Introduction to CDRT (Muskens, 1996)

October 3, 2017

1 Introduction

- Class 1 – the following dynamic equivalence established:
  
  \( \lambda y_1 \lambda x_1 \ldots x_n. \phi = \lambda y_1 \lambda x_1 \ldots x_n. \psi \)  
  (1)  
  a. \( \exists y_1 \exists x_1 \ldots x_n. \neg \phi \equiv \exists y_1 \exists x_1 \ldots x_n. \neg \psi \)

- (1-a) is the resulting formula for wh-words in G&S (82), while (1-b) involves existential quantification; on an intuitive level, this is a step forward in capturing the indefinite-interrogative affinity

- However, we are not done; we still do not provide denotations of wh-words which would parallel indefinites; and we still did not show that wh-words could be treated categorically and in a compositional way

- That’s where we are heading

- The goal is to achieve compositionality at sub-clausal level by “[] combin[ing] Montague Semantics and Discourse Representation into a formalism that is not only notationally adequate, in the sense that the working linguist need remember only a few rules and notations, but is also mathematically rigorous and based on ordinary type logic. (…) The presence of boxes in type logic permits us to fuse DRT and Montague Grammar in a rather evenhanded way: both theories will be recognizable in the result. (…) With this unification of the theories standard techniques (…) that are used in Montague Grammar become available in DRT.” (Muskens 1996: 144-145)

- The compositionality is achieved by “grafting: logics L and L’ can often be combined by (a) taking an adequate Tarski definition of L’, (b) axiomatising within L the concepts that this definition talks about, and then (c) introducing the syntactic constructs of L’ as abbreviations within L by means of a transcription of the Tarski definition. L’ can then be said to be grafted upon L.” (Muskens, 1996: 182)

- L - many-sorted type logic

- L’ - DRT

2 A quick summary of DRT

(2) A man entered. He whistled.

a. \( [x_1 | \text{man } x_1, \text{ whistle } x_1] \)

(3) DRT semantic clauses, after Muskens (1996)

a. Relations:
   \( [R(\delta_1, \ldots, \delta_n)] = \{a(\bar{\delta}_1, \ldots, \bar{\delta}_n) \in I(R)\} \)

b. DRS:
   \( [[x_1, \ldots x_n]] = \{a[a(x_1, \ldots x_n) a'] \text{ and } a' \in \{y_1 \} \cap \ldots \cap \{y_n\}\} \)

c. Operations on DRSs:
   \( [\text{not } K] = \{a' \neq \exists (a, a') \in [K]\} \)
   \( [K \text{ or } K'] = \{a' \exists (a, a') \in [K] \lor (a, a') \in [K']\} \)
   \( [K \Rightarrow K'] = \{a' \forall (a, a') \in [K] \rightarrow \exists (a', a'') \in [K']\} \)

1
$a[x_1, \ldots, x_n]a'$ – assignments $a$ and $a'$ differ at most with respect to $x_1, \ldots, x_n$

(4) a. A DRS $K$ is true under $M, a$ iff there is some embedding $a'$ s.t. $\langle a, a' \rangle \in K$
b. A DRS $K$ is true under $M$ iff $K$ is true under $M, a$ for all embeddings $a$.

(5) A man entered. He whistled.
a. $[x_1|\text{man } x_1]; \ [\text{whistle } x_1]$

(6) Semantic clause for merging
a. $[K; K'] = \{\langle a, a' \rangle | 3a'' (\langle a, a'' \rangle \in K \text{ and } \langle a'', a' \rangle \in K) \}$

(7) Merging lemma
If $x_1' \ldots x_n'$ do not occur in any $C_1 \ldots C_n$, then
$[[x_1 \ldots x_n|C_1 \ldots C_n]; [x_1' \ldots x_n'|C_1' \ldots C_n']] = [[x_1 \ldots x_n, x_1' \ldots x_n'|C_1 \ldots C_n, C_1' \ldots C_n']]$

3 Compositional DRT (CDRT)

- With (7), we have very limited compositionality (at the level of sentences)
- For sub-clausal level, we would need to adopt Montague’s strategy, using lambda abstraction and application in logical language
- We will now introduce type logic with axioms to emulate DRT

3.1 Preliminaries

The logic that underlies the entire unification project is $Ty_3$. The set of basic types is $\{t, e, s, \pi\}$ (see Tab. 1):

- type $t$ is the type of truth values: $D_t = \{0, 1\}$
- type $e$ is the type of individuals
- type $s$, state, models variable assignments
- type $\pi$ are registers (“chunks of space containing just one object”, Muskens, 1996, p. 154)

- discourse referents (drefs) – register names
  - unspecific drefs name variable registers (registers whose content can be changed)
  - specific drefs name constant registers (registers whose content remains fixed)

(8) a. $i[\delta_1 \ldots \delta_n]j$ – states $i$ and $j$ differ at most wrt $\delta_1 \ldots \delta_n$
b. $i[j] := \forall v (V(v)(i) = V(v)(j))$

- $V$ – constant function $(\pi(se))$
- inhabitant of the register $\delta$ in state $i$ – $V(\delta)(i)$

4 axioms:

(9) a. ‘Enough states’ – for each state, variable register and entity, there must be a second state which is like the first one but the entity occupies the given register
$\forall i \forall v \forall x (VAR(v) \rightarrow \exists j ([i]j \land V(v)(j) = x))$
b. Unspecific drefs refer to variable registers
$VAR(u)$, if $u$ is an unspecific dref
<table>
<thead>
<tr>
<th>Type</th>
<th>Constant</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e$</td>
<td>john, mary</td>
<td>-</td>
</tr>
<tr>
<td>$s$</td>
<td>$i, j, k$</td>
<td></td>
</tr>
<tr>
<td>$\pi$</td>
<td>$u_0, u_1, John, Mary$</td>
<td>$v$</td>
</tr>
</tbody>
</table>

Table 1: Basic types

c. Different unspecific drefs refer to different registers
   $u_n \neq u_m$, for each two different unspecific drefs $u_n$ and $u_m$
d. Constant registers always have the same inhabitant
   $\forall i (V(Tom)(i) = to\text{m})$ etc. for all names

3.2 Translations from DRT

(10) Abbreviations

a. Relations
   $R[\delta_1, \ldots, \delta_n] := \lambda i_s.R(V(\delta_1)(i), \ldots, V(\delta_n)(i))$

b. $[u_1, \ldots, u_n|C_1, \ldots, C_m] := \lambda i_s.\lambda j_s.[u_1, \ldots, u_n] \wedge C_1 \wedge \ldots \wedge C_m$

c. Operations on DRSs
   not $K := \lambda i_s.\neg \exists_j s.(Kij)$
   $K$ or $K' := \lambda i_s.\exists_j s.(Kij \lor K'ij)$
   $K \Rightarrow K' := \lambda i_s.\forall j s.(Kij \Rightarrow \exists k s.(K'jk))$

d. Merging
   $K_1; K_2 := \lambda i_s.\lambda j_s.\exists h_s.(K_1 ih \wedge K_2 hj)$

(11) Some more abbreviations

a. $\exists u_s(K) := [u_s][; K$

b. $\forall u_s(K) := \neg ([u_s][; \neg K])$
   $ := \lambda i_s.\forall h_s.(i[u_s]h \rightarrow \exists k_s.(Khk))$

(12) Truth:
A DRS $K$ (type s(st)) is true with respect to an input info state $i_s$ iff $\exists_j s.(Kij)$
A DRS $K$ (type s(st)) is true iff $\forall i \exists_j s.(Kij)$

Examples:

(13) Cross-sentential anaphora

a. A boy admired a teacher. He brought her a present.

b. $[x_1, x_2|\text{boy } x_1, \text{teacher } x_2, x_1 \text{ admired } x_2; [x_3|\text{present } x_3, \text{brought}(x_1, x_2, x_3)]$

c. $[u_1, u_2|\text{boy}[u_1], \text{teacher}[u_2], \text{admired}[u_1, u_2]; [u_3|\text{present}[u_3], \text{brought}[u_1, u_2, u_3]]$

3.3 A small fragment

- Syntax – The Y-model of syntax (Chomsky, Lectures on Government and Binding)
- Tree structures generated by phrase-structure rules + transformations
- Transformations are used to generate new trees on S-Structure, as well as new trees on LF-structure

- Semantics – conventions, see Tab. 2
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Constant</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>et</td>
<td>Static predicates</td>
<td>farmer, sleeps</td>
<td>-</td>
</tr>
<tr>
<td>⟨e(et)⟩</td>
<td>Static relations</td>
<td>loves, owns</td>
<td>-</td>
</tr>
<tr>
<td>⟨s(st)⟩</td>
<td>Dynamic propositions</td>
<td>-</td>
<td>p, q</td>
</tr>
<tr>
<td>⟨π(st)⟩</td>
<td>Dynamic predicates</td>
<td>-</td>
<td>P</td>
</tr>
<tr>
<td>⟨⟨πt⟩⟩</td>
<td>Dynamic quantifiers</td>
<td>-</td>
<td>Q</td>
</tr>
</tbody>
</table>

Table 2: Types in fragment

(14) Basic translations

a. sleep \(\sim \lambda v_n.[\lambda v_{et}[v]]\)
b. farmer \(\sim \lambda v_n.[\lambda v_{et}[v]]\)
c. Tim \(\sim \lambda P_{nt}.P(Tim)\)
d. he \(\sim \lambda P_{nt}.P(u_n)\)
e. himself \(\sim \lambda P_{nt}.P(u_n)\)
f. t \(\sim \lambda P_{nt}.P(v_n)\)
g. own \(\sim \lambda Q_{nt}\lambda v_n.Q(\lambda v'.[\lambda v_{et}[v, v']])\)
h. hate \(\sim \lambda Q_{nt}\lambda v_n.Q(\lambda v'.[\lambda v_{et}[v, v']])\)
i. every \(\sim \lambda P'_{nt}\lambda P_{nt}.[\forall u_n[P'(u_n) \Rightarrow P(u_n)]]\)
j. a \(\sim \lambda P'_{nt}\lambda P_{nt}.[\exists u_n[P'(u_n); P(u_n)]]\)

(15) Translations of nodes

a. Non-branching nodes – the mother inherits the translation of the daughter
b. Functional application – the translation of the mother is the result of applying the translation of one daughter to the translation of the other
c. Sequencing – DRSs are sequenced using ‘;’
d. Quantifying-in – If NP \(\sim a\) and S \(\sim β\), then S \(\sim a(\lambda v_n.β)\)

(16) Example

a. Every farmer hates himself.

(17) a. A boy adores every woman.
   every \(\gg a\) \([u_1[\lambda v_{et}[u_1]] \Rightarrow [u_2[\lambda v_{et}[u_2]], adore[u_1, u_2]]\]
   a \(\gg\) every \([u_2[\lambda v_{et}[u_2]], [u_1[\lambda v_{et}[u_1]]] \Rightarrow [adore[u_1, u_2]]\)

(18) Relative-clause donkey sentence

a. Every boy who admires a teacher buys her a present.
b. \([u_1[\lambda v_{et}[u_1]], teacher[u_2], admire[u_1, u_2] \Rightarrow [u_3[\lambda v_{et}[u_3]], buy[u_1, u_2, u_3]]\]
c. \(\lambda x.\forall y.(boy(x_1) \land teacher(x_2) \land admire(x_1, x_2) \rightarrow \exists y.[present(x_3) \land buy(x_1, x_2, x_3)]\)

(19) Donkey anaphora

a. If a boy admires a teacher, he buys her a present.

(20) Some notes

a. The current system only captures unselective binding.
b. This is problematic for some cases:
   In most cases if a farmer owns a donkey he beats it.
   Most farmers who own a donkey beat it.
c. The current system makes a distinction between constant and variable registers (unspecific and specific drefs); this is often dropped (Haida)
d. Often, registers are derived as type ⟨se⟩ (Brasoveanu, 2007, Haida); we’ll see that on Friday