Records and record types in semantic theory

Robin Cooper

Gteborg University

The Department of Linguistics

Contents

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`

References:
`http://www.ling.gu.se/~cooper/records`

Cooper, Robin, Austinian truth, attitudes and type theory, to appear in *Research on Language and Computation*

Cooper, Robin, Records and record types in semantic theory, *Journal of Logic and Computation*, Vol. 15 No. 2, pp. 99–112, 2005.

Cooper, Robin, A type theoretic approach to information state update in issue based dialogue management

Cooper, Robin, Dynamic generalised quantifiers and hypothetical contexts, in *Ursus Philosophicus, a festschrift for Björn Haglund*, Department of Philosophy, Göteborg University, 2004

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

1. 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:

$$\left[\begin{array}{lcl} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \ldots & & \\ l_n & = & a_n \\ \ldots & & \end{array}\right]$$

is of type:

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

a man owns a donkey

Record type:

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

Record:

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

where

a, b are of type *Ind*, individuals

p_1 is a proof of man(a)

p_2 is a proof of donkey(b)

p_3 is a proof of own(a, b)

⇐ contents

a man owns a donkey

Record type:

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

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

Records are recursive

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

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

Types as objects

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

Functions (λ-calculus)

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

a (indefinite article)

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

Dependent function types

every man owns a donkey

$$\left[\begin{array}{lcl} \mathrm{f} & : & \left(\left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathrm{man(x)} \end{array}\right]\right) \left[\begin{array}{lcl} \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_2 & : & \mathrm{donkey(y)} \\ \mathrm{c}_3 & : & \mathrm{own(x,y)} \end{array}\right] \end{array}\right]$$

Defining the type theory

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

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

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.

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 $a : T_2$.

   Similarly for $\wedge$.

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 $a : T_{o(r.\pi_1,\ldots,r.\pi_m)}$.

2. Four linguistic theories and issue based dialogue management

- Montague semantics
- DRT
- situation semantics
- HPSG
- issue based dialogue management

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

2.1. 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

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

@

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

=

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

own a donkey

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

λr_1:$\left[\text{x:}\mathit{Ind}\right]$ ($\mathcal{N}$ @ λr_2:$\left[\text{x:}\mathit{Ind}\right]$($\left[\text{c:own}(r_1\text{.x}, r_2\text{.x})\right]$))

@

λR_2:($\left[\text{x:}\mathit{Ind}\right]$)RecType $\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{donkey(par.x)} \end{bmatrix} \\ \text{scope} & : & R_2 \text{ @ par} \end{bmatrix}$

=

λr_1:$\left[\text{x:}\mathit{Ind}\right]$ ($\begin{bmatrix} \text{par} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix} \\ \text{restr} & : & \begin{bmatrix} \text{c} & : & \text{donkey(par.x)} \end{bmatrix} \\ \text{scope} & : & \begin{bmatrix} \text{c} & : & \text{own}(r_1\text{.x,par.x}) \end{bmatrix} \end{bmatrix}$)

every man

λR_1:(⌈x:*Ind*⌉)RecType

λR_2:(⌈x:*Ind*⌉)RecType

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

@

λr:⌈x:*Ind*⌉(⌈c:man(r.x)⌉)

=

λR_2:(⌈x:*Ind*⌉)RecType

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

every man owns a donkey

λR_2:($\left[\text{x:}\mathit{Ind}\right]$)RecType

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

@

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

=

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

Flattening

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

Relabelling

$$\left[\begin{array}{lcl} \text{f} & : & (r : \left[\begin{array}{lcl} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \end{array}\right]) \left[\begin{array}{lcl} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own}(r\text{.x,y}) \end{array}\right] \end{array}\right]$$

r-suppression (syntactic sugar)

$$\left[\begin{array}{lcl} \text{f} & : & (\left[\begin{array}{lcl} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \end{array}\right]) \left[\begin{array}{lcl} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{c}_3 & : & \text{own(x,y)} \end{array}\right] \end{array}\right]$$

2.2. 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 semanticss)

every man who owns a donkey beats it

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

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 *singleton type*
$b : T_a$ iff $b = a$
A manifest field in a record type is one whose type is a singleton type, e.g.
$\left[\ x\ :\ T_a\ \right]$
written for convenience as
$\left[\ x{=}a\ :\ T\ \right]$
Allows record types to be “progressively instantiated”.
We will allow dependent singleton 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 underspecification e.g. quantifier scope and that we may be able to use OT here as well.

he/him/she/her/it

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

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

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

beats it

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

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

@

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

=

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

every man who owns a donkey beats it

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

Resolution

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

r.par.x
r.restr.c.mod.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 of these is ruled out by a grammatical constraint not yet included in the grammar (reflexivity). Therefore we choose the second.

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

2.3. Situation semantics

- 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

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

Situations as records

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

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

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

A simple treatment of the attitudes

Record types as the object of the attitude.

$$\begin{bmatrix} x & : & \textit{Ind} \\ c_1 & : & \text{girl}(x) \\ p= \begin{bmatrix} y & : & \textit{Ind} \\ c_3 & : & \text{man}(y) \\ z & : & \textit{Ind} \\ c_4 & : & \text{donkey}(z) \\ c_5 & : & \text{own}(y, z) \end{bmatrix} & : & \textit{RecType} \\ c_2 & : & \text{believe}(x,p) \end{bmatrix}$$

The girl's belief is *true* of a record r just in case r is of type

$$\begin{bmatrix} y & : & \textit{Ind} \\ c_3 & : & \text{man}(y) \\ z & : & \textit{Ind} \\ c_4 & : & \text{donkey}(z) \\ c_5 & : & \text{own}(y, z) \end{bmatrix}$$

(corresponding to a judgement in type theory or an Austinian proposition in situation semantics) and true simpliciter if there is such an r (corresponding to a Russellian proposition in situation semantics).

A girl seeks a unicorn

$$\left[\begin{array}{lcl} \mathrm{x} & : & \textit{Ind} \\ \mathrm{c}_1 & : & \mathrm{girl(x)} \\ \mathrm{p}=\left[\begin{array}{lcl} \mathrm{y} & : & \textit{Ind} \\ \mathrm{c}_2 & : & \mathrm{unicorn(y)} \end{array}\right] & : & \textit{RecType} \\ \mathrm{c}_4 & : & \mathrm{seek(x,p)} \end{array}\right]$$

Suppose that r is a record of this type. r' is a *successful outcome* for r.x's search just in case r' is of type

$$\left[\begin{array}{lcl} \mathrm{y} & : & \textit{Ind} \\ \mathrm{c}_2 & : & \mathrm{unicorn(y)} \\ \mathrm{c}_3 & : & \mathrm{find}(r.\mathrm{x,y}) \end{array}\right]$$

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

$$\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{man(x)} \\ \text{p=} \begin{bmatrix} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{donkey(y)} \\ \text{z} & : & \textit{Ind} \\ \text{c}_3 & : & \text{farmer(z)} \\ \text{c}_4 & : & \text{kick(y, z)} \end{bmatrix} & : & \textit{RecType} \\ \text{c}_5 & : & \text{see(x,p)} \end{bmatrix}$$

Veridicality

$$\begin{bmatrix} \text{f} & : & (r{:} \begin{bmatrix} x & : & \textit{Ind} \\ T & : & \textit{RecType} \\ c & : & \text{see}(x, T) \end{bmatrix}) \begin{bmatrix} a & : & r.T \end{bmatrix} \end{bmatrix}$$

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.

⇐ contents

Naked infinitive perception complements

a man sees a donkey kick a farmer

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

Veridicality is required.

Also perception of the donkey-kicking-farmer situation.

Types are not enough

- Proper names
- Presuppositions

Sandy kicks a farmer

$$\begin{bmatrix} y & : & \textit{Ind} \\ c_2 & : & \text{named(y, “Sandy”)} \\ z & : & \textit{Ind} \\ c_3 & : & \text{farmer(z)} \\ c_4 & : & \text{kick(y, z)} \end{bmatrix}$$

Meanings (Frames of mind)

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

$$\lambda r : \begin{bmatrix} \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{named(y, “Sandy”)} \end{bmatrix} \begin{bmatrix} \text{z} & : & \textit{Ind} \\ \text{c}_3 & : & \text{farmer(z)} \\ \text{c}_4 & : & \text{kick}(r.\text{y}, \text{z}) \end{bmatrix}$$

A (static) meaning is a function from contexts (records, situations) to contents (record types). cf Montague/Kaplan meanings/characters.

Pierre

Frame of mind

$$\lambda r : \left[\begin{array}{lcl} \text{x} & : & \textit{Ind} \\ \text{c}_1 & : & \text{named(x, “Londres”)} \\ \text{y} & : & \textit{Ind} \\ \text{c}_2 & : & \text{named(y, “London”)} \end{array}\right] \left[\begin{array}{lcl} \text{c}_3 & : & \text{pretty}(r.\text{x}) \\ \text{c}_4 & : & \neg\text{pretty}(r.\text{y}) \end{array}\right]$$

Setting/Context

$$\left[\begin{array}{lcl} \text{x} & = & \text{london} \\ \text{c}_1 & = & \text{pf named(london “Londres”)} \\ \text{y} & = & \text{london} \\ \text{c}_2 & = & \text{pf named(london “London”)} \end{array}\right]$$

Content of Pierre's mental state

$$\left[\begin{array}{lcl} \text{c}_3 & : & \text{pretty(london)} \\ \text{c}_4 & : & \neg\text{pretty(london)} \end{array}\right]$$

⇐ contents

2.4. 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 λ-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: *Lex*, *Cat*, *Ind*.
We consider models where $A(Lex) = \{$a, every, man, donkey, owns, beats, who, he, him, she, her, it$\}$ and where $A(Cat) = \{$D, S, NP, VP, V, Det, N, NBar, RelPro, Rel, Pron$\}$
$Phon \equiv [Lex]$

$$Sign \equiv DSign \vee SSign \vee NPSign \vee VPSign \vee VSign \vee DetSign \vee NSign \vee RelProSign \vee RelSign \vee PronSign$$

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

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

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

$$[\![\ [_{\text{D}}\ \text{D S}]\ ]\!] = \left[\begin{array}{lcl} \text{d} & : & [\![\ \text{D}\ ]\!] \\ \text{s} & : & [\![\ \text{S}\ ]\!] \end{array}\right]$$

SSign ≡

$$\left[\begin{array}{lcl}
\text{phon=}\textit{Append}\text{(daughters.first.phon, daughters.rest.first.phon)} & : & \textit{Phon} \\
\text{cat=S} & : & \textit{Cat} \\
\text{daughters} & : & \left[\begin{array}{lcl} \text{first} & : & \textit{NPSign} \\ \text{rest} & : & \left[\begin{array}{lcl} \text{first} & : & \textit{VPSign} \\ \text{rest=nil} & : & [\textit{Sign}] \end{array}\right] \end{array}\right] \\
\text{content=daughters.first.content@daughters.rest.first.content} & : & \textit{RecType}
\end{array}\right]$$

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

2.5. Issue-based dialogue management

- a declarative theoretical treatment of the information state update approach to dialogue
- integration with semantics (intensionality, anaphora, . . .)

An example: Cross-speaker intensional identity

A: Sam says there's a bug in the program.
B: Pat is looking for it

Information state update

- Complex structure information states
  - gameboard
  - records
- non-monotonic updates

  e.g. questions disappear from QUD (Questions under discussion)

cf. classical dynamic or update semantics

Issue based dialogue management

Staffan Larsson (2002) *Issue-based Dialogue Management*, Ph.D. diss., Department of Linguistics, Göteborg.

- Determination of the next dialogue contribution driven largely by QUD

  almost all contributions raise or address an issue (question under discussion)

- Issues can be raised by addressing them

  giving an answer to a question that hasn't be explicitly stated (question accommodation)

A simple information state, r

Larsson, p. 56 (adapted)

$$\left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{agenda} & = & [] \\ & & [\text{raise}(?A.\text{how}(A)), \\ & & \text{findout}(?B.\text{dest\_city}(B)), \\ & & \text{findout}(?C.\text{dep\_city}(C)), \\ \text{plan} & = & \text{findout}(?D.\text{month}(D)), \\ & & \text{findout}(?E.\text{dep\_date}(E)), \\ & & \text{findout}(?F.\text{class}(F)), \\ & & \text{consultDB}(?G.\text{price}(G))] \\ \text{bel} & = & [] \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{com} & = & [] \\ \text{qud} & = & [?H.\text{price}(H)] \\ \text{lu} & = & \left[\begin{array}{lll} \text{speaker} & = & \text{usr} \\ \text{moves} & = & \{\text{ask}(?H.\text{price}(H))\} \end{array}\right] \end{array}\right] \end{array}\right]$$

⇐ contents

r is of type:

Adapted from Larsson, p. 32

$$\left[\begin{array}{lll} \text{private} & : & \left[\begin{array}{lll} \text{agenda} & : & [\mathit{Action}] \\ \text{plan} & : & [\mathit{Action}] \\ \text{bel} & : & \mathit{RecType} \end{array}\right] \\ \text{shared} & : & \left[\begin{array}{lll} \text{com} & : & \mathit{RecType} \\ \text{qud} & : & [\mathit{Question}] \\ \text{lu} & : & \left[\begin{array}{lll} \text{speaker} & : & \mathit{Participant} \\ \text{moves} & : & \{\mathit{Move}\} \end{array}\right] \end{array}\right] \end{array}\right]$$

We call this type *IS*, the type of information states.

r is *also* of type:

A *singleton type* using *manifest fields*

$$\left[\begin{array}{lll}
\text{private} : & \left[\begin{array}{lll}
\text{agenda=[]} & : & [\mathit{Action}] \\
\text{plan=} \begin{array}{l}[\text{raise}(?A.\text{how}(A)), \\ \text{findout}(?B.\text{dest_city}(B)), \\ \text{findout}(?C.\text{dep_city}(C)), \\ \text{findout}(?D.\text{month}(D), \\ \text{findout}(?E.\text{dep_date}(E), \\ \text{findout}(?F.\text{class}(F)), \\ \text{consultDB}(?G.\text{price}(G))]\end{array} & : & [\mathit{Action}] \\
\text{bel=[]} & : & \mathit{RecType}
\end{array}\right] \\
\text{shared} : & \left[\begin{array}{lll}
\text{com=[]} & : & \mathit{RecType} \\
\text{qud=}[?H.\text{price}(H)] & : & [\mathit{Question}] \\
\text{lu} & : & \left[\begin{array}{lll}
\text{speaker=usr} & : & \mathit{Participant} \\
\text{moves=}\{\text{ask}(?H.\text{price}(H))\} & : & \{\mathit{Move}\}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Another type for r:

Not a singleton type, but some fields are manifest

$$\left[\begin{array}{lll}
\text{private} & : & \left[\begin{array}{lll}
\text{agenda=[]} & : & [\textit{Action}] \\
\text{plan=} \begin{array}{l}[\text{raise}(?A.\text{how}(A)), \\ \text{findout}(?B.\text{dest_city}(B)), \\ \text{findout}(?C.\text{dep_city}(C)),\end{array} & : & [\textit{Action}] \\
\text{bel} & : & \textit{RecType}
\end{array}\right] \\
\text{shared} & : & \left[\begin{array}{lll}
\text{com} & : & \textit{RecType} \\
\text{qud=}[?H.\text{price}(H)] & : & [\textit{Question}] \\
\text{lu} & : & \left[\begin{array}{lll}
\text{speaker} & : & \textit{Participant} \\
\text{moves=}\{\text{ask}(?H.\text{price}(H))\} & : & \{\textit{Move}\}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Typing as underspecification

- there is typically more than one object of each type
- singleton types correspond to total specification
- non-deterministic updates can be expressed in terms of types

Reasoning about updates

Updates
$\text{record}_1 \Rightarrow \text{record}_2 \Rightarrow \text{record}_3$

Reasoning about updates
$\text{record type}_1 \Rightarrow \text{record type}_2 \Rightarrow \text{record type}_3$

record $\rightsquigarrow$ information state
record type $\rightsquigarrow$ information about an information state

Update rules in type theory

The basic intuition

If $r_i : T_i$, then $r_{i+1} : T_{i+1}(r_i)$
Expressed as a function from records to record types
i.e., a function of type (T_i)*RecType*

$\lambda r : T_i(T_{i+1}(r_i))$

Important: these are the same kinds of functions as our (static) meaning functions.
The discourse so far provides the context for the current utterance.

An update rule specification

integrateUsrAsk
class: integrate
Larsson, p. 40

$$\lambda r : \left[\begin{array}{ll} \text{tis} : \left[\text{is:}\left[\begin{array}{ll} \text{private} : \left[\text{agenda} : [\textit{Action}]\right] \\ \text{shared} : \left[\begin{array}{ll} \text{lu} : \left[\begin{array}{ll} \text{speaker=usr} : \textit{DP} \\ \text{moves} : \{\textit{Move}\} \end{array}\right] \\ \text{qud} : [\textit{Question}] \end{array}\right] \end{array}\right]\right] \\ \text{cond:}\left[\begin{array}{ll} \text{q} : \textit{Question} \\ \text{c} : \text{member(ask(cond.q), tis.is.shared.lu.moves)} \end{array}\right] \end{array}\right]$$

$$\left(\left[\text{tis:}\left[\text{is:}\left[\begin{array}{ll} \text{private:}\left[\text{agenda=respond}(r.\text{cond.q}) \middle| r.\text{tis.is.private.agenda} : [\textit{Action}]\right] \\ \text{shared:}\left[\text{qud=}r.\text{cond.q} \middle| r.\text{tis.is.shared.qud} : [\textit{Question}]\right] \end{array}\right]\right]\right]\right)$$

⇐ contents

Suppose current info state is of type

$$\left[\text{tis}:\left[\begin{array}{l}\text{is}\ :\left[\begin{array}{l}\text{private}:\left[\begin{array}{ll}\text{agenda}=[]\text{:}[\textit{Action}] & \\ \text{plan}=[] & :[\textit{Action}] \\ \text{bel}=[] & :\textit{RecType}\end{array}\right] \\ \text{shared}:\left[\begin{array}{l}\text{com}=[]:\textit{RecType} \\ \text{qud}=[]:[\textit{Question}] \\ \text{lu}:\left[\begin{array}{ll}\text{speaker}=\text{usr} & :\textit{DP} \\ \text{moves}=\{\text{ask}(?A.\text{time\_next\_train}(\text{Bclna},\ A))\} & :\{\textit{Move}\}\end{array}\right]\end{array}\right]\end{array}\right] \\ \ldots\end{array}\right]\right]$$

then new info state must be of type

$$\left[\begin{array}{ll}\text{tis:} & \left[\begin{array}{ll}\text{is} : & \left[\begin{array}{ll}\text{private:} & \left[\begin{array}{l}\text{agenda=[respond(?}A\text{.time\_next\_train(Bclna, }A\text{))]:[}\textit{Action}\text{]} \\ \text{plan=[]:[}\textit{Action}\text{]} \\ \text{bel=[]:}\textit{RecType}\end{array}\right] \\ \text{shared:} & \left[\begin{array}{l}\text{com=[]:}\textit{RecType} \\ \text{qud=[?}A\text{.time\_next\_train(Bclna, }A\text{)]:[}\textit{Question}\text{]} \\ \text{lu:}\left[\begin{array}{l}\text{speaker=usr:}\textit{DP} \\ \text{moves=}\{\text{ask(?}A\text{.time\_next\_train(Bclna, }A\text{))}\}:\{\textit{Move}\}\end{array}\right]\end{array}\right]\end{array}\right] \\ \ldots & \end{array}\right]$$

We know more about the type than the function expresses.

How have we used the update rule to draw this conclusion?

1. instantiation of the update function with the type of the current info state
2. relabelling of shared paths
3. compute fixed point type of resulting function
4. garbage collection of unused “support” fields

Instantiation of update function

Suppose that the update rule has the function $\lambda r : T_1(T_2(r))$
and that what we know about the information state is that it is of type T_{curr}

The rule only fires if $T_{curr} \sqsubseteq T_1$

We can *instantiate* the rule to $\lambda r : T_{curr}(T_2(r))$

Relabelling of shared paths

For the next step we will need to ensure that T_{curr} and $T_2(r)$ do not have any paths in common
which may have different values (see integrateUsrAsk)

We therefore prefix any shared path in T_{curr} with the label ‘prev(ious)’

Fixed point types of functions

If $f : (T)Type$ then T' *is a fixed point type* for f just in case $a : T'$ implies $a : f(a)$

$\left[\begin{array}{lcl} \text{x} & : & T_1 \\ \text{y} & : & T_2 \end{array}\right]$ is a fixed point type for $\lambda r : \left[\text{x}{:}T_1\right](\left[\text{y}{:}T_2\right])$

An example

Stripped down to fit on slides

$$\lambda r : \left[\begin{array}{ll} \text{prev} : \left[\text{tis:}\left[\text{is:}\left[\begin{array}{lll}\text{private} & : & [\textit{Action}]\\ \text{shared} & : & [\textit{Question}]\end{array}\right]\right]\right] & \\ \text{tis} : \left[\text{is:}\left[\text{shared:}\left[\text{lu} : \left[\begin{array}{lll}\text{speaker=usr} & : & \textit{DP}\\ \text{moves=}\{\text{ask}(?A.\text{time\_next\_train}(\text{Bclna},A))\} & : & \{\textit{Move}\}\end{array}\right]\right]\right]\right] & \\ \text{cond:}\left[\begin{array}{lll}\text{q} & : & \textit{Question}\\ \text{c} & : & \text{member}(\text{ask}(\text{cond.q}),\ \text{tis.is.shared.lu.moves})\end{array}\right] & \end{array}\right]$$

$$\left(\left[\text{tis:}\left[\text{is:}\left[\begin{array}{lll}\text{private:} & \text{agenda=respond}(r.\text{cond.q})|r.\text{prev.tis.is.private.agenda} & : [\textit{Action}]\\ \text{shared:} & \text{qud=}r.\text{cond.q}|r.\text{prev.tis.is.shared.qud} & : [\textit{Question}]\end{array}\right]\right]\right]\right.$$

The fixed point type

$$\left[\begin{array}{ll}
\text{prev}: & \left[\text{tis}: \left[\text{is}: \left[\begin{array}{lll} \text{private} & : & \left[\text{agenda=[]} \;:\; [\textit{Action}]\right] \\ \text{shared} & : & \left[\text{qud=[]} \;:\; [\textit{Question}]\right] \end{array}\right]\right]\right] \\
\text{tis} \quad : & \left[\text{is}: \left[\begin{array}{l} \text{private}: \left[\text{agenda=respond(cond.q)}\,|\,\text{prev.tis.is.private.agenda} \;:\; [\textit{Action}]\right] \\ \text{shared}: \left[\begin{array}{l} \text{qud=cond.q}\,|\,\text{prev.tis.is.shared.qud}:[\textit{Question}] \\ \text{lu}: \left[\begin{array}{l} \text{speaker=usr}:\textit{DP} \\ \text{moves}=\{\text{ask}(?A.\text{time_next_train}(\text{Bclna},A))\}:\{\textit{Move}\} \end{array}\right] \end{array}\right] \end{array}\right]\right] \\
\text{cond}: & \left[\begin{array}{lll} \text{q} & : & \textit{Question} \\ \text{c} & : & \text{member(ask(cond.q), tis.is.shared.lu.moves)} \end{array}\right]
\end{array}\right]$$

Reasoning about unique solutions

$$\left[\begin{array}{ll}
\text{prev}: & \left[\text{tis}: \left[\text{is}: \left[\begin{array}{lll} \text{private} & : & \left[\text{agenda=[]} \ : \ [\textit{Action}]\right] \\ \text{shared} & : & \left[\text{qud=[]} \ : \ [\textit{Question}]\right] \end{array}\right]\right]\right] \\
\text{tis} \quad : & \left[\text{is}: \left[\begin{array}{ll} \text{private}: & \left[\text{agenda=respond(cond.q)}\,|\,\text{prev.tis.is.private.agenda} \ : \ [\textit{Action}]\right] \\ \text{shared}: & \left[\begin{array}{ll} \text{qud=cond.q}\,|\,\text{prev.tis.is.shared.qud}: & [\textit{Question}] \\ \text{lu}: & \left[\begin{array}{l} \text{speaker=usr}:\textit{DP} \\ \text{moves}=\{\text{ask}(?A.\text{time_next_train}(\text{Bclna},A))\}:\{\textit{Move}\} \end{array}\right] \end{array}\right] \end{array}\right]\right] \\
\text{cond}: & \left[\begin{array}{ll} \text{q}=?A.\text{time_next_train}(\text{Bclna},A) & : \ \textit{Question} \\ \text{c:member(ask(cond.q), tis.is.shared.lu.moves)} & \end{array}\right]
\end{array}\right]$$

Substituting values of manifest fields

$$\left[\begin{array}{ll}
\text{prev}: & \left[\text{tis}: \left[\text{is}: \left[\begin{array}{lll} \text{private} & : & [\text{agenda}=[] \ : \ [\textit{Action}]] \\ \text{shared} & : & [\text{qud}=[] \ : \ [\textit{Question}]] \end{array}\right]\right]\right] \\
\text{tis} & : \left[\text{is}: \left[\begin{array}{ll} \text{private}: & [\text{agenda}=[\text{respond}(?A.\text{time_next_train}(\text{Bclna}, A))] \ : \ [\textit{Action}]] \\ \text{shared}: & \left[\begin{array}{l} \text{qud}=[?A.\text{time_next_train}(\text{Bclna}, A)]:[\textit{Question}] \\ \text{lu}: \left[\begin{array}{l} \text{speaker}=\text{usr}:\textit{DP} \\ \text{moves}=\{\text{ask}(?A.\text{time_next_train}(\text{Bclna}, A))\}:\{\textit{Move}\} \end{array}\right] \end{array}\right] \end{array}\right]\right] \\
\text{cond}: & \left[\begin{array}{l} \text{q}=?A.\text{time_next_train}(\text{Bclna}, A) \ : \ \textit{Question} \\ \text{c:member(ask(cond.q), tis.is.shared.lu.moves)} \end{array}\right]
\end{array}\right]$$

Garbage collection

Remove prev- and cond-paths not depended on in tis-paths

$$\left[\text{tis:}\left[\text{is:}\left[\begin{array}{l}\text{private:}\left[\text{agenda=[respond(?}A\text{.time\_next\_train(Bclna,}A\text{))]} \;:\; [\textit{Action}]\right]\\ \text{shared:}\left[\begin{array}{l}\text{qud=[?}A\text{.time\_next\_train(Bclna,}A\text{)]:[}\textit{Question}\text{]}\\ \text{lu:}\left[\begin{array}{l}\text{speaker=usr:}\textit{DP}\\ \text{moves=}\{\text{ask(?}A\text{.time\_next\_train(Bclna,}A\text{))}\}\text{:}\{\textit{Move}\}\end{array}\right]\end{array}\right]\end{array}\right]\right]\right]$$

3. Records and Tim's work

Regular types

1. if T_1, $T_2 \in$ **Type**, then $T_1T_2 \in$ **Type**
   $a : T_1T_2$ iff $a = x^\frown y$, $x : T_1$ and $y : T_2$
2. if $T \in$ **Type** then $T^+ \in$ **Type**.
   $a : T^+$ iff $a = x_1^\frown \ldots ^\frown x_n$, $n > 0$ and for i, $1 \leq i \leq n$, $x_i : T$
   ...

rain for two days

$$\left[\begin{array}{lcl} c_1 & : & \text{rain} \end{array}\right]^+$$
$$\wedge$$
$$\left[\begin{array}{lcl} c_2 & : & 0(\tau) \end{array}\right] Rec^+ \left[\begin{array}{lcl} c_2 & : & 2\text{days}(\tau) \end{array}\right]$$

$$\approx$$

$$\left[\begin{array}{lcl} c_1 & : & \text{rain} \\ c_2 & : & 0(\tau) \end{array}\right] \left[\begin{array}{lcl} c_1 & : & \text{rain} \end{array}\right]^+ \left[\begin{array}{lcl} c_1 & : & \text{rain} \\ c_2 & : & 2\text{days}(\tau) \end{array}\right]$$

Conjecture: Superposition of regular record types is the same as meet (intersection).

What is a fluent?

A *possible* definition!

f *is a fluent* iff f:(*Time*)*RecType* of the form λt:*Time*($\begin{bmatrix} \ell_1 & : & T_1(t) \\ \vdots & & \\ \ell_n & : & T_n(t) \end{bmatrix}$)

This means we should write things like:

λt:*Time* $\begin{bmatrix} \mathrm{c}_1 & : & \mathrm{rain}(t) \\ \mathrm{c}_2 & : & 0(\tau)(t) \end{bmatrix}$ λt:*Time* $\begin{bmatrix} \mathrm{c}_1 & : & \mathrm{rain}(t) \end{bmatrix}^+ \lambda t$:*Time* $\begin{bmatrix} \mathrm{c}_1 & : & \mathrm{rain}(t) \\ \mathrm{c}_2 & : & 2\mathrm{days}(\tau)(t) \end{bmatrix}$

A string of fluents could then be grounded in a sequence of times to obtain a string of record (situation) types.

We will ignore this.

Do types point out of their frames?

If $T \in$ **Type**, $T' \sqsubseteq T$ and $\ell \in$ **Label**, then $\left[\ell{:}\text{next}(T)\right]T' \in$ **Type**

$a : \left[\ell{:}\text{next}(T)\right]T'$ iff $a = x^\frown y$, x:*Rec* and $y : T'$

But it may be simpler (e.g. for simplifying meets) to regard notations such as next(T) as abbreviatory devices.

Truncation and intensionality

If $T \sqsubseteq \mathit{Rec}^+$, then ing($T$) $\in$ **Type**

s : ing(T) iff s : *Rec* Rec^+ $\exists T' \sqsubseteq \mathit{Rec}^+ \exists f : (T')Tf = \lambda s' : T'(s^\frown s')$

Alternative formulation:
s : ing(T) iff s : *Rec* Rec^+ $\exists T' \sqsubseteq \mathit{Rec}^+ \forall s' : T' s^\frown s' : T$

Pat-walk-home =

$$\left[\begin{array}{lcl} c_2 & : & \neg\text{home(p)} \end{array}\right] \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \neg\text{home(p)} \end{array}\right]^+ \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \text{home(p)} \end{array}\right]$$

ing(Pat-walk-home):

$$T' = \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \neg\text{home(p)} \end{array}\right]^+ \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \text{home(p)} \end{array}\right]$$

$$s : \left[\begin{array}{lcl} c_2 & : & \neg\text{home(p)} \end{array}\right] \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \neg\text{home(p)} \end{array}\right]^+$$

swimming out to sea on Atlantic coast $\not\approx$ swimming the Atlantic
drawing a horse $\not\approx$ drawing a unicorn

even though objects of these types may overlap in their intial segments

do observations in the overlap count as proofs of both types?

yes, but not in the same context

Make the completion type a contextually determined parameter

s : prog(T, T') iff s : *Rec* *Rec*$^+$ $\forall s' : T' s^\frown s' : T$

$[\![$ Pat walking home $]\!]$ =

$$\lambda r{:}\left[\text{compl-type}=\left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \neg\text{home(p)} \end{array}\right] + \left[\begin{array}{lcl} c_1 & : & \text{walk(p)} \\ c_2 & : & \text{home(p)} \end{array}\right] \;:\; \textit{Type}\right]$$

$$\left(\left[\text{ s } : \text{ prog(Pat-walk-home, } r.\text{compl-type)}\right]\right)$$

Note that we are using types as objects, i.e. an intensional construction.

Our completion type corresponds to the map c, Pt 2, slide 20.

Why might a completion type for Pat-walk-home be in the context?

Lots of possible reasons:

- it's known that that was Pat's intention
- it's the time of day when Pat normally walks home
- we have reason to believe that he will arrive home even though Pat intends to visit his aunt. *Pat is so absent-minded. He intended to visit his aunt but he was automatically walking home instead until he realized his mistake.*
- ...

Not the business of semantics to investigate this, though useful things can be done in limited domains.

Transitions as forces and inertia

Transition functions
$\lambda r : T_1(T_2)$
"If you have an object of type T_1, return T_2 (as the type of the next object)"

Initiation
$\lambda r{:}\left[\ell{:}\neg T\right](\left[\ell{:}T\right])$

Cessation
$\lambda r{:}\left[\ell{:}T\right](\left[\ell{:}\neg T\right])$

Other kinds transition functions? (von Wright)

Updating with transition functions
If $T \sqsubseteq T'$, then the result of updating T with $\lambda r : T_1(T_2)$ is a fixed point type of $\lambda r : T - T_2(T_2)$

If $f : (T)Type$ then T' *is a fixed point type* for f just in case $a : T'$ implies $a : f(a)$

$\left[\begin{array}{lcl} \text{x} & : & T_1 \\ \text{y} & : & T_2 \end{array}\right]$ is a fixed point type for $\lambda r : \left[\text{x}{:}T_1\right](\left[\text{y}{:}T_2\right])$

$T_1 - T_2 = T_1 - (X \cup Y)$
where
$X = \{\langle \ell, T\rangle \in T_1 \mid \exists T' \langle \ell, T'\rangle \in T_2\}$
$Y = \{\langle \ell, T\rangle \in T_1 \mid \exists \ell' T' \langle \ell', T'\rangle \in X$ and T depends on $\ell'\}$

Suppose we are updating

$$\begin{bmatrix} c_1 & : & \text{walk(p)} \\ c_2 & : & \neg\text{home(p)} \end{bmatrix}$$

with

$\lambda r{:}\left[c_2{:}\neg\text{home(p)}\right](\left[c_2{:}\text{home(p)}\right])$

(a initiation transition)

The result is a fixed point type for

$$\lambda r{:}\begin{bmatrix} c_1{:}\text{walk(p)} \\ c_2{:}\neg\text{home(p)} \end{bmatrix} - \left[c_2{:}\text{home(p)}\right] \left(\left[c_2{:}\text{home(p)}\right]\right)$$

=

$\lambda r{:}\left[c_1{:}\text{walk(p)}\right]\ (\left[c_2{:}\text{home(p)}\right])$

e.g.

$$\begin{bmatrix} c_1 & : & \text{walk(p)} \\ c_2 & : & \text{home(p)} \end{bmatrix}$$

Should there be backdate functions as well?

How should we get the effect of a force applying so that a fluent may or may not continue in the future?
Does this have to do with whether the next frame is always obtained by updating the previous frame with an update function? (Cuts from one scene to another?)