Clarification and incremental meaning/content refinement

Robin Cooper
Göteborg University
Contents

1 Clarification and meaning/content 3
2 Record type-theoretical semantics 11
3 Applying record-type theoretical semantics to clarification 18
1. Clarification and meaning/content
Simple clarification exchanges

A: Bo left
B: Bo?
A: Bo Ralph (distinguished occupant of Seat No. 2, Swedish Academy)
B: Oh

What are the consequences for notions of meaning and content?
Treatment of proper names

- Not logical constants
- Instead: $x \mid \text{named}(x, \text{“Bo”})$

*cf.* early situation semantics literature
Content tout court

A: Bo left
B: Bo?

What is the content of A’s utterance at this point in the dialogue? More useful to talk of

- the content for A (“speaker”-content)
- the content for B (“hearer”-content)

cf. early situation semantics literature
Clarification as an argument for structured meaning/content

A: Bo left
B: Bo?

In order to be able to analyze the content of B’s utterance you at least need access to the content of constituents of A’s utterance.

cf. a number of proposals for structured meanings, use of structured objects in situation semantics
Clarification and Montague/Kaplan meaning functions

A: Bo left

- At this point \( B \), not having a referent for \( Bo \), has gained the information “Somebody named Bo left”.

- Not strictly obtainable on the classical view of meaning as a function from contexts to content (since the context is not in the domain of the function).

- Need to coerce the meaning function.

- Not sufficient to say that \( B \) has not yet been able to compute a content of \( A \)’s utterance.
B: Bo?
A: Bo Ralph

- At this point $B$, having a referent for *Bo Ralph*, now has a referent for *Bo* and has gained the information “Bo Ralph left”.
- In order to compute this we need to get back the original non-coerced meaning function.
• In earlier work on record type-theoretic semantics, I was pleased that we were able to define the coercions relatively elegantly.

• Now I think there’s a better way to do it, reflecting incremental specification of content and relaxing the rigid division between meaning and content of the classical Montague/Kaplan view.
2. Record type-theoretical semantics
Ingredients from (Martin-Löf) type theory

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

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

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

is of type:

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

$\Leftarrow$ contents
a man owns a donkey

Record type:

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

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 \(\text{man}(a)\)
- \(p_2\) is a proof of \(\text{donkey}(b)\)
- \(p_3\) is a proof of \(\text{own}(a, b)\)

\(\Leftarrow\) contents
a man owns a donkey

Content (intension) is a record type:

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

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

A function from contexts (modelled as records) to record types, i.e. of type \((T)\text{RecType}\), where \(T\) is some record type.

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

of type \((\text{Rec})\text{RecType}\)
Meanings as dependent functions

Sam owns a donkey

$$\lambda r: \begin{bmatrix} x & : & Ind \\ c_1 & : & \text{named}(x, \text{“Sam”}) \end{bmatrix} \left( \begin{bmatrix} y & : & Ind \\ c_2 & : & \text{donkey}(y) \\ c_3 & : & \text{own}(r.x,y) \end{bmatrix} \right)$$

cf. Montague, Kaplan
and within type theory using type theoretical contexts Ranta, Ahn, Piwek
among many others
3. Applying record-type theoretical semantics to clarification
The coercion analysis

A: Bo left

$B$ computes meaning of $A$’s utterance:

$$
\lambda r: \left[ \begin{array}{l}
x : \text{Ind} \\
c_1 : \text{named}(x, "Bo")
\end{array} \right] \left( \left[ c_2 : \text{leave}(r.x) \right] \right)
$$

$B$ notices that her context is not of the right type to be an argument to this function, computes a coerced function by “lowering”:

$$
\lambda r: \text{Rec} \left( \left[ \begin{array}{l}
x : \text{Ind} \\
c_1 : \text{named}(x, "Bo") \\
c_2 : \text{leave}(x)
\end{array} \right] \right)
$$

$B$ applies this content to her context to obtain content:

$$
\left[ \begin{array}{l}
x : \text{Ind} \\
c_1 : \text{named}(x, "Bo") \\
c_2 : \text{leave}(x)
\end{array} \right] \rightleftharpoons \text{contents}
$$
B: Bo?
A: Bo Ralph

$B$ reaccesses original meaning of $A$’s first utterance:

$$\lambda r:\left[ \begin{array}{l}
x & : & \text{Ind} \\
c_1 & : & \text{named}(x, "Bo")
\end{array} \right] \left( \begin{array}{l}
c_2 & : & \text{leave}(r.x)
\end{array} \right)$$

Applies to updated context to obtain new content:

$$\left[ \begin{array}{l}
c_2 & : & \text{leave}(bo\_ralph)
\end{array} \right]$$
The “unification” analysis

- corresponds to the use of c-params in HPSG.
- meaning modelled as a record type rather than a function
- *i.e.* meaning and content are both record types
- important for incrementality
A: Bo left

\[ \text{B computes meaning of A's utterance:} \]
\[
\begin{array}{c}
\text{cntxt} : \begin{cases}
x : \text{Ind} \\
c_1 : \text{named}(x, "Bo") \\
c_2 : \text{leave}(\text{cntxt}.x)
\end{cases} \\
\end{array}
\]
\( (= T) \)

\( T \) is defined wrt to context \( r \) iff \( r : T.\text{cntxt} \)

\( T \) is true wrt to context \( r \) iff \( T \) is defined wrt \( r \) and inhabited

\( T \) is false wrt to context \( r \) iff \( T \) is defined wrt \( r \) and uninhabited

Note that \( T \) (the meaning) is either inhabited or not so we have a notion of truth for meanings independent of context.

\( \iff \text{contents} \)
So what context are we in?

- safe to assume we have *incomplete* information
- *i.e.*, the context is *underspecified*
- underspecified objects represented by types
A context type

\[
\begin{array}{c}
x : Ind \\
c_1 : \text{named}(x, \text{“Bo”})
\end{array}
\]

But how do you show that the context is specified?
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.

\[
\begin{array}{c}
  x : T_a \\
\end{array}
\]

written for convenience as

\[
\begin{array}{c}
  x \text{=} a : T \\
\end{array}
\]

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.
A specified context type

\[
\begin{bmatrix}
\text{x=bo_ralph} & : Ind \\
\text{c1} & : \text{named(x, “Bo”)}
\end{bmatrix}
\]
Combining meaning and context

- Can’t do function application now
- Type conjunction (meet) instead

If $T_1$ and $T_2$ are types, then $T_1 \land T_2$ is also a type.

$a : T_1 \land T_2$ iff $a : T_1$ and $a : T_2$
Simplifying meets of record types

If \( T_1 \) and \( T_2 \) are record types then there will always be a record type (not a meet) \( T_3 \) which is equivalent to \( T_1 \land T_2 \) (in the sense that \( a : T_3 \iff a : T_1 \land T_2 \)).

Some examples:

\[
[f : T_1] \land [g : T_2] \approx [f : T_1, g : T_2] \\
[f : T_1] \land [f : T_2] \approx [f : T_1 \land T_2]
\]
Continuing with the dialogue...

B: Bo?
A: Bo Ralph

B conjoins her updated context type with the meaning type for A original utterance:

\[
\begin{align*}
\text{cntxt} & : \begin{bmatrix}
  x & : & \text{Ind} \\
  c_1 & : & \text{named}(x, \text{“Bo”}) \\
  c_2 & : & \text{leave}(\text{cntxt}.x)
\end{bmatrix} \\
\approx & \\
\text{cntxt} & : \begin{bmatrix}
  x=\text{bo_ralph} & : & \text{Ind} \\
  c_1 & : & \text{named}(x, \text{“Bo”}) \\
  c_2 & : & \text{leave}(\text{cntxt}.x)
\end{bmatrix}
\end{align*}
\]
Features of the “unification” approach

- new meets can be formed as context information comes in allowing incremental specification of content

- avoids coercion of functions and retrieving uncoerced versions (or slightly less coerced versions)

- the Montague/Kaplan meaning-content dichotomy has given way to incremental specification of meaning/content

- we have not given up the traditional paraphernalia of semantics such as binding, quantification, functions (e.g. used in connection with compositional semantics) as we have to do with a unification based system

⇐ contents