King's Research Portal

DOI:

[10.1093/logcom/exi005](https://doi.org/10.1093/logcom/exi005)

Document Version

Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):

Ginzburg, J. (2005). Abstraction and Ontology: Questions as Propositional Abstracts in Type Theory with Records. *Journal of Logic and Computation*, 15(2), 113-130. <https://doi.org/10.1093/logcom/exi005>

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Abstraction and Ontology: questions as propositional abstracts in type theory with records

Jonathan Ginzburg*

Department of Computer Science

King's College, London

The Strand, London WC2R 2LS

ginzburg@dcs.kcl.ac.uk

<http://www.dcs.kcl.ac.uk/staff/ginzburg/papers>

March 3, 2005

Abstract

The paper develops a semantics for natural language interrogatives which identifies questions—the denotations of interrogatives—with propositional abstracts. The paper argues that a theory of *Questions as Propositional Abstracts* (QPA), is a simple, transparently implementable theory that has significant empirical coverage. However, until recently QPA has been abandoned in formal semantic treatments of questions, due to a number of significant problems QPA encountered when formulated within the type system of Montague Semantics. In recent work, Ginzburg and Sag provided a situation theoretic implementation of QPA that succeeded in overcoming certain of the original problems for QPA. However, Ginzburg and Sag's proposal relied on a special purpose account of λ -abstraction, raising the question to what extent QPA can be sustained using standard notions of abstraction. In this paper such doubts are allayed by implementing QPA in a version of Type Theory that provides record types. These latter allow one to develop notions of simultaneous/vacuous abstraction with restrictions and an ontology with various 'informational entities'. Moreover, the intrinsic polymorphism of this theory plays a crucial role in enabling the definition of a general type for questions, one of the main stumbling blocks for earlier versions of QPA.

1 Introduction

λ -abstracts are commonly used to model the meaning of interrogatives in NLP and AI work (for a recent example see Larsson, 2002).¹ Such a modelling was quite popular among NL formal semanticists in the late 1970s and early 1980s (see e.g. Scha, 1983; Hausser, 1983), but is, generally, avoided in contemporary work, given intrinsic problems originally discussed in (Groenendijk & Stokhof, 1984), recently reiterated in (Aloni & van Rooy, 2002). Ginzburg and Sag 2000 argued that a theory of questions as *propositional abstracts* (QPA) has a number of desirable consequences

*Earlier versions of this paper have been presented at the Barwise and Situation Semantics Workshop at Stanford, at the Lambda Calculus, Type Theory and Natural Language workshop at King's, and at a workshop on Questions in Amsterdam. I'd like to thank the audiences in all three workshops for helpful feedback. In particular, Robin Cooper for many helpful suggestions concerning Type Theory, also to Bengt Nordstrom and Aarne Ranta for some very useful advice. For comments and discussion I'd like to thank Paul Dekker, Tim Fernando, Jeroen Groenendijk, Shalom Lappin, Robert van Rooy, Remko Scha, Henk Zeevat. Two anonymous reviewers for the *Journal of Logic and Computation* made some valuable suggestions for rewriting the paper. This work is supported by grant number RES-000-23-0065 from the Economic and Social Research Council of the United Kingdom.

¹A variant on this strategy is to use open sentences. This avoids some of the problems a simple-minded view of questions as abstracts runs into, but acquires even more serious problems along the way. See Ginzburg & Sag, 2000 for discussion.

and, in particular, possesses significant advantages over the standard view of questions as encoders of exhaustive answerhood conditions (EACs). Moreover, they showed how in a situation theoretic ontology, modelled using tools from non-well-founded set theory (see Seligman & Moss, 1997), the problems for an abstract-based view could be overcome. Nonetheless, a key element of Ginzburg and Sag’s account is the use of *simultaneous abstraction*, an operation first introduced by Aczel & Lunnon, 1991. The uniform treatment of different kinds of interrogatives—including the definition of a semantic type *question* and answerhood—use this *ad hoc* notion of λ -abstraction. The question arises²—can the benefits of this account be preserved using standard notions of abstraction? In this paper, I will consider this question and offer a guarded affirmative response, using a version of Type Theory (see e.g. Betarte & Tasistro, 1998; Constable & Hickey, 2000; Coquand et al., 2003; Cooper, (forthcoming)).

2 Questions as (Propositional) Abstracts: pros and cons

The original motivation for identifying the meaning of (wh-)interrogatives with λ -abstracts comes from the phenomena of *short answers*—the nature of the question asked strongly influences the semantic type, and to some extent, the form of the short answer.

- (1) a. A: Who left?
B: Mo/No students./A friend of Jo’s.
- b. A: When did Bo leave?
B: Yesterday./At two.
- c. A: Why did Maire cross the road?
B: Because she thought no cars were passing.
- d. A: Who relies on whom (in this department)?
B: Bo on Mo./Some of my friends on each of her friends.

On a *questions as abstracts* view, we get the following basic denotations:

- (2) a. Who left $\mapsto \lambda x.leave(x)$
- b. When did Bo leave $\mapsto \lambda t.At(t, leave(b))$
- c. Why did Maire cross the road $\mapsto \lambda c.Cause(c, cross(m, r))$
- d. Who relies on whom $\mapsto \lambda x, y.rely(x, y)$.

Within many versions of typed λ -calculus, such as Montague’s Intensional Logic (IL) (Montague, 1974), this leads to several problematic consequences. These include:

Type multiplicity The type of any two interrogatives differing in arity (e.g. unary/binary/ternary *wh*-interrogative-sentences) or in the type of argument (e.g. adverbial vs. argumental *wh*-phrase) is distinct. This yields difficulties in defining Boolean operations uniformly on interrogatives.

Ontological collapse Whereas type multiplicity is a theory internal problem, a more general problem is ontological: within an IL setting, propositional abstracts are akin to properties, the denotata of (intransitive) verbs, adjectives, and common nouns, and so we are required to identify questions with the denotata of verbs, common nouns, and adjectives. This is an unpalatable ontological consequence.

- (3) a. left $\mapsto \lambda x.left(x)$

²Indeed was explicitly posed by several reviewers of Ginzburg & Sag, 2000, including Koenig, 2004; Nelken, (forthcoming).

- b. $\text{rely} \mapsto \lambda x, y. \text{rely}(x, y)$
- c. The question is who is happy—This question is unresolved.
 #The property of being happy is unresolved.
 #To be happy is unresolved.
- d. Can you tell me something about who is happy? $\nRightarrow$ Can you tell me something about being happy.
- e. Some man is happy.
 So we know that happiness and manfulness are not incompatible.
 #So we know that the question of who is happy and who is a man are not incompatible.

Polar Interrogatives An additional problem for a naive QPA approach is the fate of polar interrogatives. The distribution of polar interrogatives is essentially the same as that of *wh*-interrogatives, which suggests we would like to subsume them under a single semantic class. Moreover polar interrogatives, also exhibit a phenomenon akin to short answers:

- (4) A: Did Bo attend the meeting?
 B: Yes./Maybe./Probably/No.

The most obvious proposal would be to say that polar interrogatives constitute the limiting case where the number of uninstantiated variables is 0. In other words, a vacuous abstraction ensues. But this involves identifying polar questions with the queried proposition. The literature on interrogative semantics provides various arguments against such a move.³

3 QPA: why bother?

The previous section surveyed the original motivation for QPA. It also pointed out several technical and ontological problems it runs into. Given these problems one might wonder why bother developing a setting in which QPA is viable. The basic justification is straightforward: QPA is the simplest theory that simultaneously affords an adequate description of the most fundamental phenomena specific to questions—short answers, answerhood, and relations between questions such as dependence and influence. I suggest that it is the simplest theory because just about any theory of questions needs to associate an abstract-like object with interrogatives to explicate the resolution of short answers. Similarly, explicating answerhood involves a characterization that utilizes in some form an abstract-like object. If questions simply *are* abstracts, then the fact that abstracts need to be associated with interrogative utterances follows automatically. Finally, tractability: QPA yields a transparently implementable theory. In contrast, EAC theories that identify questions with *partitions* of propositions (e.g. Groenendijk & Stokhof, 1984 et seq.) are intractable on their most transparent implementation, as pointed out by Bos & Gabsdil, 2000.^{4,5}

4 Type Theory with Records as a semantic framework

In what follows I show how Type Theory provides a setting in which the problems for QPA noted above can be overcome and some new insights provided. All this without the need for an unorthodox abstraction operation akin to that used in Ginzburg & Sag, 2000. Before we can delve into interrogativity, we need to first sketch the basics of a version of Type Theory.

³See e.g. Ginzburg & Sag, 2000, p. 84.

⁴This is not to say such theories cannot be implemented—see e.g. Chieh Shan & ten Cate, 2002, but, arguably, this means the original semantic theory is significantly reconceptualized; in other words, the original theory is not to be construed cognitively or computationally.

⁵Apart from theoretical simplicity, Ginzburg and Sag 2000, section 7.3 provide a concrete argument for the view that questions are *propositional* abstracts, based on the distribution of *wh-in situ* phrases in English.

Ever since Ranta's pioneering work (Ranta, 1994), there has been interest in using constructive type theory (CTT) (often referred to as Martin-Löf Type Theory (MLTT)) as a framework for NL semantics (see e.g. Fernando, 2001; Krahmer & Piwek, 1999). It offers theoretically elegant accounts of anaphora and quantificational phenomena on which much semantic work has been invested in frameworks such as Discourse Representation Theory and Dynamic Semantics. The particular approach to CTT I will draw on extensively here is the Type Theory with Records framework developed in recent work by Robin Cooper, including Cooper, 1998, (forthcoming).

We face two obvious tasks: explicating what propositions are and integrating these in a structure that provides for abstraction. The latter is pretty much within our grasp on the shelf, the former somewhat less so.

4.1 Basic Notions of Type Theory with Records

The version of Type Theory used here is a type theory with records and record types (Betarte & Tasistro, 1998; Coquand et al., 2003, see also Constable & Hickey, 2000). The most fundamental notion of Type Theory is the typing *judgement* $a : T$ classifying an object a as being of type T . This can be seen as a generalization of the situation theoretic judgement $s \models \sigma$, generalization in that not only situations can figure as subjects of typing judgements. Note that the theory provides the objects and the types, but this form of judgement, as well as other forms are metatheoretical. Examples are given in (5). (5a-c) are typing judgements that presuppose the existence of types SIT, IND, REL, whose identity can be amplified. (5d) is the direct analogue of the ST statement $p \models \langle \langle RUN; b, t \rangle \rangle$; here $run(b, t)$ is a type (of a proof):⁶

- (5) a. $s : SIT$
- b. $b : IND$
- c. $run : REL$
- d. $s : run(b, t)$

A highly useful innovation in the version of CTT introduced by Betarte and Tasistro 1998 is the introduction of records and record types. A record corresponds to a number of kinds of entities in other semantic theories: the most lowly creature one might liken it to is a variable assignment, but it also bears significant resemblances to a typed feature structure and to what used to be called in ST an *abstract situation*. Technically, all a record is is an ordered tuple of the form (6), where crucially each successive field can depend on the values of the preceding fields:

$$(6) \quad \begin{bmatrix} l_i = k_i \\ l_{i+1} = k_{i+1} \\ \dots \\ l_{i+j} = k_{i+j} \end{bmatrix}$$

Together with records come record *types*. Technically, a record type is simply an ordered tuple of the form (7), where again each successive type can depend on its predecessor types within the record:

$$(7) \quad \begin{bmatrix} l_i : T_i \\ l_{i+1} : T_{i+1} \\ \dots \\ l_{i+j} : T_{i+j} \end{bmatrix}$$

⁶‘proof’ can be equally glossed as ‘observation’ or even ‘situation’, as explained by Ranta, 1994; the source of the ‘proof-based’ terminology is CTT’s initial use as a foundation for mathematics.

Record types allow us to place constraints on records: the basic typing mechanism assumed is that a record r is of type RT if all the typing constraints imposed by RT are satisfied by r . More precisely,

$$(8) \quad \text{If } a_1 : T_1, a_2 : T_2(a_1), \dots, a_n : T_n(a_1, a_2, \dots, a_{n-1}),$$

the record:

$$\left[\begin{array}{lcl} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \dots & & \\ l_n & = & a_n \end{array} \right] \text{ is of type: } \left[\begin{array}{lcl} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ \dots & & \\ l_n & : & T_n(l_1, l_2, \dots, l_{n-1}) \end{array} \right]$$

Crucially, not all the fields in r need to be ‘disciplined’ by RT . Thus, for example, the record in (9a) is of all the types (9b-e). Indeed, all records are of the empty type (9e), the type that imposes no constraints:

$$(9) \quad \text{a. } \left[\begin{array}{l} \text{runner} = \text{bo} \\ \text{time} = \text{2pm, Dec 20} \\ \text{place} = \text{batumi} \end{array} \right]$$

$$\text{(b) } \left[\begin{array}{l} \text{runner} : \text{Ind} \\ \text{time} : \text{Time} \\ \text{place} : \text{Loc} \end{array} \right] \text{ (c) } \left[\begin{array}{l} \text{runner} : \text{Ind} \\ \text{time} : \text{Time} \end{array} \right] \text{ (d) } \left[\text{runner} : \text{Ind} \right] \text{ (e) } \left[\right]$$

In particular, we can use this notion of typehood to offer the most rudimentary notion of situation, namely that it is a record which carries information about spatio-temporal extent:

$$(10) \quad \text{a. } s : \text{SIT implies } s : \left[\text{s-t-loc:ST-LOC} \right]$$

$$\text{b. } \text{SIT} =_{def} \left[\text{s-t-loc:ST-LOC} \right]$$

The type of a situation with a man riding a bicycle would then be the one in (11a). A record of this type would be as in (11b), where the required corresponding typing judgements are given in (11c):

$$(11) \quad \text{(a) } \left[\begin{array}{l} x : \text{IND} \\ c1 : \text{man}(x) \\ y : \text{IND} \\ c2 : \text{bicycle}(x) \\ c3 : \text{ride}(x,y) \\ \text{s-t-loc:STLOC} \end{array} \right] \text{ (b) } \left[\begin{array}{l} \dots \\ x = a \\ c1 = p1 \\ y = b \\ c2 = p2 \\ c3 = p3 \\ \text{s-t-loc} = \text{st0} \\ \dots \end{array} \right]$$

(c) $a : \text{IND}; c1 : \text{man}(a); b : \text{IND}; p2 : \text{bicycle}(b); p3 : \text{ride}(a,b); \text{st0} : \text{STLOC}$

4.2 Type Constructors

In order to do semantics, we need to have available various type construction operations, which allow for a recursive building up of the type theoretic universe. Ranta, 1994; Cooper, (forthcoming) list a dozen such constructors. Here I will list only a small number that are necessary for the tasks to be performed here:

$$(12) \quad \text{a. } \mathbf{function\ types:} \text{ if } T_1 \text{ and } T_2 \text{ are types, then so is } (T_1 \rightarrow T_2), \text{ the type of functions from elements of type } T_1 \text{ to elements of type } T_2. \text{ } f : (T_1 \rightarrow T_2) \text{ iff the domain of } f \text{ is } \{a | a : T_1\} \text{ and the range of } f \text{ is a subset of } \{a | a : T_2\}$$

- b. **The type of lists:** if T is a type, then $[T]$, the type of lists each of whose members is of type T , is also a type. $[a_1, \dots, a_n] : [T]$ iff for all i $a_i : T$
- c. **The unique type:** if T is a type and $x : T$, then T_x is a type. $a : T_x$ iff $a = x$.

Function types allow one to model abstraction. As Cooper points out, although abstraction in CTT works in a deceptively familiar ‘type theoretic’ way, the existence of record typing yields a rich notion of abstraction. It is simultaneous and restricted, i.e. it allows for multiple entities to be abstracted over simultaneously while encoding restrictions, and allows for vacuous abstraction.

Simultaneous abstraction with restrictions Since record types allow for multiple labels, the domain type of the abstract effects simultaneous abstraction ‘from’ the range type. Moreover, because of the successively dependent nature of the labels, restrictions can be imposed which apply to one or more labels. The simultaneous abstract in (13a) can be modelled as the function in (13b), which has the type in (13c) (assuming some typing given in terms of the types T_i for x_i):

$$\begin{aligned}
 (13) \quad & \text{a. } \lambda\{x_1, \dots, x_k\} \phi(x_1, \dots, x_k) \\
 & \quad [\psi_1(x_1, \dots, x_{n_1}), \dots, \psi_k(x_k, \dots, x_{n_k})] \\
 & \text{b. } \left[\begin{array}{l} x_1 = a_1 \\ \dots \\ x_k = a_k \\ c_1 = p_1 \dots \\ c_k = p_k \end{array} \right] \mapsto \phi(a_1, \dots, a_k) \\
 & \text{c. } r : \left[\begin{array}{l} x_1 : T_1 \\ \dots \\ x_k : T_k \\ c_1 : \psi_1(a_1, \dots, a_{n_1}) \\ \dots \\ c_k : \psi_k(a_k, \dots, a_{n_k}) \end{array} \right] \rightarrow \phi(r.x_1, \dots, r.x_k)
 \end{aligned}$$

Concretely, (a simplified) analysis of the meaning of the sentence ‘I see Bo’ would be the following function which maps a contextual assignment into a content:

$$\begin{aligned}
 (14) \quad & \text{a. context-assgn: } \alpha = \left[\begin{array}{l} x : \text{Ind} \\ t : \text{Time} \\ p1: \text{say}(x,t) \\ y: \text{Ind} \\ p2: \text{named}(\text{Bo},y) \end{array} \right] \\
 & \text{b. } \rho: \left[\text{cont} : \text{see}(\text{context-assgn}.x, \text{context-assgn}.t, \text{context-assgn}.y) \right]^7 \\
 & \text{c. context-assgn: } \alpha \rightarrow \rho: \text{ (the type of) functions from records of type } \alpha \text{ into records of type } \rho. \\
 & \text{A function of this type maps contexts specified to be a record of the following form—} \\
 & \quad \left[\begin{array}{l} \dots \\ x = x_0 \\ t = t_0 \\ p1 = c1 \\ y = y_0 \\ p2 = c2 \\ \dots \end{array} \right], \text{ where } x_0 : \text{Ind}, t_0 : \text{Time}, c1: \text{say}(x_0, t_0), y_0: \text{Ind}, \text{ and } c2: \text{named}(\text{Bo}, y_0), \\
 & \text{into a record of the type } \left[\text{cont} : \text{see}(x_0, t_0, y_0) \right]
 \end{aligned}$$

Vacuous abstraction there are a number of ways to go about this. A vacuous abstract is a constant function of some kind—where implementation varies is the domain of the function. If we think of a unary abstract as involving a domain type with one field which directly influences the ‘value’ of the function, a binary abstract as one whose domain type contains two such fields etc, then the domain type of a 0-ary type would simply be the empty type $[]$. This would make a 0-ary abstract a constant function from the universe of all records (since every record is of the type $[]$). This is a simple conception, but makes 0-ary abstracts into somewhat unwieldy entities. An alternative implementation is to arbitrarily choose some fixed entity as the domain of vacuous abstracts. For instance, we could take the domain type to be the type $[]_[]$ whose sole member is $[]$. In the absence of obvious empirical differences between the two I will stick with the former view, given its theoretical uniformity with non-vacuous abstraction:

- (15) a. $r: \alpha = []$
- b. $\rho: [\text{cont: run(a,t,l)}]$
- c. $r: [] \mapsto \rho$: function from records of type α into records of type ρ . In other words, the domain of this function is any record, whereas its range is any record of type ρ .

4.3 Propositions

One of the cornerstones of CTT is a conflation of propositions with types. There are, however, a variety of linguistic considerations, exemplified in the little inference pattern in (16), that suggest we need to treat propositions as individuals (“data elements”) that can be arguments of predicates.

- (16) a. The claim is that Bo left. Mike believes that claim. Hence, Mike believes that Bo left.

This is one feature that distinguishes the situation theoretic treatment, where propositions are first class citizens, from the standard CTT one. The situation theoretic ontology, as motivated in Ginzburg & Sag, 2000, provides a sharp distinction between propositions and situation types. Situation types are potential properties of (real world) situations, whereas propositions are the bearers of truth.

In fact, CTT offers a straightforward way of implementing the ST approach to using records. A proposition is a record of the form in (17a). The type of propositions is the record type (17b):⁸

- (17) a. $\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix}$
- b. $\begin{bmatrix} \text{sit} & : & \text{Record} \\ \text{sit-type} & : & \text{RecType} \end{bmatrix}$

The correspondence with the ST conception is quite direct. We can define truth conditions as follows:

- (18) A proposition $\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix}$ is true iff $r_0 : p_0$

We can now import an essentially intuitionist logic to underpin the Boolean structure of our space of propositions. The following are the operations on types typically assumed in CTT:⁹

- (19) a. $\neg T_0$ is the type $T_0 \rightarrow \perp$: the type $\neg T_0$ is witnessed if one can show that every situation of type T_0 is also of type $\perp$.

⁸One could, if one so wished, require *sit-type* to be an extension of the type SIT postulated in (10).

⁹See e.g. Ranta, 1994, Chapter 2.

- b. $T_1 \wedge T_2$: to show that $T_1 \wedge T_2$ is witnessed, one needs a pair $\langle p_1, p_2 \rangle$ where p_1 is of type T_1 and p_2 is of type T_2 .
- c. $T_1 \vee T_2$: a witness for $T_1 \vee T_2$ is an entity p_0 where p_0 is of type T_1 or p_0 is of type T_2 .

Given this, we can define the requisite Boolean operations on propositions:

$$(20) \quad \text{Given a proposition } p = \begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix}, \text{ its negation is the following record:}$$

$$\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & \neg p_0 \end{bmatrix}$$

A couple of remarks are worth making: just as in ST, there is a notion of partiality applicable here. If s is : *calm(jerusalem)*, it need not be : *calm(london)*. But this is not enough to allow us to infer that s : $\neg \text{calm}(\text{london})$

As a consequence of this the notion of proposition that emerges is non-classical in the sense given in (21a). Of course, it is still sound in the sense given in (21b):

- (21) a. $\text{false}(p)$ does not imply $\text{true}(\neg p)$
- b. $\text{true}(p)$ implies $\text{false}(\neg p)$

Conjunction is defined as follows:

$$(22) \quad \text{Given a proposition } p = \begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix} \text{ and a proposition } q = \begin{bmatrix} \text{sit} & = & r_1 \\ \text{sit-type} & = & p_1 \end{bmatrix},$$

their conjunction $\bigwedge\{p, q\}$ is the following record:

$$\begin{bmatrix} \text{sit} & = & \langle r_0, r_1 \rangle \\ \text{sit-type} & = & p_0 \wedge p_1 \end{bmatrix}$$

It is straightforward to show that:

$$(23) \quad p \text{ and } q \text{ are true iff } \bigwedge\{p, q\} \text{ is true.}$$

Disjunction is defined as follows:

$$(24) \quad \text{Given a proposition } p = \begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix} \text{ and a proposition } q = \begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_1 \end{bmatrix},$$

their disjunction $\bigvee\{p, q\}$ is the following record:

$$\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \vee p_1 \end{bmatrix}$$

Note that $\bigvee$ here is defined solely for propositions that have the same situational component.¹⁰ It is straightforward to show that:

$$(25) \quad \text{Either } p \text{ or } q \text{ is true iff } \bigvee\{p, q\} \text{ is true.}$$

The final notion we need concerning propositions is entailment. Assuming notions of subsumption on records and record types as given (see e.g. Betarte & Tasistro, 1998; Constable & Hickey, 2000), a sufficient condition for entailment can be described as follows:

$$(26) \quad \begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix} \rightarrow \begin{bmatrix} \text{sit} & = & r_1 \\ \text{sit-type} & = & p_1 \end{bmatrix} \text{ if } r_1 \sqsubseteq r_0 \text{ and } p_0 \sqsubseteq p_1.$$

¹⁰This has some linguistic justification from the observation, see Grice, 1989; Simmons, 2002, that disjunction only works if the two juncts ‘concern a single issue’.

5 QPA in Type Theory with Records

5.1 The Type Question

In light of my earlier comments about abstraction, a question will be a function from records into propositions. Before characterizing the type Question, let us consider some very simple examples of interrogatives and their CTT representations:

(27) a. Did Bo run

b. CTT representation: maps records $r : T_0 = []$

into propositions of the form $\left[\begin{array}{lcl} sit & = & r_1 \\ sit\text{-}type & = & [c : run(b)] \end{array} \right]$

c. who ran

d. CTT representation: maps records $r : T_{who} = \left[\begin{array}{lcl} x & : & Ind \\ rest & : & person(x) \end{array} \right]$ into propositions of the form $\left[\begin{array}{lcl} sit & = & r_1 \\ sit\text{-}type & = & [c : run(r.x)] \end{array} \right]$

e. who greeted what

f. CTT representation: maps records $r : T_{who,what} = \left[\begin{array}{lcl} x & : & Ind \\ rest & : & person(x) \\ y & : & Ind \\ rest & : & thing(x) \end{array} \right]$ into propositions of the form $\left[\begin{array}{lcl} sit & = & r_1 \\ sit\text{-}type & = & [c : greet(r.x, r.y)] \end{array} \right]$

Recall that one of the stumbling blocks facing QPA as implemented in the late 1970s was the the distinctness of types associated with interrogatives. This problem does not arise here due to the following elementary fact:¹¹

Fact 1 Function type subsumption

For any types A, A', B if $A \sqsubseteq A'$, then $(A' \rightarrow B) \sqsubseteq (A \rightarrow B)$.

Now the types which can be associated with the functions in (27) satisfy the subtype hierarchy in (28a). Given this and Fact 1, we get a correspondingly inverted hierarchy for the function types in (28b):

(28) a. $T_{who,what} \sqsubseteq T_{who} \sqsubseteq T_0$

b. $(T_0 \rightarrow Prop) \sqsubseteq (T_{who} \rightarrow Prop) \sqsubseteq (T_{who,what} \rightarrow Prop)$

Natural languages typically provide a dozen or so ‘wh-words’ (‘who’, ‘what’, ‘when’, ‘where’ etc), each of which introduces a restriction of the kind we have seen above for ‘who’ and ‘what’. For a finite set of restrictor types $wh_1, \dots, wh_k$, we can define the domain type $T_{wh_1^{n_1}, \dots, wh_k^{n_k}}$ which corresponds to a question with n_1 wh_1 elements, $\dots$, n_k wh_k elements, as in (29). $(T_{wh_1^{n_1}, \dots, wh_k^{n_k}} \rightarrow Prop)$ will subsume all other types of questions with fewer wh elements:

¹¹For a proof see e.g. Theorem 4, Constable, 2002.

$$(29) \quad T_{wh_1^{n_1}, \dots, wh_k^{n_k}} =_{def} \left[\begin{array}{ll} x_1^1 & : \text{Ind} \\ \text{rest}_1^1 & : wh_1(x) \\ \dots & \\ x_1^{n_1} & : \text{Ind} \\ \text{rest}_1^{n_1} & : wh_1(x) \\ \dots & \\ x_k^1 & : \text{Ind} \\ \text{rest}_k^1 & : wh_k(x) \\ \dots & \\ x_1^{n_k} & : \text{Ind} \\ \text{rest}_1^{n_k} & : wh_k(x) \end{array} \right]$$

We could then define the type Question to be the cartesian product $\mathbb{N} \times (T_{wh_1^{n_1}, \dots, wh_k^{n_k}} \rightarrow \text{Prop})$, where n runs over all k-tuples $\langle n_1, \dots, n_k \rangle$:

$$(30) \quad \text{Question} =_{def} \mathbb{N} \times (T_{wh_1^{n_1}, \dots, wh_k^{n_k}} \rightarrow \text{Prop})$$

The definition in (30) seems to be, with two caveats I will mention shortly, all that is needed for most linguistic or computational applications and so I will assume it here. In fact, the vast majority of queries encountered in a corpus are all to be found in $(T_{who^1, what^1, when^1, where^1, how^1, why^1})\text{Prop}$.¹² Nonetheless, there are two complications that suggest that a more general treatment is necessary to exhaustively characterize the domain type of Question. The first is due to the fact that natural languages typically contain determiners like ‘which’ and ‘whose’ that allow not only for the whole range of nouns as their complements (‘which/whose book/bicycle/violin/football boots ...’), but essentially arbitrary modification using relative clauses (‘which/whose book that S’ ...’). The second consideration is that we may wish not to restrict attention to restrictors that are denotations of NL expressions (an analogous point was emphasized for propositions by Montague when he designed his Intensional Logic).

In order to accommodate ‘which’ and ‘whose’ interrogatives with noun complements the same construction can be employed as discussed above—one simply adds all noun restrictor types, a finite set, in the pool of restrictor types. This though will not scale up to deal with restrictors emanating from relative clauses.

The obvious extension would be simply to identify Question with $(\text{RecType} \rightarrow \text{Prop})$, but this would be problematic since it would require any question to be defined on all record types. The next best move would be $(\text{RecType} \rightsquigarrow \text{Prop})$, in the notation of Constable & Smith, 1993 for the *partial* function type constructor. However, it is not clear how illuminating a characterization this is given the somewhat problematic nature of this type constructor. ? makes a related proposal, defining Question in terms of a partial function type $(\text{Rec} \rightsquigarrow \text{Prop})$, which he explicates as follows:

$$(31) \quad f : ([\] \rightsquigarrow \text{Prop}) \text{ iff } f : (T \rightarrow \text{Prop}), \text{ for some } T \sqsubseteq [\].$$

5.2 Answerhood

Answerhood is one of the essential testing grounds for a theory of questions. Abstracts can be used to *underspecify* answerhood. This is important given that NL requires a variety of answerhood notions, not merely exhaustive answerhood or notions straightforwardly definable from it. Moreover answerhood needs to be explicable theory-internally, not merely metatheoretically. This is because answerhood figures as a constituent relation of the lexical entries of resolute verbs¹³ and in rules regulating felicitous responses in dialogue management. For current purposes this means that we need to be able to define notions of answerhood as types.

¹²See Fernández, Ginzburg, & Lappin, 2004 for details.

¹³Verbs like ‘know’, ‘discover’, and ‘tell’. When such a verb embeds an interrogative complement I which denotes a question q the semantic argument of the verb is coerced to denote a fact that resolves q . See below and Ginzburg & Sag, 2000, Chapter 3, section 3.2 and Chapter 8, section 8.3.

One notion of answerhood relates to coherence: any speaker of a given language can recognize, independently of domain knowledge and of the goals underlying an interaction, that certain propositions are *about* or *directly concern* a given question. How can this notion of answerhood be characterized? The simplest notion of answerhood we can define on the basis of an abstract is one I'll call, following Ginzburg & Sag, 2000, *simple answerhood*. A simple answer is an instantiation of the abstract or negation thereof. We first define an auxiliary notion of *atomic answerhood*, specified in terms of the (family of) record type(s) in (32a). There the label *shortans* is a field for an entity of type the domain of the question,¹⁴ the label *compoundans* is a field for an instantiation of *q*, whereas the label *propans* is a field for a proposition that constitutes an atomic answer, which is a conjunct of *compoundans*;¹⁵ (32b) is then straightforward:

(32) Given a question $q : (A \rightarrow B)$:

- a. $\text{AtomAns}(q) =_{\text{def}} \left[\begin{array}{l} \text{shortans} : A \\ \text{compoundans} = q(\text{shortans}) : \text{Prop} \\ \text{propans} : \text{Prop} \\ c : \text{Conj}^*(\text{propans}, \text{compoundans}) \end{array} \right]$
- b. $\text{SimpleAns}(q) =_{\text{def}} \left[\begin{array}{l} \text{at} : \text{AtomAns}(q) \\ \text{simpleans} = \text{at.propans} \vee \neg \text{at.propans} : \text{Prop} \end{array} \right]$

Simple answerhood covers a fair amount of ground. Thus, given that a polar question is a 0-ary abstract $([])p$, its instantiations amount to the singleton $\{p\}$. Hence the simple answers of $([])p$ are p and $\neg p$. By the same token, the instantiations of a unary wh-question $([x:A])q(x)$ are all the propositions of the form $q(a)$ and $\neg q(a)$, where $a : A$. Nonetheless, simple answerhood clearly underdetermines aboutness. On the polar front, it leaves out the whole gamut of answers to polar questions that are weaker than p or $\neg p$ such as conditional answers ‘If r , then p ’ (e.g. 33a) or weakly modalized answers ‘probably/possibly/maybe/possibly not p ’ (e.g. 33b). As far as wh-questions go, it leaves out quantificational answers (33c-g), as well as disjunctive answers. These missing class of propositions, are pervasive in actual linguistic use.¹⁶

- (33) a. Christopher: Can I have some ice-cream then?
Dorothy: you can do if there is any.
- b. Anon: Are you voting for Tory?
Denise: I might.
- c. Dorothy: What did grandma have to catch?
Christopher: A bus.
- d. Elinor: Where are you going to hide it?
Tim: Somewhere you can't have it.

A simple way to enrich simple answerhood is by closing simple answerhood under disjunction:

- (34) p is about q iff p entails a (finite) disjunction of simple answers.

We can capture this in terms of the family of record types in (35); here *simpans* is a field for a list of simple answers, *disjans* is a field for their disjunction,¹⁷ and *propans* is a proposition that entails this disjunction:

¹⁴This is of course a mnemonic to the fact that short answers involve a term of this type.

¹⁵ $\text{Conj}^*(x,y)$ holds iff y is a finite conjunction of propositions, one of which is x ; we construe this relation to be reflexive. This slight complication of distinguishing compound from atomic instantiations is required for our treatment of conjoined questions presented below in section 5.3.

¹⁶These examples are all from the British National Corpus (BNC).

¹⁷This definition assumes the existence of a function that maps a list of records of type $\text{SimpleAns}(q)$ to the disjunction of the *simpleans* fields of these records.

(35) Given a question $q : (A \rightarrow B)$:

$$\text{a. Aboutness}(q) =_{\text{def}} \left[\begin{array}{l} \text{simpanslist} : [\text{SimpleAns}(q)] \\ \text{disjans} = \bigvee_{\text{simpleans}} \text{simpanslist} : \text{Prop} \\ \text{propans} : \text{Prop} \\ c : \rightarrow(\text{propans}, \text{disjans}) \end{array} \right]$$

Given that for any question, the class of simple answers always contains at least one proposition p and its negation $\neg p$, the definition in (34) is restrictive only within a logical framework in which the law of the excluded middle fails. That is, a setting in which (36) is the case:

(36) $p \vee \neg p$ is not vacuously true.

Both situation theory and CTT are such settings. Within situation theory, Ginzburg 1995, Ginzburg and Sag 2000 show that aboutness as defined in (34) seems to encompass the various classes of propositions exemplified in (33). Thus:

- By defining *possibly*(p) as involving (a) the truth of $p \vee r$, where p, r are incompatible and (b) $\neg p$ not being proven, one can show *possibly*(p) is about p ?. By monotonicity, all modalities stronger than ‘possibly’ are also about p ?
- Existence-entailing generalized quantifiers such as existentials satisfy aboutness by virtue of their entailing the truth of a positive instantiation of a given wh-question.
- Generalized quantifiers such as ‘At most one N’ or ‘Few N’ satisfy aboutness by virtue of their entailing the truth of a positive or negative instantiation of a given wh-question.¹⁸

Cooper, 2004 shows how standard generalized quantifier theory can be imported into the Type Theory with Records framework.¹⁹ Assuming such a version of GQ theory and correspondingly for modality, the situation theoretic explication of aboutness will transfer to CTT.

Answerhood in the aboutness sense is clearly distinct from a highly restricted notion of answerhood, that of being a proposition that *resolves* or constitutes *exhaustive information* about a question. This latter sense of answerhood, which is restricted to true propositions, has been explored in great detail in the formal semantics literature, since it is a key ingredient in explicating the behaviour of interrogatives embedded by resolute predicates such as ‘know’, ‘tell’ and ‘discover’. This notion of answerhood relates to intuitions of an agent possessing information about a question which forecloses discussion thereof:

- (37) a. I wondered about who/when/why Bo left, so I asked. Mo told me ..., so now I know who/when/why Bo left.
- b. An important issue is what we should do next. We’ve been discussing this in some detail. But the issue hasn’t been resolved as yet.

For current purposes, it will be sufficient to mention one simple semantic notion, strong exhaustiveness due originally to Groenendijk & Stokhof, 1984, that approximates resolvedness in cases where interaction is relatively transparent:

(38) p is a strongly exhaustive answer to q iff p is true and entails every p_i which is a true simple answer to q .

¹⁸Note that in frameworks like situation theory or CTT, a proposition such as ‘At most one student left’ will entail, for some individual ‘Kim’, the truth of the disjunction (‘Kim left’ or ‘Kim did not leave’). But it will not entail, for instance, the truth of the disjunction (‘Kim is tall’ or ‘Kim is not tall’).

¹⁹The basic idea is to define the witnessing conditions of a type theoretic GQ gq_0 which relates two functions f_1, f_2 , using the standard set theoretic GQ relation gq^* and sets that constitute the ranges of f_1, f_2 .

The strongly exhaustive answer can be constructed in terms of the family of types in (39). The functional restriction is the standard CTT way of encoding a universal quantification, akin to the classical discourse representation theoretic technique; the restriction to *true* simple answers is achieved by using the value of the *simpleans* field as a restriction on itself:

$$(39) \quad \text{StrongExhAns}(q) =_{def} \left[\begin{array}{l} \text{exhans} : \text{Prop} \\ c_0 : \left[\begin{array}{l} r : \text{SimpleAns}(q) \\ c_1 : r.\text{simpleans} \end{array} \right] \left[c_2 : \rightarrow(\text{exhans}, r.\text{simpleans}) \right] \end{array} \right]$$

The characterization in (39) seems to straightforwardly capture the resolvedness conditions for a polar question. In other words, a resolving answer needs to entail whichever of p or $\neg p$ is factual, assuming the question is decided:

$$(40) \quad \text{Bo knows whether } p \text{ (whether Mo is asleep). Hence, either Bo knows } p \text{ (that Mo is asleep) or Bo knows } \neg p \text{ (that Mo is not asleep).}$$

The situation with wh-questions is far more complicated. As discussed in *inter alia* Boër & Lycan, 1985; Ginzburg, 1995; Asher & Lascarides, 1998, there are various pragmatic factors which seem to come into the picture when evaluating whether a proposition resolves a question, in a given context. A proper treatment requires one to parametrize semantic notions of answerhood with certain pragmatic factors, an issue I will not consider here.

Finally, as indication that the account provided here is applicable to embedded interrogatives, I propose (41) as semantic entries for a question embedding verb such as ‘wonder’ and a resolute predicate such as ‘know’. (41a) is self explanatory. (41b) illustrates that although the complement ‘know’ selects is a question, the agent is predicated to know not the question as such but rather a proposition that is a strongly exhaustive answers to the question:

$$(41) \quad \begin{array}{ll} \text{a. wonder: } \lambda q : \text{Question}, x : \text{ind}([c : \text{wonder}(x, q)]) \\ \text{b. know: } \lambda q : \text{Question}, x : \text{ind} \\ \quad p = \text{StrongExhAns}(q).\text{exhans} : \text{Prop} \\ \quad (c : \text{know}(x, p)) \end{array}$$

5.3 Coordination

Conjunction and disjunction of interrogatives is an intricate issue I cannot do justice to here. The definition below shows though that this issue on the face of it does NOT pose a particular problem for this version of QPA. Indeed, a more general claim can be made, which I will not substantiate here, that conjunction and disjunction do not provide any support for an EAC account, as has often been claimed.²⁰

(42) A Conjunction Operation for Questions

$$\text{Given a question } q_1 = (T)p_1 : (T \rightarrow \text{Prop}) \text{ and a question } q_2 = (T')p_2 : (T' \rightarrow \text{Prop}) \\ q_1 \wedge q_2 =_{def} \left[\begin{array}{l} \text{conj1} : T \\ \text{conj2} : T' \end{array} \right] p_1 \wedge p_2$$

I illustrate this definition with an example. Conjoining two polar questions results in the question in (43a):

$$(43) \quad ([\Box])p_1 \wedge ([\Box])p_2 =_{def} \left[\begin{array}{l} \text{conj1} : [\Box] \\ \text{conj2} : [\Box] \end{array} \right] p_1 \wedge p_2$$

a. Jo knows whether Millie left and whether Mike arrived.

²⁰I’d like to thank Robin Cooper for an improvement on my original formulation.

The instantiation of this abstract is $p_1 \wedge p_2$. Therefore, by our definition of simple answerhood, the simple answers are $\{p_1, p_2, \neg p_1, \neg p_2\}$. This predicts that an exhaustive answer to (43a) is one that entails whichever members of this set which are true. (43b) seems to bear this out.

5.4 Grain Issues

CTT actually provides very fine grained entities and so does not run into the problems that beset traditional semantic approaches with respect to logical omniscience and various other puzzles. In fact, as Cooper, (forthcoming) discusses, this can be too much of a good thing, given that record types distinct only by their labelling are distinguished. Cooper goes on to offer a plausible criterion of type individuation of record types using Σ -types, where the corresponding ‘labels’ function as bound variables. I mention here some puzzles of semantic grain that beset previous approach to questions and are resolved simply within the current approach—in a way that is consistent with Cooper’s criterion of type identity.

There are linguistic grounds for distinguishing the denotations of positive and negative polar questions, as in (44), while ensuring the identity of their answerhood relations:

- (44) a. Did Bo leave?
b. Didn’t Bo leave?

For a start, languages like French and Georgian provide distinct words to affirm a positive polar question (*oui*, *xo*) and a negative polar question (*si*, *diax*).

In addition there appear to be distinct periphrasis possibilities between positive and negative polars:

- (45) a. A: In the end Kim and Sandy didn’t leave yesterday?
B: Right, (they’re still here.)
b. A asked whether Kim and Sandy had actually not left.
c. A checked if Kim and Sandy had not in the end left.
d. ?A asked if Kim and Sandy left.
e. ??A checked if Kim and Sandy had in the end left. (? marks inaccurate paraphrase)

Also the two types of questions elicit different response patterns in English. Whereas ‘yes’ is felicitous as a response to a positive question, it seems awkward as a response to a negative question:

- (46) A: She’s not annoyed?/Isn’t she annoyed?
B: Yes.
A: She is or she isn’t annoyed?

QPA as formulated here can distinguish the denotations of positive and negative polar questions, while ensuring that the answerhood relations specified are identical. The explanation for the former is simple: (47a) is a different function from (47b):

- (47) a. $r: \Box \mapsto p$
b. $r: \Box \mapsto \neg p$,

At the same time, as is easy to verify given our definitions above, (47a) and (47b) specify the same answerhood relations. By contrast, approaches to interrogative content based on exhaustive answerhood conditions necessarily identify the content of positive and negative polars since these have identical exhaustive answers.

A second grain issue is this: it seems desirable not to necessarily identify questions that give rise to an identical set of instantiations, such as those in (48a,b). In an IL-like type system both denote (48c):

- (48) a. Which primes smaller than five are divisors of 9?
 b. Which numbers smaller than 4 are divisors of 9?
 c. $x : \{2, 3\} \mapsto \text{Divide}(x, 9)$.

In CTT, we have no difficulty distinguishing the content of interrogatives such as (49a) and (49c):

- (49) a. which primes smaller than 5 divide 9
 b. CTT representation: a function mapping records $r : \begin{bmatrix} x & : & \text{Prime} \\ \text{rest} & : & <(x,5) \end{bmatrix}$
 into propositions of the form $\begin{bmatrix} \text{sit} & = & r_1 \\ \text{sit-type} & = & \begin{bmatrix} c : \text{divide}(r.x, 9) \end{bmatrix} \end{bmatrix}$
 c. which numbers smaller than 4 divide 9
 d. CTT representation: a function mapping records $r : \begin{bmatrix} x & : & \text{Number} \\ \text{rest} & : & <(x,4) \end{bmatrix}$
 into propositions of the form $\begin{bmatrix} \text{sit} & = & r_1 \\ \text{sit-type} & = & \begin{bmatrix} c : \text{divide}(r.x, 9) \end{bmatrix} \end{bmatrix}$

5.5 Which polyadicity

A final feature of an analysis of interrogatives using some form of simultaneous abstraction, detailed in Ginzburg & Sag, 2000, concerns the polyadic behaviour of *which*-interrogatives. Unary *which*-interrogatives (with a singular noun complement) involve a uniqueness presupposition:

- (50) A: Which student left yesterday?
 B: Bo.
 Valid inference: Bo is the student that left.
 a. A: #Which book have you been reading over the last few years?
 b. Which natural number precedes all others?

Binary and ternary *which*-interrogatives also carry a uniqueness-related presupposition. Nonetheless, it is not one of absolute uniqueness—the queried predicate is presupposed to have a unique instantiating tuple—as would be expected if interrogatives are interpreted iteratively. Rather, there is a presupposition of *n-jectiveness* in a sense I will shortly gloss:

- (51) a. [Asked of a gossip columnist] Which guest is going home with which guest? (Higginbotham & May–81, nt. 3)
 b. A: Which student read which book for which course?

- c. B: Kim read *Anna Karenina* for Russian Lit 413 and she read *To Mageio* for Greek Lit 169, whereas Sandy read *Anna Karenina* for Women’s Studies 305 and he read *White Teeth* for Women’s Studies 510. (Example due to Carl Pollard (p.c.))

n-jectivity, due to Carl Pollard, is given in (52). It is a polyadic notion, easy to associate with an *n*-ary abstract, if interrogatives arise via simultaneous abstraction:²¹

- (52) A propositional abstract $q (= \lambda\{b_1, \dots, b_n\}p)$ is *n*-jective iff the following condition holds: whenever

1. two records r_1 and r_2 are appropriate for a question q ,
2. both $q[r_1]$ and $q[r_2]$ are true propositions, and moreover
3. r_1 and r_2 are identical on $n - 1$ labels,

then r_1 and r_2 also agree on the *n*th label.

That is, if for some subset of labels $\{l_1, \dots, l_{n-1}\}$ $r_1(l_{i_j}) = r_2(x_{i_j})$, for $1 \leq j \leq n - 1$, then also $r_1(x_{i_n}) = r_2(x_{i_n})$

6 Conclusions and Future Work

Ginzburg and Sag 2000 provided a situation theoretic implementation of a theory of *Questions as Propositional Abstracts* (QPA), a simple, transparently implementable theory that has significant empirical coverage. That implementation succeeded in overcoming intrinsic problems QPA faces in its earlier implementations. Nonetheless, since Ginzburg and Sag 2000 rely on a special purpose account of abstraction, there is a question to what extent QPA can be sustained using standard notions of abstraction. In this paper I have attempted to allay such doubts by implementing QPA in a version of Type Theory that provides record types. These latter allow one to develop notions of simultaneous/vacuous abstraction with restrictions and an ontology with various ‘informational entities’. Moreover, the intrinsic polymorphism of this theory plays a crucial role in enabling the definition of a general type for questions, one of the main stumbling blocks for earlier versions of QPA.

References

- Aczel, P. & Lunnon, R. 1991. Universes and parameters. In Barwise, J., Gawron, J. M., Plotkin, G., & Tutiya, S. (Eds.), *Situation Theory and Its Applications, II*, CSLI Lecture Notes Number 26. CSLI Publications, Stanford.
- Aloni, M. & Rooy, R. van. 2002. The dynamics of question and focus. In *Proceedings of SALT 21*.
- Asher, N. & Lascarides, A. 1998. Questions in dialogue. *Linguistics and Philosophy*, 21.
- Betarte, G. & Tasistro, A. 1998. Martin-löf’s type theory with record types and subtyping. In Sambin, G. & Smith, J. (Eds.), *25 Years of Constructive Type Theory*. Oxford University Press.
- Boër, S. & Lycan, W. 1985. *Knowing Who*. Bradford Books. MIT Press, Cambridge.
- Bos, J. & Gabsdil, M. 2000. First-order inference and the interpretation of questions and answers. In Poesio, M. & Traum, D. (Eds.), *Proceedings of the GötaLog, the 4th Workshop on the Formal Semantics and Pragmatics of Dialogue*. Götaborg.

²¹This means also that this account should extend simply to phenomena of non-iterative quantification involving *similar, different* originally discussed by Ed Keenan. Thanks to a reviewer for JoL&C for pointing this out.

- Chieh Shan, C. & ten Cate, B. 2002. The partition semantics of questions, syntactically. In Nissim, M. (Ed.), *Proceedings of the ESSLLI-2002 student session* Trento. 14th European Summer School in Logic, Language and Information.
- Constable, R. 2002. Naive computational type theory. In Schwichtenberg, H. & Steinbruggen, R. (Eds.), *Proof and System-Reliability*, pp. 213–259.
- Constable, R. & Hickey, J. 2000. Nuprl’s class theory and its applications. In Bauer, F. L. & Steinbruggen, R. (Eds.), *Foundations of Secure Computation*, pp. 91–115. IOS Press, Amsterdam.
- Constable, R. & Smith, S. 1993. Computational foundations of basic recursion theory. *Theoretical Computer Science B: Logic, semantics, and theory of programming*, 121, 89–112.
- Cooper, R. Austinian truth in martin-löf type theory. *Research on Language and Computation*. Forthcoming.
- Cooper, R. 2004. Dynamic generalised quantifiers and hypothetical contexts. In *Ursus Philosophicus, a festschrift for Bjoern Haglund*, Department of Philosophy. Gothenburg University.
- Cooper, R. 1998. Mixing situation theory and type theory to formalize information states in dialogue exchanges. In Hulstijn, J. & Nijholt, A. (Eds.), *Proceedings of TwenDial 98, 13th Twente workshop on Language Technology*. Twente University, Twente.
- Coquand, T., Pollack, R., & Takeyama, M. 2003. A logical framework with dependent types. *Fundamenta Informaticae*, 20, 1–21.
- Fernández, R., and Ginzburg, J., & Lappin, S. 2004. Classifying ellipsis in dialogue: a machine learning approach. In Hulstijn, J. & Nijholt, A. (Eds.), *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland, pp. 240–246
- Fernando, T. 2001. Conservative generalized quantifiers and presupposition. In *SALT*, Vol. 11, pp. 172–191 NYU/Cornell.
- Ginzburg, J. 1995. Resolving questions, i. *Linguistics and Philosophy*, 18, 459–527.
- Ginzburg, J. & Sag, I. A. 2000. *Interrogative Investigations: the form, meaning and use of English Interrogatives*. No. 123 in CSLI Lecture Notes. CSLI Publications, Stanford: California.
- Grice, H. P. 1989. *Studies in the Ways of Words*. Harvard University Press. Reprinted from a 1957 article.
- Groenendijk, J. & Stokhof, M. 1984. *Studies on the Semantics of Questions and the Pragmatics of Answers*. Ph.D. thesis, University of Amsterdam.
- Hausser, R. 1983. The syntax and semantics of english mood. In Kiefer, F. (Ed.), *Questions and Answers*. Reidel.
- Koenig, J-P. 2004. Any questions left? review of ginzburg and sag’s interrogative investigations.. *Journal of Linguistics*, 40, 131–148.
- Krahmer, E. & Piwek, P. 1999. Presupposition projection as proof construction. In Bunt, H. & Muskens, R. (Eds.), *Computing Meaning: Current Issues in Computational Semantics*, Vol. 1. Kluwer.
- Larsson, S. 2002. *Issue based Dialogue Management*. Ph.D. thesis, Gothenburg University.
- Montague, R. 1974. The proper treatment of quantification in ordinary english. In Thomason, R. (Ed.), *Formal Philosophy*. Yale UP, New Haven.

- Neelken, R. (forthcoming). Review of interrogative investigations.. *Journal of Logic, Language, and Information*, (forthcoming).
- Ranta, A. 1994. *Type Theoretical Grammar*. Oxford University Press, Oxford.
- Scha, R. 1983. *Logical Foundations for Question Answering*. Ph.D. thesis, Rijksuniversitet Groningen.
- Seligman, J. & Moss, L. 1997. Situation theory. In van Benthem, J. & ter Meulen, A. (Eds.), *Handbook of Logic and Linguistics*. North Holland, Amsterdam.
- Simmons, M. 2002. Disjunction and alternatives. *Linguistics and Philosophy*, 25.