

Austinian truth, attitudes and type theory *

Robin Cooper (cooper@ling.gu.se)
Göteborg University

Abstract. This paper is part of a broader project whose aim is to present a coherent unified approach to natural language dialogue semantics using tools from type theory. Here we explore aspects of our approach which relate to situation theory and situation semantics. We first point out a relationship between type theory and the Austinian notion of truth. We then consider how records in type theory might be used to represent situations and how dependent record types can be used to model constraints on situations. We then sketch treatments of attitude phenomena for which Barwise and Perry proposed situation semantic analyses (perception complements, belief, the Pierre puzzle) as well as two other intensional phenomena (intensional verbs and intentional identity). Finally we give a characterisation of the type theory used and a small illustrative fragment of English.

Keywords: situation semantics, situation theory, type theory, attitudes, intensionality, records

To the memory of Jon Barwise

1. Introduction

This work is part of a broader project whose aim is to present a coherent unified approach to natural language dialogue semantics using tools from type theory. We are seeking to do this by bringing together

* I am grateful to Aarne Ranta for help in understanding basic notions of type theory and the use of record types for natural language semantics, as well as pointing out how many of my ideas relate to those presented in Ranta (1994). Bengt Nordström has helped me understand both the intuitions and technical details of type theory and in particular made an important contribution to my understanding of the relationship between record types and Σ -types. Thierry Coquand pointed out a number of problems with an earlier version of the definition of types. I have also had significant discussions with a number of other people who have contributed to my understanding and influenced my approach, including: Tim Fernando, Jonathan Ginzburg, Yiannis Moschovakis, Rich Thomason and Ray Turner. I am also grateful for thoughtful comments from an anonymous reviewer which have led to a number of changes. This work was supported by Vetenskapsrådet project number 2002-4879, Records, types and computational dialogue semantics. Earlier versions of it were presented at the Sapporo Workshop On Language, Action, and Cognition (SWOLAC 2003), 24th February, 2003, Hokkaido University and the workshop Barwise and Situation Semantics, 26th June 2003, Stanford University.

Head Driven Phrase Structure Grammar (HPSG)¹, Montague semantics², Discourse Representation Theory (DRT)³, situation semantics⁴ and issue-based dialogue management⁵ into a single type-theoretic formalism.

A survey of our approach to the semantic theories (i.e., Montague semantics, DRT and situation semantics) and HPSG can be found in Cooper(forthcoming). Other work in progress can be found on <http://www.ling.gu.se/~cooper/records>. We give a brief summary here. Record types can be used as discourse representation structures (DRSs). Truth of a DRS corresponds to there being an object of the appropriate record type and this gives us the effect of simultaneous binding of discourse references (corresponding to labels in records) familiar from the semantics of DRSs in Kamp and Reyle (1993). Dependent function types provide us with the classical treatment of donkey anaphora from DRT in a way corresponding to the type theoretic treatments proposed by Sundholm (1986) and Ranta (1994). At the same time record types can be used as feature structures of the kind found in HPSG since they have recursive structure and induce a kind of subtyping which can be used to mimic unification. Because we are using a general type theory which includes records we have functions available and a version of the λ -calculus. This means that we can use Montague's λ -calculus based techniques for compositional interpretation. From the HPSG perspective this gives us the advantage of being able to use "real" variable binding which can only be approximately simulated in pure unification based systems. From the DRT perspective this use of compositional techniques gives us an approach similar to that of Muskens (1995) and work on λ -DRT (Kohlhase *et al.*, 1996).

In this paper we explore aspects of our approach which relate to situation theory and situation semantics. Records give us a way of parcelling together elements of information in a similar manner to viewing situations as collections of infons. Dependent types give us a natural and powerful way of treating what in situation theory used to be called parametric infons. Since we are using a stratified type theory in which types are treated as first class objects we can incorporate the approach to the analysis of the attitudes that was given in Barwise and Perry (1983). While it is the first class nature of types rather than record types as such which gives us our basic treatment of the attitudes, the fact that we are using record types gets exploited in that it gives

¹ Sag *et al.* (2003)

² Montague (1974) is the classic reference.

³ Kamp and Reyle (1993), van Eijck and Kamp (1997) and much other literature.

⁴ Barwise and Perry (1983) and subsequent literature.

⁵ Larsson (2002)

us a simple way of achieving the effect of simultaneous abstraction which was important for the situation semantic treatment of context dependence. Using records also allows us to extend the Barwise and Perry approach to a treatment of intentional identity (Geach, 1967).

We will begin the paper with a consideration of a relationship between type theory and the Austinian notion of truth. We will then consider how records might be used to represent situations and how constraints on situations could be considered as constraints on universes of records. We will then go into the treatment of attitudes, sketching treatments of phenomena for which Barwise and Perry proposed analyses (perception complements, belief, the Pierre puzzle) as well as other phenomena (intensional verbs, intentional identity). We will finally give a more formal presentation of the type theory and a small illustrative fragment of English.

2. Austinian truth

Barwise and Perry paraphrase Austin (1961) on p. 160 of *Situations and Attitudes* as follows: “a statement is true when the actual situation to which it refers is of the type described by the statement.”⁶

The central idea behind this is that statements are not true or false *simpliciter* but are true or false of something, some part of the world. From this we are able, of course, to derive a notion of truth *simpliciter*, namely that there is some part of the world of which the statement is true. It is clear also that an informal notion of type is central to Austin’s view.

Something very like Austin’s idea is one of the central parts of Per Martin-Löf’s type theory. At the core of type theory is the notion that objects are of various types. But in addition to this Martin-Löf’s type theory follows the *propositions as types* principle. Propositions are regarded as types of proofs. Ranta (1994, p 53ff) has a useful discussion of what a proof might mean in the case of non-mathematical propositions. He points out that Martin-Löf construes propositions as types of proof *objects* not proof *processes*. Thus the proposition that *there is a prime number between 212 and 222* has as its proofs the set of prime numbers between 212 and 222, rather than an argument demonstrating the

⁶ The actual quotation from Austin they refer to is: “A statement is said to be true when the historic state of affairs to which it is correlated by the demonstrative conventions (the ones to which it ‘refers’) is of a type with which the sentence used in making it is correlated by the descriptive conventions.” Austin (1961), p. 123

existence of such prime numbers.⁷ For non-mathematical propositions Ranta draws a parallel with Davidson’s (1980) event-based approach. In type theory a proof of the proposition that *Amundsen flew over the North Pole* would be a flight by Amundsen over the North Pole. Mixing type theoretical and situation theoretical terminology we might say that a proof of this proposition is a situation in which Amundsen flew over the North Pole and that type theoretical propositions are types of situations. In situation theory *infons* (*states of affairs*) have been regarded as types of situations. Thus we can draw a parallel between types of proofs and infons.

3. Situations as records

In situation theory one talks of ‘ $s \models \sigma$ ’ (“situation s supports infon (soa) σ ” or “ s is of type σ ”) as a sentence of situation theory. A distinction is made between this sentence and the situation theoretic object ($s \models \sigma$), an Austinian proposition, which is true just in case $s \models \sigma$ (see e.g. Barwise and Cooper, 1991).

In type theory one has judgements of the form $a : T$ (“object a is of type T ”). Judgements are not type theoretic objects just as sentences of situation theory are not situation theoretic objects. Among the judgements are those which assign a type of proof to a proof object, e.g. $p : \text{see}(a, b)$. To say that p is of the type $\text{see}(a, b)$ is to say that p is a proof that a sees b . Taking ‘see’ here to be a relation between individuals (i.e., of arity $\langle \text{Ind}, \text{Ind} \rangle$) we have to establish that a and b are individuals in order for this to make sense. This can be done by introducing a context: $a:\text{Ind}, b:\text{Ind}$.⁸ Contexts are not normally regarded as type theoretic objects but are introduced as syntactic constructs. If we want to reify them so that they can, for example, occur in the domain of functions we can introduce records as objects in our type theoretic universe. Records belong to types which are called *record types*.

We shall use record types which have fields corresponding both to the context and the propositional judgement, e.g.

$$\left[\begin{array}{l} x \quad : \quad \text{Ind} \\ y \quad : \quad \text{Ind} \\ \text{prf} \quad : \quad \text{see}(x,y) \end{array} \right]$$

⁷ Here he is assuming an intuitionistic view of prime numbers as a pair consisting of a number and a proof that it is prime. The proof, however, would still be an object such as a function.

⁸ We may regard *Ind* as a type of individuals which characterizes what would be called a scheme of individuation in situation theory.

A record

$$\left[\begin{array}{l} x = a \\ y = b \\ \text{prf} = p \\ \dots \end{array} \right]$$

would be of this type just in case $a : Ind$, $b : Ind$ and $p : \text{see}(a, b)$. Note that the record may have more fields than the type. Thus a record may belong to several types and a notion of subtype is introduced.

Records and record types give us the possibility of grouping together collections of infon-like objects. For example, the following record types are available:

$$\left[\begin{array}{l} s : \left[\begin{array}{l} x : Ind \\ y : Ind \\ s_1 : \text{see}(x,y) \\ s_2 : \text{see}(y,x) \end{array} \right] \end{array} \right]$$

$$\left[\begin{array}{l} x : Ind \\ y : Ind \\ s : \left[\begin{array}{l} s_1 : \text{see}(x,y) \\ s_2 : \text{see}(y,x) \end{array} \right] \end{array} \right]$$

Thus we can draw a parallel between records and situations and record types and infons. The idea of situations as records corresponds to Ranta's (1991) proposal that situations should be regarded as contexts in type theory.

4. Constraints on universes of records

This kind of grouping allows us to talk about constraints on situations as constraints on universes of records. For example, the constraint *smoke means fire* discussed in Barwise and Perry (1983) meaning that if there is a situation with smoke then there is also a situation with fire could be captured by saying that any universe of records which obeys this constraint is such that if it contains a record

$$r : \left[\begin{array}{l} l : Loc \\ s : \text{smoke}(l) \end{array} \right]$$

then it also contains a record

$$r' : \left[\begin{array}{l} l : Loc \\ s : \text{fire}(l') \end{array} \right]$$

We may wish to make this constraint more precise by requiring that the location of the fire be near the location of the smoke. We can do this by making the type of r' dependent on r , i.e.

$$r' : \left[\begin{array}{l} l : Loc \\ s : \text{fire}(l) \\ c : \text{near}(l, r.l) \end{array} \right]$$

$r.l$ is used to represent the value corresponding to label 'l' in record r .

We can internalise a constraint of this kind as a function. The type of this function would be

$$(r : \left[\begin{array}{l} l : Loc \\ s : \text{smoke}(l) \end{array} \right]) \left[\begin{array}{l} l : Loc \\ s : \text{fire}(l) \\ c : \text{near}(l, r.l) \end{array} \right]$$

A function of this type maps a record r of type

$$\left[\begin{array}{l} l : Loc \\ s : \text{smoke}(l) \end{array} \right]$$

to a record of type

$$\left[\begin{array}{l} l : Loc \\ s : \text{fire}(l) \\ c : \text{near}(l, r.l) \end{array} \right]$$

Similarly the constraint *kissing involves touching* which arguably applies to single situations (you cannot perceive *a* kiss *b* without perceiving *a* touch *b*) can be captured by requiring that any record in the universe which is of type

$$\left[\begin{array}{l} x : Ind \\ y : Ind \\ s : [c_1:\text{kiss}(x,y)] \end{array} \right]$$

is also of type

$$\left[\begin{array}{l} x : Ind \\ y : Ind \\ s : [c_2:\text{touch}(x,y)] \end{array} \right]$$

Such a constraint can also be represented by a function of the type

$$(r : \left[\begin{array}{l} x : Ind \\ y : Ind \\ s : [c_1:kiss(x,y)] \end{array} \right]) \left[s=r.s : [c_2:touch(r.x,r.y)] \right]$$

A function of this type maps a record r of type

$$\left[\begin{array}{l} x : Ind \\ y : Ind \\ s : [c_1:kiss(x,y)] \end{array} \right]$$

to a record of type

$$\left[s=r.s : [c_2:touch(r.x,r.y)] \right]$$

Here we have made use of a manifest field (Coquand *et al.*, 2004). The notation

$$\left[\ell=a : T \right]$$

is a convenient variant for

$$\left[\ell : T_a \right]$$

where T_a is a type just in case $a : T$ and $b : T_a$ iff $b = a$. T_a is called a *singleton type*. This constraint requires that any record containing a proof that a kisses b will also contain a proof that a touches b .

Functions in Martin-Löf type theory are normally required to be total. This means that constraints of the kind we have discussed would be infallible constraints. One way to allow soft or fallible constraints would be to allow partial functions in the type theory. Another, probably preferable, way is to model the situation theoretic constraint as the function type itself rather than as a function of the type. Certain function types would be classified as constraints, e.g. the following type could be instantiated.

$$\left[\begin{array}{l} c=(r : \left[\begin{array}{l} x : Ind \\ s : man(x) \end{array} \right]) \left[s : snore(r.x) \right] : Type \\ s:constraint(c) \end{array} \right]$$

This corresponds to *men snore*. It introduces the function type of functions that, informally, map men to instances of their snoring and it then says that this type is a constraint. This does not commit to there being a function of this type, i.e. it does not commit to it being the case that every man snores or even that any man snores. One can, of course, discuss what exactly is to count as a proof that this type is a constraint, e.g. whether a certain number of instances of men snoring

have to have been observed, or whether it has to be a cultural norm or a physical constraint. The different kinds of proof could correspond to the different kinds of constraints that Barwise and Perry (1983) discuss.

A solution such as this requires the introduction of a type *Type*, i.e. a type to which types themselves belong. This allows us to treat types as objects in our domain. When we discuss the attitudes later we will see other reasons for doing this. It has to be done with care in order to avoid paradoxes. One standard way to do this is to introduce stratified types. Thus the type above could be written more precisely as

$$\left[\begin{array}{l} c = (r : \left[\begin{array}{l} x \quad : \quad \text{Ind} \\ s \quad : \quad \text{man}(x) \end{array} \right]) \left[\begin{array}{l} s \quad : \quad \text{snore}(r.x) \end{array} \right] : \text{Type}^0 \\ s : \text{constraint}(c) \end{array} \right]$$

where Type^0 is the type of types of order 0 which do not contain any embedded type. The whole type would be of type Type^1 , i.e. types of order 1, since it is defined in terms of a type of order 0. We will usually suppress the order superscripts in our representation of types.

Another way of thinking of constraints corresponds to the work on channel theory (Barwise and Seligman, 1997). We can, for example, consider the following type:

$$\mathcal{C} = \left[\begin{array}{l} \text{signal} \quad : \quad T_1 \\ \text{target} \quad : \quad T_2 \end{array} \right]$$

A record $r : \mathcal{C}$ is a *link classified by \mathcal{C}* . An agent attuned to \mathcal{C} on observing an object of type T_1 will predict (defeasibly) the existence of an object of type T_2 . In other words, according to \mathcal{C} an object $a : T_1$ carries the information that T_2 is inhabited. Note that the target type may depend on the signal type. For example,

$$\left[\begin{array}{l} \text{signal} \quad : \quad \left[\begin{array}{l} x \quad : \quad \text{Ind} \\ y \quad : \quad \text{Ind} \\ c_1 \quad : \quad r(x,y) \end{array} \right] \\ \text{target} \quad : \quad \left[\begin{array}{l} c_2 \quad : \quad r'(\text{signal}.x, \text{signal}.y) \end{array} \right] \end{array} \right]$$

This means we do not know what the target type is until we have determined a particular signal. Using dependent types in this way yields an interesting perspective on the problem in channel theory of relating one type to another. Essentially channel theory seems to be starting from a theory of independent types and has problems precisely because it does not have dependent types.

5. Records and attitudes

5.1. ATTITUDINAL OBJECTS AS TYPES

Our first simple treatment of attitudes has record types as the object of the attitude (corresponding approximately to a situation semantics treatment where attitudes are taken to be relations between individuals and infons). Note that this is still essentially Austinian as the “propositions” are “of something” unlike e.g. sets of possible worlds or propositions as basic objects in property theory. Austinian propositions from situation theory could be represented as pairs of records and record types (coded as a record). However, here we will explore the simpler treatment without representing the additional “Austinian situation”. The situation will, however, play a role in determining the conditions under which auxiliary notions related to the object of the attitude will hold, e.g. truth of a belief, success for a search.

Here is a type corresponding to *A girl believes that a man owns a donkey*.⁹

$$\left[\begin{array}{l} x \\ c_1 \\ p = \left[\begin{array}{l} y : Ind \\ c_3 : \text{man}(y) \\ z : Ind \\ c_4 : \text{donkey}(z) \\ c_5 : \text{own}(y, z) \end{array} \right] \\ c_2 \end{array} \right] : \begin{array}{l} Ind \\ \text{girl}(x) \\ RecType \\ \text{believe}(x, p) \end{array}$$

The girl’s belief (the record type) is *true* of a record r (the part of the world or situation which the belief is about) just in case r is of type

$$\left[\begin{array}{l} y : Ind \\ c_3 : \text{man}(y) \\ z : Ind \\ c_4 : \text{donkey}(z) \\ c_5 : \text{own}(y, z) \end{array} \right]$$

This corresponds to a judgement in type theory or an Austinian proposition in situation semantics. The belief is true *simpliciter* if and only if there is such an r . This corresponds to a Russellian proposition in situation semantics.

⁹ As with the use of record types as first class citizens in our treatment of constraints, we are suppressing the superscript representing order of stratification when we write *RecType*.

5.2. INTENSIONAL VERBS

Montague's intensional verbs were not treated in classical situation semantics (Barwise and Perry, 1983). Our proposal relates them more directly to propositional attitudes like *believe* than Montague's original treatment where *seek* was treated as a relation between individuals and properties of properties. Here is a type corresponding to *A girl seeks a unicorn*.

$$\left[\begin{array}{l} x \\ c_1 \\ p = \left[\begin{array}{l} y : Ind \\ c_2 : unicorn(y) \end{array} \right] \\ c_3 \end{array} \right. \begin{array}{l} : Ind \\ : girl(x) \\ : RecType \\ : seek(x,p) \end{array} \left. \right]$$

Suppose that r is a record of this type. r' is a *successful outcome* for $r.x$'s search just in case r' is of type (for some label c_4)

$$\left[\begin{array}{l} y : Ind \\ c_2 : unicorn(y) \\ c_4 : find(r.x,y) \end{array} \right]$$

The girl's search would be *successful* just in case there is such an r' .

It may appear that this analysis is harder to derive compositionally than the one proposed for belief. However, we will show that by using Montague's λ -calculus techniques for compositional semantics this analysis results naturally by reduction from a compositional analysis following Montague's property of properties proposal.

5.3. PERCEPTION COMPLEMENTS

One of the greatest successes of situation semantics was the analysis of perception complements where a distinction was made between *see that* (*a man sees that a donkey kicked a farmer*) and *see* followed by a naked infinitive complement (*a man sees a donkey kick a farmer*). Barwise and Perry (1983) argued that in the former the man has only to have seen evidence that the kicking took place (e.g. donkey-hoof marks on the farmer's legs) whereas in the latter the man has to have seen the actual kicking situation (even if he did not realize that was what he was seeing). In our analysis of the distinction we exploit the fact that type theory allows us not only to refer to objects of a given type but also to the type itself. The *see that* construction is analysed in the same way as *believe*, i.e. as a relation between an individual and a record type. The verb *see* followed by a naked infinitive is analysed as a relation between

an individual and a situation (record) of the corresponding type. Thus for *a man sees that a donkey kicked a farmer* we have:

$$\left[\begin{array}{l} x \\ c_1 \\ p = \left[\begin{array}{l} y : Ind \\ c_2 : donkey(y) \\ z : Ind \\ c_3 : farmer(z) \\ c_4 : kick(y, z) \end{array} \right] \\ c_5 \end{array} \right. : \begin{array}{l} Ind \\ man(x) \\ RecType \\ see_t(x, p) \end{array} \left. \right]$$

For *a man sees a donkey kick a farmer* we have:

$$\left[\begin{array}{l} x : Ind \\ c_1 : man(x) \\ s : \left[\begin{array}{l} y : Ind \\ c_2 : donkey(y) \\ z : Ind \\ c_3 : farmer(z) \\ c_4 : kick(y, z) \end{array} \right] \\ c_5 : see_i(x, s) \end{array} \right]$$

This latter requires perception of a donkey-kicking-farmer situation and thereby also requires the existence of such a situation to perceive, i.e. see_i is required to be veridical for the same reasons it is veridical in the original Barwise and Perry treatment. The analysis of see_t does not of itself require it to be veridical. If we wish it to be so we can treat it in the same way as we would treat *know* in contrast to *believe*. That is, we require that if the type $see_t(x, p)$ (or $know(x, p)$) is realised, i.e. there is something of that type, then p is also realised.

5.4. TYPES ARE NOT ENOUGH

Types, here in the form of record types, thus seem to be providing us with a reasonable view of both sentence interpretation and the objects of attitudes which is fundamentally both Austinian (in that a type holds *of* something) and “innocent” in Barwise and Perry’s sense, i.e. the interpretation of attitude complements is the same as the interpretation of non-embedded sentences¹⁰. However, types are not quite what we

¹⁰ While innocence is no doubt a desirable property I have never believed that situation semantics was distinct from classical Montague semantics in this respect even though it might appear so from Montague’s formulation of his semantics in PTQ (Montague, 1974). Montague’s semantics can be equivalently formulated as compositionally assigning intensions (corresponding to Barwise and Perry’s “content”) to phrases rather than their extensions. This is innocent in Barwise and Perry’s sense.

need. Consider *Sandy kicked a farmer* which includes the proper named *Sandy*. We might be tempted to represent the content of this as

$$\left[\begin{array}{l} y : Ind \\ c_2 : \text{named}(y, \text{"Sandy"}) \\ z : Ind \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(y, z) \end{array} \right]$$

However, this would not distinguish the content of this sentence from *Some individual named "Sandy" kicked a farmer*. What we need is a way of showing that an utterance of the proper name *Sandy* presupposes a context in which there is an individual named "Sandy". A context is a situation (here modelled as a record). An utterance of *Sandy* presupposes that there is a context of the type

$$\left[\begin{array}{l} y : Ind \\ c_2 : \text{named}(y, \text{"Sandy"}) \end{array} \right]$$

We can think of this type as representing the presupposition associated with *Sandy*, i.e. the presupposition that there is some individual named "Sandy".

The meaning of *Sandy kicked a farmer* is a function which takes a context r of this type as argument and returns the following record type:

$$\left[\begin{array}{l} z : Ind \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(r.y, z) \end{array} \right]$$

This function is represented:

$$\lambda r : \left[\begin{array}{l} y : Ind \\ c_2 : \text{named}(y, \text{"Sandy"}) \end{array} \right] \left(\left[\begin{array}{l} z : Ind \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(r.y, z) \end{array} \right] \right)$$

A function from records to record types (of which this is an example) is called a *family of record types*. Such a family has the type $(R)RecType$ where R is some record type. Using families of record types as meanings, that is functions from contexts (modelled as records) to contents (modelled as record types), gives us a view of meaning which is essentially similar to the Montague-Kaplan indexical approach (Montague, 1974, Kaplan, 1989). One important difference, however, is that the

constraints on context are indefinitely extendable (in the sense that the type of the context can have any number of fields). In the classical indexical approach there is a fixed finite number of indices, e.g. speaker, hearer, location and time of utterance etc. One of the important claims in situation semantics is that there is no upper limit on the number of dependencies the content of an utterance may have on the context and that different sentences will have different numbers and kinds of dependencies. This is particularly important in connection with the claim that each noun-phrase utterance in a sentence may introduce a new resource situation (Cooper, 1996).

Meanings construed as families of record types will have different types depending on the domain of the function. This is because, following standard practice in Martin-Löf type theory, we introduce only total function types. If we were, in addition, to include partial function types, e.g. letting $\{T_1\}T_2$ represent the type of *partial* functions from objects of type T_1 to objects of type T_2 , then each family of record types would have the partial function type $\{Rec\}RecType$ in addition to the total function type $(R)RecType$ for some record type R . This would have the consequence that the meanings of sentences would always have type $\{Rec\}RecType$ no matter how many fields the context record type contains. However, I believe that a preferable solution is to introduce polymorphic types in a way that is fairly standard in type theory. The polymorphic function type $\exists R.(R)RecType$ contains functions which map objects which belong to some record type to record types. Note that this is potentially more restrictive than allowing all partial functions from records to record types since there may be partial functions whose domain does not correspond to the extension of any type.

5.5. INTENTIONAL IDENTITY

There are arguments that the objects of the attitudes should also be treated as families. Perhaps the clearest example of this is where one agent's attitude may depend on another agent's attitude to provide the context type associated with the attitudinal object. Cases of this are known as intentional identity (Geach, 1967). Geach's original example is: *Hob thinks a witch has blighted Bob's mare, and Nob wonders whether she (the same witch) has killed Cob's sow*. Fauconnier (1985) provides examples which involve an intensional verb such as: *Sandy believes there's a mouse in the bathroom and John is looking for it*. Consider the discourse: *A man believes that a donkey kicked a farmer. A woman believes that it was angry*. We might represent the dependence of the woman's belief on the man's belief by representing the content of the discourse as follows:

$$\left[\begin{array}{l}
 x \\
 c_1 \\
 p_1 = \left[\begin{array}{l}
 y : Ind \\
 c_2 : donkey(y) \\
 z : Ind \\
 c_3 : farmer(z) \\
 c_4 : kick(y, z)
 \end{array} \right] \\
 c_5 \\
 w \\
 c_6 \\
 p_2 = \lambda r: p_1([c_7 : angry(r.y)]) \\
 c_8
 \end{array} \right. \begin{array}{l}
 : Ind \\
 : man(x) \\
 : RecType \\
 : believe(x, p_1) \\
 : Ind \\
 : woman(w) \\
 : (p_1) RecType \\
 : believe(w, p_2)
 \end{array}$$

Here the woman's belief is a family of record types depending on a record of the type characterising the man's belief, i.e. a context in which there is a farmer who kicks a donkey. The notion of dependence we discuss here follows quite closely the analysis of belief and intentional identity presented in Ranta (1994) in terms of type theoretical contexts. We can represent a case with an intensional verb in a similar way – *A man believes that a donkey kicked a farmer. A woman is looking for it.*

$$\left[\begin{array}{l}
 x \\
 c_1 \\
 p_1 = \left[\begin{array}{l}
 y : Ind \\
 c_2 : donkey(y) \\
 z : Ind \\
 c_3 : farmer(z) \\
 c_4 : kick(y, z)
 \end{array} \right] \\
 c_5 \\
 w \\
 c_6 \\
 p_2 = \lambda r: p_1([u=r.y : Ind]) \\
 c_7
 \end{array} \right. \begin{array}{l}
 : Ind \\
 : man(x) \\
 : RecType \\
 : believe(x, p_1) \\
 : Ind \\
 : woman(w) \\
 : (p_1) RecType \\
 : seek(w, p_2)
 \end{array}$$

Here the woman's search depends on the man's belief.

According to this presentation we now have two kinds of attitudinal objects: those which are non-dependent (of type *RecType*) and those which are dependent (of type $(R)RecType$) for some record type R . However, we can make the analysis more uniform by having all attitudinal objects be families of types. Those which we have so far represented as a record type will be constant functions that map *any* record (i.e., object of type *Rec*) to that type. Intensional predicates such as *believe* and *seek* will have a second argument of the polymorphic type $\exists R. (R)RecType$.

Once we see that attitudinal objects can depend on other attitudinal objects we see that there can be arbitrarily long chains of such dependencies. Consider:

A man believes a donkey kicked a farmer. A woman believes a dog chased it. A boy is looking for the dog.

Here the boy's search depends on the woman's belief which in turn depends on the man's belief. Intuitively we can think of the dog which the boy is looking for as depending on a hypothetical context built up from the man's and woman's beliefs. It is clear in the following example that objects relating to both these beliefs are available in the hypothetical context for the boy's attitude.

A man believes a donkey kicked a farmer. A woman believes a dog chased it. A boy thinks the dog belongs to the farmer.

If hypothetical contexts are to be represented by the domain restriction on families, how can we get this cumulative effect for the hypothetical context? An important notion that enables us to solve this is that of a fixed point for a family of types. If \mathcal{T} is a family of types we say that a is a *fixed point for \mathcal{T}* just in case $a : \mathcal{T}(a)$. In the case of families of record types it is straightforward to compute what the type of the fixed points should be. Suppose that \mathcal{R} is the family

$$\lambda r: \left[\begin{array}{l} y : Ind \\ c_2 : donkey(y) \\ z : Ind \\ c_3 : farmer(z) \\ c_4 : kick(y, z) \end{array} \right] \left(\left[\begin{array}{l} c_7 : angry(r.y) \end{array} \right] \right)$$

Then

$$\left[\begin{array}{l} y : Ind \\ c_2 : donkey(y) \\ z : Ind \\ c_3 : farmer(z) \\ c_4 : kick(y, z) \\ c_7 : angry(y) \end{array} \right]$$

will be the type of all and only the fixed points of \mathcal{R} . We will call this the *fixed point type of \mathcal{R}* . Intuitively, the fixed point type of a family is obtained by extending the type of the domain of the family with the dependent type that characterises its range.¹¹ We will use $\mathcal{F}(\mathcal{R})$ to

¹¹ This makes it appear that fixed point type formation corresponds to abstraction over duplex conditions in DRT (Kamp and Reyle, 1993, p. 379f).

represent the fixed point type of a family of record types \mathcal{R}^{12} . This gives us a way of incrementing hypothetical contexts. Consider the following where all attitudinal objects are treated as families:

$$\left[\begin{array}{ll} x & : \text{Ind} \\ c_1 & : \text{man}(x) \\ p_1 = \lambda r: \text{Rec} \left(\begin{array}{l} y : \text{Ind} \\ c_2 : \text{donkey}(y) \\ z : \text{Ind} \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(y, z) \end{array} \right) & : (\text{Rec})\text{RecType} \\ c_5 & : \text{believe}(x, p_1) \\ w & : \text{Ind} \\ c_6 & : \text{woman}(w) \\ p_2 = \lambda r: \mathcal{F}(p_1) \left(\begin{array}{l} u : \text{Ind} \\ c_7 : \text{dog}(u) \\ c_8 : \text{chase}(u, r.y) \end{array} \right) & : (\mathcal{F}(p_1))\text{RecType} \\ c_9 & : \text{believe}(w, p_2) \\ v & : \text{Ind} \\ c_{10} & : \text{boy}(v) \\ p_3 = \lambda r: \mathcal{F}(p_2) ([c_{11} : \text{own}(r.z, r.u)]) & : (\mathcal{F}(p_2))\text{RecType} \\ c_{12} & : \text{believe}(v, p_3) \end{array} \right]$$

Here $\mathcal{F}(p_1)$ is

$$\left[\begin{array}{l} y : \text{Ind} \\ c_2 : \text{donkey}(y) \\ z : \text{Ind} \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(y, z) \end{array} \right]$$

and $\mathcal{F}(p_2)$ is

$$\left[\begin{array}{l} y : \text{Ind} \\ c_2 : \text{donkey}(y) \\ z : \text{Ind} \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(y, z) \\ u : \text{Ind} \\ c_7 : \text{dog}(u) \\ c_8 : \text{chase}(u, y) \end{array} \right]$$

¹² In formulating this precisely we need to make sure that the domain and range types of \mathcal{R} do not have labels in common.

Much of human belief and other attitudes must depend on attitudes of others in this way and I suspect that this simple technique using families and fixed points will give us a powerful tool for analysing these dependencies¹³.

5.6. FRAMES OF MIND

Another use for families of record types is to model what Barwise and Perry call frames of mind in Chapter 10 of *Situations and Attitudes*. For example, on p. 252 they give a treatment of Kripke's (1979) Pierre who believes that Londres is pretty and that London is not pretty, not realising that Londres and London are in fact the same city. Here is Pierre's frame of mind modelled as a family of record types.

$$\lambda r : \left[\begin{array}{l} x \quad : \quad \text{Ind} \\ c_1 \quad : \quad \text{named}(x, \text{"Londres"}) \\ y \quad : \quad \text{Ind} \\ c_2 \quad : \quad \text{named}(y, \text{"London"}) \end{array} \right] \left(\left[\begin{array}{l} c_3 \quad : \quad \text{pretty}(r.x) \\ c_4 \quad : \quad \neg\text{pretty}(r.y) \end{array} \right] \right)$$

There is nothing inconsistent or irrational about this. It is possible to provide it with a context (what Barwise and Perry call a *setting*) where 'x' and 'y' are assigned to distinct cities. Contrast it with a similar frame of mind which contains a contradiction.

$$\lambda r : \left[\begin{array}{l} x \quad : \quad \text{Ind} \\ c_1 \quad : \quad \text{named}(x, \text{"Londres"}) \\ c_2 \quad : \quad \text{named}(x, \text{"London"}) \end{array} \right] \left(\left[\begin{array}{l} c_3 \quad : \quad \text{pretty}(r.x) \\ c_4 \quad : \quad \neg\text{pretty}(r.x) \end{array} \right] \right)$$

Here an agent who has this frame of mind is committed to the belief that some object is both pretty and not pretty no matter what context is provided. Now consider the intended context for Pierre's belief

$$\left[\begin{array}{l} x \quad = \quad \text{london} \\ c_1 \quad = \quad \text{pf}_1 \\ y \quad = \quad \text{london} \\ c_2 \quad = \quad \text{pf}_2 \end{array} \right]$$

where pf_1 is a proof that London is named "Londres" and pf_2 is a proof that London is named "London". This record is an appropriate argument to Pierre's frame of mind and the pair consisting of Pierre's

¹³ I suspect also that cases that have been referred to in the literature as modal subordination (Roberts, 1989 and subsequent literature) can be treated in a similar way.

frame of mind and this context (or setting) corresponds to what Barwise and Perry would call Pierre's *mental state*. If we apply Pierre's frame of mind to its context we obtain the *content* of his mental state:

$$\left[\begin{array}{l} c_3 : \text{pretty}(\text{london}) \\ c_4 : \neg\text{pretty}(\text{london}) \end{array} \right]$$

In contrast to his frame of mind this does contain a contradiction.

This approach to representing frames of mind and mental states is essentially similar to that proposed by Cooper and Ginzburg (1996) in terms of situation theory. However, the present type theoretical proposal has a number of advantages over our earlier attempt. For example, we no longer need a notion of restricted object and the effect of simultaneous abstraction is achieved by abstracting over records.

5.7. RECORD TYPES ARE A LITTLE TOO MUCH

Proposals to analyse attitudes in terms of structured semantic objects run the risk of being too finegrained, that is, creating more attitudinal objects than can be motivated intuitively. This proposal is no exception to this. Two record types are distinct if they differ only in the labels they contain, yet there is in general no motivation for considering different labellings as corresponding to different attitudinal objects. However, two record types which differ only in their labels are equivalent in the sense that they both uniquely determine the same Σ -types.

Consider the following two distinct record types:

$$\left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{man}(x) \\ y : \text{Ind} \\ c_2 : \text{donkey}(y) \\ c_3 : \text{own}(x, y) \end{array} \right]$$

$$\left[\begin{array}{l} y : \text{Ind} \\ c_3 : \text{man}(y) \\ z : \text{Ind} \\ c_4 : \text{donkey}(z) \\ c_5 : \text{own}(y, z) \end{array} \right]$$

While these are distinct record types they both intuitively correspond to the same commitment about the nature of the world. That is, if there is some record of either of these types then there is a man who owns a

donkey. This commitment is represented in type-theoretical terms by a Σ -type:

$$(\Sigma x : Ind)(\Sigma c_1 : \text{man}(x))(\Sigma y : Ind)(\Sigma c_2 : \text{donkey}(y)) \\ (\Sigma c_3 : \text{own}(x, y)) \text{True}$$

This has an obvious exact correspondence to the record type. The difference is that in a Σ -type what corresponds to the labels are in fact bound variables. Thus the following is a different representation of the same Σ -type:

$$(\Sigma y : Ind)(\Sigma c_3 : \text{man}(y))(\Sigma z : Ind)(\Sigma c_4 : \text{donkey}(z)) \\ (\Sigma c_5 : \text{own}(y, z)) \text{True}$$

However, there is a way in which Σ -types are more finegrained than our record types. Order is significant in Σ -types whereas we formulate our records as sets of fields (partially ordered by dependence). Thus

$$(\Sigma y : Ind)(\Sigma c_2 : \text{donkey}(y))(\Sigma x : Ind)(\Sigma c_1 : \text{man}(x)) \\ (\Sigma c_3 : \text{own}(x, y)) \text{True}$$

is distinct from the first Σ -type. However, these two Σ -types are equivalent in an important sense. The nature of the equivalence is made clear when we consider the conditions under which an object is a member of a Σ -type. $a : (\Sigma x : T_1)T_2(x)$ just in case $a = \langle b, p \rangle$ where $b : T_1$ and $p : T_2(b)$. Thus members of our two Σ -types are respectively of the form:

$$\langle a, \langle p_1, \langle b, \langle p_2, \langle p_3, \top \rangle \rangle \rangle \rangle \rangle \\ \langle b, \langle p_2, \langle a, \langle p_1, \langle p_3, \top \rangle \rangle \rangle \rangle \rangle$$

where a and b are of type Ind , p_1 is of type $\text{man}(a)$, p_2 is of type $\text{donkey}(b)$, p_3 is of type $\text{own}(a, b)$ and $\top : \text{True}$. If we take the result of *flattening* these two objects, i.e. the set of the basic elements in the sequences, and remove the tautology \top which does not place any constraint on the world, we obtain the same set for both Σ -types:

$$\{a, b, p_1, p_2, p_3\}$$

This corresponds to the intuition that they make the same commitment about the nature of the world: “there’s a man and a donkey and the man owns the donkey”.

We define a function σ from record types to sets of Σ -types such that if R is

$$\left[\begin{array}{l} \ell_1 \quad : \quad T_1 \\ \dots \\ \ell_n \quad : \quad T_n \end{array} \right]$$

$\sigma(R)$ is a set of Σ -types consisting of all and only Σ -types of the form

$$(\Sigma \ell_i : T_i') \dots (\Sigma \ell_m : T_m') \text{True}$$

where $T_i \dots T_m$ is a permutation of $T_1 \dots T_n$ which respects the partial order induced on $T_1 \dots T_n$ by dependency and where for any j , $T_j' \in \sigma(T_j)$ if T_j is a record type and T_j otherwise. Two record types, R_1 and R_2 are Σ -equivalent just in case $\sigma(R_1) = \sigma(R_2)$. We can then say that any two Σ -equivalent record types represent the same attitudinal object. More intuitively, perhaps, we can think of a record type as a *labelling* of an attitudinal object.

Note that this notion of Σ -equivalence is getting us very close to traditional notions of truth conditions in that it is getting us to basic components of what has to be true about the world and stripping away extraneous structure related to, among other things, the order of presentation. It does not, however, have the problems with logical equivalence that classical possible worlds semantics has. For example, record types corresponding to true mathematical propositions will not necessarily be Σ -equivalent since they may involve different objects and different types. In this sense it is very close to the kind of intensionality which situation semantics was striving after as it was presented in *Situations and Attitudes*.

6. Interpreting natural language

In order to show how compositional semantics works we will present a fragment that covers the phenomena and the analysis presented in sections 5.1–5.3.

6.1. TYPE THEORY

6.1.1. Records

A *record* is a finite set of ordered pairs (called *fields*) which is the graph of a function. If r is a record and $\langle \ell, v \rangle$ is a field in r we call ℓ a *label* and v a *value* in r and we use $r.\ell$ to denote v . We will use a tabular format to represent records. A record with fields $\langle \ell_1, v_1 \rangle, \dots, \langle \ell_n, v_n \rangle$ is displayed as

$$\begin{bmatrix} \ell_1 & = & v_1 \\ \dots & & \\ \ell_n & = & v_n \end{bmatrix}$$

A value may itself be a record and paths may extend into embedded records. A record which contains records as values is called a *complex record* and otherwise a record is *simple*. Values which are not records are called *leaves*. Consider as an example the record r which is

$$\left[\begin{array}{l} f \\ g \end{array} = \left[\begin{array}{l} f \\ g \\ h \end{array} = \left[\begin{array}{l} ff \\ gg \\ h \end{array} = \begin{array}{l} a \\ b \\ a \\ d \end{array} \end{array} \right] \right]$$

Among the paths in r are $r.f$, $r.g.h$ and $r.f.f.f$ which denote, respectively,

$$\left[\begin{array}{l} f \\ g \end{array} = \left[\begin{array}{l} ff \\ gg \end{array} = \begin{array}{l} a \\ b \end{array} \right] \right]$$

$$\left[\begin{array}{l} g \\ h \end{array} = \begin{array}{l} a \\ d \end{array} \right]$$

and a . The set of leaves of r , also known as its *extension* (those objects other than labels which it contains), is $\{a, b, c, d\}$. The bag (or multiset) of leaves of r , also known as its *multiset extension*, is $\{a, a, b, c, d\}$. A record may be regarded as a way of labelling and structuring its extension. Two records are (*multiset*) *extensionally equivalent* if they have the same (multiset) extension. Two important, though trivial, facts about records are:

Flattening. For any record r , there is a multiset extensionally equivalent simple record. We can define an operation of flattening on records which will always produce an equivalent simple record. In the case of our example, the result of flattening is

$$\left[\begin{array}{l} f.f.f.f \\ f.f.gg \\ f.g \\ g.h.g \\ g.h.h \end{array} = \begin{array}{l} a \\ b \\ c \\ a \\ d \end{array} \right]$$

Relabelling. For any record r , if $\pi_1.\ell.\pi_2$ is a path π in r , and $\pi_1.\ell'.\pi_2'$ is *not* a path in r (for any π_2'), then substituting ℓ' for the occurrence of ℓ in π results in a record which is multiset equivalent to r .

We could, for example, substitute k for the second occurrence of g in the path $g.h.g$ in our example record.

$$\left[\begin{array}{l} f \\ g \end{array} = \left[\begin{array}{l} f \\ g \end{array} = \left[\begin{array}{l} ff \\ gg \end{array} = \left[\begin{array}{l} a \\ b \end{array} \right] \right] \right] \\ \left[\begin{array}{l} g \\ h \end{array} = \left[\begin{array}{l} k \\ h \end{array} = \left[\begin{array}{l} a \\ d \end{array} \right] \right] \right] \end{array} \right]$$

6.1.2. Types and Objects

A *system of dependent types with proof and record types* is a family of quintuples indexed by the natural numbers:

$$\langle \mathbf{Type}^n, \mathbf{BType}^n, \langle \mathbf{PfType}^n, \mathbf{Pred}^n, \mathit{Ariety}^n \rangle, \langle \mathbf{RecType}^n, \mathbf{Labels} \rangle \langle A^n, F^n \rangle \rangle_{n \in \mathit{Nat}}$$

where:

1. \mathbf{Type}^n is the set of types of order n , defined by the recursive definition *A* below.
2. \mathbf{BType}^n , the set of basic types, is a subset of \mathbf{Type}^n for any n .
3. \mathbf{PfType}^n , the set of basic types of proof of order n , is a subset of \mathbf{Type}^n defined with respect to a set \mathbf{Pred}^n of predicates of order n and a function Ariety^n which assigns to each member of \mathbf{Pred}^n a finite tuple of members of \mathbf{Type}^n . If $P \in \mathbf{Pred}^n$, then $P \in \mathbf{Pred}^{n+1}$ and if $\mathit{Ariety}^n(P) = \langle T_1^n, \dots, T_m^n \rangle$, then $\mathit{Ariety}^{n+1}(P) = \langle T_1^{n+1}, \dots, T_m^{n+1} \rangle$. The sets \mathbf{PfType}^n are defined by the definition *B* below.
4. $\mathbf{RecType}^n$, the set of record types of order n , which is a subset of \mathbf{Type}^n defined with respect to a countably infinite set \mathbf{Labels} of objects used as labels. The sets $\mathbf{RecType}^n$ are defined by the recursive definition *C* below.
5. $\langle A^n, F^n \rangle$ is a model where A^n is a function from \mathbf{BType}^n to sets (of inhabitants of the basic types) and F^n is a function from \mathbf{PfType}^n to sets of objects (proofs of that type) such that if $a \in F^n(T)$ then $a \in F^{n+1}(T)$.

We use the notation $a : T$ to represent that the object a is of type T . This notion is defined recursively in association with the definitions of \mathbf{Type}^n , \mathbf{PfType}^n and $\mathbf{RecType}^n$.

Note that the set of types depends on the model whereas in standard model theory it is possible to define the set of types independently of the model.¹⁴

¹⁴ It might be suspected that this lack of independence between types and models will exclude classical modal logic based approaches to modality and intensionality.

A. Definition of \mathbf{Type}^n

1. if T is a member of \mathbf{Type}^n , then T is also a member of \mathbf{Type}^{n+1} .
2. if T is a member of \mathbf{BType}^n , then T is also a member of \mathbf{BType}^{n+1} .
3. If T is a member of \mathbf{BType}^n , then T is a member of \mathbf{Type}^n .
If T is a member of \mathbf{BType}^n , then $a : T$ iff $a \in A^n(T)$.
4. $Type^n$, the type of Types of order n , and $RecType^n$, the type of record types of order n , are members of \mathbf{Type}^{n+1} .
 $T : Type^n$ iff $T \in \mathbf{Type}^n$
 $T : RecType^n$ iff $T \in \mathbf{RecType}^n$
5. if \mathcal{P} is a member of \mathbf{PfType}^n then \mathcal{P} is also a member of \mathbf{Type}^n .
6. if R is a member of $\mathbf{RecType}^n$ then R is also a member of \mathbf{Type}^n .
7. if T_1 and T_2 are members of \mathbf{Type}^n , then $(T_1)T_2$, the type of functions from objects of type T_1 to objects of type T_2 , is a member of \mathbf{Type}^n .
 $f : (T_1)T_2$ iff f is a function whose domain is $\{a \mid a : T_1\}$ and whose range is included in $\{a \mid a : T_2\}$.
8. if T is a member of \mathbf{Type}^n and $\mathcal{F} : (T)Type^n$, then $(a : T)\mathcal{F}(a)$, the type of dependent functions from objects a of type T to $\mathcal{F}(a)$, is a member of \mathbf{Type}^{n+1} .
 $f : (a : T)\mathcal{F}(a)$ iff f is a function whose domain is $\{a \mid a : T\}$ and such that for any a in the domain of f , $f(a) : \mathcal{F}(a)$.
- 9.¹⁵ if T_1 and T_2 are members of \mathbf{Type}^n , $T_1 \vee T_2$, the join (disjunction) of T_1 and T_2 , is a member of \mathbf{Type}^n .
 $a : T_1 \vee T_2$ iff $a : T_1$ or $a : T_2$.
- 10.¹⁵ if T is a member of \mathbf{Type}^n , then $[T]$, the type of lists each of whose members are of type T , is a member of \mathbf{Type}^n .
 $[a_1, \dots, a_n] : [T]$ iff for each i , $1 \leq i \leq n$, $a_i : T$.

However, it appears we could imitate Montague's approach to intensionality and modality by letting F assign to each member of \mathbf{PfType} a function from a set of possible worlds to sets of proofs. We do not recommend this, however, as we believe that the type system as defined provides a superior approach to intensionality.

¹⁵ This clause is not used in the examples of this paper but is included here for the sake of completeness.

11. if T is a member of \mathbf{Type}^n and $x : T$, then T_x , a singleton type of x , is a member of \mathbf{Type}^n .

$$a : T_x \text{ iff } a = x$$

12. if \mathcal{T} is an expression containing one or more occurrences of R_i (i a natural number) such that the result of substituting any member S of $\mathbf{RecType}^{n-1}$ for free occurrences¹⁶ of R_i in \mathcal{T} , $\mathcal{T}[R_i/S]$, is a member of \mathbf{Type}^n , then $\exists R_i. \mathcal{T}^n$ is a member of \mathbf{Type}^n .

$$a : \exists R_i. \mathcal{T}^n \text{ iff there is some } S \text{ in } \mathbf{RecType}^{n-1} \text{ such that } a : \mathcal{T}[R_i/S].$$

B. Definition of \mathbf{PfType}^n

1. if P is a member of \mathbf{Pred} , $\text{Ariety}^n(P) = \langle T_1, \dots, T_m \rangle$ and a_1, \dots, a_m are such that $a_1 : T_1, \dots, a_m : T_m$, then $P(a_1, \dots, a_m) \in \mathbf{PfType}^n$.

$$\text{If } T = P(a_1, \dots, a_m) \text{ is a member of } \mathbf{PfType}^n, \text{ then } a : T \text{ iff } a \in F^n(T)$$

2. if T is a member of \mathbf{PfType}^n , then T is also a member of \mathbf{PfType}^{n+1} .

C. Definition of $\mathbf{RecType}^n$

For any n , $\mathbf{RecType}^n$ is itself a set of records in the general sense, i.e. a set of ordered pairs that constitute the graph of a function.

1. if R is a member of $\mathbf{RecType}^n$, then R is also a member of $\mathbf{RecType}^{n+1}$.

2. Rec^n (also represented by $[\]^n$) is a member of $\mathbf{RecType}^{n+1}$.

$$[\] \text{ (the empty record) : } \text{Rec}^0$$

$$\text{If } T \in \mathbf{RecType}^n \text{ and } r : T, \text{ then } r : \text{Rec}^n$$

3. if R is a member of $\mathbf{RecType}^n$, ℓ is a member of \mathbf{Labels} not occurring as a label in R and T is a member of \mathbf{Type}^n , then $R \cup \{ \langle \ell, T \rangle \}$ is a member of $\mathbf{RecType}^n$.

$$r : R \cup \{ \langle \ell, T \rangle \} \text{ iff } r : R, \langle \ell, a \rangle \text{ is a field in } r \text{ and } a : T.$$

¹⁶ i.e., not occurring within the scope of \exists .

4. if R is a member of **RecType**^{*n*}, ℓ is a member of **Labels** not occurring as a label in R , T_1, \dots, T_m are members of **Type**^{*n*}, $R.\pi_1, \dots, R.\pi_m$ are paths in R such that if $r : R$, then $r.\pi_1 : T_1, \dots, r.\pi_m : T_m$ and \mathcal{F} is a function of type $(a_1 : T_1) \dots (a_m : T_m) \mathbf{Type}^n$, then $R \cup \{ \langle \ell, \langle \mathcal{F}, \langle \pi_1, \dots, \pi_m \rangle \rangle \rangle \}$ is a member of **RecType**^{*n*}.

$r : R \cup \{ \langle \ell, \langle \mathcal{F}, \langle \pi_1, \dots, \pi_m \rangle \rangle \}$ iff $r : R$, $\langle \ell, a \rangle$ is a field in r and $a : \mathcal{F}(r.\pi_1, \dots, r.\pi_m)$.

For convenience we represent e.g. $\langle \lambda x \lambda y \text{ love}(x, y), \langle \pi_1, \pi_2 \rangle \rangle$ as $\text{love}(\pi_1, \pi_2)$.

- 5.¹⁵ if R is a member of **RecType**^{*n*}, ℓ is a member of **Labels** not occurring as a label in R , T, T_1, \dots, T_m are members of **Type**^{*n*}, $R.\pi_1, \dots, R.\pi_m$ are paths in R such that if $r : R$ then $r.\pi_1 : T_1, \dots, r.\pi_m : T_m$ and O is an m -place operation symbol denoting an operation o with arity $\langle T_1, \dots, T_m \rangle$ and range included in the set of objects of type T , then $R \cup \{ \langle \ell, T_{O(\pi_1, \dots, \pi_m)} \rangle \}$ is a member of **RecType**^{*n*}.

$r : R \cup \{ \langle \ell, T_{O(\pi_1, \dots, \pi_m)} \rangle \}$ iff $r : R$, $\langle \ell, a \rangle$ is a field in r and $a : T_{o(r.\pi_1, \dots, r.\pi_m)}$.

We represent a record type $\{ \langle \ell_1, T_1 \rangle, \dots, \langle \ell_n, T_n \rangle \}$ graphically as

$$\left[\begin{array}{l} \ell_1 \quad : \quad T_1 \\ \dots \\ \ell_n \quad : \quad T_n \end{array} \right]$$

In the case where T_i is a singleton type T'_x we allow a variant notation (corresponding to the manifest fields of Coquand et al., 2004)

$$\left[\ell_i=x \quad : \quad T' \right]$$

6.2. SYNTAX AND SEMANTICS

We use $f@a$ to represent application of the function f to the argument a and $\llbracket A \rrbracket$ to represent the interpretation of a phrase A (represented by a labelled bracketing according to standard linguistic conventions). *Ind* is to be the basic type of individuals. *True* is to be the basic type “truth” whose sole member is \top .

For the interpretation of common nouns and intransitive verbs we use the following notation which relates a record type to a family of record types (a function from records to record types):

$$\lambda^{\ell_i} \left[\begin{array}{l} \ell_1 \quad : \quad T_1 \\ \ell_2 \quad : \quad T_2(\ell_1) \\ \vdots \\ \ell_{i-1} \quad : \quad T_{i-1}(\ell_1, \ell_2, \dots, \ell_{i-2}) \\ \ell_i \quad : \quad T_i \\ \ell_{i+1} \quad : \quad T_{i+1}(\ell_1, \ell_2, \dots, \ell_{i-1}, \ell_i) \\ \vdots \\ \ell_n \quad : \quad T_n(\ell_1, \ell_2, \dots, \ell_{n-1}) \end{array} \right] =$$

$$\lambda r: [\ell_i: T_i] \left(\begin{array}{l} \ell_1 \quad : \quad T_1 \\ \ell_2 \quad : \quad T_2(\ell_1) \\ \vdots \\ \ell_{i-1} \quad : \quad T_{i-1}(\ell_1, \ell_2, \dots, \ell_{i-2}) \\ \ell_{i+1} \quad : \quad T_{i+1}(\ell_1, \ell_2, \dots, \ell_{i-1}, r.\ell_i) \\ \vdots \\ \ell_n \quad : \quad T_n(\ell_1, \ell_2, \dots, \ell_{i-1}, r.\ell_i, \ell_{i+1}, \dots, \ell_{n-1}) \end{array} \right)$$

where T_i is a non-dependent type. For the interpretation of transitive verbs we use a variant of the standard Montague treatment of extensional verbs corresponding to VP quantification in PTQ. We define transitive verb raising as follows:

$$\text{tvr} \left[\begin{array}{l} x \quad : \quad \text{Ind} \\ y \quad : \quad \text{Ind} \\ c \quad : \quad T(x,y) \end{array} \right] =$$

$$\lambda \mathcal{N}: (([x:\text{Ind}]) \text{RecType}) \text{RecType} \\ \lambda r_1: [x:\text{Ind}] (\mathcal{N} @ \lambda r_2: [x:\text{Ind}] ([c:T(r_1.x, r_2.x)]))$$

We define the interpretation rules:

$$\begin{aligned} \llbracket [\text{S NP VP}] \rrbracket &= \llbracket \text{NP} \rrbracket @ \llbracket \text{VP} \rrbracket \\ \llbracket [\text{VP V NP}] \rrbracket &= \llbracket \text{V} \rrbracket @ \llbracket \text{NP} \rrbracket \\ \llbracket [\text{VP V}_S \text{S}] \rrbracket &= \llbracket \text{V}_S \rrbracket @ \llbracket \text{S} \rrbracket \\ \llbracket [\text{VP V}_{NI} \text{NP VP}] \rrbracket &= \llbracket \text{V}_{NI} \rrbracket @ \llbracket \text{NP} \rrbracket @ \llbracket \text{VP} \rrbracket^{17} \\ \llbracket [\text{NP Det N}] \rrbracket &= \llbracket \text{Det} \rrbracket @ \llbracket \text{N} \rrbracket \end{aligned}$$

¹⁷ We ignore morphological details requiring the infinitive form of the complement VP in this presentation.

$$\begin{aligned}
\llbracket [\text{Det } a] \rrbracket &= \\
&\lambda R_1:([x:\text{Ind}])\text{RecType} \\
&\lambda R_2:([x:\text{Ind}])\text{RecType} \left(\begin{array}{l} \text{par} \quad : \quad [x:\text{Ind}] \\ \text{restr} \quad : \quad R_1 @ \text{par} \\ \text{scope} \quad : \quad R_2 @ \text{par} \end{array} \right) \\
\llbracket [\text{Det every}] \rrbracket &= \\
&\lambda R_1:([x:\text{Ind}])\text{RecType} \\
&\lambda R_2:([x:\text{Ind}])\text{RecType} \\
&\left(\left[f:(r : \begin{array}{l} \text{par} : [x:\text{Ind}] \\ \text{restr}:R_1 @ \text{par} \end{array}) R_2 @ r.\text{par} \right] \right) \\
\llbracket [\text{N man}] \rrbracket &= \lambda^x \begin{bmatrix} x & : & \text{Ind} \\ c & : & \text{man}(x) \end{bmatrix} \\
\llbracket [\text{N girl}] \rrbracket &= \lambda^x \begin{bmatrix} x & : & \text{Ind} \\ c & : & \text{girl}(x) \end{bmatrix} \\
\llbracket [\text{N donkey}] \rrbracket &= \lambda^x \begin{bmatrix} x & : & \text{Ind} \\ c & : & \text{donkey}(x) \end{bmatrix} \\
\llbracket [\text{N unicorn}] \rrbracket &= \lambda^x \begin{bmatrix} x & : & \text{Ind} \\ c & : & \text{unicorn}(x) \end{bmatrix} \\
\llbracket [\text{V owns}] \rrbracket &= \text{tvr} \begin{bmatrix} x & : & \text{Ind} \\ y & : & \text{Ind} \\ c & : & \text{own}(x,y) \end{bmatrix} \\
\llbracket [\text{V beats}] \rrbracket &= \text{tvr} \begin{bmatrix} x & : & \text{Ind} \\ y & : & \text{Ind} \\ c & : & \text{beat}(x,y) \end{bmatrix} \\
\llbracket [\text{V seeks}] \rrbracket &= \\
&\lambda \mathcal{N}:([x:\text{Ind}])\text{RecType}\text{RecType} \\
&\lambda r_1:[x:\text{Ind}] \\
&\left(\begin{array}{l} \text{p}=\mathcal{N} @ \lambda r_2:[x:\text{Ind}](\text{c:True}):\text{RecType} \\ \text{c:seek}(r_1.x, \text{p}) \end{array} \right) \\
\llbracket [\text{V}_S \text{ believes}] \rrbracket &= \\
&\lambda T:\text{RecType} \lambda r:[x:\text{Ind}]\left(\begin{array}{l} \text{p}=T \quad : \quad \text{RecType} \\ \text{c} \quad \quad : \quad \text{believe}(r.x, \text{p}) \end{array} \right) \\
\llbracket [\text{V}_S \text{ sees}] \rrbracket &= \lambda T:\text{RecType} \lambda r:[x:\text{Ind}]\left(\begin{array}{l} \text{p}=T \quad : \quad \text{RecType} \\ \text{c} \quad \quad : \quad \text{see}_t(r.x, \text{p}) \end{array} \right) \\
\llbracket [\text{V}_{NI} \text{ sees}] \rrbracket &= \\
&\lambda \mathcal{N}:([x:\text{Ind}])\text{RecType}\text{RecType} \\
&\lambda R:([x:\text{Ind}])\text{RecType} \\
&\lambda r:[x:\text{Ind}] \\
&\left(\begin{array}{l} \text{s} \quad : \quad \mathcal{N}@R \\ \text{c} \quad : \quad \text{see}_i(r.x, \text{s}) \end{array} \right)
\end{aligned}$$

SAMPLE DERIVATION: *a girl seeks a unicorn*¹⁸

a unicorn

$\lambda R_1:([x:Ind])RecType$

$\lambda R_2:([x:Ind])RecType \left(\begin{array}{l} \text{par} \quad : \quad [x : Ind] \\ \text{restr} \quad : \quad R_1 @ \text{par} \\ \text{scope} \quad : \quad R_2 @ \text{par} \end{array} \right)$

@

$\lambda r:[x:Ind]([c : unicorn(r.x)])$

=

$\lambda R_2:([x:Ind])RecType \left(\begin{array}{l} \text{par} \quad : \quad [x : Ind] \\ \text{restr} \quad : \quad [c : unicorn(\text{par}.x)] \\ \text{scope} \quad : \quad R_2 @ \text{par} \end{array} \right)$

¹⁸ Derivations make extensive use of β -conversion as is usual with Montague style compositional semantics. In defining β -conversion for record types one has to be careful to make sure that path-names in a record type which is embedded inside another record type are adjusted to obey the convention that all path names start from the root of the record type. We do not give an explicit definition of β -conversion in this paper.

seeks a unicorn

$$\lambda \mathcal{N}:([x:Ind]) RecType) RecType$$

$$\lambda r_1:[x:Ind]$$

$$\left(\begin{array}{l} p = \mathcal{N} @ \lambda r_2:[x:Ind]([c:True]):RecType \\ c:seek(r_1.x, p) \end{array} \right)$$

@

$$\lambda R_2:([x:Ind]) RecType \left(\begin{array}{l} par : [x : Ind] \\ restr : [c : unicorn(par.x)] \\ scope : R_2 @ par \end{array} \right)$$

=

$$\lambda r_1:[x:Ind]$$

$$\left(\begin{array}{l} p = \begin{array}{l} par : [x:Ind] \\ restr : [c:unicorn(par.x)] \\ scope: [c:True] \end{array} : RecType \\ c:seek(r_1.x, p) \end{array} \right)$$

a girl seeks a unicorn

$$\lambda R_2:([x:Ind]) RecType \left(\begin{array}{l} par : [x : Ind] \\ restr : [c : girl(par.x)] \\ scope : R_2 @ par \end{array} \right)$$

@

$$\lambda r_1:[x:Ind]$$

$$\left(\begin{array}{l} p = \begin{array}{l} par : [x:Ind] \\ restr : [c:unicorn(par.x)] \\ scope: [c:True] \end{array} : RecType \\ c:seek(r_1.x, p) \end{array} \right)$$

=

$$\left[\begin{array}{l} par : [x:Ind] \\ restr : [c:girl(par.x)] \\ scope: \begin{array}{l} p = \begin{array}{l} par : [x:Ind] \\ restr : [c:unicorn(par.x^{19})] \\ scope: [c:True] \end{array} : RecType \\ c:seek(par.x, scope.p) \end{array} \end{array} \right]$$

After flattening and removal of the field with type *True*, this type is Σ -equivalent (in the sense introduced in section 5.7) to the type introduced in section 5.2.

7. Conclusion

We have drawn a parallel between the central judgement in type theory that an object a is of type T and the Austinian notion of truth as it was used in situation semantics, i.e. the contents of truth bearing utterances are regarded as types which hold of a part of the world (an object). The rich notion of type which is to be found in Martin-Löf's type theory gives us an approach to intensionality which is more satisfactory than notion of intension as a function from possible worlds to extension as proposed by Montague and is similar in important respects to the kind of intensionality that Barwise and Perry aimed for in their presentation of situation semantics.

Ranta suggested that the notion of context in type theory can be used to correspond to situations and can be exploited in the treatment of semantic phenomena such as anaphora. Records give us explicit mathematical objects in type theory which can be used to model contexts (both in the type theoretical sense and in the intuitive sense employed in linguistic semantics). This allows contexts (i.e. records) to become first class members of type theoretic universes. For example, they can be typed and they can be included in the domains and ranges of functions. We have shown that an approach which exploits this view of context allows us to treat attitudes and perception complements following the analyses proposed by Barwise and Perry in situation semantics. In addition, we have shown that it is possible to treat intensional verbs as discussed by Montague and intentional identity as discussed by Geach.

In other work in progress²⁰ we have shown treatments for most of the phenomena in Montague's PTQ (modality, tense and adverbs remain to be worked out and there is still some discussion to be had about which version of negation and disjunction is most appropriate) and a basic fragment of DRT (including donkey anaphora and discourse

¹⁹ When types are introduced as objects (as in the treatment of intensional constructions) path names start from the root of the object type and special marking is needed to access paths elsewhere in the record type, as for example in the treatment of *A girl seeks a unicorn which loves her* or *de re* readings associated with intensional constructions. As none of our examples involve reference to paths outside of types embedded in this way we do not introduce such a convention here.

²⁰ See <http://www.ling.gu.se/~cooper/records>.

anaphora but not yet plural or tense and aspect). We have also (in Cooper, 2004) suggested how the treatment of generalised quantifiers based on classical ZF set theory could be imported into the type theoretical approach. We thus see this approach as quite promising for the development of a general semantic theory which can encompass a body of results in semantic theory which have been proposed in various frameworks since the seventies. The similarity of record types to feature structures provides the additional benefit of being able to incorporate aspects of feature-based grammar such as HPSG.

References

- Austin, J.L. (1961) Truth, in Urmsen and Warnock (1961), pp. 117–133.
- Barwise, Jon and Robin Cooper (1991) Simple Situation Theory and its Graphical Representation, Indiana University Logic Group Preprint No. IULG-91-8 and in Seligman (1991)
- Barwise, Jon and John Perry (1983) *Situations and Attitudes*, MIT Press.
- Barwise, Jon and Jerry Seligman (1997) *Information Flow: the Logic of Distributed Systems*, Cambridge Tracts in Theoretical Computer Science, 44, Cambridge University Press.
- van Benthem, Johan and Alice ter Meulen, eds., (1997) *Handbook of Logic and Language*, North Holland, Amsterdam and MIT Press, Cambridge, Mass.
- Cooper, Robin (1996) The Role of Situations in Generalized Quantifiers, in Lappin (1996).
- Cooper, Robin (2004) Dynamic generalised quantifiers and hypothetical contexts, in *Ursus Philosophicus*, a festschrift for Björn Haglund, Department of Philosophy, Göteborg University.
- Cooper, Robin (forthcoming) Records and record types in semantic theory, in *Journal of Logic and Computation*.
- Cooper, Robin and Jonathan Ginzburg (1996) A Compositional Situation Semantics for Attitude Reports, in Seligman and Westerståhl (1996)
- Coquand, Thierry, Randy Pollack and Makoto Takeyama (2004) A Logical Framework with Dependently Typed Records, *Fundamenta Informaticae*, XX, pp. 1–22.
- Davidson, Donald (1980) *Essays on Actions and Events*, Clarendon Press, Oxford.
- van Eijck, Jan and Hans Kamp (1997) Representing Discourse in Context, in van Benthem and ter Meulen (1997).
- Fauconnier, Gilles (1985) *Mental Spaces*, MIT Press, Cambridge, Mass.
- Gabbay, Dov and Franz Guenther, eds (1986) *Handbook of Philosophical Logic, Vol. III*, Reidel, Dordrecht.
- Geach, P.T. (1967) Intentional Identity, *Journal of Philosophy*, Vol. 64, pp. 627–32.
- Kamp, Hans and Uwe Reyle (1993) *From Discourse to Logic*, Kluwer, Dordrecht.
- Kaplan, David (1989) Demonstratives: An essay on the semantics, logic, metaphysics, and epistemology of demonstratives and other indexicals, in J. Almog, J. Perry, and H. K. Wettstein, editors, *Themes from Kaplan*, Oxford University Press, Oxford.

- Kohlhase, Michael, Susanna Kuschert and Manfred Pinkal (1996) A type-theoretic semantics for λ -DRT, in *Proceedings of the 10th Amsterdam Colloquium*, ed. by P. Dekker and M. Stokhof, ILLC, Amsterdam, pp. 479–498.
- Kripke, Saul (1979) A puzzle about belief, in Margalit, ed., (1979) pp. 239–83.
- Lappin, Shalom, ed., (1996) *The Handbook of Contemporary Semantic Theory*, Blackwell, Oxford.
- Larsson, Staffan (2002) *Issue-based Dialogue Management*, PhD thesis, Göteborg University.
- Margalit, A., ed (1979) *Meaning and Use*, Reidel, Dordrecht.
- Montague, Richard (1974) *Formal Philosophy: Selected Papers of Richard Montague*, ed. and with an introduction by Richmond H. Thomason, Yale University Press, New Haven.
- Muskens, Reinhard (1995) Combining Montague semantics and discourse representation, *Linguistics and Philosophy*, Vol. 19, pp. 143–186.
- Ranta, Arne (1991) Constructing possible worlds, *Theoria*, 57, pp. 77–99.
- Ranta, Arne (1994) *Type-Theoretical Grammar*, Clarendon Press, Oxford.
- Roberts, Craige (1989) Modal Subordination and Pronominal Anaphora in Discourse, *Linguistics and Philosophy*, Vol 12, No. 6, pp. 683–721.
- Sag, Ivan, Thomas Wasow and Emily Bender (2003) *Syntactic Theory*, Second Edition, CSLI Publications, Stanford.
- Seligman, Jerry, ed. (1991) *Partial and Dynamic Semantics III*, DYANA Deliverable R2.1.C, Centre for Cognitive Science, University of Edinburgh.
- Seligman, Jerry and Dag Westerståhl, eds (1996) *Logic, Language and Computation, Vol. 1*, CSLI Publications.
- Sundholm, Göran (1986) Proof Theory and Meaning, in Gabbay and Guentner (1986).
- Urmson, J.O. and G.J. Warnock, eds (1961) *Philosophical Papers*, Oxford University Press, Oxford.