THE UNTYPED λ-CALCULUS AND SEMANTICS FOR NATURAL LANGUAGE

An assessment of Property Theory with Curry Types

Will Stafford

27/07/18

University of California, Irvine

SEMANTICS FOR NATURAL LANGUAGE

- Type Theory based on Montague grammar is commonly used for natural language semantics.
- Hyperintentionality is hard to implement.
- Property theory with Curry Types (PTCT) is proposed by Fox and Lappin to over come this problem.
- PTCT consists of the untyped λ calculus with Curry types added.

THE UNTYPED λ CALCULUS

TERMS

Terms are constructed from variables, (,), and λ.

X

xy

$\lambda x.(xy)$

$$(\lambda x.(xy))(zy)$$

SYNTAX

- Formulas are constructed using = alone.
- The untyped λ-calculus consists of rules for transforming equalities.

$$(\lambda x.M)N = M[x := N] \qquad (\beta)$$

$$[\lambda y.([\lambda x.(xy)](zy))]u$$

$$[\lambda y.([\lambda x.(xy)](zy))]u$$

$$[\lambda y.([\lambda x.(xy)](zy))]u = [\lambda x.(xu)](zu)$$

$$[\lambda x.(xu)](zu) = zuu$$

CURRY TYPES

ADDING CURRY TYPES

$\mathtt{Prop}$ is the type of propositions and $\mathtt{B}$ is the type of individuals.

If $\mathsf{T}, \mathsf{S} \in \mathrm{TYPE}$ and X is a type variable, then

1. $\mathsf{T} \Rightarrow \mathsf{S}$ is a type,
2. $\Pi \mathsf{X}.\mathsf{T}$ is a type.

Examples of types:

$$\mathtt{Prop},\ \mathtt{Prop} \Rightarrow \mathtt{Prop},\ \mathtt{B} \Rightarrow (\mathtt{Prop} \Rightarrow \mathtt{Prop}),\ \Pi \mathsf{X}.[\mathsf{X} \Rightarrow (\mathsf{X} \Rightarrow \mathsf{X})]$$

- The valuation of $\Pi X.T$ is the intersection of all types generated by replacing X in T with a type that doesn't contain Π,

$$\varepsilon \Pi X.T_{g,\tau} = \bigcap_{S} \varepsilon T_{g,\tau[S/X]}.$$

- The valuation of $T \Rightarrow S$ is every element of the domain which codes a function from T to S,

$$\varepsilon T \Rightarrow S_{g,\tau} = \{d \in D \mid \forall e \in \varepsilon T_{g,\tau}\ ((d))e \in \varepsilon S_{g,\tau}\}$$

TOY EXAMPLE

Say we want to write the sentence

Sarah believes John is bored.

We need

- Sarah, John$\in \mathtt{B}$,
- bored$\in \mathtt{B} \Rightarrow \mathtt{Prop}$,
- believes$\in \mathtt{B} \Rightarrow (\mathtt{Prob} \Rightarrow \mathtt{Prop})$

TOY EXAMPLE

‘John is bored’ is bored(John)$\in$ `Prop`

‘Sarah believes...’ is believes(Sarah)$\in$ `Prop` $\Rightarrow$ `Prop`

So ‘Sarah belives John is bored’ is belives(Sarah)(bored(John))$\in$ `Prop`

HYPERINTENSIONALITY

INTENSIONALITY

Extensional settings:

- Subsitution of coreferential terms does not change truth value.
- Batman was arrested vs. Bruce was arrested.

INTENSIONALITY

Extensional settings:

- Subsitution of coreferential terms does not change truth value.
- Batman was arrested vs. Bruce was arrested.

Intensional settings:

- Subsitution of coreferential terms can change truth value.
- Vicky believes Batman is a hero vs. Vicky believes Bruce is a hero.

INTENSIONALITY

Extensional settings:

- Subsitution of coreferential terms does not change truth value.
- Batman was arrested vs. Bruce was arrested.

Intensional settings:

- Subsitution of coreferential terms can change truth value.
- Vicky believes Batman is a hero vs. Vicky believes Bruce is a hero.

Hyperintensional settings:

- Subsitution of necessarily logically equivalent terms can change truth value.

INTENSIONALITY

Extensional settings:

- Subsitution of coreferential terms does not change truth value.
- Batman was arrested vs. Bruce was arrested.

Intensional settings:

- Subsitution of coreferential terms can change truth value.
- Vicky believes Batman is a hero vs. Vicky believes Bruce is a hero.

Hyperintensional settings:

- Subsitution of necessarily logically equivalent terms can change truth value.

Example

1. Jamie knows that the law of the excluded middle is true.
2. Jamie knows that Peirce's law is true.

The untyped λ-calculus is intensional

Why?

There are models $\mathcal{M}$ and λ-terms M, N such that for all $\mathsf{a} \in \mathcal{M}$, $\mathcal{M} \vDash \mathsf{M}(\mathsf{a}) = \mathsf{N}(\mathsf{a})$ but $\mathcal{M} \nvDash \mathsf{M} = \mathsf{N}$.

COMPUTATIONAL TRACTABILITY

COMPUTATIONAL TRACTABILITY

Linguistics:

> [P]rinciples that introduce computational complexity have repeatedly been shown to be empirically false
> (Chomsky 2000, p. 15)

Computational linguistics:

> [W]e assume that to understand linguistic tasks [...] is to discover the algorithms required to perform those tasks, and to investigate their computational properties. (Pratt 2013, p. 43)

Computational semantics:

> If we are interested in building practical natural language systems, then it is appropriate to worry about the computational properties of a semantic theory.
> (Fox and Lappin 2005, p. 153)

SECOND-ORDER LOGIC

Fox and Lappin want a system which is computationally weak.

They claim higher-order systems are not suitable because:

- They do not have a complete proof system.
- They are too expressive.

However, the issues they raise are all features of the **standard** semantics, not the **Henkin** semantics.

COMPUTATION IN λ-CALCULI

If we measure the strength of λ-calculi by how much computation they model we get a very different ordering of strength from the one suggested by Fox and Lappin:

- the untyped λ-calculus is Turing complete,
- the typed λ-calculus with polymorphic types is not Turing complete,
- the simply typed λ-calculus is not Turing complete.

THE UNDECIDABILITY OF INTENSIONAL IDENTITY

It isn't decidable if $M = N$ is provable for two untyped λ-terms.

This means that on PTCT's view of intensions it is not obvious when two terms have the same intension.

Deciding whether something has the same intension appears to be an intuitive ability which native speaker possess.

Difficulties philosophers find in this task appear to be better explained as philosophical differences, rather than due to the computational difficulty of the task.

Provable equality

A set of λ-terms A is closed under equality if $t \in A$ and $\lambda \vdash t = s$ then $s \in A$.

Rice's Theorem

Given a set A of λ-terms, if A is closed under provable equality in the λ-calculus, then A is either trivial (the empty set or the set of all λ-terms) or not computable

- Fox and Lappin use types to represent grammatical categories.
- Given a term it is natural to ask if it is, for example, a noun.
- But as a result of Rice's theorem this will generally be undecidable.

CONCLUSION

Fox and Lappin propose PTCT to solve three problems:

1. Hyperintensionality
2. Computational tractability

We argue that while they may offer solutions to the first they fail to offer a more computationally tractable system than their competition.