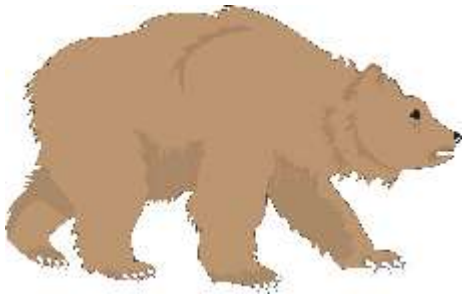
Philosophical Communications, Web Series, No. 32, pp. 47-81.
Dept. of Philosophy, Göteborg University, Sweden
ISSN 1652-0459

Ursus Philosophicus

Essays dedicated to Björn Haglund on his sixtieth birthday



Preprint, © 2004 Robin Cooper. All rights reserved

Preprint, © 2004 Robin Cooper. All rights reserved

Dynamic generalised quantifiers and hypothetical contexts

Robin Cooper

Abstract

We shall consider a formulation of generalised quantifiers using type theory with records (TTR). TTR follows closely the development of record types in Martin-Löf or constructive type theory but differs in that the type theory is defined on a classical set theoretic basis. This means that the classical set-theoretic approach to generalised quantifiers can be imported into the type theoretic framework. The result is, I believe, equivalent to the proposal for dynamic generalised quantifiers in Chierchia (1995). The use of dependent types provides us with an elegant approach to the formulation of dynamic quantifiers. We use a notion of *hypothetical context* which we have used elsewhere to give accounts of intentional identity, answers to questions and information state updates in dialogue management. We suggest that this points towards a general theory of hypothetical context in natural language. We suspect also that our analysis using records will support analyses of common noun phrase and verb-phrase anaphora and also facilitate representations which are underspecified with respect to quantifier scope, though we leave the investigation of this to future research.

1 Introduction

We[1] shall present a formulation of generalised quantifiers using type theory with records as discussed in Cooper(2003, forthcoming). TTR follows closely the development of record types in Martin-Löf or constructive type theory (Tasistro, 1997, Betarte and Tasistro, 1998, Betarte, 1998, Coquand *et al.*, 2004) but differs in that the type theory is defined on a classical set theoretic basis. This means that the classical set-theoretic approach to generalised quantifiers can be imported into the type theoretic framework. We shall first present an account of type theory with records and then show how it can be used to interpret a small fragment of English with dynamic (but not generalised) quantifiers. We will then show how non-dynamic generalised quantifiers can be added to this fragment and then present a dynamic version of these quantifiers. The resulting dynamic quantifiers appear to be equivalent to the dynamic quantifiers discussed elsewhere in the literature (as in Chierchia, 1995). Our formulation involves a notion of *hypothetical context* which we have used elsewhere (Cooper, 2003, in preparation) in the analysis of other natural language phenomena, namely intentional identity, answers to questions and information state updates in dialogue management. Our conclusion will be that our type theoretical approach connects dynamic generalised quantifiers to a general theory of hypothetical context in natural language.

2 Type theory with records (TTR)

This section[2] begins by introducing a variant of type theory which blends important aspects of Martin-Löf type theory (dependent types, proof types and record types) with elements of classical model theory used in Montague semantics (Montague, 1974).

[1]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, and for pointing me towards a way of thinking about generalised quantifiers in type theory as well as pointing out how many of my ideas relate to those presented in Ranta (1994). 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, Bengt Nordström, Rich Thomason and Ray Turner. This work was supported by Vetenskapsrådet project number 2002-4879, Records, types and computational dialogue semantics.

[2]This section contains material from Cooper(2003).

2.1 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 $< \ell, v >$ 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 $< \ell_1, v_1 >, \ldots, < \ell_n, v_n >$ is displayed as

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

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

$$\begin{bmatrix} f & = & \begin{bmatrix} f & = & \begin{bmatrix} ff & = & a \\ gg & = & b \end{bmatrix} \\ g & = & c \end{bmatrix} \\ g & = & \begin{bmatrix} h & = & \begin{bmatrix} g & = & a \\ h & = & d \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

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

$$\begin{bmatrix} f & = & \begin{bmatrix} ff & = & a \\ gg & = & b \end{bmatrix} \\ g & = & c \end{bmatrix},$$

$$\begin{bmatrix} g & = & a \\ h & = & d \end{bmatrix}$$

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

$$\begin{bmatrix} f.f.ff & = & a \\ f.f.gg & = & b \\ f.g & = & c \\ g.h.g & = & a \\ g.h.h & = & d \end{bmatrix}$$

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 $\pi_2{}'$), 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.

$$\begin{bmatrix} f & = & \begin{bmatrix} f & = & \begin{bmatrix} ff & = & a \\ gg & = & b \end{bmatrix} \\ g & = & c \end{bmatrix} \\ g & = & \begin{bmatrix} h & = & \begin{bmatrix} k & = & a \\ h & = & d \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

2.2 Types and Objects

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

$<$**Type**^{n}, **BType**^{n}, $<$**PfType**^{n}, **Pred**^{n}, *Arity*$^n>$, $<$**RecType**^{n}, **Labels**$>$, $< A^n, F^n >>_{n \in Nat}$

where:

1. **Type**^{n} is the set of types of order n, defined by the recursive definition *A* below.
2. **BType**^{n}, the set of basic types, is a subset of **Type**^{n} for any n.

3. **PfType**^{n}, the set of basic types of proof of order n, is a subset of **Type**^{n} defined with respect to a set **Pred**^{n} of predicates of order n and a function *Arity*^{n} which assigns to each member of **Pred**^{n} a finite tuple of members of **Type**^{n}. If $P \in$ **Pred**^{n}, then $P \in$ **Pred**$^{n+1}$ and *Arity*$^n(P)$=*Arity*$^{n+1}(P)$. The sets **PfType**^{n} are defined by the definition *B* below.

4. **RecType**^{n}, the set of record types of order n, which is a subset of **Type**^{n} defined with respect to a countably infinite set **Labels** of objects used as labels. The sets **RecType**^{n} are defined by the recursive definition *C* below.

5. $< A^n, F^n >$ is a model where A^n is a function from **BType**^{n} to sets (of inhabitants of the basic types) and F^n is a function from **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 is association with the definitions of **Type**^{n}, **PfType**^{n} and **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. [3]

[3] It might be suspected that this lack of independence between types and models will exclude classical modal logic based approaches to modality and intensionality. However, it appears we could imitate Montague's approach to intensionality and modality by letting F assign to each member of **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.

A. Definition of **Type**^{n}

1. if T is a member of **Type**^{n}, then T is also a member of **Type**$^{n+1}$.
2. if T is a member of **BType**^{n}, then T is also a member of **BType**$^{n+1}$.
3. If T is a member of **BType**^{n}, then T is a member of **Type**^{n}.

   If T is a member of **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 **Type**$^{n+1}$.

   $T : Type^n$ iff $T \in$ **Type**^{n}

   $T : RecType^n$ iff $T \in$ **RecType**^{n}
5. if $\mathcal{P}$ is a member of **PfType**^{n} then $\mathcal{P}$ is also a member of **Type**^{n}.
6. if R is a member of **RecType**^{n} then R is also a member of **Type**^{n}.
7. if T_1 and T_2 are members of **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 **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 **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 **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. if T_1 and T_2 are members of **Type**^{n}, $T_1 \vee T_2$, the join (disjunction) of T_1 and T_2, is a member of **Type**^{n}.

   $a : T_1 \vee T_2$ iff $a : T_1$ or $a : T_2$.
10. if T is a member of **Type**^{n}, then $[T]$, the type of lists each of whose members are of type T, is a member of **Type**^{n}.

    $[a_1, \ldots, a_n] : [T]$ iff for each i, $1 \leq i \leq n$, $a_i : T$.
11. if T is a member of **Type**^{n} and $x : T$, then T_x, a singleton type of x, is a member of **Type**^{n}.

    $a : T_x$ iff $a = x$

B. Definition of **PfType**^{n}

1. if P is a member of **Pred**, *Arity*^{n}(P) = $< T_1, \ldots, T_n >$ and $a_1, \ldots, a_n$ are such that $a_1 : T_1, \ldots, a_n : T_n$, then $P(a_1, \ldots, a_n) \in$ **PfType**^{n}.

   If $T = P(a_1, \ldots, a_n)$ is a member of **PfType**^{n}, then $a : T$ iff $a \in F^n(T)$

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

C. Definition of **RecType**^{n}

For any n, **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 **RecType**^{n}, then R is also a member of **RecType**$^{n+1}$.
2. *Rec*^{n} (also represented by $[]^n$) is a member of **RecType**$^{n+1}$.

   [] (the empty record) : *Rec*^{0}

   If $T \in$ **RecType**^{n} and $r : T$, then $r : Rec^n$
3. if R is a member of **RecType**^{n}, ℓ is a member of **Labels** not occurring as a label in R and T is a member of **Type**^{n}, then $R \cup \{< \ell, T >\}$ is a member of **RecType**^{n}.

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

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

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

We represent a record type $\{< \ell_1, T_1 >, \ldots, < \ell_n, T_n >\}$ graphically as

$$\begin{bmatrix} \ell_1 & : & T_1 \\ \cdots & & \\ \ell_n & : & T_n \end{bmatrix}$$

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)

$$\begin{bmatrix} \ell_i{=}x & : & T' \end{bmatrix}$$

In the case of dependent types introduced by clause 4 of the definition of record types, we use a convenient notation representing e.g. $< \lambda u \lambda v$ love$(u, v), < \pi_1, \pi_2 >>$ as love(π_1, π_2).

3 Using TTR to interpret English

We use $f@a$ to represent application of the function f to the argument a.

For the content 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):

$$^{\lambda \ell_1}\begin{bmatrix} \ell_1 & : & T_1 \\ \ell_2 & : & T_2(\ell_1) \\ & \vdots & \\ \ell_n & : & T_n(\ell_1, \ell_2, \ldots, \ell_{n-1}) \end{bmatrix} = \lambda r{:}\begin{bmatrix}\ell_i{:}T_i\end{bmatrix}(\begin{bmatrix} \ell_2 & : & T_2(r.\ell_1) \\ & \vdots & \\ \ell_n & : & T_n(r.\ell_1, \ell_2, \ldots, \ell_{n-1}) \end{bmatrix})$$

where T_1 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:

$$^{\text{tvr}}\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Ind} \\ \text{c} & : & T(\text{x,y}) \end{bmatrix} = \lambda\mathcal{N}{:}((\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType})\mathit{RecType}\ \lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]$$
$$(\mathcal{N}\ @\ \lambda r_2{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}T(r_1.\text{x},\ r_2.\text{x})\right]))$$

We also use a version of Montague's type raising for proper nouns and pronouns.

$$\text{If } T = \left[\text{x}{=}y{:}\mathit{Ind}\right] \text{ then } ^{\text{npr}}T = \lambda R{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x}{=}y & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & R\ @\ \text{par} \end{bmatrix}$$

We use '?' to represent metavariables which will be replaced by pronoun resolution with path names to labels on which the type where they occur may depend. The DRT notion of accessibility is given by the notion of dependence of types and thus does not have to be defined separately.

We provide rules which define the interpretation of labelled bracketings where $[\![\ A\ ]\!]$ represents the interpretation of A according to standard linguistic conventions.

Our grammar will require a system of types with the basic type *Ind* (individual) at level 0.

$[\![\ [_{\text{D}}\ \text{S}]\ ]\!] = [\![\ \text{S}\ ]\!]$

$[\![\ [_{\text{D}}\ \text{D S}]\ ]\!] = \begin{bmatrix} \text{d} & : & [\![\ \text{D}\ ]\!] \\ \text{s} & : & [\![\ \text{S}\ ]\!] \end{bmatrix}$

$[\![\ [_{\text{S}}\ \text{NP VP}]\ ]\!] = [\![\ \text{NP}\ ]\!]\ @\ [\![\ \text{VP}\ ]\!]$

$[\![\ [_{\text{VP}}\ \text{V NP}]\ ]\!] = [\![\ \text{V}\ ]\!]\ @\ [\![\ \text{NP}\ ]\!]$

$[\![\ [_{\text{NP}}\ \text{Det NBar}]\ ]\!] = [\![\ \text{Det}\ ]\!]\ @\ [\![\ \text{NBar}\ ]\!]$

$[\![\ [_{\text{Rel}}\ \text{RelPro VP}]\ ]\!] = [\![\ \text{RelPro}\ ]\!]\ @\ [\![\ \text{VP}\ ]\!]$

$[\![\ [_{\text{NBar}}\ \text{N Rel}]\ ]\!] = [\![\ \text{Rel}\ ]\!]\ @\ [\![\ \text{N}\ ]\!]$

$[\![\ [_{\text{NBar}}\ \text{N}]\ ]\!] = [\![\ \text{N}\ ]\!]$

$[\![\ [_{\text{Det}}\ \text{a}]\ ]\!] = \text{a}'$

$[\![\ [_{\text{Det}}\ \text{every}]\ ]\!] = \text{every}'$

$[\![\ [_{\text{N}}\ \text{man}]\ ]\!] = \text{man}'$

$[\![\ [_{\text{N}}\ \text{donkey}]\ ]\!] = \text{donkey}'$

$[\![\ [_{\text{V}}\ \text{owns}]\ ]\!] = \text{own}'$

$[\![\ [_{\text{V}}\ \text{beats}]\ ]\!] = \text{beat}'$

$[\![\ [_{\text{RelPro}}\ \text{who}]\ ]\!] = \text{who}'$

$[\![\ [_{\text{NP}}\ \alpha]\ ]\!] = \alpha'$ where $\alpha \in \{\text{he, him, she, her, it}\}$

$$\text{a}' = \lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & R_1\ @\ \text{par} \\ \text{scope} & : & R_2\ @\ \text{par} \end{bmatrix}$$

$$\text{every}' = \lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & R_1\ @\ \text{par} \end{bmatrix})\ R_2\ @\ r.\text{par} \end{bmatrix}$$

$$\text{man}' = {}^{\lambda\text{x}}\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \text{man(x)} \end{bmatrix}$$

$$\text{donkey}' = {}^{\lambda\text{x}}\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \text{donkey(x)} \end{bmatrix}$$

$$\text{own}' = {}^{\text{tvr}}\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Ind} \\ \text{c} & : & \text{own(x,y)} \end{bmatrix}$$

$$\text{beat}' = {}^{\text{tvr}}\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Ind} \\ \text{c} & : & \text{beat(x,y)} \end{bmatrix}$$

$$\text{who}' = \lambda R_1{:}(\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\lambda R_2{:}(\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\lambda r{:}\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix} \text{c} & : & \begin{bmatrix} \text{pred} & : & R_2 \mathbin{@} r \\ \text{mod} & : & R_1 \mathbin{@} r \end{bmatrix} \end{bmatrix})$$

$$\text{he}' = \text{him}' = \text{she}' = \text{her}' = \text{it}' = \overset{\text{npr}}{\begin{bmatrix}\text{x=?}{:}\mathit{Ind}\end{bmatrix}}$$

The definitions so far will assign (functions with range) *underspecified* record types to phrases of English, where an underspecified record type is one that contains a metavariable. An underspecified record type ρ is *resolved* to a record type R by substituting all occurrences of metavariables ν with paths π meeting the following conditions:

1. π is of the form ....par.x
2. π is either a path in R or of the form $r.\pi'$ where π' is a path in record r.
3. in an underspecified type

$$\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x}{=}\nu & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & P(\pi_1, \ldots, \pi_i, \ldots, \pi_n) \end{bmatrix} \\ \ldots & & \end{bmatrix}$$

where π_i is a path to par.x in this type, then the subsitution π for ν is distinct from each $\pi_1, \ldots, \pi_n$.

The last condition is for non-reflexive pronouns. In a more complete grammar we would need to make a distinction between reflexive and non-reflexive pronouns and we leave a more precise formulation of resolution constraints until later. The plan is to take an optimality theoretic approach to the formulation of these constraints along the lines of Beaver (2004).

This grammar will assign record types to sentences of English. By flattening and relabelling these record types can be reduced to equivalent record types which are easier to read.

3.1 Sample derivation: *every man owns a donkey*

a donkey

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

@

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

=

$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\begin{bmatrix} \text{par} & : & \left[\ \text{x} \ : \ \mathit{Ind}\ \right] \\ \text{restr} & : & \left[\ \text{c} \ : \ \text{donkey(par.x)}\ \right] \\ \text{scope} & : & R_2 \text{ @ par} \end{bmatrix}$$

own a donkey

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

@

$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\begin{bmatrix} \text{par} & : & \left[\ \text{x} \ : \ \mathit{Ind}\ \right] \\ \text{restr} & : & \left[\ \text{c} \ : \ \text{donkey(par.x)}\ \right] \\ \text{scope} & : & R_2 \text{ @ par} \end{bmatrix}$$

=

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]\ (\begin{bmatrix} \text{par} & : & \left[\ \text{x} \ : \ \mathit{Ind}\ \right] \\ \text{restr} & : & \left[\ \text{c} \ : \ \text{donkey(par.x)}\ \right] \\ \text{scope} & : & \left[\ \text{c} \ : \ \text{own}(r_1.\text{x,par.x})\ \right] \end{bmatrix})$$

every man

$$\lambda R_1{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\ \lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & R_1 \text{ @ par} \end{bmatrix}) R_2 \text{ @ } r.\text{par} \end{bmatrix}$$

@

$$\lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:man}(r.\text{x})\end{bmatrix})$$

=

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{man(par.x)} \end{bmatrix} \end{bmatrix}) R_2 \text{ @ } r.\text{par} \end{bmatrix}$$

every man owns a donkey

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{man(par.x)} \end{bmatrix} \end{bmatrix}) R_2 \text{ @ } r.\text{par} \end{bmatrix}$$

@

$$\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\ (\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{donkey(par.x)} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{own}(r_1.\text{x,par.x}) \end{bmatrix} \end{bmatrix})$$

=

$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{man(par.x)} \end{bmatrix} \end{bmatrix}) \begin{bmatrix} \text{par} & : \begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix} \\ \text{restr} & : \begin{bmatrix}\text{c:donkey(par.x)}\end{bmatrix} \\ \text{scope} & : \begin{bmatrix}\text{c:own}(r.\text{par.x,par.x})\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Flattening

$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par.x} & : & \textit{Ind} \\ \text{restr.c} & : & \text{man(par.x)} \end{bmatrix}) \begin{bmatrix} \text{par.x} & : & \textit{Ind} \\ \text{restr.c} & : & \text{donkey(par.x)} \\ \text{scope.c} & : & \text{own}(r\text{.par.x,par.x}) \end{bmatrix} \end{bmatrix}$$

Relabelling

$$\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \end{bmatrix}) \begin{bmatrix} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own}(r\text{.x,y}) \end{bmatrix} \end{bmatrix}$$

3.2 Sample derivation: *a man owns a donkey*

a donkey

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

@

$$\lambda r{:}\left[\text{x}{:}\textit{Ind}\right](\left[\text{c}{:}\text{man}(r\text{.x})\right])$$

=

$$\lambda R_2{:}(\left[\text{x}{:}\textit{Ind}\right])\textit{RecType} \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \textit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{man(par.x)} \end{bmatrix} \\ \text{scope} & : & R_2 \text{ @ par} \end{bmatrix}$$

a man owns a donkey

$$\lambda R_2{:}(\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x} & : & \mathit{Ind}\end{bmatrix} \\ \text{restr} & : & \begin{bmatrix}\text{c} & : & \text{man(par.x)}\end{bmatrix} \\ \text{scope} & : & R_2 \text{ @ par}\end{bmatrix}$$

@

$$\lambda r_1{:}\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix}\left(\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x} & : & \mathit{Ind}\end{bmatrix} \\ \text{restr} & : & \begin{bmatrix}\text{c} & : & \text{donkey(par.x)}\end{bmatrix} \\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{own}(r_1\text{.x, par.x})\end{bmatrix}\end{bmatrix}\right)$$

=

$$\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x} & : & \mathit{Ind}\end{bmatrix} \\ \text{restr} & : & \begin{bmatrix}\text{c} & : & \text{man(par.x)}\end{bmatrix} \\ \text{scope} & : & \begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x} & : & \mathit{Ind}\end{bmatrix} \\ \text{restr} & : & \begin{bmatrix}\text{c} & : & \text{donkey(scope.par.x)}\end{bmatrix} \\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{own(par.x, scope.par.x)}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Flattening

$$\begin{bmatrix}\text{par.x} & : & \mathit{Ind} \\ \text{restr.c} & : & \text{man(par.x)} \\ \text{scope.par.x} & : & \mathit{Ind} \\ \text{scope.restr.c} & : & \text{donkey(scope.par.x)} \\ \text{scope.scope.c} & : & \text{own(par.x, scope.par.x)}\end{bmatrix}$$

Relabelling

$$\begin{bmatrix}\text{x} & : & \mathit{Ind} \\ \text{c}_1 & : & \text{man(x)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own(x, y)}\end{bmatrix}$$

3.3 Sample derivation: *every man who owns a donkey beats it*

who owns a donkey

$$\lambda R_1{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\ \lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\ \lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c} & : & \begin{bmatrix}\text{pred} & : & R_2\ @\ r\\ \text{mod} & : & R_1\ @\ r\end{bmatrix}\end{bmatrix})$$

@

$$\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\ (\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x} & : & \mathit{Ind}\end{bmatrix}\\ \text{restr} & : & \begin{bmatrix}\text{c} & : & \text{donkey(par.x)}\end{bmatrix}\\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{own}(r_1\text{.x,par.x})\end{bmatrix}\end{bmatrix})$$

=

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\ \lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}R_2\ @\ r\\ \text{mod:}\begin{bmatrix}\text{par} & :\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr} & :\begin{bmatrix}\text{c:donkey(c.mod.par.x)}\end{bmatrix}\\ \text{scope:}\begin{bmatrix}\text{c:own}(r\text{.x, c.mod.par.x})\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix})$$

man who owns a donkey

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\ \lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}R_2\ @\ r\\ \text{mod:}\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr} & : & \begin{bmatrix}\text{c:donkey(c.mod.par.x)}\end{bmatrix}\\ \text{scope} & : & \begin{bmatrix}\text{c:own}(r\text{.x, c.mod.par.x})\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix})$$

@

$$\lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:man}(r\text{.x})\end{bmatrix})$$

=

$$\lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}\begin{bmatrix}\text{c:man}(r\text{.x})\end{bmatrix}\\ \text{mod:}\begin{bmatrix}\text{par} & :\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr} & :\begin{bmatrix}\text{c:donkey(c.mod.par.x)}\end{bmatrix}\\ \text{scope:}\begin{bmatrix}\text{c:own}(r\text{.x, c.mod.par.x})\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix})$$

every man who owns a donkey

$$\lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}$$
$$(\begin{bmatrix} \text{f} & : & (r : \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & R_1 \text{ @ par} \end{bmatrix}) \ R_2 \text{ @ } r.\text{par} \end{bmatrix})$$

@

$$\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\begin{bmatrix} \text{c} & : & \begin{bmatrix} \text{pred} & : & \begin{bmatrix} \text{c} & : & \text{man}(r.\text{x}) \end{bmatrix} \\ \text{mod} & : & \begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{donkey(c.mod.par.x)} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{own}(r.\text{x, c.mod.par.x}) \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix})$$

=

$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}$$
$$(\begin{bmatrix} \text{f:}(r : \begin{bmatrix} \text{par} & : \left[\text{x}{:}\mathit{Ind}\right] \\ \text{restr:} & \begin{bmatrix} \text{c:} & \begin{bmatrix} \text{pred:} & \left[\text{c:man(par.x)}\right] \\ \text{mod:} & \begin{bmatrix} \text{par} & : \left[\text{x}{:}\mathit{Ind}\right] \\ \text{restr} & : \left[\text{c:donkey(restr.c.mod.par.x)}\right] \\ \text{scope:} & \left[\text{c:own(par.x, restr.c.mod.par.x)}\right] \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}) \ R_2 \text{ @ } r.\text{par} \end{bmatrix})$$

beats it

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

@

$$\lambda R{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ (\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x=?} & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & R \text{ @ par} \end{bmatrix})$$

=

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]\ (\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x=?} & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{beat}(r_1.\text{x, par.x}) \end{bmatrix} \end{bmatrix})$$

every man who owns a donkey beats it

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$\begin{bmatrix}\text{f:}(r:\begin{bmatrix}\text{par}:\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr:}\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}\begin{bmatrix}\text{c:man(par.x)}\end{bmatrix}\\ \text{mod:}\begin{bmatrix}\text{par}:\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr}:\begin{bmatrix}\text{c:donkey(restr.c.mod.par.x)}\end{bmatrix}\\ \text{scope:}\begin{bmatrix}\text{c:own(par.x, restr.c.mod.par.x)}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix})\ R_2\ @\ r.\text{par}\end{bmatrix}$$

@

$$\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\ (\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x=?} & : & \mathit{Ind}\end{bmatrix}\\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{beat}(r_1\text{.x, par.x})\end{bmatrix}\end{bmatrix})$$

=

$$\begin{bmatrix}\text{f:}(r:\begin{bmatrix}\text{par}:\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr:}\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}\begin{bmatrix}\text{c:man(par.x)}\end{bmatrix}\\ \text{mod:}\begin{bmatrix}\text{par}:\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ \text{restr}:\begin{bmatrix}\text{c:donkey(restr.c.mod.par.x)}\end{bmatrix}\\ \text{scope:}\begin{bmatrix}\text{c:own(par.x, restr.c.mod.par.x)}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix})\\ \begin{bmatrix}\text{par}:\begin{bmatrix}\text{x=?:}\mathit{Ind}\end{bmatrix}\\ \text{scope:}\begin{bmatrix}\text{c:beat}(r\text{.par.x, par.x})\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Resolution

We find candidate paths for the resolution of the metavariable ‘?’ by looking for paths of the form . . . .par.x:*Ind*. The candidates are

r.par.x
r.restr.c.mod.par.x
par.x
and in addition if there were any r' defined (representing context) then any path r'. . . . .par.x would be included in the list (provided x:*Ind*)

The first and third of these are ruled out by grammatical constraints not yet included in

the grammar. Therefore we choose the second.

$$\begin{bmatrix} \text{f:}(r: \begin{bmatrix} \text{par} : \begin{bmatrix} \text{x:}\mathit{Ind} \end{bmatrix} \\ \text{restr:} \begin{bmatrix} \text{c:} \begin{bmatrix} \text{pred:} \begin{bmatrix} \text{c:man(par.x)} \end{bmatrix} \\ \text{mod:} \begin{bmatrix} \text{par} : \begin{bmatrix} \text{x:}\mathit{Ind} \end{bmatrix} \\ \text{restr} : \begin{bmatrix} \text{c:donkey(restr.c.mod.par.x)} \end{bmatrix} \\ \text{scope:} \begin{bmatrix} \text{c:own(par.x, restr.c.mod.par.x)} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \text{par} : \begin{bmatrix} \text{x=}r\text{.restr.c.mod.par.x:}\mathit{Ind} \end{bmatrix} \\ \text{scope:} \begin{bmatrix} \text{c:beat(}r\text{.par.x, par.x)} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Flattening

$$\begin{bmatrix} \text{f} : (r: \begin{bmatrix} \text{par.x} & : & \mathit{Ind} \\ \text{restr.c.pred.c} & : & \text{man(par.x)} \\ \text{restr.c.mod.par.x} & : & \mathit{Ind} \\ \text{restr.c.mod.restr.} & : & \text{donkey(restr.c.mod.par.x)} \\ \text{restr.c.mod.scope.c} & : & \text{own(par.x, restr.c.mod.par.x)} \end{bmatrix}) \\ \begin{bmatrix} \text{par.x=}r\text{.restr.c.mod.par.x} & : & \mathit{Ind} \\ \text{scope.c} & : & \text{beat(}r\text{.par.x, par.x)} \end{bmatrix} \end{bmatrix}$$

Relabelling

$$\begin{bmatrix} \text{f} : (r: \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_1 & : & \text{man(x)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own(x, y)} \end{bmatrix}) \\ \begin{bmatrix} \text{z=}r\text{.y} & : & \mathit{Ind} \\ \text{c}_5 & : & \text{beat(}r\text{.x, z)} \end{bmatrix} \end{bmatrix}$$

Manifest field elimination (plus relabelling of c_5)

$$\begin{bmatrix} \text{f} \quad : \quad (r : \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \\ \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own(x, y)} \end{bmatrix}) \\ \begin{bmatrix} \text{c}_4 & : & \text{beat}(r.\text{x}, r.\text{y}) \end{bmatrix} \end{bmatrix}$$

3.4 Sample derivation: *A man owns a donkey. He beats it.*

A man owns a donkey.

$$\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \\ \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own(x, y)} \end{bmatrix}$$

He beats it.

$$\begin{bmatrix} \text{x=?} & : & \textit{Ind} \\ \text{y=?} & : & \textit{Ind} \\ \text{c}_3 & : & \text{beat(x, y)} \end{bmatrix}$$

A man owns a donkey. He beats it.

$$\begin{bmatrix} \text{d} \quad : & \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(d.x)} \\ \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(d.y)} \\ \text{c}_3 & : & \text{own(d.x, d.y)} \end{bmatrix} \\ \text{s} \quad : & \begin{bmatrix} \text{x=?} & : & \textit{Ind} \\ \text{y=?} & : & \textit{Ind} \\ \text{c}_3 & : & \text{beat(s.x, s.y)} \end{bmatrix} \end{bmatrix}$$

Resolution

$$\begin{bmatrix} d : & \begin{bmatrix} x & : & \textit{Ind} \\ c_1 & : & \text{man}(d.x) \\ y & : & \textit{Ind} \\ c_2 & : & \text{donkey}(d.y) \\ c_3 & : & \text{own}(d.x, d.y) \end{bmatrix} \\ s : & \begin{bmatrix} x=d.x & : & \textit{Ind} \\ y=d.y & : & \textit{Ind} \\ c_3 & : & \text{beat}(s.x, s.y) \end{bmatrix} \end{bmatrix}$$

Flattening

$$\begin{bmatrix} d.x & : & \textit{Ind} \\ d.c_1 & : & \text{man}(d.x) \\ d.y & : & \textit{Ind} \\ d.c_2 & : & \text{donkey}(d.y) \\ d.c_3 & : & \text{own}(d.x, d.y) \\ s.x=d.x & : & \textit{Ind} \\ s.y=d.y & : & \textit{Ind} \\ s.c_3 & : & \text{beat}(s.x, s.y) \end{bmatrix}$$

Relabelling

$$\begin{bmatrix} x & : & \textit{Ind} \\ c_1 & : & \text{man}(x) \\ y & : & \textit{Ind} \\ c_2 & : & \text{donkey}(y) \\ c_3 & : & \text{own}(x, y) \\ z=x & : & \textit{Ind} \\ w=y & : & \textit{Ind} \\ c_4 & : & \text{beat}(z, w) \end{bmatrix}$$

Manifest field elimination

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathrm{man(x)} \\ \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_2 & : & \mathrm{donkey(y)} \\ \mathrm{c}_3 & : & \mathrm{own(x,\ y)} \\ \mathrm{c}_4 & : & \mathrm{beat(x,\ y)} \end{bmatrix}$$

4 A treatment of generalised quantifiers

The natural language quantifiers we have introduced so far correspond to the existential and universal quantifiers of classical logic except that because of the dependent types of our type theory we have a dynamic version of them which allows for the treatment of donkey anaphora. The classical type theoretic treatment of existential quantification, relying on a type being instantiated ("true") just in case there is something of that type corresponds to the unselective binding associated with discourse representation structures. Similarly, the classical type theoretic treatment of universal quantification in terms of functions corresponds to the DRT use of implication corresponding to universal quantification.

However, if we wish to treat quantifiers other than those which can be defined in terms of existential and universal quantifiers we need to move to a generalised quantifier analysis (cf Barwise and Cooper, 1981, and a good deal of subsequent literature). This also corresponds to the addition of generalised quantifiers in DRT (Kamp and Reyle, 1993).

Our basic strategy for treating generalised quantifiers in type theory is to consider them as two place predicates whose arguments are functions corresponding to properties of individuals[4], that is, in our record theoretic terms, functions of type $(\left[\mathrm{x}{:}\mathit{Ind}\right])\mathit{RecType}$. Thus if q is such a predicate we can represent the type of proofs that "q women run" as

$$q(\lambda r : \left[\mathrm{x}{:}\mathit{Ind}\right](\left[\mathrm{c}{:}\mathrm{woman}(r.\mathrm{x})\right]),\ \lambda r : \left[\mathrm{x}{:}\mathit{Ind}\right]\ (\left[\mathrm{c}{:}\mathrm{run}(r.\mathrm{x})\right]))$$

[4] I am grateful to Aarne Ranta for pointing out to me that this is the correct strategy.

Suppose that q is such a predicate and f_1 and f_2 are functions that are appropriate arguments to it. What kind of object would be a proof of type $q(f_1, f_2)$?

Classical generalised quantifier theory (e.g. Barwise and Cooper, 1981) tells us that q corresponds to a relation between sets. We will call this set theoretic relation $q*$. We can obtain a set from a function $f : (\left[\text{x:}\mathit{Ind}\right])\mathit{RecType}$ as follows[5]:

$$\{a|\exists r : \left[\text{x:}\mathit{Ind}\right][f(r)\text{true} \wedge a = r.\text{x}]\}$$

We can now say that

$$p : q(f_1, f_2)$$

just in case

$$p =< \{a|\exists r : \left[\text{x:}\mathit{Ind}\right][f_1(r)\text{true} \wedge a = r.\text{x}]\}, \{a|\exists r : \left[\text{x:}\mathit{Ind}\right][f_2(r)\text{true} \wedge a = r.\text{x}]\} >$$
and $q*$ holds between p_1 and p_2.

For example, if q is a quantifier predicate 'most', corresponding to English *most*, then 'most*' may be defined as[6]:

most*(X, Y) iff $\frac{|X \cap Y|}{|X|} > \frac{1}{2}$

Classical quantifiers can also be defined as generalised quantifiers:

every*(X, Y) iff $X \subseteq Y$
a*(X, Y) iff $X \cap Y \neq \emptyset$

This gives us a treatment of classical non-dynamic generalised quantifiers. That is, it will allow us to treat examples like *most men own a donkey* but not *most men who own a donkey beat it* since there will be no way of making *it* depend on *a donkey*. In

[5]Following standard type theoretical notation we will write "T true" to mean "T is instantiated", i.e. there is some b such that $b : T$

[6]One can discuss whether this represents the actual meaning of *most*. I favour a more complex analysis where the ratio varies depending on context.

order to incorporate a treatment of this kind of donkey anaphora we will need to make our quantifier predicates 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.[7]

If q is a quantifier predicate then

1. arity(q) = $< (X \sqsubseteq \left[\text{x:}\mathit{Ind}\right))\mathit{RecType}, (X \sqsubseteq \left[\text{x:}\mathit{Ind}\right))\mathit{RecType} >$
2. q is associated with a relation between sets $q*$ such that

   $p : q(f_1 : (T_1)RecType, f_2 : (T_2)RecType)$
   iff
   $p =< \{a|\exists r : T_1[f_1(r)\text{true} \wedge a = r.\text{x}]\}, \{a|\exists r : T_2[f_2(r)\text{true} \wedge a = r.\text{x}]\} >$ 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 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

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

Then

[7]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.

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

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 $\mathcal{R}$[8].

Thus the first argument of the quantifier provides a *hypothetical context* for the second argument. That is, 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.

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

Thus a record type corresponding to the content of *most men who own a donkey beat it* could be as follows:

$$\begin{bmatrix} \text{restr}=\lambda r{:}\left[\text{x}{:}\textit{Ind}\right]\left(\begin{bmatrix} \text{c}_1 & : & \text{man}(r.\text{x}) \\ \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own}(r.\text{x,y}) \end{bmatrix}\right) & : & (\left[\text{x}{:}\textit{Ind}\right])\textit{RecType} \\ \text{scope}=\lambda r{:}\mathcal{F}(\text{restr})(\left[\text{c}_4{:}\text{beat}(r.\text{x},r.\text{y})\right]) & : & (\mathcal{F}(\text{restr}))\textit{RecType} \\ \text{c}_5 & : & \text{most(restr,scope)} \end{bmatrix}$$

This corresponds to a reading where only *most* is treated as a generalised quantifier. If we also treat *a* as a generalised quantifier we obtain the following result according to

[8]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.

compositional interpretation provided by our grammar defined below. This is before resolution of *it*.

$$\begin{bmatrix} \text{restr}=\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right] \\ \quad (\begin{bmatrix} \text{c}: \begin{bmatrix} \text{pred}: \left[\text{c:man}(r_1.\text{x})\right] \\ \text{mod}: \begin{bmatrix} \text{restr}=\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c:donkey}(r.\text{x})\right]){:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}=\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c:own}(r_1.\text{x},r.\text{x})\right]){:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType} \\ \text{c:a(c.mod.restr,c.mod.scope)} \end{bmatrix} \end{bmatrix} \end{bmatrix}){:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}=\lambda r{:}\mathcal{F}(\text{restr})(\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x=?} & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{beat}(r.\text{x, par.x}) \end{bmatrix} \end{bmatrix}{:}(\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c:most(restr,scope)} \end{bmatrix}$$

As we have now used a generalised quantifier for the indefinite article we cannot simply pick out a path in the hypothetical context as we did before when we chose r.y to correspond to *it*. We need a notion of witness for an existential generalised quantifier. We will define a notion of *witness type* for a record which is an existential generalised quantifier.[9]

If

$$r : \begin{bmatrix} \text{restr} & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope} & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c} & : & \text{a(restr,scope)} \end{bmatrix}$$

then the *witness type* for r, $\mathcal{W}(r)$, i.e. the type of all and only the witnesses for the quantifier, is

$\mathcal{F}(r.\text{scope})$

Thus if

$$r : \begin{bmatrix} \text{restr}=\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}_1\text{:donkey}(r_1.\text{x})\right]) & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}=\lambda r_2{:}\mathcal{F}(\text{restr})(\left[\text{c}_2\text{:work}(r_2.\text{x})\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c}_3 & : & \text{a(restr,scope)} \end{bmatrix}$$

then $\mathcal{W}(r)$ is

[9] Witnesses for other quantifiers will be defined rather differently and we will not concern ourselves with a more general definition here.

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathrm{donkey(x)} \\ \mathrm{c}_2 & : & \mathrm{work(x)} \end{bmatrix}$$

The duality of types as types of objects and proposition-like objects means that the witness type is in fact the non-generalised quantifier analysis of existentials. This means that the function $\mathcal{W}$ gives us a way of converting back to the non-generalised quantifier analysis and will as expected give us a way of resolving *it*. We need to introduce a witness of the existential quantifier. We can do this either in the domain of the function which is the scope of *most* (yielding what is known in the literature as a *strong* reading "most men beat all donkeys they own") or in the body of this function (yielding a *weak* reading "most men beat some donkey they own").[10] The two possibilities are:

$$\begin{bmatrix} \mathrm{restr}=\lambda r_1{:}\left[\mathrm{x}{:}\mathit{Ind}\right] \\ \quad (\begin{bmatrix} \mathrm{c}: \begin{bmatrix} \mathrm{pred}: \left[\mathrm{c{:}man}(r_1.\mathrm{x})\right] \\ \mathrm{mod}: \begin{bmatrix} \mathrm{restr}=\lambda r{:}\left[\mathrm{x}{:}\mathit{Ind}\right](\left[\mathrm{c{:}donkey}(r.\mathrm{x})\right]){:}(\left[\mathrm{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \mathrm{scope}=\lambda r{:}\mathcal{F}(\mathrm{c.mod.restr})(\left[\mathrm{c{:}own}(r_1.\mathrm{x},r.\mathrm{x})\right]){:}(\mathcal{F}(\mathrm{c.mod.restr}))\mathit{RecType} \\ \mathrm{c{:}a(c.mod.restr,c.mod.scope)} \end{bmatrix} \end{bmatrix} \end{bmatrix}){:}(\left[\mathrm{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \mathrm{scope}=\lambda r{:}\mathcal{F}(\mathrm{restr})\cup\left[\mathrm{w}{:}\mathcal{W}(\mathrm{c.mod})\right](\begin{bmatrix} \mathrm{par} & : & \begin{bmatrix} \mathrm{x=r.w.x} & : & \mathit{Ind} \end{bmatrix} \\ \mathrm{scope} & : & \begin{bmatrix} \mathrm{c} & : & \mathrm{beat}(r.\mathrm{x},\ \mathrm{par.x}) \end{bmatrix} \end{bmatrix}{:}(\mathcal{F}(\mathrm{restr}))\mathit{RecType} \\ \mathrm{c{:}most(restr,scope)} \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{restr}=\lambda r_1{:}\left[\mathrm{x}{:}\mathit{Ind}\right] \\ \quad (\begin{bmatrix} \mathrm{c}: \begin{bmatrix} \mathrm{pred}: \left[\mathrm{c{:}man}(r_1.\mathrm{x})\right] \\ \mathrm{mod}: \begin{bmatrix} \mathrm{restr}=\lambda r{:}\left[\mathrm{x}{:}\mathit{Ind}\right](\left[\mathrm{c{:}donkey}(r.\mathrm{x})\right]){:}(\left[\mathrm{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \mathrm{scope}=\lambda r{:}\mathcal{F}(\mathrm{c.mod.restr})(\left[\mathrm{c{:}own}(r_1.\mathrm{x},r.\mathrm{x})\right]){:}(\mathcal{F}(\mathrm{c.mod.restr}))\mathit{RecType} \\ \mathrm{c{:}a(c.mod.restr,c.mod.scope)} \end{bmatrix} \end{bmatrix} \end{bmatrix}){:}(\left[\mathrm{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \mathrm{scope}=\lambda r{:}\mathcal{F}(\mathrm{restr})(\begin{bmatrix} \mathrm{w} & : & \mathcal{W}(r.\mathrm{c.mod}) \\ \mathrm{par} & : & \begin{bmatrix} \mathrm{x=w.x} & : & \mathit{Ind} \end{bmatrix} \\ \mathrm{scope} & : & \begin{bmatrix} \mathrm{c} & : & \mathrm{beat}(r.\mathrm{x},\ \mathrm{par.x}) \end{bmatrix} \end{bmatrix}{:}(\mathcal{F}(\mathrm{restr}))\mathit{RecType} \\ \mathrm{c{:}most(restr,scope)} \end{bmatrix}$$

We can achieve this (and similar interpretations for *a* and *every*) in our compositional grammar by adding:

[10] For discussion of strong and weak readings see e.g Chierchia (1995) and literature cited there.

$$
\begin{aligned}
&[\![\ [_{\text{Det}}\ \text{most}]\ ]\!] = \text{most}' \\
&\text{most}' = \lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad\qquad\qquad (\begin{bmatrix} \text{restr}{=}R_1 & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(R_2\ @\ \left[\text{x}{=}r.\text{x}\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c} & : & \text{most(restr,scope)} \end{bmatrix})
\end{aligned}
$$

Similarly, *a′* and *every′* are redefined:

$$
\begin{aligned}
&\text{a}' = \lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad\qquad\qquad (\begin{bmatrix} \text{restr}{=}R_1 & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(R_2\ @\ \left[\text{x}{=}r.\text{x}\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c} & : & \text{a(restr,scope)} \end{bmatrix}) \\
&\text{every}' = \lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\
&\qquad\qquad\qquad\qquad (\begin{bmatrix} \text{restr}{=}R_1 & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(R_2\ @\ \left[\text{x}{=}r.\text{x}\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c} & : & \text{every(restr,scope)} \end{bmatrix})
\end{aligned}
$$

4.1 Sample derivation: *most men who own a donkey beat it*

who own a donkey

$$\lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\begin{array}{lcl}\text{c} & : & \left[\begin{array}{lcl}\text{pred} & : & R_2\ @\ r\\ \text{mod} & : & R_1\ @\ r\end{array}\right]\end{array}\right])$$

@

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]\ (\left[\begin{array}{lcl}\text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\begin{array}{lcl}\text{c} & : & \text{donkey}(r.\text{x})\end{array}\right]) & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(\left[\begin{array}{lcl}\text{c} & : & \text{own}(r_1.\text{x},r.\text{x})\end{array}\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType}\\ \text{c} & : & \text{a(restr,scope)}\end{array}\right])$$

=

$$\begin{array}{l}\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\\ \quad\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]\\ \quad(\left[\begin{array}{l}\text{c}{:}\left[\begin{array}{l}\text{pred}{:}R_2\ @\ r_1\\ \text{mod}{:}\left[\begin{array}{ll}\text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}\text{donkey}(r.\text{x})\right]) & :(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c}{:}\text{own}(r_1.\text{x},r.\text{x})\right]) & :(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType}\\ \text{c} & :\text{a(c.mod.restr,c.mod.scope)}\end{array}\right]\end{array}\right]\end{array}\right])\end{array}$$

men who own a donkey

$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]$$
$$\left(\left[\text{c}{:}\left[\begin{array}{ll}\text{pred}{:}R_2\ @\ r_1 & \\ \text{mod}{:}\left[\begin{array}{ll}\text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}\text{donkey}(r.\text{x})\right]) & {:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c}{:}\text{own}(r_1.\text{x},r.\text{x})\right]) & {:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType} \\ \text{c} & {:}\text{a(c.mod.restr,c.mod.scope)}\end{array}\right]\end{array}\right]\right])$$

@

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

=

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]$$
$$\left(\left[\text{c}{:}\left[\begin{array}{ll}\text{pred}{:}\left[\text{c}{:}\text{man}(r_1.\text{x})\right] & \\ \text{mod}{:}\left[\begin{array}{ll}\text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}\text{donkey}(r.\text{x})\right]) & {:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c}{:}\text{own}(r_1.\text{x},r.\text{x})\right]) & {:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType} \\ \text{c} & {:}\text{a(c.mod.restr,c.mod.scope)}\end{array}\right]\end{array}\right]\right])$$

most men who own a donkey

$$\lambda R_1{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}$$
$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}$$
$$(\begin{bmatrix} \text{restr}{=}R_1 & : & (\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(R_2\ @\ \left[\text{x}{=}r.\text{x}\right]) & : & (\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c} & : & \text{most}(\text{restr},\text{scope}) \end{bmatrix})$$

@

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]$$
$$(\begin{bmatrix} \text{c}{:} \begin{bmatrix} \text{pred}{:}\left[\text{c}{:}\text{man}(r_1.\text{x})\right] \\ \text{mod}{:} \begin{bmatrix} \text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}\text{donkey}(r.\text{x})\right]) & {:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c}{:}\text{own}(r_1.\text{x},r.\text{x})\right]) & {:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType} \\ \text{c} & {:}\text{a}(\text{c.mod.restr},\text{c.mod.scope}) \end{bmatrix} \end{bmatrix} \end{bmatrix})$$

=

$$\lambda R_2{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}$$
$$(\begin{bmatrix} \text{restr}{=}\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right] \\ (\begin{bmatrix} \text{c}{:} \begin{bmatrix} \text{pred}{:}\left[\text{c}{:}\text{man}(r_1.\text{x})\right] \\ \text{mod}{:} \begin{bmatrix} \text{restr}{=}\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right](\left[\text{c}{:}\text{donkey}(r.\text{x})\right]){:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\left[\text{c}{:}\text{own}(r_1.\text{x},r.\text{x})\right]){:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType} \\ \text{c:a}(\text{c.mod.restr},\text{c.mod.scope}) \end{bmatrix} \end{bmatrix} \end{bmatrix}){:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType} \\ \text{scope}{=}\lambda r{:}\mathcal{F}(\text{restr})(R_2\ @\ \left[\text{x}{=}r.\text{x}\right]){:}(\mathcal{F}(\text{restr}))\mathit{RecType} \\ \text{c:most}(\text{restr},\text{scope}) \end{bmatrix})$$

beat it

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

@

$$\lambda R{:}(\left[\text{x}{:}\mathit{Ind}\right])\mathit{RecType}\ \left(\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x=?} & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & R\ @\ \text{par} \end{bmatrix}\right)$$

=

$$\lambda r_1{:}\left[\text{x}{:}\mathit{Ind}\right]\ \left(\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x=?} & : & \mathit{Ind} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{beat}(r_1.\text{x},\ \text{par.x}) \end{bmatrix} \end{bmatrix}\right)$$

most men who own a donkey beat it

$$\lambda R_2{:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}$$
$$(\begin{bmatrix}\text{restr}=\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ (\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}\begin{bmatrix}\text{c:man}(r_1.\text{x})\end{bmatrix}\\ \text{mod:}\begin{bmatrix}\text{restr}=\lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:donkey}(r.\text{x})\end{bmatrix}){:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\\ \text{scope}=\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\begin{bmatrix}\text{c:own}(r_1.\text{x},r.\text{x})\end{bmatrix}){:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType}\\ \text{c:a(c.mod.restr,c.mod.scope)}\end{bmatrix}\end{bmatrix}\end{bmatrix}){:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\\ \text{scope}=\lambda r{:}\mathcal{F}(\text{restr})(R_2 \mathbin{@} \begin{bmatrix}\text{x}=r.\text{x}\end{bmatrix}){:}(\mathcal{F}(\text{restr}))\mathit{RecType}\\ \text{c:most(restr,scope)}\end{bmatrix}){:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\end{bmatrix}$$

@

$$\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x=?} & : & \mathit{Ind}\end{bmatrix}\\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{beat}(r_1.\text{x, par.x})\end{bmatrix}\end{bmatrix})$$

=

$$\begin{bmatrix}\text{restr}=\lambda r_1{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\\ (\begin{bmatrix}\text{c:}\begin{bmatrix}\text{pred:}\begin{bmatrix}\text{c:man}(r_1.\text{x})\end{bmatrix}\\ \text{mod:}\begin{bmatrix}\text{restr}=\lambda r{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{c:donkey}(r.\text{x})\end{bmatrix}){:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\\ \text{scope}=\lambda r{:}\mathcal{F}(\text{c.mod.restr})(\begin{bmatrix}\text{c:own}(r_1.\text{x},r.\text{x})\end{bmatrix}){:}(\mathcal{F}(\text{c.mod.restr}))\mathit{RecType}\\ \text{c:a(c.mod.restr,c.mod.scope)}\end{bmatrix}\end{bmatrix}\end{bmatrix}){:}(\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix})\mathit{RecType}\\ \text{scope}=\lambda r{:}\mathcal{F}(\text{restr})(\begin{bmatrix}\text{par} & : & \begin{bmatrix}\text{x=?} & : & \mathit{Ind}\end{bmatrix}\\ \text{scope} & : & \begin{bmatrix}\text{c} & : & \text{beat}(r.\text{x, par.x})\end{bmatrix}\end{bmatrix}{:}(\mathcal{F}(\text{restr}))\mathit{RecType}\\ \text{c:most(restr,scope)}\end{bmatrix}$$

Resolution

Resolution is carried out by adding a witness field and specifying the metavariable as given in the text above.

5 Conclusions

I believe that this treatment of dynamic quantifiers is equivalent to other treatments in the literature (for example, Kamp and Reyle, 1993, Chierchia, 1995). What makes this formulation attractive is that the use of fixed point types relates it to similar uses of fixed point types that we have discussed in Cooper (2003, in preparation) for the analysis of intentional identity, questions and information state updates in dialogue management. It seems to be pointing towards a general theory of hypothetical context in natural lan-

guage. We suspect also that the use of record types with separate fields for restriction and scope arguments for generalised quantifiers will give support for analyses of common noun phrase and verb-phrase anaphora and also facilitate representations which are underspecified with respect to quantifier scope. We leave the investigation of this to future research.

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

References

Barwise, Jon and Robin Cooper (1981) Generalized quantifiers and natural language, *Linguistics and Philosophy*, Vol. 4, pp. 159–219.

Beaver, David (2004) The Optimization of Discourse Anaphora, *Linguistics and Philosophy*, 27(1), pp. 3–56.

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 and Smith (1998).

Chierchia, Gennaro (1995) *Dynamics of Meaning: Anaphora, Presupposition, and the Theory of Grammar*, University of Chicago Press, Chicago.

Cooper, Robin (2003) Austinian truth, attitudes and type theory, paper presented at the workshop Barwise and Situation Semantics, Stanford, Cal., 26 June 2003, `http://www.ling.gu.se/~cooper/records/stanford-paper.ps`

Cooper, Robin (forthcoming) Records and record types in semantic theory, in *Journal of Logic and Computation*, `http://www.ling.gu.se/~cooper/records/london-paper.ps`

Cooper, Robin (in preparation) *Records and Dialogue*,
`http://www.ling.gu.se/~cooper/records/report.ps`

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

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

Sambin, Giovanni and Jan Smith, eds (1998) *Twenty-Five Years of Constructive Type Theory*, Oxford Logic Guides, 36, Oxford University 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.