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

http://www.ling.gu.se/~cooper/records
Ingredients from (Martin-Löf) type theory

- records and record types
- dependent types
- “propositions” as types (of proofs)
- types as objects
- functions ($\lambda$-calculus)
- dependent function types
Records and record types

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

\[
\begin{bmatrix}
  l_1 &= a_1 \\
  l_2 &= a_2 \\
  \vdots \\
  l_n &= a_n \\
  \vdots
\end{bmatrix}
\]

is of type:

\[
\begin{bmatrix}
  l_1 & : & T_1 \\
  l_2 & : & T_2(l_1) \\
  \vdots \\
  l_n & : & T_n(l_1, l_2, \ldots, l_{n-1})
\end{bmatrix}
\]
a man owns a donkey

Record type:

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

Record:

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

where

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

Record type:

\[
\begin{array}{l}
\text{x} : \text{Ind} \\
\text{c}_1 : \text{man}(x) \\
\text{y} : \text{Ind} \\
\text{c}_2 : \text{donkey}(y) \\
\text{c}_3 : \text{own}(x,y)
\end{array}
\]

- a record of this type may have additional fields
- the types man(x), donkey(y), own(x,y) are dependent types of proofs
Records are recursive

\[
\begin{align*}
  f &= \begin{bmatrix} 
  ff &= a \\
  gg &= b 
  \end{bmatrix} \\
  g &= c \\
  h &= \begin{bmatrix} 
  g &= a \\
  h &= d 
  \end{bmatrix}
\end{align*}
\]

\emph{r} are \emph{r.f, r.g.h} and \emph{r.f.f.f} are \emph{paths} in this record
Types as objects

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

donkey

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

$a$ (indefinite article)

$\lambda R_1 : ([x : Ind] \text{RecType}) \lambda R_2 : ([x : Ind] \text{RecType})$

\[
\begin{array}{l}
\text{par} : [x : \text{Ind}] \\
\text{restr} : R_1 @ \text{par} \\
\text{scope} : R_2 @ \text{par}
\end{array}
\]
Dependent function types

evory man owns a donkey

\[
\begin{array}{c}
f : (\begin{array}{c}
x : \text{Ind} \\
c_1 : \text{man}(x)
\end{array})
\begin{array}{c}
y : \text{Ind} \\
c_2 : \text{donkey}(y) \\
c_3 : \text{own}(x,y)
\end{array}
\end{array}
\]
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 (← DRT)
- improved treatment of intensionality including perception (← situation semantics)
- improved treatment of context dependence, resources (← situation semantics)
Compositionality

Sample derivation: *every man owns a donkey*

*a donkey*

\[
\lambda R_1:([x:Ind])\text{RecType} \lambda R_2:([x:Ind])\text{RecType}
\]

\[
\begin{array}{c}
\text{par} : [x : Ind] \\
\text{restr} : R_1 \otimes \text{par} \\
\text{scope} : R_2 \otimes \text{par}
\end{array}
\]

@

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

= 

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

\[
\begin{array}{c}
\text{par} : [x : Ind] \\
\text{restr} : [c : \text{donkey}(\text{par}.x)] \\
\text{scope} : R_2 \otimes \text{par}
\end{array}
\]
own a donkey

\[ \lambda N : (([x:Ind])RecType)RecType \]
\[ \lambda r_1 : [x:Ind] \ (N \ @ \ \lambda r_2 : [x:Ind] ([c:own(r_1.x, r_2.x)])) \]

@

\[ \lambda R_2 : ([x:Ind])RecType \]
\[ \begin{align*}
&\text{par} : \ [x : Ind] \\
&\text{restr} : \ [c : \text{donkey}(\text{par}.x)] \\
&\text{scope} : \ R_2 \ @ \ \text{par}
\end{align*} \]

= \\

\[ \lambda r_1 : [x:Ind] \ (\begin{align*}
&\text{par} : \ [x : Ind] \\
&\text{restr} : \ [c : \text{donkey}(\text{par}.x)] \\
&\text{scope} : \ [c : \text{own}(r_1.x, \text{par}.x)]
\end{align*}) \]
every man

\[ \lambda R_1 : \left( \left[ x : \text{Ind} \right] \right) \text{RecType} \]

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

\[ \left[ f : \begin{array}{ll}
  r : & \left[ \begin{array}{l}
    \text{par} : \left[ x : \text{Ind} \right] \\
    \text{restr} : \text{R}_1 \odot \text{par}
  \end{array} \right] \\
  \text{R}_2 \odot r.\text{par}
\end{array} \right] \]

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

\[ = \]

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

\[ \left[ f : \begin{array}{ll}
  r : & \left[ \begin{array}{l}
    \text{par} : \left[ x : \text{Ind} \right] \\
    \text{restr} : \left[ c : \text{man}(\text{par}.x) \right]
  \end{array} \right] \\
  \text{R}_2 \odot r.\text{par}
\end{array} \right] \]
every man owns a donkey

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

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

= 

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

\[
\begin{align*}
&f : (r : \begin{bmatrix}
  \text{par.x} : \text{Ind} \\
  \text{restr.c} : \text{man(par.x)}
\end{bmatrix}) \begin{bmatrix}
  \text{par.x} : \text{Ind} \\
  \text{restr.c} : \text{donkey(par.x)} \\
  \text{scope.c} : \text{own(r.par.x,par.x)}
\end{bmatrix} \\
&f : (r : \begin{bmatrix}
  \text{x} : \text{Ind} \\
  \text{c}_1 : \text{man(x)}
\end{bmatrix}) \begin{bmatrix}
  \text{y} : \text{Ind} \\
  \text{c}_2 : \text{donkey(y)} \\
  \text{c}_3 : \text{own(r.x,y)}
\end{bmatrix}
\end{align*}
\]

Relabelling

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

r-suppression (syntactic sugar)

\[
\begin{align*}
&f : (\begin{bmatrix}
  \text{x} : \text{Ind} \\
  \text{c}_1 : \text{man(x)}
\end{bmatrix}) \begin{bmatrix}
  \text{y} : \text{Ind} \\
  \text{c}_2 : \text{donkey(y)} \\
  \text{c}_3 : \text{own(x,y)}
\end{bmatrix}
\end{align*}
\]
DRT

Dynamic binding

- “λ-DRT” (← Montague semantics)
- improved treatment of intensionality including perception (← situation semantics)
- improved treatment of context dependence, resources (← situation semantess)
every man who owns a donkey beats it

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

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

Metavariabes 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 underspecification e.g. quantifier scope and that we may be able to use OT here as well.
npr \[ x=? : \text{Ind} \]

If \( T = \begin{bmatrix} x=y : \text{Ind} \end{bmatrix} \)

then \( \text{npr} T = \lambda R : (\begin{bmatrix} x : \text{Ind} \end{bmatrix}) \text{RecType} \begin{bmatrix} \text{par} : T \\ \text{scope} : R @ \text{par} \end{bmatrix} \)

beats it

\( \lambda N : ( (\begin{bmatrix} x : \text{Ind} \end{bmatrix}) \text{RecType}) \text{RecType} \)
\( \lambda r_1 : [x : \text{Ind}] (N @ \lambda r_2 : [x : \text{Ind}] ([c : \text{beat}(r_1.x, r_2.x)])) \)

\( \begin{bmatrix} \text{par} : x=? : \text{Ind} \\ \text{scope} : R @ \text{par} \end{bmatrix} \)

= 

\( \lambda r_1 : [x : \text{Ind}] (\begin{bmatrix} \text{par} : x=? : \text{Ind} \\ \text{scope} : c : \text{beat}(r_1.x, \text{par}.x) \end{bmatrix}) \)
every man who owns a donkey beats it
**Resolution**

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

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

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

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

\[
\begin{array}{l}
\left[ \begin{array}{l}
\text{par} : [ x : \text{Ind} ] \\
\end{array} \right]
\end{array}
\begin{array}{l}
\left[ \begin{array}{l}
pred : [ c : \text{man}(\text{par}.x) ] \\
\text{par} : [ x : \text{Ind} ] \\
\end{array} \right]
\end{array}
\begin{array}{l}
\left[ \begin{array}{l}
\text{mod} : [ \text{restr} : [ c : \text{donkey}(\text{restr}.c\cdot\text{mod}.\text{par}.x) ] ] \\
\text{scope} : [ c : \text{own}((\text{par}.x, \text{restr}.c\cdot\text{mod}.\text{par}.x)) ] \\
\end{array} \right]
\end{array}
\begin{array}{l}
\left[ \begin{array}{l}
\text{par} : [ x=\text{restr}.c\cdot\text{mod}.\text{par}.x : \text{Ind} ] \\
\text{scope} : [ c : \text{beat}(r \cdot \text{par}.x, \text{par}.x) ] \\
\end{array} \right]
\end{array}
\]
A man owns a donkey. He beats it.

| d  | x  : Ind            |
|    | c₁ : man(d.x)      |
|    | y  : Ind            |
|    | c₂ : donkey(d.y)    |
|    | c₃ : own(d.x, d.y)  |
|    | x=? : Ind           |

| s  | y=? : Ind           |
|    | c₃ : beat(s.x, s.y) |

Resolution

| d  | x  : Ind            |
|    | c₁ : man(d.x)      |
|    | y  : Ind            |
|    | c₂ : donkey(d.y)    |
|    | c₃ : own(d.x, d.y)  |
|    | x=d.x : Ind         |

| s  | y=d.y : Ind         |
|    | c₃ : beat(s.x, s.y) |
Flattening

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

Relabelling

\[
\begin{array}{c}
\text{x : } \text{Ind} \\
\text{c}_1 : \text{man(x)} \\
\text{y : } \text{Ind} \\
\text{c}_2 : \text{donkey(y)} \\
\text{c}_3 : \text{own(x, y)} \\
\text{z=x : } \text{Ind} \\
\text{w=y : } \text{Ind} \\
\text{c}_4 : \text{beat(z, w)} \\
\end{array}
\]
“Demanifestation”

\[
\begin{align*}
x & : \text{Ind} \\
c_1 & : \text{man}(x) \\
y & : \text{Ind} \\
c_2 & : \text{donkey}(y) \\
c_3 & : \text{own}(x, y) \\
c_4 & : \text{beat}(x, y)
\end{align*}
\]
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

\[ Amundsen \textit{flew over the North Pole} \]

is

a flight by Amundsen over the North Pole
In terms of situation theory:

A proof of

\[ \text{Amundsen flew over the North Pole} \]

is

a \textbf{situation} in which Amundsen flies over the North Pole
Infons (states of affairs, soas) – types of situations

<table>
<thead>
<tr>
<th>TT</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>types of proofs (propositions)</td>
<td>infons</td>
</tr>
</tbody>
</table>

Sentences of situation theory

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

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: \( a:Ind, b:Ind \)
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.

\[
\begin{bmatrix}
  x : Ind \\
  y : Ind \\
  s_1 : \text{see}(x, y) \\
  s_2 : \text{see}(y, x)
\end{bmatrix}
\]

\[
\begin{bmatrix}
  x : Ind \\
  y : Ind \\
  s : \begin{bmatrix}
    s_1 : \text{see}(x, y) \\
    s_2 : \text{see}(y, x)
  \end{bmatrix}
\end{bmatrix}
\]
A simple treatment of the attitudes

Record types as the object of the attitude.

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

The girl’s belief is \textit{true} of a record \textit{r} just in case \textit{r} is of type

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

(corresponding to a judgement in type theory or an Austinian proposition in situation semantics) and true simpliciter if there is such an \textit{r} (corresponding to a Russellian proposition in situation semantics).
A girl seeks a unicorn

\[
\begin{align*}
\begin{array}{c}
x \\
c_1 \\
p = \begin{bmatrix}
y & : & \text{Ind} \\
c_2 & : & \text{unicorn}(y) \\
c_4 & : & \text{seek}(x, p)
\end{bmatrix} \\
& : & \text{RecType}
\end{array}
\end{align*}
\]

Suppose that \( r \) is a record of this type. \( r' \) is a successful outcome for \( r \)'s search just in case \( r' \) is of type

\[
\begin{align*}
\begin{array}{c}
y & : & \text{Ind} \\
c_2 & : & \text{unicorn}(y) \\
c_3 & : & \text{find}(r \cdot x, y)
\end{bmatrix}
\end{align*}
\]

The girl’s search would be successful just in case there is such an \( r' \).
Perception attitudes

*a man sees that a donkey kicked a farmer*

\[
\begin{align*}
  & x : \text{Ind} \\
  c_1 & : \text{man}(x) \\
  & y : \text{Ind} \\
  c_2 & : \text{donkey}(y) \\
  p = & z : \text{Ind} \\
  c_3 & : \text{farmer}(z) \\
  c_4 & : \text{kick}(y,z) \\
  c_5 & : \text{see}(x,p)
\end{align*}
\]

**Veridicality**

\[
\begin{align*}
  & f : (r: T : \text{RecType})[ a : r.T ] \\
  & c : \text{see}(x,T)
\end{align*}
\]

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*

\[
\begin{align*}
\text{x} &: \text{Ind} \\
\text{c}_1 &: \text{man(x)} \\
\quad \text{y} &: \text{Ind} \\
\quad \text{c}_2 &: \text{donkey(y)} \\
\text{s} &: \text{z} &: \text{Ind} \\
\quad \text{c}_3 &: \text{farmer(z)} \\
\quad \text{c}_4 &: \text{kick(y, z)} \\
\text{c}_5 &: \text{see(x,s)}
\end{align*}
\]

Veridicality is required.

Also perception of the donkey-kicking-farmer situation.
Three apparent problems

- Does treating types as objects lead to paradoxes?
- Too finegrained - most (but not all) beliefs individuated by corresponding $\sigma$-types
- We need families of types rather than types as attitudinal objects
Types are not enough

- Proper names
- Presuppositions

*Sandy kicks a farmer*

\[
\begin{array}{ll}
y & : Ind \\
c_2 & : named(y, "Sandy") \\
z & : Ind \\
c_3 & : farmer(z) \\
c_4 & : kick(y, z) \\
\end{array}
\]
Frames of mind

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

\[ \lambda r : \begin{cases} y : \text{Ind} \\ c_2 : \text{named}(y, \text{“Sandy”}) \\ z : \text{Ind} \\ c_3 : \text{farmer}(z) \\ c_4 : \text{kick}(r.y, z) \end{cases} \]

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

\[ \lambda r : \begin{align*}
& x : \text{Ind} \\
& c_1 : \text{named}(x, \text{"Londres"}) \\
& y : \text{Ind} \\
& c_2 : \text{named}(y, \text{"London"}) \\
& \quad \begin{align*}
& c_3 : \text{pretty}(r.x) \\
& c_4 : \neg\text{pretty}(r.y)
\end{align*}
\end{align*} \]

Setting

\[ \begin{align*}
& x = \text{london} \\
& c_1 = \text{pf named(london "Londres")} \\
& y = \text{london} \\
& c_2 = \text{pf named(london "London")}
\end{align*} \]

Content of Pierre’s mental state

\[ \begin{align*}
& c_3 : \text{pretty(london)} \\
& c_4 : \neg\text{pretty(london)}
\end{align*} \]
Frames of mind

Double role as

• objects of attitudes
• information states
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 $T$

$$a : r \models T \text{ just in case for some } \ell, r : [\ell:T] \text{ and } r.\ell = a$$
Anchored resources (specific)

\[ \lambda r_1 : \text{Rec} \]
\[ \lambda r_2 : \text{Rec} \]
\[ f : (r : \begin{cases} x : r_1 \models \text{Ind} \\ c_1 : r_1 \models \text{man}(x) \end{cases} \) \]
\[ \begin{cases} y : r_2 \models \text{Ind} \\ c_2 : r_2 \models \text{donkey}(y) \\ c_3 : \text{own}(r.x,y) \end{cases} \]
\[ \text{at } @ \text{res}_1 @ \text{res}_2 \]

"Every man in res_1 owns a donkey in res_2."

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

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

"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)

\[
\begin{array}{c}
\text{res}_1 : \text{Rec} \\
\text{x} : \text{res}_1 \models \text{Ind} \\
\text{c}_1 : \text{res}_1 \models \text{man}(x)
\end{array}
\quad
\begin{array}{c}
\text{res}_2 : \text{Rec} \\
\text{y} : \text{res}_2 \models \text{Ind} \\
\text{c}_2 : \text{res}_2 \models \text{donkey}(y) \\
\text{c}_3 : \text{own}(r.x,y)
\end{array}
\]

“For every resource \( \text{res}_1 \) and every man \( x \) in \( \text{res}_1 \) there is a resource \( \text{res}_2 \) such that \( x \) owns a donkey in \( \text{res}_2 \).”
Records (situations) as modules

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

If $r : \begin{bmatrix} a : r_1 \models T \end{bmatrix}$
then $r : \begin{bmatrix} a : T \end{bmatrix}$

$r$ 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 (← type theory apparatus)
- direct treatment of semantics, functions, binding (← type theory apparatus)
- dynamic binding, discourse (← DRT)
- compositionality with $\lambda$-calculus (← Montague semantics)
- more faithful treatment of situation semantics (← situation semantics)
Towards an HPSG style grammar

Our grammar will require a system of types with the following basic types: \textit{Lex, Cat, Ind.}

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

\( \text{Phon} \equiv [\text{Lex}] \)

\( \text{Sign} \equiv D\text{Sign} \lor S\text{Sign} \lor N\text{PSign} \lor V\text{PSign} \lor V\text{Sign} \lor D\text{etSign} \lor N\text{Sign} \lor R\text{elProSign} \lor R\text{elSign} \lor P\text{ronSign} \)
\[ DSgn \equiv \]
\[
\begin{aligned}
&\text{phon=daughters.first.phon} : \text{Phon} \\
&\text{cat=D} : \text{Cat} \\
&\text{daughters} : \begin{cases}
&\text{first} : SSign \\
&\text{rest}=\text{nil} : [\text{Sign}] 
\end{cases} \\
&\text{content=daughters.first.content} : \text{RecType} \\
&\text{phon=\text{Append}(daughters.first.phon, daughters.rest.first.phon)} : \text{Phon} \\
&\text{cat=D} : \begin{cases}
&\text{first} : DSgn \\
&\text{rest} : \begin{cases}
&\text{first} : SSign \\
&\text{rest}=\text{nil} : [\text{Sign}] 
\end{cases}
\end{cases} \\
&\text{content=} \begin{cases}
&d : \text{daughters.first.content} \\
&s : \text{daughters.rest.first.content}
\end{cases} : \text{RecType}
\end{aligned}
\]

\( DSgn \) corresponds to the following two rules for interpreting labelled bracketings:

\[
\begin{bmatrix}
[D & S]
\end{bmatrix} = \begin{bmatrix}
S
\end{bmatrix}
\]

\[
\begin{bmatrix}
[D & D & S]
\end{bmatrix} = \begin{bmatrix}
d : [D] \\
S
\end{bmatrix}
\]
\[ S Sign \equiv \begin{cases} 
\text{phon}=\text{Append}(\text{daughters.first.phon, daughters.rest.first.phon}) & : \text{Phon} \\
\text{cat}=S & : \text{Cat} \\
\text{daughters} & : \begin{cases} 
\text{first} : \text{NPSign} \\
\text{rest} : \begin{cases} 
\text{first} : \text{VPSign} \\
\text{rest}=\text{nil} : [\text{Sign}] 
\end{cases} 
\end{cases} \\
\text{content}=\text{daughters.first.content}@\text{daughters.rest.first.content} & : \text{RecType} 
\end{cases} 
\]

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