Curry Typing, Polymorphism, and Fine-Grained Intensionality*

Shalom Lappin

King's College London shalom.lappin@kcl.ac.uk

A draft chapter for the Wiley-Blackwell *Handbook of Contemporary Semantics — second edition*, edited by Shalom Lappin and Chris Fox. This draft formatted on 5th June 2014.

1 Introduction

Two of the central elements of Montague semantics (Montague, 1974b) are (i) a higher-order intensional logic IL that incorporates Church's simple theory of types (STT, (Church, 1940)), and (ii) a model theory that uses a Kripke frame semantics. The latter gives a modalized treatment of intensions based on Carnap's view of an intension as a function from possible worlds to denotations (Carnap, 1947).[1] These formal devices have continued to play an influential role in semantic theory even in many of the revisions of Montague semantics and the alternative semantic theories that have emerged in the past thirty years.

Montague's framework remains a seminal achievement in formal semantic theory. However, several of its foundational assumptions encounter serious problems when this framework is extended beyond the small fragment of English that Montague formalized. In this chapter I will examine several of these problems, and I will consider alternatives to Montague's type theory and his characterization of intensions in order to deal with these problems.

In Section 2, I give a brief summary of the architecture of IL, and take up some of the difficulties that it raises. Section 3 describes Property Theory with Curry Typing (PTCT, (Fox & Lappin, 2005, 2010)), a first-order semantic representation system that uses Curry typing with weak polymorphism. In Section 4, I discuss how PTCT provides a formal solution to the problem of fine-grained intensionality through its typed treatment of identity vs. equivalence. I then extend this solution to a computational account of intensional difference which accounts for intensions without using possible worlds. I argue that worlds are not effectively representable, and so it is necessary to develop a cognitively viable intensional semantics on foundations that do not rely on them. Section 5 presents some programmatic ideas on how one could move beyond classical categorial semantic theories to a probabilistic system that accommodates the pervasive gradience of semantic properties. Such an approach also provides a framework for addressing the nature of semantic

* The main ideas in this chapter were presented in my NASSLLI 2012 course Alternative Paradigms of Computational Semantics, which, in turn, developed out of many years of fruitful joint work with Chris Fox. I am grateful to the participants in this course for stimulating feedback on these ideas, and to Chris for wonderful cooperation, that has shaped my thinking on the foundational questions of semantic theory. I would also like to thank Robin Cooper, Simon Dobnik, Jan van Eijck, Jonathan Ginzburg, Noah Goodman, Staffan Larsson, Peter Sutton, and Dag Westerståhl for very helpful discussion of some of the issues addressed in this chapter.

[1] More precisely Montague (1973) uses translation of natural language into IL to induce a model theoretic interpretation of the expressions of NL. Montague (1974a) applies rules of model theoretic interpretation directly to the typed expressions of English. My concern here is with the type and model theories common to both versions of this approach.

learning. Finally, Section 6 states conclusions and suggests some directions for future work on the issues considered in this chapter.

2 Higher-Order Intensional Logic

2.1 The Syntax and Semantics of IL

Montague (1973) interprets the expressions of a fragment of English by specifying a formal syntax for this fragment and mapping its syntactic categories into the types of IL. This mapping is required to be a homomorphism, which assigns all elements of a category C to the same type τ. The homomorphism is many-to-one, with several distinct categories going to the same type. So, for example, common nouns and verb phrases are assigned the type $\langle\langle s, e\rangle, t\rangle$. This type is a function from individual intensions (functions from worlds to entities of type e) to truth-values (type t).

The types of IL are defined recusively as follows:

Basic Types:

1a. t (truth-values)
b. e (individual entities)

Exponential Types:

2. If a, b are types, then $\langle a, b\rangle$ is a type.
(($\langle a, b\rangle$ corresponds to $a \to b$ in more standard type theoretic notation.)

Intensional Types:

3. If a is a type, then $\langle s, a\rangle$ is a type (the type of the intension of a).

The set ME (meaningful expressions) of well-formed expressions of IL is defined recursively as a family of typed expressions.

***ME* of IL**

1a. Every variable of type a is in ME_a.
b. Every constant of type a is in ME_a.
2. If $\alpha \in ME_a$ and u is a variable in ME_b, then $\lambda u\alpha \in ME_{\langle b,a\rangle}$.
3. If $\alpha \in ME_{\langle a,b\rangle}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.
4. If $\alpha, \beta \in ME_a$, then $\alpha = \beta \in ME_t$.
5. If $\phi, \psi \in ME_t$, then so are
   a. $\neg\phi$
   b. $\phi \vee \psi$
   c. $\phi \wedge \psi$
   d. $\phi \to \psi$
   e. $\phi \leftrightarrow \psi$.
6. If $\phi \in ME_t$ and u is a variable in ME_a, then $\forall u\phi$ and $\exists u\phi \in ME_t$.
7. If $\phi \in ME_t$, then $\diamond\phi \in ME_t$.
8. If $\alpha \in ME_a$, then $^\wedge\alpha \in ME_{\langle s,a\rangle}$.
9. If $\alpha \in ME_{\langle s,a\rangle}$, then $^\vee\alpha \in ME_a$.

A model $M = \langle A, W, F\rangle$ for IL is an ordered triple such that (i) A is the (non-empty) domain of individuals, (ii) W is the (non-empty) set of possible worlds, and (iii) for each type a of IL, F is a function from the non-logical constants of ME_a to interpretations of these constants. For each constant $c \in ME_a, F(c)$ is a function $f_c : W \to D_a$, where D_a is the domain of possible denotations for expressions of type a.

Each non-logical constant of type a has a domain of possible denotations D_a. The set of these domains is defined recursively as follows.

The Domain D of Denotations for the Types of IL

1. $D_e = A$
2. $D_t = \{0, 1\}$
3. $D_{\langle a,b\rangle} = {D_b}^{D_a}$ (where ${D_b}^{D_a}$ is the set of functions from D_a to D_b)
4. $D_{\langle s,a\rangle} = {D_a}^{W}$

Let g be an assignment function such that for any variable $x \in a$ of IL, $g(x) \in D_a$. For a model M of IL, let $w \in W$. The parameterized interpretation function $\|\alpha\|^{M,w,g}$ determines the value of an expression $\alpha \in ME$ relative to M, w, and g.

A set of semantic rules provides a recursive definition of $\|\alpha\|^{M,w,g}$, where the recursion specifies the semantic values of expressions for each type in IL.

1. If $\phi \in ME_t$, then $\|\phi\|^{M,w} = t$ iff $\|\phi\|^{M,w,g} = t$ for all g.
2. If $\phi \in ME_t$, then ϕ is valid in IL iff $\|\phi\|^{M,w} = t$ for all M and w.
3. If Γ is a set of $\phi_i \in ME_t$ and $\psi \in ME_t$, then $\Gamma \models \psi$ iff for every M, w where all $\phi_i \in \Gamma$ are such that $\|\phi_i\|^{M,w} = t$, $\|\psi\|^{M,w} = t$.

2.2 Generalized Quantifiers and Modification in IL

Noun phrases in natural language correspond to generalized quantifiers (GQs) in IL. Abstracting away from intensions in IL for the moment, these are expressions of the type $\langle\langle e, t\rangle, t\rangle$, which denote sets of properties.

(1)a. John sings.
b. John $\Rightarrow \lambda P_{\langle e,t\rangle}[P(john_{\langle e\rangle})]_{\langle\langle e,t\rangle,t\rangle}$
c. sings $\Rightarrow \lambda x_{\langle e\rangle}[sing'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}$
d. John sings $\Rightarrow \lambda P[P(john)]_{\langle\langle e,t\rangle,t\rangle}(\lambda x[sings'(x)]_{\langle e,t\rangle}) =$
e. $\lambda x[sings'(x)]_{\langle e,t\rangle}(john_{\langle e\rangle}) =$
f. $sing'(john)_{\langle t\rangle}$

(2)a. Every girl programs.
b. every $\Rightarrow \lambda Q_{\langle e,t\rangle}\lambda P_{\langle e,t\rangle}[\forall x[Q(x) \to P(x)]_{\langle t\rangle}]_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$
c. girl $\Rightarrow \lambda x_{\langle e\rangle}[girl'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}$
d. every girl $\Rightarrow \lambda Q_{\langle e,t\rangle}\lambda P_{\langle e,t\rangle}[\forall x[Q(x) \to P(x)]_{\langle t\rangle}]_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$
$(\lambda x_{\langle e\rangle}[girl'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}) =$

e. $\lambda P_{\langle e,t\rangle}[\forall x[girl'(x) \rightarrow P(x)]_{\langle t\rangle}]_{\langle\langle e,t\rangle,t\rangle}$
f. programs $\Rightarrow \lambda x_{\langle e\rangle}[program'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}$
e. Every girl programs $\Rightarrow \lambda P_{\langle e,t\rangle}[\forall x[girl'(x) \rightarrow P(x)]_{\langle t\rangle}]_{\langle\langle e,t\rangle,t\rangle}$ $(\lambda x_{\langle e\rangle}[program'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}) =$
g. $\forall x[girl'(x) \rightarrow program'(x)]_{\langle t\rangle}$

Montague did not represent higher-order GQs like *most* in IL.[2] However, if we add a type n for natural numbers and a cardinality predicate *card* of type $\langle\langle e,t\rangle,n\rangle$ (a function from properties to numbers), then we can represent these quantifiers.

(3)a. Most students sing.
b. most $\Rightarrow \lambda Q_{\langle e,t\rangle}[\lambda P_{\langle e,t\rangle}[card_{\langle\langle e,t\rangle,n\rangle}(\lambda x[Q(x)\wedge P(x)]) > card_{\langle\langle e,t\rangle,n\rangle}(\lambda x[Q(x)\wedge \neg P(x)])]]_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$
c. most students $\Rightarrow \lambda P_{\langle e,t\rangle}[card(\lambda x[student'(x) \wedge P(x)]) > card(\lambda x[student'(x) \wedge \neg P(x)])]_{\langle\langle e,t\rangle,t\rangle}$
d. most students sing $\Rightarrow \lambda P_{\langle e,t\rangle}[card(\lambda x[student'(x) \wedge P(x)]) > card(\lambda x[student'(x) \wedge \neg P(x)])]_{\langle\langle e,t\rangle,t\rangle}(\lambda x_{\langle e\rangle}[sing'_{\langle e,t\rangle}(x)]_{\langle e,t\rangle}) =$
e. $card(\lambda x[student'(x) \wedge sing'(x)]) > card(\lambda x[student'(x) \wedge \neg sing'(x)])_{\langle t\rangle}$

Reinstating intensions, modifiers (adjectives and adverbs) are of type $\langle\langle s,\tau\rangle,\tau\rangle$, functions from intensions of type τ to extensions of type τ. So, for example, nominal adjectives and verb phrase adverbs are of type $\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle$. These are functions from properties of individuals in intension (functions from worlds to sets of individuals) to extensions of such properties (sets of individuals). This type allows for a unified treatment of intensional and extensional (intersective) modifiers.

(4)a. Mary bought a green car.
b. $\exists x((green'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}({}^\wedge car'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(x) \wedge bought'(mary,x))$

(5)a. Rosa dances beautifully.
b. $(beautifully'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}({}^\wedge dance'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(rosa)$

(6)a. Mary bought a toy car.
b. $\exists x((toy'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}({}^\wedge car'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(x) \wedge bought'(mary,x))$

(7)a. Rosa allegedly dances
b. $(allegedly'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}({}^\wedge dance'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(rosa)$

Montague employs meaning postulates to sustain inferences like the following for extensional modifiers.

(8)a. $\exists x((green'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}({}^\wedge car'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(x) \wedge bought'(mary,x)) \Rightarrow$
b. $\exists x(car'(x) \wedge green'(x) \wedge bought'(mary,x))$

[2] See (Barwise & Cooper, 1981; Keenan & Westerståhl, 1997; Westerståhl, this volume) on GQs in general, and higher-order GQs in particular.

(9)a. $(beautifully'_{\langle\langle s,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}(^\wedge dance'_{\langle s,\langle e,t\rangle\rangle}))_{\langle e,t\rangle}(rosa) \Rightarrow$
b. $dance'(rosa)$

The meaning postulates will not support such inferences for intensional modifiers, like those in (6) and (7).

2.3 Problems with IL

The first problem to note is that IL does not accommodate fine-grained intensionality. Montague's characterization of intensions as functions from worlds (indices of worlds and times) to denotations reduces intensional identity to equivalence of denotation across possible worlds. Logically equivalent expressions are semantically indistinguishable. This is too coarse a criterion for semantic identity. Logical equivalence is not a sufficient condition for intersubstitutability in all contexts.

(10)a. Every prime number is divisible only by itself and 1. $\Leftrightarrow$
b. If $A \subseteq B$ and $B \subseteq A$, then $A = B$.

(11)a. John believes that every prime number is divisible only by itself and 1. $\not\Leftrightarrow$
b. John believes that if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

The second problem is that by adopting Church's STT, Montague commits IL to an inflexible and relatively impoverished type system. In natural language verbs, coordinating expressions, and other function denoting terms apply to arguments of distinct types, which is a kind of polymorphism of semantic type. Attempting to capture this fact within IL has resulted in the addition of awkward type changing and type coercion operations. IL requires that all non-basic types are exponential, and so additional types (product types, comprehension types, subtypes, and lists) are not available. They could, of course, be added, which would involve a significant extension of the type system and the model theory.

A third, and related difficulty, is that lexical semantic relations are expressed only through meaning postulates, which are, effectively, constraints on possible models. In a richer type system, at least some of these elements of meaning could be expressed through subtypes (such as intensional vs. extensional modifiers). Given Montague's strict homomorphism condition on the mapping of natural language syntactic categories to the types of IL, subtyping would require the proliferation of additional syntactic subcategories corresponding to subtypes.

Fourth, IL has significant expressive power by virtue of being higher-order. But there is a price to pay for this power. In general the set of theorems of a higher-order logic is not recursively enumerable, and so its proof theory is incomplete. Constructing a computationally viable higher-order theorem prover for a subpart of a system like IL that would be adequate for natural language semantics is a difficult task.

Fifth, gradience in semantic properties (such as predication and entailment) is pervasive throughout all aspects of interpretation, and it is plausible to regard it as intrinsic to the semantics of natural language. Montague semantics excludes gradient semantic properties, and it can only accommodate gradience as a performance factor.

Finally, it is not clear how a representation system like IL could be the target of efficient semantic learning that is largely data and observation driven. On the other hand, there does not appear to be convincing psychological evidence to suggest that IL encodes universal constraints (prior learning biases) which define the hypothesis class for semantic learning. So IL does not offer a psychologically plausible class of representations for the semantic part of language acquisition.

2.4 A Representability Problem for Possible Worlds

The last problem for IL cited in Section 2.3 raises a more general issue for the model theory that Montague uses. Can we represent possible worlds in a computationally tractable and cognitively plausible way? Montague regards the study of syntax, semantics, and pragmatics as a branch of mathematics, rather than of psychology. From this perspective, representing worlds is not an issue. We can take them to be unanalysed elements of a set, as in the Kripke frame semantics (Kripke, 1963) that Montague uses.

However, if we wish to account for semantic learning and the procedures through which speakers/hearers actually compute interpretations for expressions in their language, then it is a central problem. Tractability in learning and representation are basic requirements for cognitively viable theories of syntax, morphology, and phonology. These conditions also apply to psychologically plausible theories of interpretation.

Carnap (1947); Kripke (1959); Jonsson & Tarski (1951) characterize worlds as maximally consistent sets of propositions. Fox *et al.* (2002); Fox & Lappin (2005); Pollard (2008) generalize this approach to define worlds as untrafilters in the prelattice of propositions, and they take the truth of a proposition, relative to a world, to be its membership in such an ultrafilter. If the logic of propositions is higher-order, then the problem of determining membership in such a set is not complete. If the logic is classically first-order, then the membership problem is complete, but undecidable.

Let's radically simplify the representation of worlds by limiting ourselves to propositional logic, and assuming that we can generate a maximally consistent set of propositions from a single finite proposition in Conjunctive Normal Form (CNF, a conjunction of disjunctions) by extending this proposition through the addition of new conjuncts. It is not clear what (finite) set of rules or procedures we could use to decide which conjuncts to add to this CNF proposition in order to generate a full description of a world in a systematic way. It is not obvious at what point the conjunction will constitute a complete description of the world (Rescher, 1999). Notice the contrast with syntax, morphology, and phonology here. In these systems we can generate an infinite set of well formed representations from a finite vocabulary of basic elements, through a finite set of precisely defined combinatorial operations.

Consistency reduces to the satisfiability of a CNF proposition P. We extend P by adding new conjucts to it to give P', and testing P' for satisfiability. All the propositions that P entails must be added to it, and all the the propositions with which P is inconsistent must be excluded, in order to obtain the maximal consistent set of propositions that describe a world. This is the kSAT problem, which consists in attempting to identify a set of truth-value assignments to the literals (propositional variables or their negations) of a formula P in CNF that satisfies P. There are are well known complexity results for different types of kSAT problems (Papadimitriou, 1995). If the number k of literals for each conjunct in P is such that $3 \leq k$, then the satisfiability problem for P is, in the general case, NP-complete, and so it is computationally intractable.

Given that we have to include all of the entailments of P in the maximal consistent set that represents a world w, and exclude all of the sentences with which P is incompatible, we have no basis for restricting the cardinality of k to less than 3 for the extension P' of P that we use to encode w. Therefore, even given the (radically) simplifying assumptions that we have made concerning the representation of a world with a finite extendable formula of propositional logic in CNF, verifying that this formula is consistent, at each point of its construction, is an intractable problem. It follows that individual worlds are not effectively representable, and, therefore, neither is the set of worlds.

Notice that the problem is not avoided by using a Kripke frame semantics (as in Montague (1974b)) in which worlds are simple elements of a set W. In a system of this kind a model is an ordered k-tuple $\langle D, W, F, R\rangle$, where D is the domain of objects, F is an interpretation function that assigns intensions to the constants of a language, and R is an accessibility relation on W. Intensions are functions from worlds to denotations of the appropriate type. Propositions are functions from worlds to truth-values, and so every $w_i \in W$ is in a one-to-one correspondence with the maximal set $Prop_{w_i}$ of propositions that are true at w_i. But then each w_i is identified by its corresponding set of maximal propositions, and the problem of representing w_i reduces to that of determining membership in $Prop_{w_i}$.

Some formal semantic theories have characterised modality and intensions in terms of the set of possible situations rather than the set of worlds (Heim, 1990; Lappin & Francez, 1994; Lappin, 2000; Kratzer, 2014). Possible situations are parts of worlds, and so they are not (necessarily) maximal. One might think, at first glance, that the non-maximality of situations allows us to avoid the problem of effective representability that arises for worlds.

In fact, positing the set of possible situations as the basis for an intensional semantics makes the problem considerably worse. If a world is a maximal consistent set of propositions, each situation in that world will be a subset of this maximal set. Each world w_i yields a power set $\mathcal{P}(w_i)$ of the set of propositions that defines w_i. As the maximal set of propositions that specify a w_i is infinite, the cardinality of $\mathcal{P}(w_i)$ is uncountably infinite (by Cantor's theorem for the cardinality of power sets). The set of possible situations is the union of $\mathcal{P}(w_i)$ (more accurately, of $\mathcal{P}(w_i) - \emptyset$) for all w_i. This set is not recursively enumerable.

It seems, then, that neither possible worlds nor possible situations are appropriate elements for a cognitively plausible theory of semantic learning and interpretation. We need to characterise both intensions and modality in terms that do not depend upon these objects in our semantic theory.

3 Property Theory with Curry Typing

3.1 The Untyped λ-Calculus, Curry Typing, and First-Order Logic

In Church's STT every expression of a language L is specified as the element of a type. Therefore its type is intrinsic to its status as an expression of L. By contrast, in Curry typing the expressions of a language are introduced independently of their types, and type membership statements are part of the language.[3] IL uses a strictly typed version of the λ-calculus, which requires that each expression be uniquely typed. Curry typing permits polymorphism in which the same expressions may inhabit several types.

Property Theory with Curry Typing uses the untyped λ-calculus as the combinatorial system for generating the terms of its representation language. The untyped λ-calculus is a "simple" calculus of substitutions that uses function application (the application of a function t to an argument t', $t(t')$) and abstraction (the formation of a function $\lambda x(t)$ from a term t) to derive normal forms from terms in the language. The terms in this calculus can be thought of as programs. In principle, all computer programs can be encoded as terms in the calculus. Executing a program consists in performing substitutions on terms. This combinatorial system is the basis for functional programming languages, like Lisp, Scheme, ML, Haskell, and Miranda.

PTCT factors the semantic representation language into three distinct components: (i) an untyped λ-calculus of terms for the combinatorial engine that generates representations of meaning, (ii) Curry-types corresponding to natural language semantic types, and (iii) a first-order logic of well-formed formulas for reasoning about the truth of propositions that terms belonging to the appropriate type represent. This federated approach allows for greater flexibility in coordinating the formal power and the expressive resources of the theory with the semantic properties of natural language.[4] The terms of untyped λ-calculus are intensions, and these are assigned Curry-types. The logic, which includes a truth predicate, determines an entailment relation among propositions, represented by propositional terms.

Because PTCT uses the untyped λ-calculus it is necessary to impose constraints on the interaction of the different components of the system to avoid paradoxes. These constraints consist in restrictions on the quantification over type variables, where this restricted quantification yields a weak form of polymorphism, and the exclusion of self-application for λ-terms (a function cannot take itself as an argument.)

[3] See (Turner, 1997) for a discussion of the distinction between Church and Curry typing.

[4] Fox & Lappin (2014) present a version of PTCT in which the mechanisms handled by these three distinct languages are encoded in a single typed predicate logic. This allows for a fully unified representation of the framework.

3.2 Syntax of PTCT

Term Language:

Logical Constants

$l ::= \hat{\sim} \,|\hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon T$

Terms:

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

The language of terms is the untyped λ-calculus, enriched with logical constants. It is used to *represent* the interpretations of natural language expressions, and so its expressions are the intensions of the system. It has no internal logic, but the logic is imposed "externally" through the typing and logic components. The identity criteria for terms are those of the λ-calculus, which are the α, β, and η conditions.

Types:

$T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S \mid X \mid \{x \in T.\varphi'\} \mid \{x.\varphi'\} \mid \mathit{\Pi} X.T \mid S \otimes T$

The type system includes propositional (Prop), functional ($T \Longrightarrow S$), separation ($\{x \in T.\varphi'\}$), comprehension ($\{x.\varphi'\}$), and product ($S \otimes T$) types, as well as restricted quantification over type variables ($\mathit{\Pi} X.T$). φ' is a term representable fragment of wff. The type quantification allows types to be polymorphic. The restriction on quantification over types consists in the requirement that type variables X range over non-polymorphic types. This constraint avoids impredicativity (self-application and the paradoxes that it generates). The languages of types and terms are combined with appropriate rules and axioms to produce a Curry-typed λ-calculus.

Well-Formed Formulas:

Atomic Wffs

$\alpha ::= (t =_T s) \mid t \in T \mid t \cong_T s \mid {}^{\mathsf{T}}(t)$

Wffs

$\varphi ::= \alpha \mid \sim\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi) \mid (\forall x \varphi) \mid (\exists x \varphi) \mid (\forall X \varphi) \mid (\exists X \varphi)$

The first-order language of wffs is used to formulate type judgements for terms, and truth conditions for those terms judged to be in Prop. If a term t represents a proposition, then $^{\mathsf{T}}(t)$ is a wff that denotes its truth conditions. The identity criteria of wffs are those of their truth conditions. Type variables X range only over term representable types (types that can be represented as terms in the term language).

It is important to distinguish between the notion of a proposition itself in the language of wffs, 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, $^{\mathsf{T}}(t)$, where $^{\mathsf{T}}(t)$ will be a true wff whenever the proposition represented by t is true, and 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 $^{\mathsf{T}}(t)$. Terms of type Prop have no intrinsic logic. Their identity criteria are those of the untyped λ-calculus.

3.3 A Proof Theory for PTCT

Proof rules and axioms govern the logical behaviour of PTCT. The connectives of the wffs have standard classical first-order behaviour. The usual rules of the untyped λ-calculus (α, β, and η equivalence) define the identity of terms $=_T$. The rules of the Curry-typed calculus, augmented by rules for those terms that represent propositions (Prop) specify the typing of λ-terms. Additional rules for the language of wffs that govern the truth conditions of terms in Prop (which represent propositions) give the truth conditions for propositions. The theory has a notion of extensional equivalence, $\cong_T$, which is distinct from intensional identity, $=_T$.

As an example of the principles of the proof theory consider the rules of inference for conjunction, both as it applies in the language of wffs ($\wedge$), and in the language of terms ($\hat{\wedge}$).[5]

1. The basic connectives of the wff

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge_i \quad \frac{\varphi \wedge \psi}{\varphi}\wedge_e \quad \frac{\varphi \wedge \psi}{\psi}\wedge_e$$

2. Typing rules for λ-terms

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

3. Truth conditions for Propositions

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

[5] In Fox & Lappin (2005) these principes are encoded as tableaux rules, but here they are given in the format of sequent calculus and inference rules for ease of presentation.

The proof theory defines identity and equivalence of terms, typing and type inference, and the rules of truth for the first-order logic of wffs. It is the primary vehicle for determining the interpretation of expressions in each of the three components of PTCT, and the relations among these components. It is through the proof theory that a logic is imposed on the language of terms, specifically those terms that belong to the type Prop, and so represent propositions. The model theory, briefly sketched in Section 3.5, is designed to sustain the soundness and the completeness of the proof theory.

3.4 Polymorphism and Subtyping

The expressions of natural language frequently correspond to more than one type. So, for example, the same verb can take subjects of distinct types, as in (12), and a conjunction can be type heterogenous, as in (13).

(12)a. John finished early.
b. Voting finished early.
c. The concert finished early.

(13)a. Continuous functions and quarks have interesting formal properties.
b. Mary sees a connection between Bach concertos and fractals.

In a strictly typed representation language like IL it is necessary to assign each such expression to a distinct functional type. The weak polymorphism of PTCT allows the same expressions to belong to more than one type, as functional types can take arguments of distinct types. Type variables can be used for such arguments. A verb like *finish* can have the type $\Pi X.X \Longrightarrow \mathsf{Prop}$, permitting it to take type distinct subject arguments. Similarly, *and* can be assigned to the type $\Pi X.X \Longrightarrow X \Longrightarrow X$, which allows it to apply to different types of conjuncts. Therefore, PTCT, in contrast to IL, does not require type changing or type coercion rules to accommodate these expressions.

The typing component of PTCT includes separation types of the form

$$z \in \{x \in T : \varphi'\} \leftrightarrow (z \in T \wedge \varphi'[z/x]).$$

This allows us to define intersection, union and difference types. To ensure that the theory is first-order, separation types are required to be term representable, and so φ' must be term representable. To sustain this result Fox & Lappin (2005) define a term representable fragment of the language of wffs. The separation, comprehension, and product types of PTCT give it greater expressive power than IL by allowing for an enriched subtyping system. In this respect PTCT adopts a *rich* type system, similar in approach to the one that Type Theory with Records (TTR, (Cooper, 2012; Cooper & Ginzburg, this volume)) uses.

Fox & Lappin (2005) use separation types for dynamic treatments of anaphora and ellipsis. They characterise both anaphora and ellipsis resolution as a process of identifying the value of a type parameter with a subtype condition on an individual variable in PTCT. The content of the condition is recovered from the representation of the antecedent. For a discussion of pronominal anaphora see Kehler (this volume), and for an account of ellipsis see Kempson *et al.* (this volume).

Fox & Lappin (2010) use product types and the weak polymorphism of PTCT to generate *underspecified* scope representations for sentences containing quantified NPs. They define a family of functions $perms\_scope_k$ (where $k > 1$) that generate all $k!$ indexed permutation products of a k-ary indexed product term $\langle t_1, \ldots, t_k \rangle$ as part of the procedure for generating the set of possible scope readings of a sentence. This function is interpreted by a tree construction algorithm that produces the set of all possible permutations of scope-taking elements to which $perms\_scope_k$ applies.

More specifically, $perms\_scope_k$ applies to a k-ary product of scope-taking elements (by default, in the order in which they appear in the surface syntax) and a k-ary relation representing the core proposition as its arguments. The scope-taking elements and the core representation can be combined into a single product, for example as a pair consisting of the k-tuples of quantifiers as its first element and the core relation as its second. The elements of the k-tuple of scoping taking terms can be type heterogenous, and so the permutation function $perms\_scope_k$ is polymorphic. It applies to k-tuples (effectively lists) of different types, and for each such list it returns the $k!$-ary product of scoped readings in the lists as its value. When a k-tuple of quantifiers is permuted, the λ-operators that bind the quantified argument positions in the core relation are effectively permuted in the same order as the quantifiers in the k-tuple. This correspondence is necessary to preserve the connection between each GQ and its argument position in the core relation across scope permutations.

Consider the example

(14) Every student knows a professor.

The GQs interpreting the subject NP, the object NP and the core relation are given as the PTCT terms

(15)a. $Q_1 = \lambda P \hat{\forall} x \epsilon \mathsf{e}(\mathsf{student}'(x) \hat{\rightarrow} P(x))$
b. $Q_2 = \lambda Q \hat{\exists} y \epsilon \mathsf{e}(\mathsf{professor}'(y) \hat{\wedge} Q(y))$
c. $\lambda uv[\mathsf{knows}' uv]$, where $\mathsf{knows}' \epsilon\ \Pi X.X \Longrightarrow \mathsf{e} \Longrightarrow \mathsf{Prop}$
d. $perms\_scope_2 \langle \langle Q_1, Q_2 \rangle, \lambda uv.\mathsf{knows}' uv \rangle$

Notice the fact that knows' is assigned to a functional type that is polymorphic in its initial (object NP) position permits it to takes objects of distinct types, as in (16)a, where the value of X in the type of knows' is n rather than e, and (15)b, where it is Prop.

(16)a. Every student knows a prime number.
b. Every student knows that the assignment is due tomorrow.

The permutations of the quantifiers and the core representation that $perms\_scope_k$ produces for (14) are given by the following.

(17) $perms\_scope_2\langle\langle Q_1, Q_2\rangle, \lambda uv.\mathsf{knows}' uv\rangle =$
$\langle\langle\langle Q_1, Q_2\rangle, \lambda uv[\mathsf{knows}' uv]\rangle, \langle\langle Q_2, Q_1\rangle, \lambda vu.\mathsf{knows}' uv\rangle\rangle$

Applying relation-reduction to each of the representations of the scope orderings gives a pair of propositional terms corresponding to the two readings.

(18) $(\hat{\forall} x\epsilon\mathsf{e}(\mathsf{student}'(x) \hat{\rightarrow} \hat{\exists} y\epsilon\mathsf{e}(\mathsf{professor}'(y) \hat{\wedge} \mathsf{knows}'(x, y))),$
$\hat{\exists} y\epsilon\mathsf{e}(\mathsf{student}'(y) \hat{\wedge} \hat{\forall} x\epsilon\mathsf{e}(\mathsf{professor}'(x) \hat{\rightarrow} \mathsf{knows}'(x, y))))$

An underspecified expression of the form (19) is a λ-term of PTCT.

(19) $perms\_scope_k\langle\langle Q_1, Q_2, ..., Q_k\rangle, \lambda u_1, ..., u_k.\phi'\rangle$

It is possible to specify constraints on possible scope relations among the elements of $\langle Q_1, Q_2, ..., Q_k\rangle$ by applying additional λ-terms as filters to the $perms\_scope_k$ term.

3.5 Semantics of PTCT

Following Meyer (1982) we define a model of the untyped λ-calculus (e.g. General Functional Models) as $\mathcal{D} = \langle D, [D \rightarrow D], \Phi, \Psi\rangle$ where D is isomorphic to $[D \rightarrow D]$ and

Ia. D is a non-empty set,
b. $[D \rightarrow D]$ is some class of functions from D to D,
c. $\Phi : D \rightarrow [D \rightarrow D]$,
d. $\Psi : [D \rightarrow D] \rightarrow D$, and
e. $\Psi(\Phi(d)) = d$ for all $d \in D$.

We interpret the types of PTCT as terms in D that correspond to subsets of D.

A model of PTCT is $\mathcal{M} = \langle \mathcal{D}, \mathsf{T}, \mathsf{P}, \mathsf{B}, \mathcal{B}, \mathcal{T}', \mathcal{T}\rangle$, where

IIa. $\mathcal{D}$ is a model of the λ-calculus.
b. $\mathsf{T} : D \rightarrow \{0, 1\}$ models the truth predicate $^{\mathsf{T}}$.
c. $\mathsf{P} \subset D$ models the class of propositions.
d. $\mathsf{B} \subset D$ models the class of basic individuals.
e. $\mathcal{B}(\mathsf{B})$ is a set of sets whose elements partition B into equivalence classes of individuals.
f. $\mathcal{T}' \subset \mathcal{T}$ models the term representation of non-polymorphic types.
g. $\mathcal{T} \subset D$ models the types.

Sufficient structural constraints are imposed on T, P and $\mathcal{T}$ to validate the rules of the proof theory for PTCT.

Although PTCT achieves much of the expressive richness of a higher-order type theory, Fox & Lappin (2005) demonstrate that it remains first-order in its formal power. They show that its tableau proof theory is sound and complete. The soundness proof proceeds by showing the downward correctness of tableaux through induction on the tableaux rules. The completeness proof proceeds by establishing the upward correctness of tableaux through induction on the rules. Unlike IL, the theorems of PTCT are recursively enumerable.

4 Fine-Grained Intensionality

4.1 Distinguishing Intensional Identity and Provable Equivalence

As noted in Section 3.3, there are two equivalence notions in PTCT: intensional identity and extensional equivalence, which can apply to expressions of the same type. The proposition $t \cong_T s$ states that the terms t, s are extensionally equivalent in type T. In the case where two terms t, s are propositions ($t, s \in \mathsf{Prop}$), then $t \cong_{\mathsf{Prop}} s$ corresponds to $t \leftrightarrow s$. If 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, that is $\forall x(x \in T \rightarrow (^{\mathsf{T}}t(x) \leftrightarrow {}^{\mathsf{T}}s(x)))$..

The proposition $t =_T s$ states that two terms are intensionally identical in type T. As noted, the rules for intensional identity are essentially those of the $\lambda\alpha\beta\eta$-calculus. We are able to derive $t =_T s \rightarrow t \cong_T s$ for all types inhabited by t, s, but not $t \cong_T s \rightarrow t =_T s$. Therefore PTCT avoids the reduction of provable equivalence to intensional identity. Two terms can be provably equivalent by the proof theory, but not identical. In this case, they remain intensionally distinct.

PTCT allows us to sustain both the logical equivalence of (10)a and (10)b, and the non-equivalence of (11)a and (11)b. The former are provably equivalent, but they correspond to non-identical propositional terms in PTCT.

(10)a. Every prime number is divisible only by itself and 1. $\Leftrightarrow$
b. If $A \subseteq B$ and $B \subseteq A$, then $A = B$.

(11)a. John believes that every prime number is divisible only by itself and 1. $\not\Leftrightarrow$
b. John believes that if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

The proof theory of PTCT induces a prelattice on the terms in Prop. In this prelattice the members of an equivalence class of mutually entailing propositional terms (terms that encode mutually entailing propositions) are non-identical and so correspond to distinct propositions.[6] While this result achieves the formal property of fine-grained intensionality, it does not, in itself, explain what intensional non-identity consists in beyond the fact that two distinct expressions in the language of terms are identified with different intensions. This leaves us with what we can describe as a problem of ineffability. Intensional difference is posited as (a certain kind of) inscriptional distinctness in the λ-calculus of terms, but this reduction does not offer a substantive explanation of the semantic properties that ground the distinction. Intensional difference remains ineffable.

[6] (Fox *et al.*, 2002; Fox & Lappin, 2005; Pollard, 2008) construct higher-order hyperintensional semantic systems using an extended version of Church's SST and a prelattice of propositions in which the entailment relation is a preorder.

4.2 A Computational Account of Intensional Difference

It is possible to characterize the distinction between intensional identity and provable equivalence computationally by invoking the contrast between operational and denotational semantics in programming language. Two simple examples illustrate this contrast.[7]

For the first example take the function $predecessorSet(x)$, which maps an object in an ordered set into the set of its predecessors. So, for example, if $x \in \{0, 1, 2, 3, 4, 5\}$, then, with numeric ordering, $predecessorSet(x) = PredSet_x \subset \{0, 1, 2, 3, 4, 5\}$ such that $\forall y \in Pred_x(y < x)$.

It is possible to define (at least) two variants of this function, $predSet_a$ and $predSet_b$, that are denotationally equivalent but operationally distinct. $predSet_a$ is specified directly in terms of an immediate predecessor relation, while $predSet_b$ depends upon a successor relation.

(20)a. $predecessorSet_a(x) = PredSet_x$, if
$\forall y(y \in PredSet_x \rightarrow predecessor(y, x))$.

b. $predecessor(y, x)$ if
$predecessor_{immediate}(y, x)$; else
  $predecessor(y, x)$ if
  $predecessor_{immediate}(y, z)$, and
  $predecessor(z, x)$.

(21)a. $predecessorSet_b(x) = PredSet_x$, if
$\forall y(y \in PredSet_x \rightarrow successor(x, y))$.

b. $successor(x, y)$ if
$successor_{immediate}(x, y)$; else
  $successor(x, y)$ if
  $successor_{immediate}(x, z)$, and
  $successor(z, y)$.

The second example involves functions $g : \Sigma^* \rightarrow \{1, 0\}$, that is functions from Σ^*, the set of strings formed from the alphabet of a language, to the Boolean values 1 and 0, where $g(s) = 1$ if $s \in L$, and 0 otherwise. Such a function recognises all and only the strings in a language defined on the alphabet Σ. Let g_{csg1} be defined by the Definite Clause Grammar (DCG) in (22), and g_{csg2} by the DCG in (23).[8]

[7] For earlier discussions of the main ideas in this Section see Lappin (2012, 2013).

[8] See Pereira & Shieber (1987) for an explanation of Definite Clause Grammars. The DCG in (22) is from Gazdar & Mellish (1989). Matt Purver and I constructed the DCG in (23) as a Prolog programming exercise for a computational linguistics course that I gave in the Computer Science Department at King's College London in 2002.

In these examples sequences starting with a capital (A, Bn, etc.) are non-terminal symbols, S is the start or initial symbol, which covers a full string in the language, and lower case letters are terminal symbols.

(22) $S \rightarrow [a],\ S(i).$
$S(I) \rightarrow [a],\ S(i(I)).$
$S(I) \rightarrow Bn(I),\ Cn(I).$
$Bn(i(I)) \rightarrow [b],\ Bn(I).$
$Bn(i) \rightarrow [b].$
$Cn(i(I)) \rightarrow [c],\ Cn(I).$
$Cn(i) \rightarrow [c].$

(23) $S \rightarrow A(I),\ B(I),\ C(I).$
$A(i) \rightarrow [a].$
$A(i(I)) \rightarrow [a],\ A(I).$
$B(i) \rightarrow [b].$
$B(i(I)) \rightarrow [b],\ B(I).$
$C(i) \rightarrow [c].$
$C(i(I)) \rightarrow [c],\ C(I).$

Both these DCGs define the same context-sensitive language

$$\{a^n b^n c^n |\ 1 \leq n\},$$

the language whose strings consist of n occurrences of a, followed by n bs, and then n cs. The number of as, bs, and cs match in all strings. Each DCG uses a counting argument I for a non-terminal symbol to build up a stack of indices i that gives the successive number of occurrences of as, bs, and cs in a string. But the grammar in (22) counts from the bottom up, adding an i for each non-terminal that the recognizer encounters. By contrast the grammar in (23) imposes the requirement that the three stacks for the non-terminals A, B, and C be identical, and then it computes the indices top down. The two grammars are computationally distinct, and using each of them to recognize a string can produce different sequences of operations, of different lengths and relative efficiency. Therefore, g_{csg1} and g_{csg2} are operationally distinct, but denotationally equivalent. They compute the same string set through different sets of procedures.

Recall that the terms of PTCT are λ-expressions that encode computable functions. We have identified these with the intensions of words and phrases in a natural language. Given the distinction between denotational and operational meaning we can now interpret the non-identity of terms in the representation language as an operational difference in the functions that these terms express. But a class of such terms can still be provably equivalent in the sense that they yield the same values for the same arguments by virtue of the specifications of the functions that they correspond to. This provides a straightforward account of fine-grained intensionality in PTCT which avoids taking intensional difference as ineffable.

Muskens (2005) suggests a similar approach to hyperintensionality. He identifies the intension of an expression with an algorithm for determining its extension.[9] There are two major points of difference between Musken's theory and the one proposed here. First, he embeds his account in a logic programming approach, which he seems to take as integral to his explanation of hyperintensionality, while I have developed my analysis in a functional programming framework. This is, in fact, not an issue of principle. The same algorithm can be formulated in any programming language. So, for example, the definitions of $predSet_a$ and $predSet_b$ correspond to two Horn clause definitions in Prolog for variant predecessor predicates, `predecessorA(Y, X)` and `predecessorB(Y, X)`.

(24)
```
predecessorA(Y, X) :- predecessorImmediate(Y, X).
predecessorA(Z, X) :-
predecessorImmediate(Y, X),
predecessorA(Y, Z).
```

(25)
```
predecessorB(Y, X) :- successor(X, Y).
successor(X, Y) :- successorImmediate(X, Y).
successor(X, Z) :-
successorImmediate(X, Y),
successor(Y, Z).
```

Similarly, the DCGs in (22) and (23) that we used to define g_{csg1} and g_{csg2}, respectively, are (close to) Prolog executable code.

However, the functional programming formulation of the operational view of fine-grained intensionality follows straightforwardly from PTCT, where the untyped λ-calculus generates the intensional terms of the semantic representation language, and these encode computable functions. As we have seen PTCT also offers rich Curry typing with weak polymorphism, and a logic of wffs for reasoning about truth and entailment, within a first-order system. The fact that it implies the operational account of intensional difference without further stipulation renders it attractive as a framework for developing computational treatments of natural language semantic properties.

The second, more substantive point of difference concerns the role of modality (possible worlds) in characterizing intensions. Muskens develops his hyperintensional semantics on the basis of Thomason (1980)'s Intentional Logic. In this logic Thomason proposes a domain of propositions as intensional objects, where the set of propositions is recursively defined with intensional connectives and quantifiers. He posits a homomorphism that maps propositions (and their constituents) to their extensions, and he constrains this homo-

[9] Duží *et al.* (2010) also adopt an operational view of hyperintensionality within Tichý (1988)'s Transparent Intensional Logic. However, the computational details of their account are left largely unspecified. Both Muskens (2005) and Duží *et al.* (2010) regard their respective proposals as working out Frege's notion that an intension is a rule for identifying the denotation of an expression.

morphism with several meaning postulates that restrict this mapping.[10] Muskens modifies and extends Thomason's logic by specifying a homomorphism between the intensional expressions of the logic and their extensions across the set of possible worlds. Propositions are mapped to the set of worlds in which they are true. As the homomorphism can be many-to-one, distinct propositions can receive the same truth-value across worlds.[11]

By contrast, PTCT adopts Thomason's non-modal strategy of mapping propositions to truth-values. It does this by using a truth predicate to form a wff $^{\mathsf{T}}(\phi)$ to assert the truth of the proposition that the term $\phi \in \mathsf{Prop}$ represents. Therefore, like Thomason's Intentional Logic, PTCT de-modalizes intensions. This is a positive result. It is not clear why, on the fine grained view, possible worlds must be essentially connected with the specification of intensions.

Moschovakis (2006) suggests an operational treatment of meaning within the framework of the typed λ-calculus. He constructs a language L^{λ}_{ar} as an extension of Gallin (1975)'s Ty_2. He specifies acyclic recursive procedures for reducing the terms of L^{λ}_{ar} to unique cannonical forms. Moschovakis identifies the meaning (*referential intension*) of a term in this language with the *abstract algorithm* for computing its denotation.

Moschovakis specifies a Kripke frame semantics for L^{λ}_{ar} which is a variant of Montague's possible worlds models (he refers to them as *Carnap states*). Carnap states are n-tuples of indices corresponding to worlds, times, speakers, and other parameters of context. Intensions are characterized as algorithmic procedures for determining the denotation of a term relative to a world and the other elements of such an n-tuple. The arguments that were brought against this view in Muskens' case apply with equal force here.

On both Musken's and Moschovakis' accounts, and the one proposed here, the content of an intension is the set of computational operations through which it determines its denotational value, where these need not make essential reference to possible worlds. In the case of a proposition, the denotation that it determines is a truth-value, rather than a truth-value relative to a world.

Worlds are not required for an adequate explanation of fine-grained intensionality. On the contrary, such an account must dispense with the original characterization of intensions as functions from worlds to extensions

[10] Fox & Lappin (2005) point out that Thomason's logic is problematic because it does not characterize the algebraic structure of the domain of propositions. It does not offer a proof theory that defines entailment for propositions, and so it leaves the relation between intentional identity and extensional equivalence crucially under determined.

[11] (Fox *et al.*, 2002; Fox & Lappin, 2005; Pollard, 2008) adopt a similar view for the fine-grained higher-order logics that they construct. They define worlds as untrafilters in the prelattice of propositions, and they take the truth of a proposition, relative to a world, to be its membership in such an ultrafilter. As entailment in the prelattice is defined by a preorder, distinct propositions can belong to the same set of ultrafilters.

in order to explain the persistence of intensional difference beyond provable equivalence. Therefore, the radically non-possible worlds view of fine-grained intensionality offered here provides the cleaner approach.

5 Probabilistic Semantics

In this Section I offer some brief programmatic speculations on the sorts of radical revisions of classical semantic theory that may be required in order to accommodate both gradience and learning.

5.1 Gradience and Semantic Learning

Formal semantic theories like IL and PTCT model both lexical and phrasal meaning through categorical rules and algebraic systems that cannot accommodate gradience effects. This approach is common to theories which sustain compositionality and those with employ underspecified representations. It effectively invokes the same strong version of the competence-performance distinction that categorical models of syntax assume. This view of linguistic knowledge has dominated linguistic theory for the past fifty years.

Gradient effects in representation are ubiquitous throughout linguistic and other cognitive domains. Any appeal to performance factors to explain gradience has no explanatory content unless it is supported by a precise account of how the interaction of competence and performance generates these effects in each case. By contrast, gradience is intrinsic to the formal models that information theoretic methods use to represent events and processes.

Lexically mediated relations like synonymy, antinomy, polysemy, and hyponymy are notoriously prone to clustering and overlap effects. They hold for pairs of expressions over a continuum of degrees $[0, 1]$, rather than Boolean values $\{1, 0\}$. Moreover, the denotations of major semantic types, like the predicates corresponding to Nouns, Adjective Phrases, and Verb Phrases, can rarely, if ever, be identified as sets with determinate membership.

It is also unclear how these representations could be learned from the primary linguistic data (PLD) of language acquisition. The abstract formal structures that they posit are not easily inferred from observable data. On the other hand, there does not seem to be much evidence that they correspond to biologically conditioned learning biases or categories of perception. Most work in formal learning for natural languages has focussed on syntax (grammar induction), morphology, and phonology. The problem of developing a plausible account of efficient learnability of appropriate target representations is as important for semantics as it is for other types of linguistic knowledge.

One way of accommodating both gradience and semantic learning is to abandon the categorical view of competence and adopt a probabilistic model of linguistic representation. Stochastic models assign gradient probability values to judgements. They are the target representations of probabilistic learning theories. There is a fair amount of evidence to suggest that language acquisition in general crucially relies on probabilistic learning (see (Clark & Lappin, 2011)). The case for a probabilistic approach to both representation and learning is at least as strong in semantics as it is in syntax, as well as in other parts of the grammar.[12]

5.2 Type Theory in Probabilistic Semantics: A Top-Down Approach

There are two obvious strategies for constructing a probabilistic semantics. On the top-down approach one sustains classical (categorical) type and model theories, and then specifies a function that assigns probability values to the possible worlds that the model provides. The probability value of a sentence relative to a model M is the sum of the probabilities of the worlds in which it is true. On the bottom-up approach one defines a probabilistic type theory and characterizes the probability value of a sentence as the output of the function that applies to the probabilistic semantic type judgements associated with its syntactic constituents.

In their proposal van Eijck & Lappin (2012) adopt the top-down strategy. They retain a classical type theory and the specification of intensions for each type as functions from worlds to extensions. They define a *probabilistic model* M as a tuple $\langle D, W, P\rangle$ with D a domain, W a set of worlds for that domain (predicate interpretations in that domain), and P a probability function over W, i.e., for all $w \in W$, $P(w) \in [0, 1]$, and $\sum_{w\in W} P(w) = 1$.

An interpretation of a language L in a $M = \langle D, W, P\rangle$ is given in terms of the standard notion $w \models \phi$, as follows:

$$[\![\phi]\!]^M := \sum P(w)\ s.t\ w \in W \wedge w \models \phi$$

This definition of a model entails that

$$[\![\neg\phi]\!]^M = 1 - [\![\phi]\!]^M.$$

Also, if

$$\phi \models \neg\psi, \text{ i.e., if } W_\phi \cap W_\psi = \emptyset,$$

then

$$[\![\phi \vee \psi]\!]^M = \textstyle\sum_{w\in W_{\phi\vee\psi}} P(w) = \sum_{w\in W_\phi} P(w) + \sum_{w\in W_\psi} P(w) = [\![\phi]\!]^M + [\![\psi]\!]^M,$$

as required by the axioms of Kolmogorov (1950)'s probability calculus.

This theory has several attractive properties. It retains a classical type system and model theory to compute the value of a sentence in a world, and

[12] See Manning (2003), Cohen (2003), and some of the other papers in that collection for detailed arguments in support of a probabilistic approach to linguistic representation.

then it applies a standard probability calculus to compute the probability of a sentence. Therefore, it uses well understood formal systems at both levels of representation. It also proposes the outline of a theory of semantic learning for simple one-place predicate classifiers, where this could be generalized to a richer representation language.

However, it suffers from the disadvantage that it requires probabilities to be assigned to entire worlds in the model, with sentences receiving probability values derivatively from these assignments. As we saw in Section 2.4, worlds are not tractably representable, and so this approach does not offer a cognitively plausible framework for developing a probabilistic semantics.

5.3 Type Theory in Probabilistic Semantics: A Bottom-Up Approach

A bottom-up model assigns probabilities to individual type judgements as classifier applications. The probability of a sentence is computed directly from the probabilities of its constituent types. This approach avoids the holism of the top-down view. It can be applied to rich type theories like PTCT or TTR, transforming them into gradient models of classification and predication. Such a probabilistic type theory would also offer the basis for an account of semantic learning in which individual classifiers are acquired probabilistically through observation driven Bayesian inference and update rules.

Cooper *et al.* (2014) propose a probabilistic version of TTR in which type judgements are assigned probability values. Central to standard formulations of rich type theories (for example, (Martin-Löf, 1984)) is the notion of a judgement $a : T$, that object a is of type T. Cooper *et al.* (2014) represent the probability of this judgement as $p(a : T)$. Their system (based on (Cooper, 2012)) includes the types of TTR, and equations for computing the probability values of judgements for each of these types.

Probability theorists working in AI often describe probability judgements as involving distributions over worlds. In fact, they tend to limit such judgements to a restricted set of outcomes or events, each of which corresponds to a partial world which is, effectively, a type of situation (Halpern, 2003; Goodman & Lassiter, this volume). A classic example of the reduction of worlds to situation types in probability theory is the estimation of the likelihood of heads vs tails in a series of coin tosses. Here the world is held constant except along the dimension of a binary choice between a particular set of possible outcomes. A slightly more complex case is the probability distribution for possible results of throwing a single die, which allows for six possibilities corresponding to each of its numbered faces. This restricted range of outcomes constitutes the sample space.

Cooper *et al.* (2014) make explicit the assumption, common to most probability theories used in AI, with clearly defined sample spaces, that probability is distributed over situation types (Barwise & Perry, 1983), rather than over sets of entire worlds, or the set of all possible situations. An Austinian proposition is a judgement that a situation is of a particular type, and they treat it as probabilistic. In fact, it expresses a subjective probability in that it encodes the belief of an agent concerning the likelihood that a situation is of that type. The core of an Austinian proposition is a type judgement of the form $s : T$, which states that a situation s is of type T. On their account this judgement is expressed probabilistically as $p(s : T) = r$, where $r \in [0,1]$. In the probabilistic type system that Cooper *et al.* (2014) propose situation types are intensional objects over which probability distributions are specified. This allows one to reason about the likelihood of alternative states of affairs without invoking possible worlds or possible situations.

The theory assumes only actual situations, and an intensional type system. Types are not sets of situations. They can be as large or as small as we require them to be. It is not necessary to represent the full set of situations (actual or possible) in order to acquire these types. They are classifiers of situations that can be learned through sampling of actual situations, and probabilistic reasoning concerning the types to which they belong. Therefore, the problems of tractable representation that we encountered with worlds, and with the set of possible situations do not arise in this framework.

Cooper *et al.* (2014) specify compositional rules for computing the probability values of Austinian propositions expressed by declarative sentences from the interpretations of their syntactic constituents. They also give an outline of a learning theory for naive Bayesian classifiers, where these support the acquisition of the basic types of the semantics. The type system provides the interface between observation based learning of perceptual classifiers and the combinatorial semantic procedures that generate the interpretations of complex expressions in natural language.

5.4 Uncertainty and Vagueness

Identifying the interpretations of sentences with their probability conditions permits us to model the uncertainty that characterizes some judgements concerning semantic relations and predications for a language. This sort of uncertainty accounts for an important element of gradience in semantic knowledge. It captures the defeasibility of implications, and the graded nature of synonymy (co-intensionality) and meaning intersection. However, it is unclear whether all species of semantic vagueness can be subsumed by the uncertainty that probabilistic judgements express. The vagueness that infects the application of degree modifiers (*ajar, open, tall, fast*) does not seem directly reducible to the uncertainty that probability measures.[13]

[13] See Lassiter (this volume) for a discussion of degree modifiers and Goodman & Lassiter (this volume) for a probabilistic treatment of vagueness and uncertainty in predication.

Edgington (1997) suggests that vagueness and uncertainty (Bayesian probability) share the same formal structure, but that they are distinct phenomena. She does not explain the apparent isomorphism between uncertainty and vagueness, and so it remains coincidental on her account. Lassiter (2011) seeks to reduce vagueness to probability, but at the cost of treating a vague predicate as ambiguous among an unbounded disjunction of semantically determinate variants over which probability is distributed. Neither Edgington nor Lassiter offer a type theory for computing the graded semantic value of a sentence. They also do not propose an account of semantic learning, nor do they consider its connection to vagueness.[14]

It may be possible to account for vagueness as an effect of semantic learning. Learners estimate the likelihood that competent speakers will assent to the application of a predicate (modifier) to a class of objects or events. In the absence of disambiguating evidence, a probability distribution over situation types for a range of predicate applications may survive learning to be incorporated into the model of mature speakers. Uncertainty in learning becomes vagueness in the intensions of predicates for mature speakers. On this view, vagueness is, then, the residue of probabilistic learning that survives into the mature representation systems of semantically competent speakers. No additional facts of the sort that drive semantic learning will decide among the gradient judgements that are left over once the learning process has converged on mature semantic representation. While gradience starts out as the product of a learner's probability distribution over judgements that competent speakers will accept a predication in a given context, it ends up as an intrinsic feature of the predication itself, due to the fact that the data of learning does not fully resolve the uncertainty of these judgements.

[14] Sutton (2013) presents detailed critical discussions of Edgington's and Lassiter's respective treatments of vagueness, as well as Williamson (1994)'s epistemicist view. He proposes an alternative probabilistic account that shares some of the main features of Cooper *et al.* (2014)'s approach.

6 Conclusions and Future Work

We have considered the architecture of classical Montague semantics, which has been the dominant influence in formal semantic theory for the past thirty-five years, and we have identified several foundational problems with it. Its reliance on a rigid and impoverished system of Church typing prevents it from handling the polymorphism and subtyping exhibited in natural language. Its modalized treatment of intensions does not allow for an appropriately fine-grained account of semantic difference. The categorical nature of Montague semantics excludes the gradience that is pervasive in natural language semantic properties. The representations that it generates do not to lend themselves to a plausible account of semantic learning. Its reliance on possible worlds to model intentionality prevents it from offering a cognitively viable semantic framework because worlds are not effectively representability.

By adopting a flexible Curry typing framework PTCT achieves a constrained polymorphism that is adequate for the type heterogenity of natural language expressions of functional type. It incorporates a rich system of subtyping and product types that covers some of the fine-grained intensionality that IL cannot express. PTCT uses the untyped λ-calculus to generate terms that correspond to intensions, and a proof theory that sustains the distinction between provable equivalence and intensional identity. In this way it formally models fined-grained intensional difference.

I propose a computational interpretation of this model on which an intension of a computable function is the sequence of procedures that it applies to compute its value. This permits us to express intensional non-identity as operational difference. Provable equivalence is sameness of denotational value for two functions, entailed by their respective specifications. This equivalence is compatible with the functions being operationally distinct. The computational interpretation offers anl account of fine-grained intensionality that avoids the problem of ineffability, and which does not rely on possible worlds.

Even representation languages with flexible, rich type theories, like PTCT, cannot accommodate gradience or address semantic learning, when they incorporate the same categorical algebraic methods as classical frameworks like IL. A promising way of approaching these problems is to reconstruct a rich type theory as a system of probabilistic judgements that situations are of a particular type. The gradience of a semantic property is derived from the semantic learning process. Learners assign probabilities to type judgements on the basis of the likelihood that semantically competent speakers endorse these judgements. Learning consists in converging on the representation of property types that mature speakers have achieved, and vagueness is the residue of unresolved uncertainty that survives semantic learning.

In order for this program for constructing a probabilistic type theory to be successful it must devise appropriate principles for computing the probability values of distinct type judgements, and an effective set of combinatorial procedures for deriving the probabilistic interpretations of complex expressions from their constituent types. Cooper *et al.* (2014) have made made a promising start on this task, but it remains the primary challenge for future research on probabilistic semantics.

References

Barwise, J. & R. Cooper (1981), Generalised quantifiers and natural language, *Linguistics and Philosophy* 4:159–219.

Barwise, Jon & John Perry (1983), *Situations and Attitudes*, Bradford Books, MIT Press, Cambridge, Mass.

Carnap, R. (1947), *Meaning and Necessity*, University of Chicago Press, Chicago.

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

Clark, A. & S. Lappin (2011), *Linguistic Nativism and the Poverty of the Stimulus*, Wiley-Blackwell, Chichester, West Sussex, and Malden, MA.

Cohen, A. (2003), Probabilistic approaches to semantics, in R. Bod, Jennifer Hay, & Stephanie Jannedy (eds.), *Probabilistic Linguistics*, MIT Press, Cambridge, MA, (343–380).

Cooper, R. (2012), Type theory and semantics in flux, in R. Kempson, T. Fernando, & N. Asher (eds.), *Philosophy of Linguistics*, Elsevier, North Holland, Oxford, Amsterdam, (271–323).

Cooper, R., S. Dobnik, S. Lappin, & S. Larsson (2014), A probabilistic rich type theory for semantic interpretation, in *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, Association of Computational Linguistics, Gothenburg, Sweden, (72–79).

Cooper, R. & J. Ginzburg (this volume), Type theory with records for natural language semantics, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Duží, M., B. Jespersen, & P. Materna (2010), *Procedural Semantics for Hyperintensional Logic*, Springer, Dordrecht, New York.

Edgington, Dorothy (1997), Vagueness by degrees, in R. Keefe & P. Smith (eds.), *Vagueness: A Reader*, MIT Press, Cambridge, MA, (294–316).

van Eijck, J. & S. Lappin (2012), Probabilistic semantics for natural language, in Z. Christoff, P. Galeazzi, N. Gierasimszuk, A. Marcoci, & S. Smets (eds.), *Logic and Interactive Rationality (LIRA), Volume 2*, ILLC, University of Amsterdam.

Fox, C. & S. Lappin (2005), *Foundations of Intensional Semantics*, Blackwell, Oxford.

Fox, C. & S. Lappin (2010), Expressiveness and complexity in underspecified semantics, *Linguistic Analysis, Festschrift for Joachim Lambek* 36:385–417.

Fox, C. & S. Lappin (2014), Type-theoretic logic with an operational account of intensionality, *Synthese* 191.

Fox, C., S. Lappin, & C. Pollard (2002), A higher-order, fine-grained logic for intensional semantics, in G. Alberti, K. Balough, & P. Dekker (eds.), *Proceedings of the Seventh Symposium for Logic and Language*, Pecs, Hungary, (37–46).

Gallin, D. (1975), *Intensional and Higher-Order Modal Logic*, North-Holland, Amsterdam.

Gazdar, G. & C. Mellish (1989), *Natural Language Processing in Prolog*, Addison-Wesley, Waltham, MA.

Goodman, N. & D. Lassiter (this volume), Probabilistic semantics and pragmatics: Uncertainty in language and thought, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Halpern, J. (2003), *Reasoning About Uncertainty*, MIT Press, Cambridge MA.

Heim, I. (1990), E-type pronouns and donkey anaphora,, *Linguistics and Philosophy* 13:137–177.

Jonsson, B. & A. Tarski (1951), Boolean algebras with operators, *American Journal of Mathematics* 73:891–939.

Keenan, E. & D. Westerståhl (1997), Generalized quantifiers in linguistics and logic, in J. van Benthem & A. ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, Amsterdam, (838–893).

Kehler, Andrew (this volume), Reference in discourse, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Kempson, Ruth, Ronnie Cann, Arash Eshghi, Eleni Gregoromichelaki, & Matthew Purver (this volume), Ellipsis, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Kolmogorov, A.N. (1950), *Foundations of Probability*, Chelsea Publishing, New York.

Kratzer, A. (2014), Situations in natural language semantics, in *Stanford Encylopedia of Philosophy*, Stanford University.

Kripke, S. (1959), A completeness theorem in modal logic, *Journal of Symbolic Logic* 24:1–14.

Kripke, S. (1963), Semantical considerations on modal logic, *Acta Philosophica Fennica* 16:83–89.

Lappin, S. (2000), An intensional parametric semantics for vague quantifiers, *Linguistics and Philosophy* 23:599–620.

Lappin, S. & N. Francez (1994), E-type pronouns, I-sums, and donkey anaphora, *Linguistics and Philosophy* 17:391–428.

Lappin, Shalom (2012), An operational approach to fine-grained intensionality, in Thomas Graf, Denis Paperno, Anna Szabolcsi, & Jos Tellings (eds.), *Theories of Everything: In Honor of Ed Keenan*, UCLA Working Papers in Linguistics 17, (Creative Commons Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/)).

Lappin, Shalom (2013), Intensions as computable functions, *Linguistic Issues in Language Technology* 9:1–12.

Lassiter, D. (this volume), Adjectival modification and gradation, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Lassiter, Daniel (2011), Vagueness as probabilistic linguistic knowledge, in R. Nouwen, R. van Rooij, U. Sauerland, & H-C Schmitz (eds.), *Vagueness in Communication*, Springer, Berlin, (127–150).

Manning, C. (2003), Probabilistic syntax, in R. Bod, Jennifer Hay, & Stephanie Jannedy (eds.), *Probabilistic Linguistics*, MIT Press, Cambridge, MA, (289–342).

Martin-Löf, Per (1984), *Intuitionistic Type Theory*, Bibliopolis, Naples.

Meyer, A. (1982), What is a model of the lambda calculus?, *Information and Control* 52:87–122.

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*, D. Reidel, Dordrecht, synthese Library.

Montague, R. (1974a), English as a formal language, in R. H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, CT/London, UK, (188–221).

Montague, R. (1974b), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, CT/London, UK, edited with an introduction by R. H. Thomason.

Moschovakis, Yiannis N. (2006), A logical calculus of meaning and synonymy, *Linguistics and Philosophy* 29:27–89.

Muskens, R. A. (2005), Sense and the computation of reference, *Linguistics and Philosophy* 28:473–504.

Papadimitriou, C. (1995), *Computational Complexity*, Addison-Wesley Publishing Co., Readin, MA.

Pereira, F. & S. Shieber (1987), *Prolog and Natural-Language Analysis*, CSLI, Stanford, CA.

Pollard, Carl (2008), Hyperintensions, *Journal of Logic and Computation* 18:257–282.

Rescher, Nicholas (1999), How many possible worlds are there?, *Philosophy and Phenomenological Research* 59(2):pp. 403–420.

Sutton, P. (2013), *Vagueness, Communication, and Semantic Information*, Ph.D. thesis, Department of Philosophy, King's College London.

Thomason, R. (1980), A model theory for propositional attitudes, *Linguistics and Philosophy* 4:47–70.

Tichý, P. (1988), *The Foundations of Frege's Logic*, De Gruyter, Berlin.

Turner, R. (1997), Types, in J. van Benthem & A. ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, (535–586).

Westerståhl, D. (this volume), Generalised quantifiers in natural language semantics, in S. Lappin & C. Fox (eds.), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford.

Williamson, Timothy (1994), *Vagueness*, Routledge, London.