$) as examples, together with the notions of intensional identity (=) and extensional equivalence ($\cong$). First we have the formation rules for expressions in π.

$$\frac{t : \pi \quad s : \pi}{(t \wedge s) : \pi} \wedge F$$

$$\frac{\Gamma, x : T \vdash t : \pi \quad \Gamma \vdash T : \tau}{\Gamma \vdash (\forall x \epsilon T \cdot t) : \pi} \forall F$$

$$\frac{t : T \quad s : T \quad T : \tau}{(t =_T s) : \pi} = F$$

$$\frac{t : T \quad s : T \quad T : \tau}{(t \cong_T s) : \pi} \cong F$$

Type T constrained to an object level type (i.e. T must be a type in τ).

The logical rules for conjunction and universal quantification follow.

$$\frac{s \quad t}{(s \wedge t)} \wedge + \quad \frac{(s \wedge t)}{s} \wedge -_1 \quad \frac{(s \wedge t)}{t} \wedge -_2$$

$$\frac{\Gamma, x : T \vdash t \quad \Gamma \vdash T : \tau}{\Gamma \vdash \forall x \epsilon T \cdot t} \forall + \quad \frac{\forall x \epsilon T \cdot t \quad s : T}{t[s/x]} \forall -$$

Intensional identity is a fine-grained notion. We can take it to be term identity in Λ.[6]

$$\frac{T : \tau \quad s : T \quad t : T \quad s =_\Lambda t}{s =_T t} =_T$$

Equivalence is a more coarse grained notion. For propositions it can be characterised as truth conditional equivalence.

$$\frac{s : \pi \quad t : \pi \quad s \leftrightarrow t}{s \cong_\pi t} \cong_\pi + \quad \frac{s \cong_\pi t}{s \leftrightarrow t} \cong_\pi -$$

The notion can be extended to "operational" extensional equivalence

$$\frac{x : S \quad f \cong_{S \Rightarrow T} g}{fx \cong_T gx} \cong_{S \Rightarrow T} -$$

As an example, properties will be equivalent if they apply to the same individuals.

[6]Intensional identity could be interpreted as operational identity of the λ-calculus, given that the latter can be taken to be a theory of computation.

2.4 Additional Types

We can have rules for PTCT types, including formation and membership rules for polymorphic types and separation types.[7] Here are the rules for polymorphic types.

$$\frac{\Gamma, X : \sigma \vdash T : \tau}{\Gamma \vdash \Pi X \cdot T : \tau}\ \Pi F$$

A term t belongs to object-level type $\Pi X \cdot T$ iff t belongs to T for all "safe" values of X

$$\frac{\Gamma, X : \sigma \vdash t : T}{\Gamma \vdash t : \Pi X \cdot T}\ \Pi{+} \quad \frac{t : \Pi X \cdot T \quad S : \sigma}{t : T[S/X]}\ \Pi{-}$$

Separation types are a form of subtype that combines types with propositions.

$$\frac{\Gamma \vdash T : \tau \quad \Gamma, x : T \vdash s : \pi}{\Gamma \vdash \{x \epsilon T \cdot s\} : \tau}\ \textit{SepF}$$

A term t will be of the type $\{x \epsilon T \cdot s\}$ iff t is in T, and $s[x/t]$ is true.

$$\frac{\{x \epsilon T \cdot s\} : \tau \quad t : T \quad s[x/t]}{t : \{x \epsilon T \cdot s\}}\ \textit{Sep}{+}$$

$$\frac{t : \{x \epsilon T \cdot s\}}{t : T}\ \textit{Sep}{-}_1 \quad \frac{t : \{x \epsilon T \cdot s\}}{s[x/t]}\ \textit{Sep}{-}_2$$

2.5 An example

Without going into details of the compositional analysis, we can illustrate how a sentence may be represented within this framework. If we assume something like a Montagovian-like translation (Montague, 1973), *modulo* the use of PTCT, the sentence

"*Every man loves a woman*"

can be given the representation

$$\forall x \epsilon B \cdot \textit{man}'(x) \rightarrow \exists y \epsilon B \cdot \textit{woman}'(y) \wedge \textit{loves}'(x, y)$$

[7] Function types were introduced implicitly with the rules for λ-application and λ-abstraction.

where

$$\begin{aligned} &\mathit{man}' : B \Rightarrow \pi \\ &\mathit{woman}' : B \Rightarrow \pi \\ &\mathit{loves}' : B \Rightarrow B \Rightarrow \pi \end{aligned}$$

There is no need for "quoting" or an overt truth-operator, unlike original formulation of PTCT, and no need for Montague's "up" and "down" ($^\wedge$,$^\vee$) operators (Montague, 1973, 1974).

In the case of the sentence

"*John believes every man loves a woman*"

we can have the representation

$$\mathit{believes}'(\mathit{John}')(\forall x \epsilon B \cdot \mathit{man}'(x) \rightarrow \exists y \epsilon B \cdot \mathit{woman}'(y) \wedge \mathit{loves}'(x, y))$$

where

$$\mathit{believes}' : B \Rightarrow (\pi \Rightarrow \pi)$$

Here, the "content" of the believe relation is *not* collapsed to a truth value, nor to a set of worlds, and again there is no need to "quote" the proposition, unlike in the original formulation of PTCT, nor use up and down operators to translate between intensional and extensional forms.

3 Paradoxes and Impredicativity

In some respects, this formulation of PTCT in TPL can be viewed as a "flattened" version of a stratified logic, such as the one proposed by Turner (2005). Here we show how the simple layering of the types of PTCT, and other restrictions, help avoid paradoxes and impredicativity.

3.1 Paradoxes

We wish to avoid Russell-like paradoxes. One way is to block various forms of self-application, or arrange things so that their "truth" is never considered. Here we review two potentially problematic cases, (i) a universal type, and (ii) free-standing type-judgements.

In PTCT we have no universal type, and do not allow Λ to appear within PTCT expressions. To see why, consider expression rr where

$$r =_{\text{def}} \lambda x.\exists y \epsilon (\Lambda \Rightarrow \mathrm{P}) \cdot x =_{\Lambda \Rightarrow \mathrm{P}} y \wedge \neg xy$$

The problem is avoided by excluding Λ (or a universal type) from the object language.

PTCT also does not permit "Free-standing" type judgements. The reason can be seen by considering $r'r'$ where

$$r' =_{\text{def}} \{x \epsilon T \cdot \neg\, x \epsilon x\}$$

This problem is avoided if the use of $x \epsilon S$ is constrained. In both cases, other solutions to these potential pitfalls are possible.

PTCT is intended for the semantic interpretation of natural language. For this intended use, the lack of universal type and free-standing type-membership may not be an issue, as such expressions do not appear to be required to represent any natural utterance.

3.2 Impredicativity

In PTCT we have a form of explicit polymorphism ($\Pi X.T$). It is useful for conjunction, which we can type as $\Pi X.X \Rightarrow (X \Rightarrow X)$, and perhaps for and type general predicates (e.g. "*is fun*" : $\Pi X.X \longrightarrow \pi$, cf. Chierchia, 1982).

But polymorphic types can lead to an impredicative theory. If the type T is defined by the following,

$$T =_{\text{def}} \Pi X.S$$

we may wonder whether X range over T itself, the type that is being defined. If it does, then the definition is impredicative. In PTCT, this impredicativity is avoided by a two-level stratification:

1. τ — the types that can appear within PTCT
2. σ — the smaller class of τ over which Π quantifies.

The class of types σ excludes the polymorphic types themselves.

3.3 A Truth Predicate?

Finally, we can consider whether it is feasible to add a truth "predicate", or judgement, to the object language. Care would need to be taken to avoid logical paradoxes. If $\tau(t)$ were a PTCT expression corresponding to "t True", then we might wish to constrain it so that it can only form a proposition if t itself is a proposition (cf. Turner, 1990).

4 Other Issues

Other issues we can reflect on are the consistency of PTCT, and the use of TPL as a more general unifying framework.

4.1 Interpretation and Consistency

A modal for PTCT has already given elsewhere (Fox & Lappin, 2005). This treats intensional identity as inscriptional. An alternative is to think of intensions as procedures, so intensional identity is characterised as operational identity, and extensional equivalence is denotational equivalence. Note that PTCT already contains untyped λ-calculus, which can be interpreted as a model of computation. This proposal is explored in (Fox & Lappin, 2013).

4.2 Unified Framework

One aim of formulating PTCT in TPL is to provide a unified framework that should make it easier to incorporate accounts of non-propositional phenomena. Previous work on imperatives (Fox, 2012a), obligations and permissions (Fox, 2012b), and questions (Fox, 2013) has appealed to logical characterisations involving a range of different judgements, including commanding, satisfying asking, answering. These might all be incorporated into TPL, or an extension of it.

Another unification that suggests itself is the formulation of natural language syntax in TPL, alongside its semantic interpretation in, for example, PTCT. The most natural grammar frameworks for might be along the lines of the Lambek calculus (Lambek, 1958), or some other type-logical grammar (Morrill, 1994; Steedman, 1993, 2000) — although there are no intrinsic constraints in TPL that would require us to adopt such formalisations. This would then provide an analysis of a fragment of natural language syntax and semantics in a uniform framework. The goal would be similar to that of Cooper (2012), where semantics and syntax are to be given a uniform analysis in terms of record types. But with TPL there would be no commitments or obligations to adhere to any particular kind of structured type.

5 Conclusions

We have reprised some core aspects of Property Theory with Curry Typing (PTCT), and presented a reformulation within Typed Predicate Logic (TPL). This gives an elegant presentation of the theory at a more abstract

level, and avoids the need to invoke distinct levels of representation in the semantic analysis of natural language. PTCT is then captured within a single language in place of federation of languages for terms, types, and wffs. The notion of judgement in TPL may be adapted and extended to include judgements that are relevant for non-propositional utterances. The way in which the grammatical and logical rules are given in a uniform presentation in TPL may be extended to give rules governing both the syntax and semantic interpretation of natural language within a single unifying framework.

References

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

Church, A. (1940, June). A formulation of the Simple Theory of Types. *The Journal of Symbolic Logic*, *5*(2), 56–68.

Cooper, R. (2012). Type theory and semantics in flux. In R. Kempson, T. Fernando, & N. Asher (Eds.), *Philosophy of linguistics* (pp. 271–323). Amsterdam: Elsevier.

Curry, H. B., & Feys, R. (1958). *Combinatory logic* (Vol. 1). North Holland.

Fox, C. (2000). *The ontology of language*. Stanford: CSLI.

Fox, C. (2012a). Imperatives: a judgemental analysis. *Studia Logica*, *100*(4), 879–905.

Fox, C. (2012b). Obligations and permissions. *Language and Linguistics Compass*, *6*(9), 593–610.

Fox, C. (2013). Axiomatising questions. In V. Puncochar & P. Svarny (Eds.), *Logica year book 2012* (pp. 23–34). College Publications.

Fox, C. (2014). *The meaning of formal semantics.* (To appear in P. Stalmaszczyk (Ed.), *Semantics and Beyond. Philosophical and Linguistic Investigations*, Ontos Verlag.)

Fox, C., & Lappin, S. (2005). *Formal foundations of intensional semantics*. Oxford: Blackwell.

Fox, C., & Lappin, S. (2013). *Type-theoretic logic with an operational account of intensionality.* (Manuscript.)

Fox, C., & Turner, R. (2012). In defense of axiomatic semantics. In P. Stalmaszczyk (Ed.), *Philosophical and formal approaches to linguistic*

analysis (pp. 145–160). Ontos Verlag.
Lambek, J. (1958). The mathematics of sentence structure. *Mathematical Monthly*, *65*, 154–169.
Montague, R. (1973). The proper treatment of quantification in ordinary English. In K. J. J. Hintikka, J. M. E. Moravcsik, & P. Suppes (Eds.), *Approaches to natural language* (pp. 221–242). Dordrecht: D. Reidel. (Synthese Library. Also in *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, 1974, edited by R. H. Thomason)
Montague, R. (1974). *Formal philosophy*. edited by R. Thomason, New Haven, CT: Yale University Press.
Morrill, G. (1994). *Type logical grammar: Categorial logic of signs*. Dordrecht: Kluwer Academic Publishers.
Steedman, M. (1993). Categorial grammar (tutorial overview). *Lingua*, *90*, 221–258.
Steedman, M. (2000). *The syntactic process*. Cambridge: MIT Press.
Turner, R. (1990). *Truth and modality for knowledge representation*. Pitman.
Turner, R. (1992). Properties, propositions and semantic theory. In M. Rosner & R. Johnson (Eds.), *Computational linguistics and formal semantics* (pp. 159–180). Cambridge: Cambridge University Press.
Turner, R. (2005). Semantics and stratification. *Journal of Logic and Computation*, *15*(2), 145–158.
Turner, R. (2008). Computable models. *Journal of Logic and Computation*, *18*(2), 283-318.
Turner, R. (2009). *Computable models*. London: Springer-Verlag.

Chris Fox
University of Essex
e-mail: foxcj@essex.ac.uk
URL: http://chris.foxearth.org/