Records and record types in semantic theory

Robin Cooper
Göteborg University

January 14, 2005
to appear in Journal of Logic and Computation
1 Introduction

In this paper we will explore possibilities for formulating linguistic semantics in terms of records and record types of the kind used in recent developments of Martin-Löf type theory [21, 3, 4, 9]. We will suggest that they give us the tools to develop a theory which includes aspects of Montague semantics, using the lambda calculus\(^1\), Discourse Representation Theory (DRT)\(^2\), situation semantics\(^3\) and Head-Driven Phrase Structure Grammar (HPSG)\(^4\) in a single theory. We will also argue that formulating these theories in terms of record types may provide us not only with a unified approach but also with certain improvements over the individual theories.

The ingredients from type theory that we will use, in addition to records and record types, are:

- dependent types (including dependent function types)
- the notion of “propositions” as types of proofs
- the first class status of types as objects

The approach that we will outline has the following features:

- compositionality using the \(\lambda\)-calculus (from Montague semantics)
- dynamic binding (from DRT)
- a treatment of intensionality including perception complements and intensional verbs like seek (from situation semantics)
- a treatment of context dependence including resource situations (from situation semantics)
- a sign-based approach to the relation between syntax and semantics (from HPSG)

While some of these features have been combined in other approaches (e.g. [18, 14, 10], we want to argue that records in type theory provide an elegant and powerful way of combining them all.

In the rest of the paper we will first give an informal account of records and record types based on the more explicit account given in [8]. We will then show how these tools can be exploited in Montague semantics, DRT, situation semantics and HPSG.

2 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 [8] and work in progress on the project can be found on http://www.ling.gu.se/~cooper/records. While the type theoretical machinery is based on work carried out in the Martin-Löf

\(^1\) [17] is the classic reference.

\(^2\) [12, 10] and much other literature.

\(^3\) [1] and subsequent literature.

\(^4\) For a recent account see [11].
approach we are making a serious attempt to give it a foundation in standard set theory using Montague style recursive definitions of semantic domains. There are two main reasons for this. The first is that we think it important to show the relationship between the Montague model theoretic tradition which has been developed for natural language semantics and the proof-theoretic tradition associated with type theory. We believe that the aspects of this kind of type theory that we need can be seen as an enrichment of Montague’s original programme. The second reason is that we are interested in exploring to what extent intuitionistic and constructive approaches are appropriate or necessary for natural language. For example, we make important use of the notion “propositions as types” which is normally associated with an intuitionistic approach. However, we suspect that our Montague-like approach to defining the type theory to some extent decouples the notion from intuitionism. We would like to see type theory as providing us with a powerful collection of tools for natural language analysis which ultimately do not commit one way or the other to philosophical notions associated with intuitionism.

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:

\[
\begin{align*}
  l_1 &= a_1 \\
  l_2 &= a_2 \\
  \ldots \\
  l_n &= a_n \\
  \ldots
\end{align*}
\]

is of type:

\[
\begin{align*}
  l_1 : T_1 \\
  l_2 : T_2(l_1) \\
  \ldots \\
  l_n : T_n(l_1, l_2, \ldots, l_{n-1})
\end{align*}
\]

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

\[ a \text{ man owns a donkey} \]

corresponds to a record type:

\[
\begin{align*}
  x & : \text{Ind} \\
  c_1 & : \text{man}(x) \\
  y & : \text{Ind} \\
  c_2 & : \text{donkey}(y) \\
  c_3 & : \text{own}(x,y)
\end{align*}
\]
A record of this type will be:

\[
\begin{bmatrix}
  x &= a \\
  c_1 &= p_1 \\
  y &= b \\
  c_2 &= p_2 \\
  c_3 &= p_3 \\
\end{bmatrix}
\]

where

- \(a, b\) are of type \(\text{Ind}\), individuals
- \(p_1\) is a proof of \(\text{man}(a)\)
- \(p_2\) is a proof of \(\text{donkey}(b)\)
- \(p_3\) is a proof of \(\text{own}(a, b)\).

Note that the record may have had additional fields and still be of this type. The types \(\text{man}(x), \text{donkey}(y), \text{own}(x,y)\) 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 \(\text{man}(x)\) 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 chosen 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 [19], p 53ff. We discuss it in more detail in [8].

There is an obvious correspondence between this record type and a discourse representation structure (DRS) as characterised in [12]. 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)\(^5\). For example,

\[
r = \begin{bmatrix}
f &= \begin{bmatrix} f &= \begin{bmatrix} ff &= a \\ gg &= b \end{bmatrix} \\ g &= \begin{bmatrix} c \\ h &= \begin{bmatrix} g &= a \\ h &= d \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
g &= \begin{bmatrix} h &= \begin{bmatrix} g &= \begin{bmatrix} f &= \begin{bmatrix} ff &= T_1 \\ gg &= T_2 \end{bmatrix} \\ g &= T_3 \\ h &= \begin{bmatrix} g &= T_1 \\ h &= T_4 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}
\]

is of type

\[
R = \begin{bmatrix}
f & : & \begin{bmatrix} f & : & \begin{bmatrix} ff & : & T_1 \\ gg & : & T_2 \end{bmatrix} \\ g & : & T_3 \\ h & : & \begin{bmatrix} g & : & T_1 \\ h & : & T_4 \end{bmatrix} \end{bmatrix} \end{bmatrix}
\]

given that \(a : T_1, b : T_2, c : T_3\) and \(d : T_4\). We can use \textit{path-names} in records and record types to designate values in particular fields, e.g.

\(^5\)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 [8] for more discussion.
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.

Another important aspect of the type theory we are using is that types themselves can also be treated as objects. A simple example of how this can be exploited is the following representation for a girl believes that a man owns a donkey. This is a simplified version of the treatment discussed in [8].

\[
\begin{align*}
\text{x} & : \text{Ind} \\
\text{c}_1 & : \text{girl}(\text{x}) \\
\text{c}_2 & : \text{believe}(\text{x}, \begin{align*}
\text{y} & : \text{Ind} \\
\text{c}_3 & : \text{man}(\text{y}) \\
\text{z} & : \text{Ind} \\
\text{c}_4 & : \text{donkey}(\text{z}) \\
\text{c}_5 & : \text{own}(\text{y}, \text{z})
\end{align*})
\end{align*}
\]

The treatment of types as first class objects in this way is a feature which this type theory has in common which situation theory and it is an important component in allowing us to incorporate analyses from situation semantics in our type theoretical treatment.

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 \(\lambda\)-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 \([x:\text{Ind}]\) (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:

\[
\lambda r : [x:\text{Ind}](\text{c:donkey}(r.x))
\]

The type of this function is

\[
P = ([x:\text{Ind}])\text{RecType}
\]

This corresponds to Montague’s type \(<e, t>\) (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). Using the power of the \(\lambda\)-calculus we can treat determiners Montague-style as functions which take two arguments of type \(P\) and return a record type. For example, we represent the indefinite article by

\footnote{In order to do this safely we stratify the types. We define the type system as a family of type systems of order \(n\) for each natural number \(n\). The idea is that types which are not defined in terms of other types are of order 0 and that types which are defined in terms of types of order \(n\) are of order \(n + 1\). We will not discuss this in detail here but rely on the discussion in [8]. In this paper we will suppress reference to order in the specification of our types.}
\( \lambda R_1;([x: \text{Ind}]) \text{RecType} \lambda R_2;([x: \text{Ind}]) \text{RecType} (\begin{array}{l}
\text{par : } [x : \text{Ind}] \\
\text{restr : } R_1 \circ \text{par} \\
\text{scope : } R_2 \circ \text{par}
\end{array}) \)

Here we use \( F \circ a \) to represent the result of applying function \( F \) to argument \( a \).

The last important feature of the type theory that we will take up in this section is that it includes dependent function types. These can be used to give a classical treatment of universal quantification corresponding to DRT’s ‘⇒’. For example, an interpretation of every man owns a donkey can be the following record type:

\[
\begin{array}{l}
\text{f : } (r : [x : \text{Ind}]) \text{RecType} (\begin{array}{l}
\text{c_1 : man(x)} \\
\text{c_2 : donkey(y)} \\
\text{c_3 : own(r,x,y)}
\end{array})
\end{array}
\]

Records of the type

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

map records \( r \) of type

\[
\begin{array}{l}
x : \text{Ind} \\
c_1 : \text{man(x)}
\end{array}
\]

to records of type

\[
\begin{array}{l}
y : \text{Ind} \\
c_2 : \text{donkey(y)} \\
c_3 : \text{own(r,x,y)}
\end{array}
\]

Our interpretation of every man owns a donkey requires that there exist a function of this type. Why do we use the record type with the label ‘f’ rather than the function type itself as the interpretation of the sentence? One reason is to achieve a uniform treatment where the interpretation of a sentence is always a record type. Another reason is that the label gives us a handle which can be used to anaphorically refer to the function. This can, for example, be exploited in so-called paycheck examples [13] such as Everybody receives a paycheck. Not everybody pays it into the bank immediately, though.

3 Four linguistic theories

In this section we will look in a little more detail at how you can get aspects of the four linguistic theories we are concerned with to work together simultaneously in a single approach. We will take each of the four theories in turn and start with a summary of what the theory provides to our overall approach and what it gains from the other theories that are incorporated. These summaries are representative rather than exhaustive and are meant to give some basic motivation for trying to combine the different approaches in the way that we are suggesting.
3.1 Montague semantics

This theory provides our approach to compositionality exploiting the λ-calculus using techniques familiar from Montague’s original work and the large literature which has built on it. The theory gains:

- dynamic binding (from DRT)
- an improved treatment of intensionality including perception (from situation semantics)
- an improved treatment of context dependence, resources (from situation semantics)

In order to illustrate the compositional treatment we sketch a sample derivation of every man owns a donkey. We show only the combinations of the semantic representation corresponding to each constituent of the sentence rather than giving a formal definition of syntax and semantics.

In order to interpret the noun-phrase a donkey we apply the function corresponding to the indefinite article a to the interpretation of donkey. The result is simplified by β-conversion:

\[ a \text{ donkey} \]

\[
\lambda R_1:([x:Ind]) \text{RecType} \lambda R_2:([x:Ind]) \text{RecType} \\
\begin{cases}
\text{par:}[x:Ind] \\
\text{restr:} R_1 \otimes \text{par} \\
\text{scope:} R_2 \otimes \text{par}
\end{cases}
\]

\[ @ \]

\[
\lambda r: [x:Ind] ([c: \text{donkey}(r.x)])
\]

\[ = \]

\[
\lambda R_2:([x:Ind]) \text{RecType} \\
\begin{cases}
\text{par:}[x:Ind] \\
\text{restr:} [c: \text{donkey}(\text{par}.x)] \\
\text{scope:} R_2 \otimes \text{par}
\end{cases}
\]

The function corresponding to the transitive verb own takes noun-phrase interpretations as its argument and returns a result (here represented after β-conversion) which corresponds to a one-place predicate of individuals, i.e. in our terms, a function from records which introduce an individual to record types.

\[ \text{own a donkey} \]

\[
\lambda N:([x:Ind]) \text{RecTypeRecType} \\
\lambda r_1: [x:Ind] (N \otimes \lambda r_2: [x:Ind] ([c: \text{own}(r_1.x, r_2.x)]))
\]

\[ @ \]

\[
\lambda R_2:([x:Ind]) \text{RecType} \\
\begin{cases}
\text{par:}[x:Ind] \\
\text{restr:} [c: \text{donkey}(\text{par}.x)] \\
\text{scope:} R_2 \otimes \text{par}
\end{cases}
\]

\[ = \]

\[
\lambda r_1: [x:Ind] ([\text{par:}[x:Ind] \\
\text{restr:} [c: \text{donkey}(\text{par}.x)] \\
\text{scope:} [c: \text{own}(r_1.x, \text{par}.x)]])
\]
The interpretation of the determiner *every* is of the same type as that of the
determiner *a*. However, the record type it returns introduces a function corresponding
to universal quantification.

\[
\text{every man} \\
\lambda R_1: ([x: Ind]) \text{RecType} \\
\lambda R_2: ([x: Ind]) \text{RecType} \\
\left[ f: (\begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } R_1 @ \text{par}
\end{array}) R_2 @ r.\text{par} \right]
\]\n
\[\lambda r: [x: Ind] ([c: \text{man}(r.x)]) = \lambda R_2: ([x: Ind]) \text{RecType} \]
\[
\left[ f: (\begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } [c: \text{man}(\text{par.x})]
\end{array}) R_2 @ r.\text{par} \right]
\]

Finally, the interpretation of the whole sentence is the result of applying the
interpretation of *every donkey* to the interpretation of *owns a donkey*. The result is
represented after \(\beta\)-conversion.

\[
\text{every man owns a donkey} \\
\lambda R_2: ([x: Ind]) \text{RecType} \\
\left[ f: (\begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } [c: \text{man}(\text{par.x})]
\end{array}) R_2 @ r.\text{par} \right]
\]\n
\[\lambda r_1: [x: Ind] ( \begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } [c: \text{donkey}(\text{par.x})]
\end{array} ) \]
\[
= \left[ f: (\begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } [c: \text{man}(\text{par.x})]
\end{array}) \right]
\]
\[
\left[ f: (\begin{array}{c}
\text{par: } [x: Ind] \\
\text{restr: } [c: \text{donkey}(\text{par.x})]
\end{array}) \right]
\]
\[
\left[ \text{scope: } [c: \text{own}(r_1.x, \text{par.x})] \right]
\]

This record type is equivalent\(^7\), though not identical, to record types which have
a simpler structure and are easier to read. For any record type we can obtain an
equivalent record type by replacing each path with a single label. This procedure we
call *flattening*. We show its application to this example.

\[
\text{Flattening} \\
\left[ f: (\begin{array}{c}
\text{par: } x.\text{Ind} \\
\text{restr: } [c: \text{man}(\text{par.x})]
\end{array}) \right]
\]
\[
\left[ f: (\begin{array}{c}
\text{par: } x.\text{Ind} \\
\text{restr: } [c: \text{donkey}(\text{par.x})]
\end{array}) \right]
\]
\[
\left[ \text{scope: } [c: \text{own}(r.\text{par.x}, \text{par.x})] \right]
\]

Finally, relabelling of a record type will also give an equivalent record type\(^8\)

---

\(^7\)The notion of equivalence used here is that of \(\Sigma\)-equivalence as introduced in [8].

\(^8\)Both flattening and relabelling are discussed in more detail in [8].
3.2 DRT

DRT provides us with our approach to dynamic binding which enables us to treat donkey anaphora and discourse anaphora. It gains:

- a compositional treatment using the $\lambda$-calculus (from Montague semantics). The result is closely related to other combinations of Montague’s approach with DRT such as “$\lambda$-DRT” [18, 14, 10].
- an improved treatment of intensionality including perception (from situation semantics)
- an improved treatment of context dependence, resources (from situation semantics)

The approach to donkey anaphora exploits the fact that we have dependent function types and follows the type theoretical proposals which have been made in [20, 19].

The classical analysis of every man who owns a donkey beats it, i.e. the one that analyses it as “for every man-donkey pair $<x, y>$ such that $x$ owns $y$, it is the case that $x$ beats $y$, is represented as:

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

There are two techniques we exploit in order to treat anaphora compositionally: manifest fields and metavariables.

**Manifest fields.** This notion is introduced in [9]. It builds on the notion of singleton type. If $a : T$, then $T_a$ is a *singleton type* and $b : T_a$ iff $b = a$. A manifest field in a record type is one whose type is a singleton type, e.g.

\[
\left[ \begin{array}{c}
  x : T_a
\end{array} \right]
\]

written for convenience as

\[
\left[ \begin{array}{c}
  x = a : T
\end{array} \right]
\]

This notion allows record types to be “progressively instantiated”, i.e. intuitively, for values to be specified within a record type. A record type that only contains manifest fields is completely instantiated and there will be exactly one record of that type. We will allow dependent singleton types, where $a$ in $T_a$ can be represented by a path in a record type. Manifest fields are important not only for the treatment of anaphora but also for using record types to represent feature structures as used in linguistic theories like HPSG.
Metavariables. Metavariables have been used in type theoretical approaches to proof editing [16, 15]. We will use occurrences of metavariables (anonymous variables) ‘?’ in manifest fields in order to treat anaphoric constructions. They will be resolved to paths. In the future we plan to use a variant of David Beaver’s [2] optimality theoretic (OT) version of centering theory for resolution and we suspect that metavariables can be used for other kinds of underspecification such as quantifier scope and that we may be able to use OT here as well.

We let the interpretation of pronouns he/him/she/her/it⁹ be:

\[ \text{npr} \left[ x = ? : \text{Ind} \right] \]

The superscript \text{npr} stands for NP-raising which is defined as follows:¹⁰

\[
\text{If } T = \left[ x = y : \text{Ind} \right] \\
\text{then } \text{npr}\, T = \lambda R: ([x:\text{Ind}] \text{RecType}) [\text{par} : T \text{ scope} : R @ \text{par}] \\
\]

The interpretation of the verb-phrase beats it is thus derived compositionally as follows (showing the result of the function application after \( \beta \)-conversion).

\[
\lambda \text{N} : ([x:\text{Ind}] \text{RecType}) \text{RecType} \\
\lambda r_1 : [x:\text{Ind}] (\text{N} @ \lambda r_2 : [x:\text{Ind}] ([c:\text{beat}(r_1.x, r_2.x)]) @ \\
\lambda R : ([x:\text{Ind}] \text{RecType} ([\text{par} : x = ? : \text{Ind} \text{ scope} : R @ \text{par}]) = \\
\lambda r_1 : [x:\text{Ind}] ([\text{par} : x = ? : \text{Ind} \text{ scope} : c : \text{beat}(r_1.x, \text{par}.x)]) \\
\]

The representation for every man who owns a donkey beats it obtained by compositional semantics is

\[
\begin{align*}
\text{f} : (r : & \left[ \text{par} : [x:\text{Ind}] \right] \\
& \text{restr} : c : \left[ \text{pred} : [c:\text{man}(\text{par}.x)] \right] \left[ \text{par} : [x:\text{Ind}] \right] \\
& \text{mod} : \left[ \text{restr} : [c:\text{donkey}(\text{restr}.c.\text{mod}\.\text{par}.x)] \right] \left[ \text{scope} : [c:\text{own}(\text{par}.x, \text{restr}.c.\text{mod}\.\text{par}.x)] \right] \\
& \text{par} : [x = ? : \text{Ind}] \\
& \text{scope} : [c:\text{beat}(r.\text{par}.x, \text{par}.x)] \\
\end{align*}
\]

We find candidate paths for the resolution of the metavariable ‘?’ by looking for paths of the form \ldots \text{par} : x : \text{Ind}. The candidates are

\[
\begin{align*}
& r.\text{par}.x \\
& r.\text{restr}.c.\text{mod}\.\text{par}.x \\
\end{align*}
\]

and in addition if there were any \( r' \) defined (e.g. representing context) then any path \( r'. \ldots \text{par} : x : \text{Ind} \) would be included in the list (provided \( x : \text{Ind} \))

⁹We ignore gender and case in this treatment.
¹⁰Corresponding to Montague style raising of an individual variable \( x \) to a generalized quantifier denotation \( \lambda \text{FP}(x) \) corresponding to a noun-phrase interpretation.
In a complete treatment the first of these would be ruled out by a grammatical constraint on non-reflexive pronouns ruling out co-arguments to a predicate reducing to the same path. Therefore we choose the second. The resulting type can be shown to be equivalent to that presented on page 8 by flattening, relabelling and elimination of manifest fields representing equalities. 

3.3 Situation semantics

Situation semantics contributes:

- a treatment of the attitudes and other intensional constructions (including perception complements)
- a treatment of context using resource situations

It gains:

- a rigorous and precise approach to compositional interpretation (from Montague semantics)
- dynamic semantics (from DRT)
- a way of connecting the relational theory of meaning to HPSG

We have discussed the treatment of the attitudes in some detail in [8] and will therefore not discuss it further here. We will briefly present how the notion of resource situation can be incorporated into our type-theoretical approach. It is well known in the literature that quantificational domains need to be restricted in examples such as *every man owns a donkey* and *the man owns a donkey*. For one discussion of this restriction in terms of resource situations see [6]. The essential idea is to interpret a noun-phrase such as *every man* as “everything which is a man in situation s”. In our record-based approach we use records to model situations. Thus the interpretation becomes “everything which is a man in record r”. In order to achieve this we introduce a new way of constructing types which is similar to the construction of singleton types discussed on page 8. We will use the notation $T \mid r$ to represent type $T$ restricted to record $r$. If $T$ is a type and $r$ is a record (of some type) then $T \mid r$ is a type. $a : T \mid r$ just in case for some $\ell$ occurring in $r$, $r.\ell = a$ and $a : T$ (equivalently, $r : (\ell : T)$). Intuitively, $T \mid r$ is the type of objects of type $T$ which occur as a value in $r$. We will refer to $r$ in $T \mid r$ as a resource. We can use resources in various ways. We can introduce specific resources:

$$
\begin{align*}
  f &: (r : \left[ x : Ind \mid res_1, c_1 : \text{man}(x) \mid res_1 \right]) \\
  y &: Ind \mid res_2 \\
  c_2 &: \text{donkey}(y) \mid res_2 \\
  c_3 &: \text{own}(r.\cdot, y)
\end{align*}
$$

This corresponds to an intuitive reading: “Every man in $res_1$ owns a donkey in $res_2$.” Note that a notation such as $\text{man}(x) \mid res_1$ corresponds intuitively to what a

---

[11] The “classical” DRT treatment of donkey anaphora presented here has its shortcomings as has been pointed out in the literature on this topic (e.g. [5]). Weak readings can be provided and the proportion problem fixed by introducing generalized quantifiers into our approach [7].
situation theorist would write as $res_1 \models \langle \langle \text{man}, x; 1 \rangle \rangle$. That is, the type of proofs man(x) corresponds to a situation theoretic infon (type of situation)\(^{12}\) and the notion of belonging to a restricted type corresponds to the situation theoretic notion of support.

We can introduce resources which are existentially quantified, obtaining a kind of non-specific reading:

\[
\begin{align*}
\text{res}_1 & : \text{Rec} \\
\text{res}_2 & : \text{Rec} \\
f & : (r : \left[ \begin{array}{l}
x : \text{Ind}|\text{res}_1 \\
c_1 : \text{man}(x)|\text{res}_1 \\
y : \text{Ind}|\text{res}_2 \\
c_2 : \text{donkey}(y)|\text{res}_2 \\
c_3 : \text{own}(r,x,y)
\end{array} \right])
\end{align*}
\]

This corresponds to the intuitive reading “There are resources $\text{res}_1$, $\text{res}_2$ such that every man in $\text{res}_1$ owns a donkey in $\text{res}_2$.”

We can also quantify over resources, obtaining a kind of generic reading:

\[
\begin{align*}
\text{res}_1 & : \text{Rec} \\
x & : \text{Ind}|\text{res}_1 \\
c_1 & : \text{man}(x)|\text{res}_1 \\
\text{res}_2 & : \text{Rec} \\
y & : \text{Ind}|\text{res}_2 \\
c_2 & : \text{donkey}(y)|\text{res}_2 \\
c_3 & : \text{own}(r,x,y)
\end{align*}
\]

This corresponds to the intuitive reading “For every resource $\text{res}_1$ and every man $x$ in $\text{res}_1$ there is a resource $\text{res}_2$ such that $x$ owns a donkey in $\text{res}_2$."

There is an important way in which the notion of resource introduced here differs from the notion of resource situation in situation semantics. Types restricted to a resource import information from the resource in a similar fashion to the way modules are imported into other modules in programming languages. Suppose that we have a record $r : [\ell : T \mid r']$. This will require that for some $\ell'$, $r' : [\ell' : T]$. But note now that $r : [\ell : T]$ also and $r$ can itself be used as a resource. We can regard $r$ as having imported information from $r'$. The information is present in $r$ as well and we can refer to it without having to refer to $r'$ when we use $r$ as a resource. This notion of importation was not present in situation theory, in part because the notion of resource situation was a way of using situations in situation semantics but was not a notion defined in situation theory as such.

### 3.4 HPSG

HPSG provides our approach with

- feature structures
- the notion of sign, a relational, single level and constraint based approach to the relationship between phonology, syntax and semantics

HPSG gains

- the ability to represent both types and objects. Feature structures in HPSG can be viewed either as something like types or as something like objects but using records and record types gives us a formalism in which both the types and the objects can be represented. (from type theory)

\(^{12}\)cf. the discussion of situations as proof objects on page 3.
• a direct treatment of semantics rather than a coding of it in terms of a unification based formalism. Our type theoretical approach includes the usual kind of paraphernalia which formal semanticist use, including functions and binding. (from type theory)

• dynamic binding and discourse interpretation (from DRT)

• compositionality using the \( \lambda \)-calculus (from Montague semantics)

• a more faithful treatment of situation semantics than is possible in a unification based formalism (from situation semantics)

We will illustrate our approach to constructing HPSG-like grammars by giving examples of the way in which such a grammar can be constructed.

Our grammar will require a system of types with the following basic types: \( \text{Lex} \) (lexical items), \( \text{Cat} \) (categories), \( \text{Ind} \) (individuals). Thus our types now include not only types of objects (like individuals) which are used in order to construct semantic interpretations but also types of linguistic objects like words and categories. To make this concrete for our example, we consider models where \( \text{Lex} \) is assigned the set \{a, every, man, donkey, owns, beats, who, he, him, she, her, it\} and where \( \text{Cat} \) is assigned the set \{D(iscourse), S, NP, VP, V, Det, N, NBar, RelPro, Rel, Pron\}.

We will need to exploit list types which are standard in type theory. For example, we will follow the HPSG tradition of using lists of lexical items in place of real phonology. Thus we define the type \( \text{Phon} \) as follows: \( \text{Phon} \equiv [\text{Lex}] \)

Similarly we will introduce disjunction in order to be able to treat the kind of type hierarchies that HPSG uses. Thus, for example, we can define the type \( \text{Sign} \) as follows:

\[
\text{Sign} \equiv D\text{Sign} \lor S\text{Sign} \lor N\text{PSign} \lor V\text{PSign} \lor V\text{Sign} \lor D\text{etSign} \lor N\text{Sign} \\
\lor R\text{elProSign} \lor R\text{elSign} \lor P\text{ronSign}
\]

In the classical approach to syntax and semantics we define syntactic rules (e.g. phrase-structure rules) and semantic interpretation recursively on these rules or structures generated by these rules, e.g. in the following standard format for a rule combining a noun-phrase and a verb-phrase to form a sentence:

\[
[ [S \text{ NP VP} ] ] = [ [ \text{ NP} ] ] \circ [ \text{ VP} ]
\]

In contrast to this the HPSG approach defines that type of signs which are sentences simultaneously characterising both syntactic and semantic aspects. Corresponding to this rule we define \( S\text{Sign} \) to be the following type:

\[
13\text{See } [8] \text{ for an account of the relevant notion of model.}
14\text{This is a more classical assortment of categories than is normally found in HPSG grammars, which take a more abstract approach to the notion of category. However, it helps us to see the relationship between HPSG and standard Montague semantics and DRT. We do not foresee any problems with following HPSG analyses more closely once we have understood the basic strategy for setting up HPSG-like grammars in our approach.}
15\text{See } [8].
16\text{The disjunction we use is not the kind of disjunction used in Martin-Löf type theory since we use a standard join rather than disjoint union. I believe that Martin-Löf’s disjunction could be used and that the only cost would be an increased complexity in the notation of the grammar. See } [8] \text{ for a precise characterisation of the notion of disjunction.}
\]
Note that we are making crucial use of manifest fields which we introduced earlier in connection with the treatment of anaphora in DRT. Disjunction is also important here since there is in general more than one way of building an object of a given category, e.g. sentences can also be formed by placing and between two sentences so we would need an additional syntactic rule (and corresponding interpretation rule). So in general types like SSgn will be a disjunction of a number of types.

An important consequence of including HPSG in our collection of theories is that our type theoretical approach thereby becomes not just a way of formulating a semantic theory but rather a general linguistic theory encompassing phonology, syntax and semantics. It is appealing that the same theory of types should underly all aspects of linguistic analysis, so that for example a choice made in the type theory to support a phonological analysis may have consequences for semantics. In work in progress (see Using Type Theory with Records for HPSG on http://www.ling.gu.se/~cooper/records) we have begun to explore a view of robust parsing utilising the strong intensionality of the record types we have discussed in this paper. The idea has to do with the fact that two distinct record types can have the same extension. This is important for the semantic analysis of belief and other attitudes, as is well known. It also provides an interesting view of robust parsing. Distinct empty types containing rich information about ungrammatical input can be used to explain how we manage to reason about utterances of phrases that do not belong to the language. We speculate that the kind of intensionality needed for the analysis of the attitudes is the same as that needed for robust parsing and that this can be captured by taking this type theoretical approach to a theory like HPSG.

4 Conclusion

In this paper, we have given an overview of how a type theory including records can be used to combine four linguistic theories into a single theory which has advantages from all of them. At the core of the theory is the notion of type theoretical judgement, i.e. the judgement that a given object is of a given type. This central notion and the type theory built around it provides us both with our treatment of syntax and of semantics. Within semantics it is this central notion which provides us with our treatment of basic extensional semantics, intensionality and discourse anaphora. In classical Montague semantics when moving from an extensional semantics to an intensional semantics we need to add additional semantic objects and techniques (essentially possible worlds). In this theory the notion of type we define explains both basic extensional semantics, intensional semantics and also DRT style anaphora. Thus the underlying theory of types in our approach is doing more work for us that any of the individual components of the various theories that we are combining. This makes for a theory which is explanatory in Chomsky’s sense. The choices we make in our type theory will make
predictions for various different aspects of semantics and syntax. For example, the
troduction of singleton types and manifest fields has consequences both for the
treatment of anaphora and the HPSG-style treatment of signs. This economy in our
theory also has computational implications. It means that an implementation of our
type theory will serve as the computational tool for implementation of various aspects
of natural language and that we reduce the need for interfacing subsystems defined
using different computational tools.

Acknowledgements

This work was supported by Vetenskapsrådet project number 2002-4879 Records, types
I am grateful to Thierry Coquand, Jonathan Ginzburg, Fritz Hamm, Staffan Larsson,
Uwe Mönich, Yiannis Moschovakis, Bengt Nordström and Arne Ranta for helpful
comments.

References

[2] Beaver, David. The Optimization of Discourse Anaphora. Linguistics and Phi-
[3] Betarte, Gustavo. Dependent Record Types and Algebraic Structures in Type The-
Chalmers University of Technology, 1998.
with record types and subtyping. In Sambin, Giovanni and Jan Smith, eds.
Twenty-Five Years of Constructive Type Theory, Oxford Logic Guides, 36, Ox-
[6] Cooper, Robin. The Role of Situations in Generalized Quantifiers. In Lappin,
Shalom, ed. The Handbook of Contemporary Semantic Theory, Blackwell, Oxford,
1996.
Ursus Philosophicus, a festschrift for Björn Haglund. Department of Philosophy,
and Computation, forthcoming.


