

Records and record types in semantic theory

**Robin Cooper
Göteborg University**

Work on records in Göteborg

Incorporating records into Martin-Löf type theory.

Work in Göteborg: Betarte, Tasistro, Coquand

Currently involved in Göteborgs Records and Dialogue Semantics project:

Robin Cooper

Thierry Coquand

Staffan Larsson

Peter Ljunglöf

Bengt Nordström

Aarne Ranta

Ingredients from (Martin-Löf) type theory

- records and record types
- dependent types
- “propositions” as types (of proofs)
- types as objects
- functions (λ -calculus)
- dependent function types

Records and record types

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

the record:

$$\left[\begin{array}{l} l_1 = a_1 \\ l_2 = a_2 \\ \dots \\ l_n = a_n \\ \dots \end{array} \right]$$

is of type:

$$\left[\begin{array}{l} l_1 : T_1 \\ l_2 : T_2(l_1) \\ \dots \\ l_n : T_n(l_1, l_2, \dots, l_{n-1}) \end{array} \right]$$

a man owns a donkey

Record type:

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

Record:

$$\left[\begin{array}{l} x = a \\ c_1 = p_1 \\ y = b \\ c_2 = p_2 \\ c_3 = p_3 \end{array} \right]$$

where

a, b are of type *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 $\text{own}(a, b)$

a man owns a donkey

Record type:

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

- a record of this type may have additional fields
- the types $\text{man}(x)$, $\text{donkey}(y)$, $\text{own}(x,y)$ are dependent types of proofs

Records are recursive

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

r are $r.f$, $r.g.h$ and $r.f.f.f$ are *paths* in this record

Types as objects

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

Functions (λ -calculus)

donkey

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

a (indefinite article)

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

Dependent function types

every man owns a donkey

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

Four linguistic theories

- Montague semantics
 - dynamic binding
 - improved treatment of intensionality (including perception)
 - improved treatment of context dependence (resources)
- DRT
 - λ -DRT
 - improved treatment of intensionality (including perception)
 - improved treatment of context dependence (resources)
- situation semantics
 - compositional treatment
 - dynamic binding
 - type discipline, for better or worse ...
- HPSG
 - binding and functions
 - both types and objects
 - no need to “code” semantics

Main advantage: you can get aspects of all four theories going at the same time.

Montague semantics

Compositionality

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

Compositionality

Sample derivation: *every man owns a donkey*

a donkey

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

@

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

=

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

own a donkey

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

@

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

=

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

every man

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

@

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

=

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

every man owns a donkey

$\lambda R_2:([x:Ind])\text{RecType}$

$$\left[\text{f} : (r : \left[\begin{array}{l} \text{par} : [x : Ind] \\ \text{restr} : [c : \text{man}(\text{par}.x)] \end{array} \right]) R_2 @ r.\text{par} \right]$$

@

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

=

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

Flattening

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

Relabelling

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

r-suppression (syntactic sugar)

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

DRT

Dynamic binding

- “ λ -DRT” (\leftarrow Montague semantics)
- improved treatment of intensionality including perception (\leftarrow situation semantics)
- improved treatment of context dependence, resources (\leftarrow situation semantics)

every man who owns a donkey beats it

$$\left[f : (r : \left[\begin{array}{l} x : \mathit{Ind} \\ c_1 : \mathit{man}(x) \\ y : \mathit{Ind} \\ c_2 : \mathit{donkey}(y) \\ c_3 : \mathit{own}(x, y) \end{array} \right]) \left[c_4 : \mathit{beat}(r.x, r.y) \right] \right]$$

Two techniques exploited in compositional treatment of anaphora

- manifest fields (Coquand)
- metavariables (Göteborg work on proof editing)

Manifest fields

If $a : T$, then T_a is a *unique type*

$b : T_a$ iff $b = a$

A manifest field in a record type is one whose type is a unique type, e.g.

$[x : T_a]$

written for convenience as

$[x=a : T]$

Allows record types to be “progressively instantiated”.

We will allow dependent unique types, i.e. where a can be represented by a path in a record type.

Metavariables

We will use metavariables (anonymous variables) ‘?’ in manifest fields in order to treat anaphoric constructions.

Metavariables will be *resolved* to paths.

Ultimately we plan to use a variant of David Beaver’s OT version of centering theory for resolution.

We suspect that metavariables can be used for other kinds of under-specification e.g. quantifier scope and that we may be able to use OT here as well.

he/him/she/her/it

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

If $T = \left[\text{x=y} : \text{Ind} \right]$

then $\text{npr}T = \lambda R:([\text{x:Ind}])\text{RecType} \left[\begin{array}{l} \text{par} : T \\ \text{scope} : R @ \text{par} \end{array} \right]$

beats it

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

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

@

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

=

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

every man who owns a donkey beats it

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

Resolution

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

$r.\text{par.x}$

$r.\text{restr.c.mod.par.x}$

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

The first of these is ruled out by a grammatical constraint not yet included in the grammar (reflexivity). Therefore we choose the second.

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

A man owns a donkey. He beats it.

$$\left[\begin{array}{l} \text{d} : \left[\begin{array}{l} \text{x} : \textit{Ind} \\ \text{c}_1 : \text{man}(\text{d.x}) \\ \text{y} : \textit{Ind} \\ \text{c}_2 : \text{donkey}(\text{d.y}) \\ \text{c}_3 : \text{own}(\text{d.x}, \text{d.y}) \end{array} \right] \\ \text{s} : \left[\begin{array}{l} \text{x=?} : \textit{Ind} \\ \text{y=?} : \textit{Ind} \\ \text{c}_3 : \text{beat}(\text{s.x}, \text{s.y}) \end{array} \right] \end{array} \right]$$

Resolution

$$\left[\begin{array}{l} \text{d} : \left[\begin{array}{l} \text{x} : \textit{Ind} \\ \text{c}_1 : \text{man}(\text{d.x}) \\ \text{y} : \textit{Ind} \\ \text{c}_2 : \text{donkey}(\text{d.y}) \\ \text{c}_3 : \text{own}(\text{d.x}, \text{d.y}) \end{array} \right] \\ \text{s} : \left[\begin{array}{l} \text{x=d.x} : \textit{Ind} \\ \text{y=d.y} : \textit{Ind} \\ \text{c}_3 : \text{beat}(\text{s.x}, \text{s.y}) \end{array} \right] \end{array} \right]$$

Flattening

d.x	:	<i>Ind</i>
d.c ₁	:	man(d.x)
d.y	:	<i>Ind</i>
d.c ₂	:	donkey(d.y)
d.c ₃	:	own(d.x, d.y)
s.x=d.x	:	<i>Ind</i>
s.y=d.y	:	<i>Ind</i>
s.c ₃	:	beat(s.x, s.y)

Relabelling

x	:	<i>Ind</i>
c ₁	:	man(x)
y	:	<i>Ind</i>
c ₂	:	donkey(y)
c ₃	:	own(x, y)
z=x	:	<i>Ind</i>
w=y	:	<i>Ind</i>
c ₄	:	beat(z, w)

“Demanifestation”

$$\left[\begin{array}{l} x : \textit{Ind} \\ c_1 : \text{man}(x) \\ y : \textit{Ind} \\ c_2 : \text{donkey}(y) \\ c_3 : \text{own}(x, y) \\ c_4 : \text{beat}(x, y) \end{array} \right]$$

Situation semantics

Situations and Austinian propositions

Perception complements

Attitudes

Resources (resource situations)

- compositionality (← Montague semantics)
- dynamic semantics (← DRT)
- no problems with restrictions and parameters (← type theoretical apparatus)
- “relational theory of meaning” related to HPSG (← HPSG)

Austinian truth

Barwise and Perry (*Situations and Attitudes*) paraphrase Austin:

a statement is true when the actual situation to which it refers is of the type described by the statement.

Austin's original ('Truth', 1961)

A statement is said to be true when the historic state of affairs to which it is correlated by the demonstrative conventions (the ones to which it 'refers') is of a type with which the sentence used in making it is correlated by the descriptive conventions.

- Statements are true of something (part of the world) not true *simpliciter*
- Truth *simpliciter* can be derived: there is some part of the world of which the statement is true
- Statements are correlated with types

Martin-Löf Type Theory

- Objects are of types
- Propositions are regarded as types of proofs (“propositions as types” principle)
- Proofs are objects, e.g. the proofs of *there is a prime number between 212 and 222* are the prime numbers between 212 and 222

Proofs of non-mathematical propositions

Ranta in *Type-Theoretical Grammar* draws a parallel with Davidson's event-based approach:

A proof of

Amundsen flew over the North Pole

is

a flight by Amundsen over the North Pole

In terms of situation theory:

A proof of

Amundsen flew over the North Pole

is

a **situation** in which Amundsen flies over the North Pole

Infons (states of affairs, soas) – types of situations

TT	ST
types of proofs (propositions)	infons

Sentences of situation theory

$s \models \sigma$ (“situation s supports infon (soa) σ ” or “ s is of type σ ”)

Situation theoretic objects

$(s \models \sigma)$, an Austinian proposition, true just in case $s \models \sigma$

Judgements

Type theoretical judgements

$a : T$ (“object a is of type T ”)

Type theoretical objects

an object a , of type $T - a : T$

No objects corresponding to judgements as such

Judgements concerning proofs

$p : \text{see}(a, b)$

p is a proof that a sees b

context: $a:\text{Ind}, b:\text{Ind}$

A note on individuals

What is the type *Ind*?

Schemes of individuation

Situations as records

Records and record types give us the possibility of grouping together collections of infon-like objects.

$$\left[\begin{array}{l} \mathbf{s} : \left[\begin{array}{l} \mathbf{x} : \mathit{Ind} \\ \mathbf{y} : \mathit{Ind} \\ \mathbf{s}_1 : \mathit{see}(\mathbf{x},\mathbf{y}) \\ \mathbf{s}_2 : \mathit{see}(\mathbf{y},\mathbf{x}) \end{array} \right] \end{array} \right]$$

$$\left[\begin{array}{l} \mathbf{x} : \mathit{Ind} \\ \mathbf{y} : \mathit{Ind} \\ \mathbf{s} : \left[\begin{array}{l} \mathbf{s}_1 : \mathit{see}(\mathbf{x},\mathbf{y}) \\ \mathbf{s}_2 : \mathit{see}(\mathbf{y},\mathbf{x}) \end{array} \right] \end{array} \right]$$

Records and attitudes

A simple treatment of the attitudes

Record types as the object of the attitude (corresponding to a ST treatment where attitudes are relations between individuals and in-fons).

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

- The object of belief is Austinian.
- Austinian propositions could be represented as pairs of records and record types (coded as a record).

Perception attitudes

a man sees that a donkey kicked a farmer

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

Veridicality

$$\left[f : (r : \left[\begin{array}{l} x : Ind \\ T : \text{RecType} \\ c : \text{see}(x, T) \end{array} \right]) \left[a : r.T \right] \right]$$

Note that this does not require that the man saw a donkey-kicking-farmer situation – he may just have seen donkey hoof marks on the farmer’s legs.

Naked infinitive perception complements

a man sees a donkey kick a farmer

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

Veridicality is required.

Also perception of the donkey-kicking-farmer situation.

Types are not enough

- Proper names
- Presuppositions

Sandy kicks a farmer

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

Frames of mind

A *family of types* – function from records of a given type to a record type.

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

A *setting* for a frame of mind is a record in its domain.

The *presupposition* associated with a frame of mind is the type characterizing its domain.

A *mental state* is a pair consisting of a frame of mind (agent internal) and a setting (agent external).

The *content of a mental state* is the result of applying the frame of mind to the setting.

Advantage over Cooper and Ginzburg (1996) : no restricted objects

Pierre

Frame of mind

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

Setting

$$\left[\begin{array}{l} x = \text{london} \\ c_1 = \text{pf named}(\text{london} \text{"Londres"}) \\ y = \text{london} \\ c_2 = \text{pf named}(\text{london} \text{"London"}) \end{array} \right]$$

Content of Pierre's mental state

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

Records as resources

Domain restrictions

every/the man owns a donkey

Resource situations in situation semantics

“everything which is a man in situation s ”

“everything which is a man in record r ”

We use

$$r \models T$$

to represent the type of objects in record r which are of type T

$a : r \models T$ just in case for some $\ell, r : [\ell:T]$ and $r.\ell = a$

Anchored resources (specific)

$\lambda r_1 : Rec$

$\lambda r_2 : Rec$

$$\left[\begin{array}{l} \mathbf{f} : (r : \left[\begin{array}{l} \mathbf{x} : r_1 \models Ind \\ \mathbf{c}_1 : r_1 \models \text{man}(x) \end{array} \right]) \left[\begin{array}{l} \mathbf{y} : r_2 \models Ind \\ \mathbf{c}_2 : r_2 \models \text{donkey}(y) \\ \mathbf{c}_3 : \text{own}(r.x,y) \end{array} \right] \end{array} \right] \\ @ res_1 @ res_2$$

“Every man in res_1 owns a donkey in res_2 .”

Non-anchored resources (non-specific, existentially quantified)

$$\left[\begin{array}{l} res_1 : Rec \\ res_2 : Rec \\ \mathbf{f} : (r : \left[\begin{array}{l} \mathbf{x} : res_1 \models Ind \\ \mathbf{c}_1 : res_1 \models \text{man}(x) \end{array} \right]) \left[\begin{array}{l} \mathbf{y} : res_2 \models Ind \\ \mathbf{c}_2 : res_2 \models \text{donkey}(y) \\ \mathbf{c}_3 : \text{own}(r.x,y) \end{array} \right] \end{array} \right]$$

“There are resources res_1, res_2 such that every man in res_1 owns a donkey in res_2 .”

Quantification over resources (a kind of generic)

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

“For every resource res_1 and every man x in res_1 there is a resource res_2 such that x owns a donkey in res_2 .”

Records (situations) as modules

Types restricted to resources import information from the resource (cf importing from modules in programming languages).

If $r : [a : r_1 \models T]$
then $r : [a : T]$

r can be used as a resource for another record.

This kind of “information spreading” not present in situation theory.

HPSG

Feature structures: phonology, syntax, semantics (constraint based, relational)

- allows us to represent both types and objects (\leftarrow type theory apparatus)
- direct treatment of semantics, functions, binding (\leftarrow type theory apparatus)
- dynamic binding, discourse (\leftarrow DRT)
- compositionality with λ -calculus (\leftarrow Montague semantics)
- more faithful treatment of situation semantics (\leftarrow situation semantics)

Towards an HPSG style grammar

Our grammar will require a system of types with the following basic types: *Lex*, *Cat*, *Ind*.

We consider models where $A(Lex) = \{a, \text{every}, \text{man}, \text{donkey}, \text{owns}, \text{beats}, \text{who}, \text{he}, \text{him}, \text{she}, \text{her}, \text{it}\}$ and where $A(Cat) = \{D, S, NP, VP, V, Det, N, NBar, RelPro, Rel, Pron\}$

$Phon \equiv [Lex]$

$Sign \equiv DSign \vee SSign \vee NP\text{Sign} \vee VP\text{Sign} \vee V\text{Sign} \vee Det\text{Sign} \vee N\text{Sign} \vee RelPro\text{Sign}$

$$\begin{aligned}
DSign \equiv & \left[\begin{array}{l} \text{phon}=\text{daughters.first.phon} \quad : \quad Phon \\ \text{cat}=\text{D} \quad : \quad Cat \\ \text{daughters} \quad : \quad \left[\begin{array}{l} \text{first} \quad : \quad SSign \\ \text{rest}=\text{nil} \quad : \quad [Sign] \end{array} \right] \\ \text{content}=\text{daughters.first.content} \quad : \quad RecType \end{array} \right] \\
\vee & \left[\begin{array}{l} \text{phon}=\text{Append}(\text{daughters.first.phon}, \text{daughters.rest.first.phon}) \quad : \quad Phon \\ \text{cat}=\text{D} \quad : \quad Cat \\ \text{daughters} \quad : \quad \left[\begin{array}{l} \text{first} \quad : \quad DSign \\ \text{rest} \quad : \quad \left[\begin{array}{l} \text{first} \quad : \quad SSign \\ \text{rest}=\text{nil} \quad : \quad [Sign] \end{array} \right] \end{array} \right] \\ \text{content}=\left[\begin{array}{l} \text{d} \quad : \quad \text{daughters.first.content} \\ \text{s} \quad : \quad \text{daughters.rest.first.content} \end{array} \right] \quad : \quad RecType \end{array} \right]
\end{aligned}$$

DSign corresponds to the following two rules for interpreting labelled bracketings:

$$[[[D S]]] = [S]$$

$$[[[D D S]]] = \left[\begin{array}{l} \text{d} \quad : \quad [[D]] \\ \text{s} \quad : \quad [S] \end{array} \right]$$

SSign ≡

phon= <i>Append</i> (daughters.first.phon, daughters.rest.first.phon)	:	<i>Phon</i>																	
cat=S	:	<i>Cat</i>																	
daughters	:	<table style="border-collapse: collapse; margin-left: 10px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">first</td> <td style="padding-left: 5px;">:</td> <td><i>NPSign</i></td> <td style="border-right: 1px solid black;"></td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">rest</td> <td style="padding-left: 5px;">:</td> <td> <table style="border-collapse: collapse; margin-left: 10px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">first</td> <td style="padding-left: 5px;">:</td> <td><i>VPSign</i></td> <td style="border-right: 1px solid black;"></td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">rest=nil</td> <td style="padding-left: 5px;">:</td> <td>[<i>Sign</i>]</td> <td style="border-right: 1px solid black;"></td> </tr> </table> </td> <td style="border-right: 1px solid black;"></td> </tr> </table>	first	:	<i>NPSign</i>		rest	:	<table style="border-collapse: collapse; margin-left: 10px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">first</td> <td style="padding-left: 5px;">:</td> <td><i>VPSign</i></td> <td style="border-right: 1px solid black;"></td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">rest=nil</td> <td style="padding-left: 5px;">:</td> <td>[<i>Sign</i>]</td> <td style="border-right: 1px solid black;"></td> </tr> </table>	first	:	<i>VPSign</i>		rest=nil	:	[<i>Sign</i>]			
first	:	<i>NPSign</i>																	
rest	:	<table style="border-collapse: collapse; margin-left: 10px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">first</td> <td style="padding-left: 5px;">:</td> <td><i>VPSign</i></td> <td style="border-right: 1px solid black;"></td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">rest=nil</td> <td style="padding-left: 5px;">:</td> <td>[<i>Sign</i>]</td> <td style="border-right: 1px solid black;"></td> </tr> </table>	first	:	<i>VPSign</i>		rest=nil	:	[<i>Sign</i>]										
first	:	<i>VPSign</i>																	
rest=nil	:	[<i>Sign</i>]																	
content=daughters.first.content@daughters.rest.first.content	:	<i>RecType</i>																	

[[*S* NP VP]] = [NP] @ [VP]

Conclusions

Records in Martin-Löf (like) type theory

- allow us to combine various natural language “technologies”
 - Montague lambda technology
 - DRT, dynamic semantics
 - situation semantics
 - typed feature structures (HPSG)
- exploit key features of MLTT
 - dependent types
 - propositions as types
 - types as objects