Using Type Theory with Records for HPSG

Robin Cooper
Göteborg University

The Department of Linguistics

GU DIALOGUE SYSTEMS LAB

The Records and Dialogue Semantics Project

http://www.ling.gu.se/~cooper/records/ (These slides are there)

- Records, types and computational dialogue semantics
- Funded by VETENSKAPSRÅDET THE SWEDISH RESEARCH COUNCIL
- 2003–2005
- CHALMERS GÖTEBORG UNIVERSITY Computing Science
  Computer Science and Engineering – Chalmers University of Technology and Göteborg University
- Robin Cooper, Thierry Coquand, Staffan Larsson, Peter Ljunglöf, Bengt Nordström, Aarne Ranta

⇐ contents

Four linguistic theories + dialogue management

- Montague semantics
- DRT
- situation semantics
- HPSG
- issue based dialogue management (Larsson)

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

HPSG in type theory with records

Collaboration with Jonathan Ginzburg, King's College, London.

cf. Ginzburg's paper at HPSG in Leuven

Contents

1. Records in type theory

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:

$$\begin{bmatrix} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \ldots & & \\ l_n & = & a_n \\ \ldots & & \end{bmatrix}$$

is of type:

$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ \ldots & & \\ l_n & : & T_n(l_1, l_2, \ldots, l_{n-1}) \end{bmatrix}$$

a man owns a donkey

Record type:

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

Record:

$$\begin{bmatrix} \text{x} & = & a \\ \text{c}_1 & = & p_1 \\ \text{y} & = & b \\ \text{c}_2 & = & p_2 \\ \text{c}_3 & = & p_3 \end{bmatrix}$$

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)

a man owns a donkey
Content (*intension*) is a record type:

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

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

Manifest fields

Coquand, Pollack and Takeyama

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.
$\left[\begin{array}{lll} x & : & T_a \end{array}\right]$
written for convenience as
$\left[\begin{array}{lll} x{=}a & : & T \end{array}\right]$

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.

2. Records as linguistic objects

man

$$\begin{bmatrix} \text{phon} & = & [\text{man}] \\ \text{cat} & = & \text{n} \\ \text{agr} & = & \begin{bmatrix} \text{num} & = & \text{sg} \\ \text{gen} & = & \text{masc} \\ \text{pers} & = & \text{third} \end{bmatrix} \end{bmatrix}$$

fish

$$\begin{bmatrix} \text{phon} & = & \text{[fish]} \\ \text{cat} & = & \text{n} \\ \text{agr} & = & \begin{bmatrix} \text{num} & = & \text{sg} \\ \text{gen} & = & \text{neut} \\ \text{pers} & = & \text{third} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{phon} & = & \text{[fish]} \\ \text{cat} & = & \text{n} \\ \text{agr} & = & \begin{bmatrix} \text{num} & = & \text{pl} \\ \text{gen} & = & \text{neut} \\ \text{pers} & = & \text{third} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{phon} & = & [\text{fish}] \\ \text{cat} & = & \text{n} \\ \text{agr} & = & \begin{bmatrix} \text{num} & = & \text{sg} \\ \text{gen} & = & \text{masc} \\ \text{pers} & = & \text{third} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{phon} & = & [\text{fish}] \\ \text{cat} & = & \text{n} \\ \text{agr} & = & \begin{bmatrix} \text{num} & = & \text{sg} \\ \text{gen} & = & \text{fem} \\ \text{pers} & = & \text{third} \end{bmatrix} \end{bmatrix}$$

...

Part of a type system (ignoring semantics)

$$Noun \equiv \begin{bmatrix} \text{phon} & : & Phon \\ \text{cat=n} & : & Cat \\ \text{agr} & : & \begin{bmatrix} \text{num} & : & Number \\ \text{gen} & : & Gender \\ \text{pers} & : & Person \end{bmatrix} \end{bmatrix}$$

$Phon \equiv [Lex]$

the, a, fish, man, . . . : *Lex*

n, det, np, v, vp, s, . . . : *Cat*

sg, pl : *Number*

masc, fem, neut : *Gender*

first, second, third : *Person*

$$Det \equiv \begin{bmatrix} \text{phon} & : & \mathit{Phon} \\ \text{cat=det} & : & \mathit{Cat} \\ \text{agr} & : & \begin{bmatrix} \text{num} & : & \mathit{Number} \\ \text{gen} & : & \mathit{Gender} \\ \text{pers} & : & \mathit{Person} \end{bmatrix} \end{bmatrix}$$

$$Agr \equiv \begin{bmatrix} \text{num} & : & \mathit{Number} \\ \text{gen} & : & \mathit{Gender} \\ \text{pers} & : & \mathit{Person} \end{bmatrix}$$

A singleton (fully specified) type

$$\left[\begin{array}{lcl} \text{phon=[man]} & : & \mathit{Phon} \\ \text{cat=n} & : & \mathit{Cat} \\ \text{agr} & : & \left[\begin{array}{lcl} \text{num=sg} & : & \mathit{Number} \\ \text{gen=masc} & : & \mathit{Gender} \\ \text{pers=third} & : & \mathit{Person} \end{array}\right] \end{array}\right]$$

This type identifies a unique linguistic object.
It is a *subtype* (or specification) of *Noun*.

A type with several elements (underspecification of linguistic objects)

$$\left[\begin{array}{lcl} \text{phon=[fish]} & : & \mathit{Phon} \\ \text{cat=n} & : & \mathit{Cat} \\ \text{agr} & : & \left[\begin{array}{lcl} \text{num} & : & \mathit{Number} \\ \text{gen} & : & \mathit{Gender} \\ \text{pers=third} & : & \mathit{Person} \end{array}\right] \end{array}\right]$$

This type identifies a set of linguistic objects with phonology [fish] and category noun.

It is also a subtype (specification) of *Noun*.

Two determiners with different degrees of specification

$$\left[\begin{array}{lll} \mathrm{phon}{=}[\mathrm{a}] & : & \mathit{Phon} \\ \mathrm{cat}{=}\mathrm{det} & : & \mathit{Cat} \\ \mathrm{agr} & : & \left[\begin{array}{lll} \mathrm{num}{=}\mathrm{sg} & : & \mathit{Number} \\ \mathrm{gen} & : & \mathit{Gender} \\ \mathrm{pers}{=}\mathrm{third} & : & \mathit{Person} \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{lll} \mathrm{phon}{=}[\mathrm{the}] & : & \mathit{Phon} \\ \mathrm{cat}{=}\mathrm{det} & : & \mathit{Cat} \\ \mathrm{agr} & : & \left[\begin{array}{lll} \mathrm{num} & : & \mathit{Number} \\ \mathrm{gen} & : & \mathit{Gender} \\ \mathrm{pers}{=}\mathrm{third} & : & \mathit{Person} \end{array}\right] \end{array}\right]$$

Subtypes of *Det*.

3. A case where we don’t seem to need unification

Noun phrases (with number agreement)

$$
\mathit{NP} \equiv
\left[\begin{array}{lll}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)} & : & \mathit{Phon} \\
\text{cat=np} & : & \mathit{Cat} \\
\text{daughters:}\left[\begin{array}{lll}
\text{first} & : & \mathit{Det} \\
\text{rest} & : & \left[\begin{array}{lll}
\text{first} & : & \mathit{Noun} \\
\text{rest=nil} & : & [\mathit{Sign}]
\end{array}\right]
\end{array}\right] & & \\
\text{agr=daughters.rest.first.agr} & : & \mathit{Agr} \\
\text{c:eq(}\mathit{Number}\text{, daughters.first.agr.num, daughters.rest.first.agr.num)} & &
\end{array}\right]
$$

$\mathit{Sign} \equiv \mathit{Det} \vee \mathit{Noun} \vee \mathit{NP} \vee \ldots$

eq(*Number*, daughters.first.agr.num, daughters.rest.first.agr.num) is a type of proof (objects). An object of this type would be a pair of objects $< a, b >$ such that a, b : *Number* and $a = b$.

the fish

phon=append(daughters.first.phon, daughters.rest.first.phon):*Phon*
cat=np:*Cat*
daughters:
first:
phon=[the]:*Phon*
cat=det :*Cat*
agr : num : *Number*
gen : *Gender*
pers=third : *Person*
rest:
first :
phon=[fish]:*Phon*
cat=n:*Cat*
agr: num : *Number*
gen : *Gender*
pers=third : *Person*
rest=nil:[*Sign*]
agr=daughters.rest.first.agr:*Agr*
c:eq(*Number*, daughters.first.agr.num, daughters.rest.first.agr.num)

the man

$$
\left[\begin{array}{l}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon}\\
\text{cat=np}:\mathit{Cat}\\
\text{daughters}: \left[\begin{array}{ll}
\text{first}: & \left[\begin{array}{ll}
\text{phon=[the]}:\mathit{Phon} & \\
\text{cat=det} & :\mathit{Cat}\\
\text{agr} & : \left[\begin{array}{lll} \text{num} & : & \mathit{Number}\\ \text{gen} & : & \mathit{Gender}\\ \text{pers=third} & : & \mathit{Person}\end{array}\right]
\end{array}\right]\\
\text{rest}: & \left[\begin{array}{ll}
\text{first} & : \left[\begin{array}{l} \text{phon=[man]}:\mathit{Phon}\\ \text{cat=n}:\mathit{Cat}\\ \text{agr}: \left[\begin{array}{lll} \text{num=sg} & : & \mathit{Number}\\ \text{gen=masc} & : & \mathit{Gender}\\ \text{pers=third} & : & \mathit{Person}\end{array}\right]\end{array}\right]\\
\text{rest=nil}:[\mathit{Sign}] &
\end{array}\right]
\end{array}\right]\\
\text{agr=daughters.rest.first.agr}:\mathit{Agr}\\
\text{c}:\text{eq}(\mathit{Number}, \text{daughters.first.agr.num}, \text{daughters.rest.first.agr.num})
\end{array}\right]
$$

Singularity coming from *man*.

a fish

$$
\left[\begin{array}{l}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon} \\
\text{cat=np}:\mathit{Cat} \\
\text{daughters}: \left[\begin{array}{ll}
\text{first}: & \left[\begin{array}{lcl}
\text{phon=[a]}:\mathit{Phon} & & \\
\text{cat=det} & : & \mathit{Cat} \\
\text{agr} & : & \left[\begin{array}{lcl}
\text{num=sg} & : & \mathit{Number} \\
\text{gen} & : & \mathit{Gender} \\
\text{pers=third} & : & \mathit{Person}
\end{array}\right]
\end{array}\right] \\
\text{rest}: & \left[\begin{array}{lcl}
\text{first} & : & \left[\begin{array}{ll}
\text{phon=[fish]}:\mathit{Phon} & \\
\text{cat=n}:\mathit{Cat} & \\
\text{agr}: & \left[\begin{array}{lcl}
\text{num} & : & \mathit{Number} \\
\text{gen} & : & \mathit{Gender} \\
\text{pers=third} & : & \mathit{Person}
\end{array}\right]
\end{array}\right] \\
\text{rest=nil}:[\mathit{Sign}] & &
\end{array}\right]
\end{array}\right] \\
\text{agr=daughters.rest.first.agr}:\mathit{Agr} \\
\text{c:eq}(\mathit{Number}\text{, daughters.first.agr.num, daughters.rest.first.agr.num)}
\end{array}\right]
$$

Singularity coming from *a*.

*a men

$$
\left[\begin{array}{ll}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon} & \\
\text{cat=np}:\mathit{Cat} & \\
\text{daughters}: & \left[\begin{array}{ll}
\text{first}: & \left[\begin{array}{ll}
\text{phon=[a]}:\mathit{Phon} & \\
\text{cat=det} & :\mathit{Cat} \\
\text{agr} & : \left[\begin{array}{lll} \text{num=sg} & : & \mathit{Number} \\ \text{gen} & : & \mathit{Gender} \\ \text{pers=third} & : & \mathit{Person} \end{array}\right]
\end{array}\right] \\
\text{rest}: & \left[\begin{array}{ll}
\text{first} & : \left[\begin{array}{ll}
\text{phon=[men]}:\mathit{Phon} & \\
\text{cat=n}:\mathit{Cat} & \\
\text{agr}: & \left[\begin{array}{lll} \text{num=pl} & : & \mathit{Number} \\ \text{gen=masc} & : & \mathit{Gender} \\ \text{pers=third} & : & \mathit{Person} \end{array}\right]
\end{array}\right] \\
\text{rest=nil}:[\mathit{Sign}] &
\end{array}\right]
\end{array}\right] \\
\text{agr=daughters.rest.first.agr}:\mathit{Agr} & \\
\text{c}:\text{eq}(\mathit{Number}, \text{daughters.first.agr.num}, \text{daughters.rest.first.agr.num}) &
\end{array}\right]
$$

An empty type – useful for robust parsing?

Intensionality and robust parsing

- types can be empty (in our case, no linguistic objects of that type)
- this does not mean that they are ill-formed types
- there are lots of distinct empty types (intensionality)
- even though a type is empty we still have a lot of information about what would have been objects of the type (contradictions are local, cf unification failure)
- e.g. if the type includes a representation of content we would still be able to interpret ungrammatical input
- potential for building a parser which assigns empty types to ungrammatical strings (in addition to types to grammatical strings)
- same intensionality that gives us an improved treatment of propositional attitudes in semantics
- therefore potentially saying something general about information and cognition, blah, blah
- cf. Frederik Fouvry's recent work on constraint relaxation with weighted feature structures (Essex PhD 2003)

Extracting out the head feature principle

$$
\mathit{NP} \equiv
\left[\begin{array}{lll}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)} & : & \mathit{Phon} \\
\text{cat=np} & : & \mathit{Cat} \\
\text{hd-daughter=daughters.rest.first} & : & \mathit{Noun} \\
\text{daughters:} \left[\begin{array}{lll}
\text{first} & : & \mathit{Det} \\
\text{rest} & : & \left[\begin{array}{lll}
\text{first} & : & \mathit{Noun} \\
\text{rest=nil} & : & [\mathit{Sign}]
\end{array}\right]
\end{array}\right] & & \\
\text{agr} & : & \mathit{Agr} \\
\text{c:eq(}\mathit{Number}\text{, daughters.first.agr.num, daughters.rest.first.agr.num)} & &
\end{array}\right]
$$

$$\wedge$$

$$\mathit{HFP}$$

$$
\mathit{HFP} \equiv
\left[\begin{array}{lll}
\text{hd-daughter} & : & \mathit{Sign} \\
\text{agr=hd-daughter.agr} & : & \mathit{Agr}
\end{array}\right]
$$

Double specification over long distances

wem Hans begegnete
(I wonder) who [dat] Hans met

$$\left[\begin{array}{l}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon}\\
\text{cat=s}:\mathit{Cat}\\
\text{daughters}:\left[\begin{array}{l}\text{first}:\mathit{WH}\\ \text{rest}:\left[\begin{array}{l}\text{first}:\mathit{S}\\ \text{rest=nil}:[\mathit{Sign}]\end{array}\right]\end{array}\right]\\
\text{c}_1:\text{eq}(\mathit{Cat}, \text{daughters.first.cat, daughters.rest.first.slash.cat})\\
\text{c}_2:\text{eq}(\mathit{Agr}, \text{daughters.first.agr, daughters.rest.first.slash.agr})
\end{array}\right]$$

Agr now defined to include a case feature. *Cat* and *Agr* might be zipped together (part of local?) so that we only need one constraint.

4. Putting content into TTR-HPSG

$\mathbf{S} \longrightarrow \mathbf{NP\ VP}$

$$S \equiv \begin{bmatrix} \text{phon}=\text{append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon} \\ \text{cat}=\text{s}:\mathit{Cat} \\ \text{daughters}: \begin{bmatrix} \text{first}:\mathit{NP} \\ \text{rest}: \begin{bmatrix} \text{first}:\mathit{VP} \\ \text{rest}=\text{nil}:[\mathit{Sign}] \end{bmatrix} \end{bmatrix} \\ \text{content}=\text{daughters.first.content@daughters.rest.first.content}:\mathit{RecType} \\ \text{c}:\text{eq}(\mathit{Agr}\text{, daughters.first.agr.num, daughter.rest.first.agr.num)} \end{bmatrix}$$

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

VP $\longrightarrow$ V NP

$$\left[\begin{array}{l}\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon}\\ \text{cat=vp}:\mathit{Cat}\\ \text{daughters}:\left[\begin{array}{l}\text{first}:\mathit{V}\\ \text{rest}:\left[\begin{array}{l}\text{first}:\mathit{NP}\\ \text{rest=nil}:[\mathit{Sign}]\end{array}\right]\end{array}\right]\\ \text{agr=daughters.first.agr}:\mathit{Agr}\\ \text{content=daughters.first.content@daughters.rest.first.content}:(\left[\text{x}:\mathit{Ind}\right])\mathit{RecType}\end{array}\right]$$

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

Compositionality

a donkey

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

@

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

=

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

own a donkey

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

$$@$$

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

$$=$$

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

5. A case where we seem to need not to have unification

Information structure

Vallduví, Engdahl and Vallduví

- [The president hates the Delft china set $_{\mathrm{F}}$]
- The president [hates the Delft china set $_{\mathrm{F}}$]
- The president hates [the Delft china set $_{\mathrm{F}}$]
- The president [hates $_{\mathrm{F}}$] the Delft china set

The problem

In a classical HPSG approach using unification the content of the verb is the same as the content of the sentence

The syntactic strategy

Engdahl and Vallduví

Mark focus on syntactic constituents

Still leaves us with no representation of focus information

The MRS strategy

Wilcock
every dog chased [some cat $_F$]

1:every(x,3,5), 3:dog(x), 7:cat(y), 5:some(y,7,4), 4:chase(e,x,y)
TOP-HANDLE:1, LINK:1, FOCUS:5

You seem to get different focus information depending on how you plug things together? Notice that scope and information structure are independent.

The separate semantic formalism strategy

Ericsson
From the unification formalism point of view semantic representations (e.g. using the λ-calculus) are atoms

This works fine because no unification is going on in the semantic representation.
But it misses the integration of syntax and semantics that HPSG wants to have.

GU Dialogue Systems Lab

Towards a TTR-HPSG analysis

We show some expressive possibilities, though not yet an general analysis.

VP $\rightarrow$ V [$_F$ NP]

$$\begin{bmatrix}
\text{phon=append(daughters.first.phon, focus(daughters.rest.first.phon))}:\mathit{Phon} \\
\text{cat=vp}:\mathit{Cat} \\
\text{daughters}: \begin{bmatrix} \text{first}:\mathit{VSign} \\ \text{rest}: \begin{bmatrix} \text{first}:\mathit{NPSign} \\ \text{rest=nil}:[\mathit{Sign}] \end{bmatrix} \end{bmatrix} \\
\text{content=daughters.first.content@daughters.rest.first.content}:\mathit{RecType} \\
\text{info-struct}: \begin{bmatrix} \text{focus=daughters.first.rest.first.content}:\mathit{Quant} \\ \text{ground=daughters.first.content}:(\mathit{Quant})\mathit{Property} \end{bmatrix}
\end{bmatrix}$$

$$\left[\begin{array}{l}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\mathit{Phon}\\
\text{cat=s}:\mathit{Cat}\\
\text{daughters}:\left[\begin{array}{l}\text{first}:\mathit{NPSign}\\ \text{rest}:\left[\begin{array}{l}\text{first}:\mathit{VPSign}\\ \text{rest=nil}:[\mathit{Sign}]\end{array}\right]\end{array}\right]\\
\text{content=daughters.first.content@daughters.rest.first.content}:\mathit{RecType}\\
\text{info-struct}:\left[\begin{array}{l}\text{focus=daughters.first.rest.first.focus}:\mathit{Quant}\\ \text{ground=}\lambda q{:}\mathit{Quant}\\ \qquad\qquad\text{(daughters.first.content}\\ \qquad\qquad\qquad\text{@(daughters.rest.first.ground@}q\text{))}:\mathit{RecType}\end{array}\right]
\end{array}\right]$$

6. Conclusions

Type theory with records provides us with a powerful formalism for

- feature-based analyses
- "unification" phenomena
- semantics
- "functional" phenomena

and, importantly, an *integrated* approach.

A. Abstract and concrete syntax

Names for abstract and concrete syntax

abstract syntax	concrete syntax	Ranta GF, compiler technology
tectogrammar	phenogrammar	Curry, Dowty
categorial grammar derivation	strings at nodes in derivation	Montague's CG based grammar

Properties of systems which distinguish abstract and concrete syntax

- Concrete syntax is a *compositional* projection from abstract syntax
- Abstract syntax can be a semantic representation, but may be a more abstract syntactic representation (cf Montague Grammar)
- A grammar can be regarded as a function from expressions of abstract syntax to sets of expressions of concrete syntax (Ranta)
- There may be several concrete syntaxes corresponding to a single abstract syntax (Ranta)
- Grammars (functions) can be composed (Ranta)

Related compositional notions

Abstract	Concrete
CF-backbone	HPSG feature structure
C-structure	F-structure
C-structure	Phonology projection

An intuitively related non-compositional notion

Abstract	Concrete
D-structure	S-structure
D-structure	LF
D-structure	P-structure

Extracting the categorial abstract syntax from HPSG signs

Use the cat and daughters attributes.

$$[\![\ [_\mathrm{S}\ \mathrm{NP}\ \mathrm{VP}]\ ]\!] = \begin{bmatrix} \text{phon=append(daughters.first.phon, daughters.rest.first.phon):}\mathit{Phon} \\ \text{cat=s:}\mathit{Cat} \\ \text{daughters:} \begin{bmatrix} \text{first=}[\![\ \text{NP}\ ]\!]\text{:}\mathit{NPSign} \\ \text{rest:} \begin{bmatrix} \text{first=}[\![\ \text{VP}\ ]\!]\text{:}\mathit{VPSign} \\ \text{rest=nil:[}\mathit{Sign}\text{]} \end{bmatrix} \end{bmatrix} \\ \text{content=daughters.first.content@daughters.rest.first.content:}\mathit{RecType} \end{bmatrix}$$

Using a more GF-like notation

predS np vp =

$$\left[\begin{array}{ll}
\text{phon=append(daughters.first.phon, daughters.rest.first.phon):}\textit{Phon} \\
\text{cat=s:}\textit{Cat} \\
\text{daughters:} & \left[\begin{array}{ll} \text{first=np:}\textit{NPSign} \\ \text{rest:} & \left[\begin{array}{l} \text{first=vp:}\textit{VPSign} \\ \text{rest=nil:[}\textit{Sign}\text{]} \end{array}\right] \end{array}\right] \\
\text{content=daughters.first.content@daughters.rest.first.content:}\textit{RecType}
\end{array}\right]$$

Extracting function argument abstract syntax from HPSG signs

Use the content attributes.

apply np vp =

$$\left[\begin{array}{l}\text{phon=append(daughters.first.phon, daughters.rest.first.phon)}:\textit{Phon}\\ \text{cat=s}:\textit{Cat}\\ \text{daughters}:\left[\begin{array}{l}\text{first=np}:\textit{NPSign}\\ \text{rest}:\left[\begin{array}{l}\text{first=vp}:\textit{VPSign}\\ \text{rest=nil}:[\textit{Sign}]\end{array}\right]\end{array}\right]\\ \text{content=daughters.first.content@daughters.rest.first.content}:\textit{RecType}\end{array}\right]$$

Projecting the content and phonology

(apply np vp).content = np.content@vp.content
(apply np vp).phon = append(np.phon, vp.phon)

HPSG and LFG

- HPSG provides data structures in which abstract syntax is hidden (but can be extracted)
- LFG uses C-structure as abstract syntax and projects to different representations (which could be zipped together in a single data structure)
- A true grammatical framework will allow us to convert grammars between these and other possibilities (e.g. alternative abstract syntaxes)
- It will also allow high level reasoning about grammars (alternative concrete syntaxes, grammar composition, grammar extraction)