Philosophical Communications, Web Series, No. 35, pp. 75-88
Dept. of Philosophy, Göteborg University, Sweden
ISSN 1652-0459

Kvantifikator för en Dag

Essays dedicated to Dag Westerståhl on his sixtieth birthday

A record type theoretic account of copredication and dynamic generalized quantification

Robin Cooper

Abstract

In a recent paper Asher and Pustejovsky propose a type theoretical approach to account for cases of copredication which had motivated Pustejovsky to introduce dot types in the Generative Lexicon. In this paper I will propose an alternative treatment to that given by Asher and Pustejovsky using type theory with records. I will suggest that using record types not only gives us a simple and intuitive account of dot types but also makes an important connection between copredication and the use of hypothetical contexts in dynamic generalized quantifiers I have proposed in earlier work. More generally it allows us to preserve the intuitive feature structure based accounts of lexical analysis proposed in Pustejovsky's original work on the Generative Lexicon while at the same time introducing functions and binding in the way that is necessary for giving formal accounts of compositional semantics.

1 Introduction

In[1] smart house dialogue system applications we want to be able to talk with networked devices such as stereos, video machines, lights, fire alarms etc. We want the language facility of the system to be centralised to at least some degree so that we can give instructions to more than one device at the same time, e.g. *if the phone rings turn down the volume on the stereo*. At the same time we wish to have plug and play facilities on our network so that we do not have to reprogram the whole system if we wish to add a new kind of device to the network.

[1]I am grateful for useful comments to Nicholas Asher, Tim Fernando, Stuart Shieber, two anonymous reviewers for the GSLT internal conference and participants in the LT seminar in Göteborg. In particular I am grateful to Dag Westerståhl for stimulating discussion not only of this work but of natural language semantics in general (and situation semantics and generalized quantifiers in particular) over the long period of our association. This work was supported by Vetenskapsrådet project number 2002-4879 *Records, types and computational dialogue semantics*, `http://www.ling.gu.se/~cooper/records/`.

Nevertheless, a new kind of device should add new language capabilities to the central system. One imagines that each device comes with its own language module which can somehow be compiled together with the central language processing unit for the house. How should something like this be organized? One way would be to organize devices into various types which determine the kinds of things you can say to them. For example, stereos, videocorders and lights all belong to the *OnOff* type meaning that you can turn them on and off. Stereos and TVs belong to the *Volume* type which means that you can turn the volume up or down, but you cannot modify the volume on lights. Note that stereos belong to both types. As a consequence one can say (1a) but (1b) does not make much sense if what is meant that you should turn up the volume on the lights.

(1) a. Turn on the stereo and turn up the volume (on it)

b. ?Turn on the light and turn up the volume (on it)

In a recent paper Asher and Pustejovsky (2005) propose a type theoretical approach to account for cases which are related to this example which had motivated Pustejovsky (1995) to introduce dot types. The intuitive idea behind dot types is that they are compositions of two types which nevertheless allow the two individual types to be recovered. Consider the examples in (2), based on examples given in Pustejovsky (1995) and Asher and Pustejovsky (2005).

(2) a. The lunch was delicious

b. The lunch took forever

delicious is a predicate that can be used of food but not events.

(3) a. The blancmange was delicious

b. ?IFK Göteborg's last game was delicious

The "?" in (3b) indicates that we have to work harder to get an intepretation of the sentence. Certainly, it is not impossible to think in terms of something like the taste of victory but games in general are not something we talk of as being delicious. Pustejovsky would say that *game* and/or *delicious* are being coerced to another type in order to get an interpretation. Similarly *take forever* is a predicate which holds of events and not of food.

(4) a. ?The blancmange took forever

b. IFK's last game took forever

Again we can certainly get intepretations out of (4a) but it involves some kind of coercion. It is not the actual food that takes forever but something like the eating of it, the preparing of it or waiting for it. In the case of *lunch* we do not feel that coercion is necessary. This word seems equally easy to interpret as representing food or as representing an event. Pustejovsky therefore associates it with a dot type *Food·Event*. This is meant to represent that it will simultaneously behave as something of type *Food* and something of type *Event* and accept predicates of food and predicates of events.

A natural view might be that this is simply a case of polysemy, that is, that *lunch* is ambiguous between a food interpretation and an event interpretation. If this were the case it would be hard to see what the motivation of the dot type would be. There would simply be two types one associated with each meaning of the word. However, Asher and Pustejovsky (2005) discuss in detail cases of copredication where one occurrence of the word simultaneously has both interpretations.[2] Compare (5a) with (5b) (both examples from Asher and Pustejovsky).

(5) a. The lunch was delicious but took forever

b. [!]The bank specializes in IPO's and is being quickly eroded by the river

The word *bank* is ambiguous between meaning a financial institution and the ground at the side of a river. A single occurrence of the word cannot be used in both senses. However, no such problems seem to occur with *lunch*. It seems that in some way natural language manages to treat lunch as a single object which is simultaneously an event and food.

In this paper I will propose an alternative treatment to that given by Asher and Pustejovsky using type theory with records of the kind that I have developed in Cooper (2004, 2005, forthcoming) based on work in Martin-Löf type theory (Tasistro, 1997, Betarte, 1998, Betarte and Tasistro 1998, Coquand *et al.*, 2004). I will suggest that using record types not only gives us a simple and intuitive account of dot types but also makes an important connection between copredication and the use of hypothetical contexts in dynamic generalized quantifiers proposed in Cooper (2004). More generally it allows us to preserve the original intuitive feature structure based accounts of lexical analysis proposed in Pustejovsky (1995) while at the same time introducing functions and binding in the way that is necessary for giving formal accounts of compositional semantics.

The rest of the paper is organized as follows. In section 2 I will attempt to argue against

[2]I believe that such examples were first reported in the literature by McCawley (1968).

an assumption that Asher and Pustejovsky make concerning the inconsistency of constituent types in dot types. In section 3 I will give a brief introduction to the kind of type theory we are employing and then, in section 4, show how this type theory can be used to give an account of dynamic generalized quantifiers. In section 5 we will show how the account can be extended to deal with the examples we have discussed in this introduction. Finally, we will draw some general conclusions for the Generative Lexicon.

2 Nondistributive predication and dot types

Asher (pc) has pointed out that part of the motivation for analyzing words like *lunch* in terms of dot types rather than conjunctive types is the assumption that the intersection of the two components T_1 and T_2 in a dot type $T_1 \cdot T_2$ is $\perp$. That is, T_1 and T_2 are inconsistent with each other. Certainly there is a robust intuition that anything which is food, i.e., objects and masses which might appear on a plate, is not an event and similarly that events are not food. You cannot, for example, put events on a plate. It is this basic intuition that would lead Asher and Pusteyovsky to analyze (6a) as (6b).

(6) a. A lunch was delicious

b. $\exists x : \textit{Food}\ \exists v : \textit{Food} \cdot \textit{Event}\ [\text{lunch}(v) \wedge \text{O-Elab}(x, v) \wedge \text{delicious}(x)]$

According to (6b) it is not the lunch which is delicious but an "object elaboration" of the lunch. Asher and Pustejovsky's claim is that object elaborations are aspects of an object and that we need to quantify over these in order to analyze examples like (6a). I would prefer not to be forced to introduce such objects into the analysis and to say that the lunch is delicious in virtue of the food which is part of the lunch being delicious. It is common in natural language for us to make predications of objects in terms of predications that hold of some of their parts, though not all of them. Consider (7a) in contrast to (7b).

(7) a. The house is locked

b. The door/lock is locked

We say that the house (which for simplicity we will assume has exactly one external door) is locked in virtue of its door being locked. The door, arguably, is locked in virtue of its lock being locked. We apply the predicate *locked* to the house, without requiring that all of its parts

be locked (e.g. the chimney is not locked, neither are the walls). This non-distributive nature of predication becomes important when we consider nouns representing collections. Consider the examples in (8).

(8) a. The choir/singers sang the Hallelujah Chorus. Mildred had a tickle in her throat and didn't sing a note.

b. The (players of the) Gothenburg Symphony Orchestra played Mahler's second and a Mozart symphony. There were many more people playing for the Mahler.

Even though Mildred, a member of the choir, was standing at her place with the rest of the choir with her copy of *Messiah* open at the right page, she could not sing. This does not mean that the choir had not sung the Hallelujah Chorus even though Mildred had not sung a note of it. Many more members of the Gothenburg Symphony Orchestra are required for a Mahler symphony than a Mozart symphony. A performance by the orchestra does not require all the members of the orchestra to be playing. Exactly when you are allowed to predicate of the whole when only a part is actually involved is similar in character to the generic problem originally posed by Carlson (1977a, 1977b): how many bad drivers do you need to make *Boston drivers are bad* be true? Thus (9)

(9) The GSO played Elgar's Intro and Allegro for Strings

sounds acceptable though one might be tempted to say *the strings of the GSO* while

(10) The GSO played a solo violin partita by Bach

sounds odd even if it was the leader of the GSO playing the violin. Though if the partita is one item in a concert with a lot of orchestral music it might perhaps be improved.

When we consider such examples, it seems quite reasonable to think of lunch as an object which has both an event part and a food part and to apply predicates to the whole object which properly apply only to one of the parts. I will not give an analysis of object structure in this paper but will argue for record types as a useful way of representing the fact that an object can have different kinds of predicates applied to it.

3 Records and record types

In this section we give a very brief intuitive introduction to the kind of type theory we are employing. A more detailed and formal account can be found in Cooper (forthcoming) and work in progress on the project can be found on `http://www.ling.gu.se/~cooper/records`.

The central idea of records and record types can be expressed informally as follows, where $T(a_1, \ldots, a_n)$ represents a type T which depends on the objects $a_1, \ldots, a_n$.

If $a_1 : T_1, a_2 : T_2(a_1), \ldots, a_n : T_n(a_1, a_2, \ldots, a_{n-1})$, a record:

(11)
$$\begin{bmatrix} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \ldots & & \\ l_n & = & a_n \\ \ldots & & \end{bmatrix}$$

is of type:

(12)
$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ \ldots & & \\ l_n & : & T_n(l_1, l_2, \ldots, l_{n-1}) \end{bmatrix}$$

A record is to be regarded as a set of fields consisting of a label and an object. A record type is to be regarded as a set of fields consisting of a label and type. A record is of this type just in case for each field in the type there is a corresponding field in the record (with the same label) and the object in the record field is of the type in the type field. Notice that the record may have additional fields not mentioned in the type. Thus a record will generally belong to several record types and any record will belong to the empty record type. This gives us a notion of subtyping.

Let us see how this notion can be applied to a simple linguistic example. We will take the content of a sentence to be modelled by a record type. The sentence

(13) *a man owns a donkey*

corresponds to a record type:

$$(14)\quad \begin{bmatrix} x & : & \textit{Ind} \\ c_1 & : & \text{man}(x) \\ y & : & \textit{Ind} \\ c_2 & : & \text{donkey}(y) \\ c_3 & : & \text{own}(x,y) \end{bmatrix}$$

A record of this type will be:

$$(15)\quad \begin{bmatrix} x & = & a \\ c_1 & = & p_1 \\ y & = & b \\ c_2 & = & p_2 \\ c_3 & = & p_3 \end{bmatrix}$$

where

a, b are of type *Ind*, individuals
p_1 is a proof of man(a)
p_2 is a proof of donkey(b)
p_3 is a proof of own(a, b).

Note that the record may have had additional fields and still be of this type. The types man(x), donkey(y), own(x,y) are dependent types of proofs. The use of types of proofs for what in other theories would be called propositions is often referred to as the notion of "propositions as types". Exactly what type man(x) is depends on which individual you choose in your record to be labelled by x. If the individual a is chosen then the type is the type of proofs that a is a man. If another individual d is chose then the type is the type of proofs that d is a man, and so on. What is a proof? Martin-Löf considers proofs to be objects rather than arguments or texts. For non-mathematical propositions proofs can be regarded as situations or events. For useful discussion of this see Ranta (1994), p 53ff. We discuss it in more detail in Cooper (forthcoming).

There is an obvious correspondence between this record type and a discourse representation structure (DRS) as characterised in Kamp and Reyle (1993). The characterisation of what it means for a record to be of this type corresponds in an obvious way to the standard embedding semantics for such a DRS which Kamp and Reyle provide.

Records (and record types) are also recursive in the sense that the value corresponding

to a label in a field can be a record (or record type)[3]. For example,

(16) $r = \left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} ff & = & a \\ gg & = & b \end{array}\right] \\ g & = & c \end{array}\right] \\ g & = & \left[\begin{array}{lcl} h & = & \left[\begin{array}{lcl} g & = & a \\ h & = & d \end{array}\right] \end{array}\right] \end{array}\right]$

is of type

(17) $R = \left[\begin{array}{lcl} f & : & \left[\begin{array}{lcl} f & : & \left[\begin{array}{lcl} ff & : & T_1 \\ gg & : & T_2 \end{array}\right] \\ g & : & T_3 \end{array}\right] \\ g & : & \left[\begin{array}{lcl} h & : & \left[\begin{array}{lcl} g & : & T_1 \\ h & : & T_4 \end{array}\right] \end{array}\right] \end{array}\right]$

given that $a : T_1, b : T_2, c : T_3$ and $d : T_4$. We can use *path-names* in records and record types to designate values in particular fields, e.g.

(18) a. $r.f = \left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} ff & = & a \\ gg & = & b \end{array}\right] \\ g & = & c \end{array}\right]$

b. $R.f.f.ff = T_1$

The recursive nature of records and record types is important for capturing aspects of linguistic theories which use feature structures, for example, in writing HPSG style grammars.

The theory of records and record types is embedded in a general type theory. This means that we have functions and function types available giving us a version of the λ-calculus. We can thus use Montague's techniques for compositional interpretation. For example, we can interpret the common noun *donkey* as a function which maps records r of the type $\left[\text{x:}\mathit{Ind}\right]$ (i.e. records which introduce an individual labelled with the label 'x') to a record type dependent on r. We notate the function as follows:

(19) $\lambda r{:}\left[\text{x:}\mathit{Ind}\right](\left[\text{c:donkey}(r.\text{x})\right])$

[3]There is a technical sense in which this recursion is non-essential. These records could also be viewed as non-recursive records whose labels are sequences of atomic labels. See Cooper (forthcoming) for more discussion.

The type of this function is

(20) $(\left[\text{x:}\mathit{Ind}\right])\text{RecType}$

This correponds to Montague's type $\langle e, t\rangle$ (the type of functions from individuals (entities) to truth-values). In place of individuals we use records introducing individuals with the label 'x' and in place of truth-values we use record types which, as we have seen above, correspond to an intuitive notion of proposition (in particular a proposition represented by a DRS).

4 Dynamic generalized quantifiers

Following the proposal for the treatment of generalized quantifiers in Cooper (2004) we will treat determiners such as *the* as predicates which hold between two such functions. Thus *the donkey runs* could be represented as

(21) $\left[\ \text{c} \ : \ \text{the}(\lambda r{:}\left[\text{x:}\mathit{Ind}\right](\left[\text{c:donkey}(r.\text{x})\right]), \lambda r{:}\left[\text{x:}\mathit{Ind}\right](\left[\text{c:run}(r.\text{x})\right]))\ \right]$

In our treatment of dynamic quantifiers in Cooper (2004) we required such quantifier predicates to be polymorphic. We will use the notation $X \sqsubseteq T$ to represent a variable over types which are subtypes of the record type T, that is, record types which contain at least fields with the same labels as T and each type corresponding to a particular label, ℓ, corresponds to a subtype of the type in the ℓ-field of T.[4]

If q is a quantifier predicate then

1. arity(q) = $\langle (X \sqsubseteq \left[\text{x:}\mathit{Ind}\right])\mathit{RecType}, (X \sqsubseteq \left[\text{x:}\mathit{Ind}\right])\mathit{RecType}\rangle$
2. q is associated with a relation between sets $q*$ (the relation between sets from classical generalized quantifier theory, Peters and Westerståhl, forthcoming) such that

   $p : q(f_1 : (T_1)RecType, f_2 : (T_2)RecType)$
   iff
   $p = \langle \{a | \exists r : T_1[f_1(r)\text{is non-empty} \wedge a = r.\text{x}]\}, \{a | \exists r : T_2[f_2(r)\text{is non-empty} \wedge a = r.\text{x}]\}\rangle$
   and $q*$ holds between p_1 and p_2

In order to achieve dynamic quantification we introduce a notion of fixed point type and use the fixed point type of the first argument to q as the domain type of the second argument

[4]The presence of dependent types means that we have to be careful with the exact formulation of the subtype relation, but for present purposes we will just use this intuitive informal formulation.

to q. If $\mathcal{T}$ is a function from records to record types (a family of record 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

$$(22) \quad \lambda r{:}\begin{bmatrix} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{z} & : & \textit{Ind} \\ \text{c}_3 & : & \text{farmer(z)} \\ \text{c}_4 & : & \text{kick(y, z)} \end{bmatrix}.\begin{bmatrix} \text{c}_7 & : & \text{angry}(r.\text{y}) \end{bmatrix}$$

Then

$$(23) \quad \begin{bmatrix} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{z} & : & \textit{Ind} \\ \text{c}_3 & : & \text{farmer(z)} \\ \text{c}_4 & : & \text{kick(y, z)} \\ \text{c}_7 & : & \text{angry(y)} \end{bmatrix}$$

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. We will use $\mathcal{F}(\mathcal{R})$ to represent the fixed point type of a family of record types $\mathcal{R}$[5].

This notion of fixed point type is exploited in order to make the first argument of the quantifier provide a *hypothetical context* for the second argument. The second argument becomes a function which requires as argument (i.e. context) a record which is of the fixed point type of the first argument. We call it hypothetical because it does not require that there be such a context. It just characterises the domain of the function[6]. This is represented in (24).

$$(24) \quad q(f_1 : (T)RecType, f_2 : (\mathcal{F}(f_1))RecType)$$

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

[6]A similar analysis of generalized quantifiers exploiting contexts in type theory is given in Fernando (2001)

5 Treating copredication

Allowing generalized quantifier predicates to be polymorphic enables us to add additional constraints on the domains of functions corresponding to nouns and verb-phrases. For example, we require that records which are arguments to *take forever* in addition to introducing an individual also introduce a proof that that individual is an event:

$$(25) \quad \lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{event}(\mathrm{x}) \end{bmatrix}\left(\left[\mathrm{c}_2{:}\mathrm{take\_forever}(r.\mathrm{x})\right]\right)$$

Similarly *be delicious* will require that its subject is food.

$$(26) \quad \lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{food}(\mathrm{x}) \end{bmatrix}\left(\left[\mathrm{c}_2{:}\mathrm{be\_delicious}(r.\mathrm{x})\right]\right)$$

The conjunction *be delicious and take forever* needs to require that its subject is both food and an event:

$$(27) \quad \lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{food}(\mathrm{x}) \\ \mathrm{c}_2{:}\mathrm{event}(\mathrm{x}) \end{bmatrix}\left(\begin{bmatrix} \mathrm{c}_3{:}\mathrm{be\_delicious}(r.\mathrm{x}) \\ \mathrm{c}_4{:}\mathrm{take\_forever}(r.\mathrm{x}) \end{bmatrix}\right)$$

Similarly, nouns will place additional constraints on their domains.

$$(28) \quad \lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{food}(\mathrm{x}) \end{bmatrix}\left(\left[\mathrm{c}_2{:}\mathrm{blancmange}(r.\mathrm{x})\right]\right)$$
$$\lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{event}(\mathrm{x}) \end{bmatrix}\left(\left[\mathrm{c}_2{:}\mathrm{game}(r.\mathrm{x})\right]\right)$$
$$\lambda r{:}\begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c}_1{:}\mathrm{food}(\mathrm{x}) \\ \mathrm{c}_2{:}\mathrm{event}(\mathrm{x}) \end{bmatrix}\left(\left[\mathrm{c}_3{:}\mathrm{lunch}(r.\mathrm{x})\right]\right)$$

Lunch, it will be noted, holds of objects which are both food and an event.

Following the theory of dynamic quantifiers sketched in section 4, the function which is the first argument to the quantifier predicated is used to further restrict the domain of the second function. Thus *the game took forever* would actually be interpreted in terms of

(29) $\text{the}(\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{event(x)}\end{bmatrix}(\left[\text{c}_2{:}\text{game}(r.\text{x})\right]),$
$\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{event(x)} \\ \text{c}_2{:}\text{game(x)}\end{bmatrix}(\left[\text{c}_3{:}\text{took\_forever}(r.\text{x})\right]))$

Thus the first argument to the quantifier predicate provides a context in which the second argument is interpreted by restricting the domain of the second argument. Normally the information passed on by the first argument is a subtype of the domain type of the original second argument. However, if we try to say that the game is delicious, this will not be the case since the second argument will require that objects in its domain are food whereas *game* will only provide the event restriction but not the food restriction. Thus some coercion is needed. There are three main strategies that seem to be available. The first two strategies involve radical coercions. Either we make *game* be food or we change the restriction on *delicious* so that it does not presuppose that its argument is food. The third strategy is a mixture of the two and is probably the most natural. We find a way in which games are food-like and we change the restriction on *delicious* so that it applies to food-like things. Notice that the first strategy is something like presupposition accommodation going on within a predication.

With *lunch* we can use either verb-phrases requiring food or events since in either case the function corresponding to *lunch* will provide us with a subtype of the domain type of the second argument to the quantifier.

We can now return to the example from the beginning of the paper concerning dialogue systems for networked devices. We can represent the predicates *light*, *stereo*, *turn on* and *turn up volume* as in (30).

(30) $\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{on-off(x)}\end{bmatrix}(\left[\text{c}_2{:}\text{light}(r.\text{x})\right])$
$\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{on-off(x)} \\ \text{c}_2{:}\text{volume(x)}\end{bmatrix}(\left[\text{c}_3{:}\text{stereo}(r.\text{x})\right])$
$\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{on-off(x)}\end{bmatrix}(\left[\text{c}_2{:}\text{turn\_on}(r.\text{x})\right])$
$\lambda r{:}\begin{bmatrix}\text{x} : \textit{Ind} \\ \text{c}_1{:}\text{volume(x)}\end{bmatrix}(\left[\text{c}_2{:}\text{turn\_up\_volume}(r.\text{x})\right])$

(For convenience I am ignoring the fact that *turn_on* and *turn_up_volume* should properly be two-place predicates.)

Thus the system will allow you to say (31a) but will object to (31b).

(31) a. Turn on the stereo and turn up the volume

b. Turn on the light and turn up the volume

The difference between stereos and lights is parallel to the difference between lunch and blancmange in terms of their respective types in our analysis. New devices plugged into the system need to declare their types and language libraries need to be available to tell the system how to talk to devices of various types.

6 Conclusion

We have suggested that a record type theoretical approach to semantics gives us a simple approach to different aspects of objects without using dot types or dot objects. The approach provides a straightforward treatment of copredication examples. From the point of view of the Generative Lexicon project the approach is perhaps attractive because record types provide us with feature structure like objects which are important elsewhere in the Generative Lexicon.

Robin Cooper
Department of Linguistics, Göteborg University
Box 200
S-405 30 Göteborg
Sweden
cooper@ling.gu.se

References

Asher, Nicholas and James Pustejovsky (2005) Word Meaning and Commonsense Metaphysics, in course materials for Type Selection and the Semantics of Local Context, ESSLLI 2005.

Betarte, Gustavo (1998) *Dependent Record Types and Algebraic Structures in Type Theory*. Ph.D. thesis. Department of Computing Science, Göteborg University and Chalmers University of Technology.

Betarte, Gustavo and Alvaro Tasistro (1998) Extension of Martin-Löf's type theory with record types and subtyping. In Sambin, Giovanni and Jan Smith, eds. *Twenty-Five Years of Constructive Type Theory*, Oxford Logic Guides, 36, Oxford University Press, Oxford.

Carlson, G.N. (1977) A unified analysis of the English bare plural. *Linguistics and Philosophy*, Vol. 1, No. 3, pp. 413–458.

Carlson, G.N. (1977) *Reference to Kinds in English*, Ph.D. thesis, Dept. of Linguistics, University of Massachusetts, Amherst.

Cooper, Robin (2004) Dynamic generalised quantifiers and hypothetical contexts. In *Ursus Philosophicus*, a festschrift for Björn Haglund. Department of Philosophy, Göteborg University. Available from `http://www.phil.gu.se/posters/festskrift/`.

Cooper, Robin (2005) Records and Record Types in Semantic Theory, *Journal of Logic and Computation*, Vol. 15 No. 2, pp. 99–112.

Cooper, Robin (forthcoming) Austinian truth, attitudes and type theory. *Research on Language and Computation*.

Coquand, Thierry, Randy Pollock and Makoto Takeyama (2004) A Logical Framework with Dependently Typed Records. *Fundamenta Informaticae*, **XX**, 1–22.

Fernando, Tim (2001) Conservative generalized quantifiers and presupposition, in *Semantics and Linguistic Theory XI*, NYU/Cornell, pp 172–191.

Kamp, Hans and Uwe Reyle (1993) *From Discourse to Logic*, Kluwer, Dordrecht.

McCawley, James D. (1968) The Role of Semantics in a Grammar, in *Universals in Linguistic Theory*, ed. by Emmon Bach and Robert T. Harms, Holt, Rinehart and Winston, New York, 1968, pp. 125–69.

Peters, S., and Westerståhl, D. (forthcoming) *Quantifiers in Language and Logic*, Oxford University Press, Oxford.

Pustejovsky, James (1995) *The Generative Lexicon*, MIT Press, Cambridge, Mass.

Ranta, Aarne (1994) *Type-Theoretical Grammar*, Clarendon Press, Oxford.

Tasistro, Alvaro (1997) *Substitution, record types and subtyping in type theory, with applications to the theory of programming*. PhD thesis, Department of Computing Science, Göteborg University and Chalmers University of Technology.