A Fine-Grained Intensional First-Order Logic with Flexible Curry Typing

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

University of Essex

KING'S College LONDON Founded 1829

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

>>

Functional Types and HOL

First Assumption: Functional Types and Higher-Order Logic

- Much work in formal and computational semantics has assumed that higher-order logic and type theory are necessary to achieve the expressive power required to represent the semantic properties of natural language.
- The functional types of a higher-order system are required to express generalized quantifiers and modifiers.
- Montague (1974), Gallin (1975), Barwise and Cooper (1981), and Keenan and Stavi (1986) develop higher-order systems of semantic representation based on this assumption.

Formal Power

- If we are interested in a computationally viable theory, then we should be concerned about the formal power of the theory.
- In Fox, Lappin, and Pollard (2002) we define a Fine-Grained Intensional Logic (FIL), which is a higher-order theory with Church typing.
- However, we can achieve similar expressive potential (i.e. fine-grained intensionality with a rich system of types) without going higher-order.
- Property Theory with Curry Typing (PTCT) contains (i) a language of terms, (ii) a language of types, and (iii) a language of wffs in which the truth conditions are specified.
- We use the language of terms to represent intensions, the language of types to add typing to the language of terms, and the language of wffs to express type judgements for terms and specify truth-conditions for propositional terms.

Intensions and Possible Worlds

Second Assumption: Intensions are Functions from Worlds to Extensions

- The view that characterizes intensions as functions from possible worlds (situations) to extensions has been influential at least since Carnap (1947).
- It achieves detailed formal expression in Montague (1974).
- As has been frequently noted, this treatment of intensions yields a theory of meaning that is not sufficiently fine-grained.
- It entails that logically equivalent expressions are cointensional and so intersubstitutable in all contexts.

Equivalence and Identity

Equivalence and Identity of Intension

1. Every prime number is divisible only by itself and 1.

   $\Leftrightarrow$

2. If $A \subseteq B$ and $B \subseteq A$, then $A = B$.
3. John believes that every prime number is divisible only by itself and 1.

   $\neq$

4. John believes that if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

Impossible Worlds

Fine-Grained Theories of Meaning with Impossible Worlds

- Most attempts to construct fine-grained intensional theories have followed one of two approaches.
- The first involves positing impossible worlds or situations in which at least some classically valid formulas do not hold.
- This enlargement of the set of worlds/situations permits distinctions to be made between expressions that are equivalent across the set of possible worlds/situations.
- Variants of this view are presented in Barwise and Etchemendy (1990), Barwise (1997), Muskens (1995), and Gregory (2002).
- A serious disadvantage of this approach is that it requires us to characterize meaning in epistemic terms, where impossible worlds represent states of partial or imperfect knowledge (as in Barwise (1997)).

Alternatives

Fine-Grained Theories of Meaning with an Unspecified Proof Theory

- On the second approach, a hyperintensional model theory is constructed which appears to permit distinctions among logically equivalent expressions.
- However, a proof theory is not specified.
- As a result it is is not clear how the system prevents intensional identity from collapsing into logical equivalence.
- This difficulty arises with the theories proposed in Thomason (1980), Cresswell (1985), Landman (1986), and Larson and Segal (1995).

Bealer's Intensional Logic

- One exception to approaches with impossible worlds, or with a poorly specified proof-theory, is Bealer's (1980) first-order intensional logic.
- This logic contains an abstraction operator that forms names of intensional entities.
- Its model theory posits a domain of intensions (properties, k-ary relations, and propositions).
- its proof theory (on one version) is designed to prevent the reduction of intensional identity to logical equivalence.
- Possible worlds and situations play no role in the theory.
- Bealer's logic does not contain types beyond those of classical first-order logic.
- Therefore, it lacks the expressive power required for NL semantic representation.

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

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

Truth conditions for Propositions

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

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. *[We won't go into the details here.]*

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 using Polymorphic *Kinds*. Quantification can range over types, but not the Polymorphic Kinds themselves.

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.

>>

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

Summary of subtopics

- Anaphora and Type Theory
- Limitations of Ranta's Analysis
- Alternative Account of Anaphora
- Quantified NP Antecedents
- Names and Existential NPs
- Donkey Anaphora
- Existential Readings
- Proportional Donkey Sentences

Anaphora and Type Theory

A Type-Theoretical Approach to Anaphora

- Ranta (1994) uses Martin-Löf Type Theory (MLTT) to represent cases of pronominal anaphora which have been previously used to motivate DRT (Kamp, 1981); Kamp and Reyle, 1993) and dynamic logic (Groenendik and Stokhof, 1990 and 1991).
- He represents
  Every man who owns a donkey beats it
  as (intuitionistic) universal quantification over products pairs.
- $\Pi z : (\Sigma x : man)(\Sigma y : donkey)(x\ owns\ y))(p(z)\ beats\ p(q(z))$
- z is a variable over product pairs, and p and q are left and right projections, respectively, on the product pair. △

Anaphora and Type Theory

- Ranta's compositional rules for constructing these projections generate dependent type conditions on z such that
  $p(z) : man$,
  $q(z) : (\Sigma y : donkey)(p(z)\ owns\ y)$,
  $p(q(z)) : donkey$,
  $q(q(z)) : (p(z)\ owns\ p(q(z)))$.
- This representation entails that for every pair containing a man and a donkey that he owns, the man beats the donkey. △

Limitations of Ranta's Analysis

Limitations of Ranta's Analysis of Donkey Anaphora

- As Ranta acknowledges, by adopting the DRT treatment of donkey anaphora as (intuitionistic) universal quantification over pairs his analysis inherits the proportion problem (Heim, 1990; Kadmon, 1990).
- It provides the wrong results for cases like *"Most men who own a donkey, beat it."*
- On the pair quantification view this sentence is true in a model in which 10 men own donkeys, 9 men own a donkey each and do not beat it, and 1 men owns 10 donkeys and beats all of them.
- This analysis also does not capture the existential reading of sentences like *"Every person who had a quarter in his/her pocket put it in the parking meter."* (Pelletier and Schubert, 1989). △

Limitations of Ranta's Analysis

Limitations of Ranta's Analysis of Donkey Anaphora

- Ranta treats pronominal discourse anaphora like *"A man arrived. He sang."* as (intuitionistic) existential quantification over a product variable.

$$\Sigma z : (\Sigma x : man)(x\ arrived))(p(z)\ sang)$$

  where $p(z) : man$.

- It is not obvious how this approach can be extended to plural anaphora as in *"Every man arrived. They sang."* △

Alternative Account of Anaphora

An Alternative Type-Theoretic Account of Pronominal Anaphora

- We assume that all quantified NPs are represented as cardinality relations on the model of our treatment of *"most"* in PTCT.
- *"Every student sang."*
- $|\{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} sang'(x)\}|_B$
  $\cong_{Num} |\{x \in B.^{\mathsf{true}} student'(x)\}|_B$
- Pronouns are represented as appropriately typed free variables.
- If the free pronoun is within the scope of a set forming operator that specifies a subtype and it meets the same typing constraints as the variable bound by the set operator, the variable can be interpreted as bound by the operator through substitution under α identity. △

Alternative Account of Anaphora

An Alternative Type-Theoretic Account of Pronominal Anaphora

- This interpretation yields the bound reading of the pronoun.

  "Every man loves his mother."

  $$|\{x \in B.^{\mathsf{true}} man'(x) \wedge {}^{\mathsf{true}} love'(x, mother\text{-}of'(y))\}|_B$$
  $$\cong_{Num} |\{x \in B.^{\mathsf{true}} man'(x)\}|_B$$
  $$\Rightarrow$$
  $$|\{x \in B.^{\mathsf{true}} man'(x) \wedge {}^{\mathsf{true}} love'(x, mother\text{-}of'(x))\}|_B$$
  $$\cong_{Num} |\{x \in B.^{\mathsf{true}} man'(x)\}|_B$$

- Representations of this kind are generated by compositional semantic operations as described in (for example) Lappin (1989), and Lappin and Francez (1994).

△

Quantified NP Antecedents

Unbound Pronouns with Quantified NP Antecedents

- When the pronoun is interpreted as dependent upon an NP which does not bind it, we represent the pronoun variable as constrained by a dependent type obtained from the predicative part of the antecedent clause representation.
- In the default case, the pronoun variable is bound by a universal quantifier in the language of wffs.
- *"Every student arrived."*
  $|\{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\}|_B$
  $\cong_{Num} |\{x \in B.^{\mathsf{true}} student'(x)\}|_B$
- *"They sang."*
  $\forall y \in B.({}^{\mathsf{true}} sang'(y))$
  $(y \in \{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\})$ △

Names and Existential NPs

Proper Name and Existentially Quantified NP Antecedents

- *"John arrived."*

  $^{\text{true}}arrived'(john)$

  - *"He sang."*

    $\forall y \in B.(^{\text{true}}sang'(y))$

    $(y \in \{x \in B.x = john\})$

- *"Some man arrived."*

  $|\{x \in B.^{\text{true}}man'(x) \wedge {}^{\text{true}}arrived'(x) \wedge {}^{\text{true}}\phi(x)\}|_B >_{Num} 0$

  - *"He sang."*

    $\forall y \in B.(^{\text{true}}sang'(y))$

    $(y \in \{x \in B.^{\text{true}}man'(x) \wedge {}^{\text{true}}arrived'(x) \wedge {}^{\text{true}}\phi(x)\})$

  - ϕ is a predicate that is specified in context and uniquely identifies a man who arrived in that context.

△

Donkey Anaphora in PTCT

- *"Every man who owns a donkey beats it."*

$$|\{x \in B.^{\mathsf{true}} man'(x) \wedge (|\{y \in B.^{\mathsf{true}} own'(x,y) \wedge {}^{\mathsf{true}} donkey'(y)\}|_B >_{Num} 0) \wedge \forall z({}^{\mathsf{true}} beat'(x,z))\}|_B$$

$$\cong_{Num}$$

$$|\{x \in B.^{\mathsf{true}} man'(x) \wedge (|\{y \in B.^{\mathsf{true}} own'(x,y) \wedge {}^{\mathsf{true}} donkey'(y)\}|_B >_{Num} 0)\}|_B$$

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

- The type constraint on the pair (x, z) could be formulated in terms of a a curried type which applies in succession to each variable.
- The representation asserts that every man who owns at least one donkey beats all of the donkeys that he owns. △

Existential Readings

Existential Readings of Donkey Sentences

- The existential reading of a donkey sentence can be obtained by binding the variable representing the pronoun by an existential quantifier.
- *"Every person who had a quarter put it in a parking meter."*

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

$$\cong_{Num} |\{x \in B.^{\mathsf{true}} person'(x) \wedge (|\{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}|_B >_{Num} 0)\}|_B$$

$$((x,z) \in \{(y,w) \in B \otimes B.^{\mathsf{true}} had'(y,w) \wedge {}^{\mathsf{true}} quarter'(w)\})$$

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

Proportional Donkey Sentences

- The proportionality problem does not arise on our account.
- *"Most"* is represented as a cardinality relation (generalized quantifier) in which quantification is over the elements of the set corresponding to the subject restriction rather than over pairs.
- *"Most men who own a donkey beat it."*

$$|\{x \in B.{}^{\mathsf{true}}man'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0) \wedge \forall z \neg({}^{\mathsf{true}}beat'(x,z))\}|_B <_{\mathsf{Num}} |\{x \in B.{}^{\mathsf{true}}man'(x) \wedge (|\{y \in B.{}^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0) \wedge \forall z({}^{\mathsf{true}}beat'(x,z))\}|_B$$

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

- This representation states that most men who own a donkey beat all of the donkeys they own, and so it is false in the model which causes problems for the universal quantification over pairs analysis. △

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.