Property Theory with Curry Typing

An Intensional Logic for Natural Language Semantics

Chris Fox

Dept. of Computer Science

University of Essex

foxcj@essex.ac.uk

Shalom Lappin

Dept. of Computer Science

King's College London

lappin@dcs.kcl.ac.uk

Overview

1. Two dominant assumptions in formal and computational semantics
2. Property Theory with Curry Typing (PTCT): An expressive first-order logic with fine-grained intensionality
3. Syntax and proof theory
4. Model theory
5. Restricted polymorphic types
6. An intensional number theory and generalized quantifiers
7. *[Using subtypes and dependent types for pronominal anaphora resolution]*
8. *[A type-theoretical approach to dynamic anaphora]*
9. Conclusions and future work

Background Summary

- Functional Types and HOL
- Formal Power
- Intensions and Possible Worlds
- Equivalence and Identity
- Impossible Worlds
- Alternatives

>>

PTCT

An Intensional First-Order Logic with Flexible Types

- As in Bealer's (1980) logic, the model theory for PTCT takes intensions to be basic entities.
- Intensions are represented independently of modality and without (im)possible worlds.
- The proof theory prevents the reduction of provable equivalence to identity.
- However, PTCT supports functional (as well as separation and comprehension) types and (restricted) polymorphism.

Polymorphism in NL

Polymorphism in Natural Language

- NL expressions act as if they belong to more than one semantic type: *playing tennis is fun*, *to play tennis is fun*, *tennis is fun* (Chierchia 1982, Turner 1997).
- Some NL expressions (such as conjunctions) can combine expressions of different types. There are constraints on the types of the combined expressions and the resultant expression.
- More “natural” treatments of such phenomena are possible if we allow a flexible system of types (such as Curry-style typing), where
  1. expressions can belong to more than one type
  2. functional types can apply to arguments of several types.

PTCT: Basic Syntax

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

Terms $t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$

(logical constants) $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$

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

Wff $\varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi) \mid (\forall x \varphi) \mid (\exists x \varphi) \mid {}^{\mathsf{true}}t$

(atomic wff) $\alpha ::= t =_T s \mid \bot \mid t \in T \mid t \cong_T s$

- The language of terms is the untyped λ-calculus, enriched with logical constants. It is used to *represent* the interpretations of natural language expressions. It has no internal logic!
- The languages of types and terms can be combined with appropriate rules and axioms to produce a Curry-typed λ-calculus (Turner 1997).
- The first-order language of wffs will be used to formulate type judgements for terms, and truth conditions for those terms judged to be in Prop.

PTCT: Propositions and Terms

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).*

- All wffs are propositions in the usual sense of classical first-order logic.
  - They are true or false.
  - Their identity criteria are those of their truth conditions.
- Some terms *represent* propositions (and predicates).
  - In PTCT, such terms are of type Prop.
  - They have no intrinsic logic.
  - Their identity criteria are those of the λ-calculus.

PTCT: Propositions and Terms

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).*

- If term t represents a proposition, then we can form a wff, $^{\mathsf{true}}(t)$.
  - $^{\mathsf{true}}(t)$ will be a true wff whenever the proposition represented by 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 ($^{\mathsf{true}}(t)$).

PTCT: Rules and Axioms

Rules and axioms govern the logical behaviour of PTCT. They can be classified 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 rules/axioms of the untyped λ-calculus.

Typing of λ-terms These are essentially the rules/axioms 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 notion of extensional equivalence which is given the expected behaviour.

PTCT: Rules and Axioms

Here we exemplify some of these kinds of rules as they apply to conjunction, both as it appears in the language of wff ($\wedge$), and in the language of terms ($\hat{\wedge}$)

The basic connectives of the wff

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

Typing rules for λ-terms

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

Truth conditions for Propositions

$$t \in \mathsf{Prop} \wedge t' \in \mathsf{Prop} \rightarrow ({}^{\mathsf{true}}(t \mathbin{\hat{\wedge}} t') \leftrightarrow {}^{\mathsf{true}}t \wedge {}^{\mathsf{true}}t')$$

PTCT: Alternative Formulation

We also specify tableau-style *rules for the system, giving an effective algorithm for reasoning in* PTCT.

Rules for Wffs

Conjunction

$$\frac{(s \wedge t)^*}{\begin{array}{l} s \\ t \end{array}}$$

Negated Conjunction

$$\frac{\sim(s \wedge t)^*}{\sim s \quad \sim t}$$

Type Inference Rules

Conjunctive Propositions

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

Truth Rules

Conjunction

$$\frac{{}^{\mathsf{true}}(s \mathbin{\hat{\wedge}} t)^*}{{}^{\mathsf{true}}s \wedge {}^{\mathsf{true}}t}$$

where $s, t \in \mathsf{Prop}$ *is on the path*

Equivalence

There are two equivalence notions in this theory: intensional identity and extensional equivalence.

$t \cong_T s$ states that the terms t, s are extensionally equivalent in type T

- This equivalence predicate has to be type subscripted to indicate under which type t and s are judged extensionally equivalent.
- Extensional equivalence is represented in the language of terms by $t \hat{\cong}_T s$

Identity

There are two equivalence notions in this theory: intensional identity and extensional equivalence.

$t =_T s$ states that two terms are intensionally identical in type T.

- The rules for intensional identity are essentially those of the $\lambda\alpha\beta\eta$-calculus.
- It is necessary to type the intensional identity predicate to avoid semantic paradoxes.
- It is represented in the language of terms by $t \mathrel{\hat{=}}_T s$

Note: 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 $t \cong_T s \rightarrow t =_T s$.

Propositions and Predicates

Extensional equivalence of propositions and predicates correspond to some existing notions.

Propositions: in the case where two terms t, s are propositions ($t, s \in \mathsf{Prop}$), then

$$t \cong_{\mathsf{Prop}} s$$

corresponds to

$$t \leftrightarrow s$$

Predicates: in the case where two predicates of T are extensionally equivalent

$$t \cong_{(T \Longrightarrow \mathsf{Prop})} s$$

then t, s each hold of the same elements of T. Therefore

$$\forall x(x \in T \rightarrow ({}^{\mathsf{true}}t(x) \leftrightarrow {}^{\mathsf{true}}s(x)))$$

Separation Types

Separation types (or subtypes) can be used to formulate a treatment of anaphora.

- Add $\{x \in T : \varphi'\}$ to the types
- 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.

Term Representable Wffs

- First, we introduce syntactic sugar for typed quantification in the wffs.
  1. $\forall_T x\varphi =_{\mathsf{def}} \forall x(x \in T \to \varphi)$
  2. $\exists_T x\varphi =_{\mathsf{def}} \exists x(x \in T \wedge \varphi)$
- 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':
- $\varphi' ::= \alpha' \mid (\varphi' \wedge \psi') \mid (\varphi' \vee \psi') \mid (\varphi' \to \psi') \mid (\varphi' \leftrightarrow \psi')$
  $\mid (\forall_T x\varphi') \mid (\exists_T x\varphi) \mid {}^{\mathsf{true}}t$
  (atomic representable wffs) $\alpha' ::= \bot \mid t =_T s \mid t \cong_T s$

Translations of Wffs to Terms

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

1. $\lceil a \wedge b \rceil = \lceil a \rceil \hat{\wedge} \lceil b \rceil$
2. $\lceil a \vee b \rceil = \lceil a \rceil \hat{\vee} \lceil b \rceil$
3. $\lceil a \rightarrow b \rceil = \lceil a \rceil \hat{\rightarrow} \lceil b \rceil$
4. $\lceil a \leftrightarrow b \rceil = \lceil a \rceil \hat{\leftrightarrow} \lceil b \rceil$
5. $\lceil a \cong_T b \rceil = \lceil a \rceil \hat{\cong}_T \lceil b \rceil$
6. $\lceil a =_T b \rceil = \lceil a \rceil \hat{=}_T \lceil b \rceil$
7. $\lceil \bot \rceil = \hat{\bot}$
8. $\lceil {}^{\mathsf{true}}t \rceil = t$
9. $\lceil \forall_T x.a \rceil = \hat{\forall} x \epsilon T \lceil a \rceil$
10. $\lceil \exists_T x.a \rceil = \hat{\exists} x \epsilon T \lceil a \rceil$

Separation Types Revisited

- Now we can express separation types as $\{x \in S.\varphi'\}$, which can be taken to be sugar for $\{x \epsilon S.\lceil \varphi' \rceil\}$.
- The following theorem is an immediate consequence of the recursive definition of representable wffs and their term representations.
- **Theorem** (Representability) $\lceil \varphi' \rceil \in \mathsf{Prop}$ for all representable wffs φ', and furthermore ${}^{\mathsf{true}}\lceil \varphi' \rceil \leftrightarrow \varphi'$.

Comprehension

Usually comprehension can be derived from SP and UT (universal type). We must forgo UT to avoid paradoxes, and define comprehension independently. The same arguments apply as for SP concerning representability.

- Add $\{x : \varphi'\}$ to the types.
- COMP: $z \in \{x : \varphi\} \leftrightarrow \varphi[z/x]$
- Given that COMP = SP + UT, where UT is the Universal Type $\Delta = \{x : x \mathrel{\hat{=}} x\}$, we would derive a paradox if $=$ was not typed.
- This is because in PTCT Prop is a type. So rr, where $r = \lambda x.\hat{\exists} y \in (\Delta \Longrightarrow \mathsf{Prop})[x \mathrel{\hat{=}} y \mathrel{\hat{\wedge}} \hat{\sim} xy]$ 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.

Polymorphic Types

- Enrich the language of types to include type variables X, and the wffs to include quantification over types $\forall X\varphi, \exists X\varphi$.
- Add $\Pi X.T$ to the language of types, governed by the following axiom:

  PM $f \in \Pi X.T \leftrightarrow \forall X(f \in T)$
- Note that PM is impredicative (the type quantification ranges over the types that are being defined).
- This can be avoided by adding a language of Kinds (K) to PTCT:

  Kinds $K ::= T \mid \Pi X.K$

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

  where X only ranges over types.

Applications of Polymorphism

Polymorphic types can be of use in analysing the types of some verbs, and the types of conjunctive expressions.

- Such polymorphic types represent infinite intersections of types. For 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 be given the type $\Pi X.(X \Longrightarrow \mathsf{Prop})$.
- NL conjunction 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.
- It appears that the weaker formulation of PM′ is sufficient for NL. >>

Model Theory for PTCT

Sketch of a model

- A model of the untyped λ-calculus (e.g. General Functional Models), $\mathcal{D} = \langle D, [D \to D], \Phi, \Psi \rangle$ where D is isomorphic to $[D \to D]$
  1. D is a non-empty set,
  2. $[D \to D]$ is some class of functions from D to D,
  3. $\Phi : D \to [D \to D]$,
  4. $\Psi : [D \to D] \to D$,
  5. $\Psi(\Phi(d)) = d$ for all $d \in D$

  (Meyer 1982).
- Interpret the types as terms in D that correspond to subsets of D.

A Model for PTCT

- A model of PTCT is $\mathcal{M} = \langle \mathcal{D}, \mathsf{T}, \mathsf{P}, \mathsf{B}, \mathcal{B}, \mathcal{T}, \mathcal{K} \rangle$, where
  1. $\mathcal{D}$ is a model of the λ-calculus
  2. $\mathsf{T} : D \rightarrow \{0, 1\}$ models the truth predicate $^{\mathsf{true}}$
  3. $\mathsf{P} \subset D$ models the class of propositions
  4. $\mathsf{B} \subset D$ models the class of basic individuals
  5. $\mathcal{B}(\mathsf{B})$ is a set of sets whose elements partition B into equivalence classes of individuals
  6. $\mathcal{T} \subset \mathcal{K}$ models the class of types
  7. $\mathcal{T} \subset D$ models the term representation of types

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

>>

Kinds and Types

- Kinds and types could 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 $^{\cup} S$.

Kinds and Types

The types in the model, and the interpretation of PTCT's types are specified by rules like those in the following examples.
Here, τ is an assignment from type variables to elements of $\mathcal{T}$, and $\lceil\cdot\rceil$ is as previously defined, and Φ is as defined in the model of the λ-calculus.

1. For all $t \in \mathcal{K}$, ${}^{\cup}t \subset D$
2. $[\![X]\!]_{g,\tau} = \tau(X) \in \mathcal{T}$
3. If $[\![S]\!]_{g,\tau}, [\![U]\!]_{g,\tau} \in \mathcal{T}$, then $[\![S \Longrightarrow U]\!]_{g,\tau} \in \mathcal{T}$
4. If $[\![S]\!]_{g,\tau} \in \mathcal{T}$ and $[\![\lceil\varphi'\rceil]\!]_{g,\tau} \in$ P then $[\![\{x \in S.\varphi'\}]\!]_{g,\tau} \in \mathcal{T}$
5. ${}^{\cup}[\![S \Longrightarrow U]\!]_{g,\tau} = \{d \in D : \forall e \in {}^{\cup}[\![S]\!]_{g,\tau}.\Phi(d)e \in {}^{\cup}[\![U]\!]_{g,\tau}\}$
6. ${}^{\cup}[\![\{x \in S.\varphi'\}]\!]_{g,\tau} = \{d \in {}^{\cup}[\![S]\!]_{g,\tau}.\mathcal{M}_{g[d/x],\tau} \models \varphi'\}$

Propositions

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

1. If $[\![t]\!]_{g,\tau} \in$ P and $[\![s]\!]_{g,\tau} \in$ P, then $[\![t \mathbin{\hat{\wedge}} s]\!]_{g,\tau} \in$ P.
2. If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, and $[\![t]\!]_{g[d/x],\tau} \in$ P for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$, then $[\![\hat{\forall} x \epsilon S.t]\!]_{g,\tau} \in$ P.
3. If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, then $[\![t \mathbin{\hat{\cong}}_S s]\!]_{g,\tau} \in$ P iff $[\![s]\!]_{g,\tau}, [\![t]\!]_{g,\tau} \in {}^{\cup}[\![S]\!]_{g,\tau}$.
4. If $[\![S]\!]_{g,\tau} \in \mathcal{T}$, then $[\![t \mathbin{\hat{=}}_S s]\!]_{g,\tau} \in$ P iff $[\![s]\!]_{g,\tau}, [\![t]\!]_{g,\tau} \in {}^{\cup}[\![S]\!]_{g,\tau}$.

Truth

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

1. If $[\![t]\!]_{g,\tau} \in \mathsf{P}$ and $[\![s]\!]_{g,\tau} \in \mathsf{P}$, then $\mathsf{T}([\![t \mathbin{\hat{\wedge}} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![t]\!]_{g,\tau}) = 1$ and $\mathsf{T}([\![s]\!]_{g,\tau}) = 1$.
2. 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 $d \in {}^{\cup}[\![S]\!]_{g,\tau}$.
3. (a) If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in \mathsf{B}$ then $\mathsf{T}([\![t \mathbin{\hat{\cong}}_{\mathsf{B}} s]\!]_{g,\tau}) = 1$ iff there is an A such that $A \in \mathcal{B}(\mathsf{B})$ and $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau}$ are elements of A.

   (b) If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in \mathsf{P}$ then $\mathsf{T}([\![t \mathbin{\hat{\cong}}_{\mathsf{Prop}} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![t]\!]_{g,\tau}) = \mathsf{T}([\![s]\!]_{g,\tau})$.

   (c) If $[\![t]\!]_{g,\tau}, [\![s]\!]_{g,\tau} \in [\![S \Longrightarrow U]\!]_{g,\tau}$, where $[\![S]\!]_{g,\tau}, [\![U]\!]_{g,\tau} \in \mathcal{T}$ then $\mathsf{T}([\![t \mathbin{\hat{\cong}}_{(S \Longrightarrow U)} s]\!]_{g,\tau}) = 1$ iff $\mathsf{T}([\![tx \mathbin{\hat{\cong}}_{U} sx]\!]_{g[d/x],\tau}) = 1$ for all $d \in {}^{\cup}[\![S]\!]_{g,\tau}$.

Well Formed Formulæ

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

- $\mathcal{M}_{g,\tau} \models s =_T t$ 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 $[\![t]\!]_{g,\tau} \in {}^{\cup}[\![T]\!]_{g,\tau}$ [a]

  $\mathcal{M}_{g,\tau} \models \varphi \wedge \psi$ 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 $S \in \mathcal{T}$, $\mathcal{M}_{g,\tau[S/X]} \models \varphi$

Soundness and Completeness

We have sketches of the proofs of soundness and completeness.

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.

Soundess of PTCT If PTCT $\vdash \phi$, then ϕ is valid. (Proof by induction on the downward correctness of the tableau rules of PTCT.)

Completeness of PTCT If ϕ is valid, then PTCT $\vdash \phi$. (Proof by induction on the upward correctness of the tableau rules.)

Intensional Number Theory

We can add an intensional number theory to PTCT

Terms $0 \mid succ \mid pred \mid add \mid mult \mid \hat{most} \mid \mid \cdot \mid_B$

Types Num

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

Axioms for Num The usual Peano axioms, adapted to PTCT

Axioms for $<_{\mathsf{Num}}$

$$y \in \mathsf{Num} \rightarrow 0 <_{\mathsf{Num}} succ(y)$$

$$x \in \mathsf{Num} \rightarrow x \not<_{\mathsf{Num}} 0$$

$$x \in \mathsf{Num} \wedge y \in \mathsf{Num} \rightarrow (succ(x) <_{\mathsf{Num}} succ(y) \leftrightarrow x <_{\mathsf{Num}} y)$$

Proportional Quantifiers

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

Cardinality of properties $|p|_B$

1. $p \in (B \Longrightarrow \mathsf{Prop}) \wedge \sim \exists x(x \in B \wedge {}^{\mathsf{true}}px) \rightarrow$
   $|p|_B \cong_{\mathsf{Num}} 0$
2. $p \in (B \Longrightarrow \mathsf{Prop}) \wedge b \in B \wedge {}^{\mathsf{true}}pb \rightarrow$
   $|p|_B \cong_{\mathsf{Num}} add(|p - \{x.x \mathrel{\hat{=}}_B b\}|_B)(succ(0))$

Analysis of $most(p)(q)$

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

Anaphora and PTCT

Cardinality and separation types allow us to implement an alternative account of anaphora, for example, as described by Lappin (1989), and Lappin and Francez (1994). This is an alternative to Ranta's (1994) type-theoretic approach.

The account can deal with the following range of phenomena:

- Quantified NP Antecedents.
- Names and Existential NPs.
- Donkey Anaphora.
- Existential Readings.
- Proportional Donkey Sentences.

Computational Semantics

Relevance of PTCT to Computational Semantics

- 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 would be preferable, provided it is sufficiently expressive for the relevant domain.
- Basic first-order logic by itself does not provide the features that are usually thought to be required for natural language semantics. It is insufficiently expressive for this purpose.

Computational Semantics

Relevance of PTCT to Computational Semantics

- PTCT is a first-order system whose expressiveness is designed for natural language semantics. It 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 of PTCT (without the intensional number theory), and have proved the soundness and completeness of the rules relatiave to the model theory.

Conclusions

- We have constructed a first-order fine-grained intensional logic with flexible Curry typing, PTCT, for the semantic representation of natural languages.
- PTCT contains general 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.
- We have developed a model theory for PTCT using extensional models for the untyped λ-calculus enriched with interpretations of Curry types.
- Unlike some alternative frameworks, this logic distinguishes among provably equivalent propositions without resorting to impossible worlds to sustain the distinction.

Conclusions

In addition:

- The incorporation of Curry typing into the logic allows us to sustain polymorphism.
- Separation types (subtypes) permit a us to develop a uniform type-theoretical account of pronominal anaphora with wide empirical coverage.

Future Work

- Refine the model theory for PTCT.
- Explore theorem proving for PTCT as a component of an implemented NL interpretation system.
- Study the extent to which higher-order intensional systems like FIL can be reduced to PTCT
- Extend the empirical coverage of PTCT to natural language semantic phenomena like ellipsis interpretation.