Computational Semantics (SCA-UE-908C3)

Tuesday 24th January 2012

For every question, we remind you ***to explain*** *and* ***to precisely justify*** *all the reasonings that lead you to the answer. The quality, the clarity and the precision of these elements will be taken into account (remember that clarity and concision often go hand in hand). Don't forget to say what the formula you write down stands for and what the relations between the formula are.*

And don't forget to read all the examination questions before starting to answer.

Consider the following abstract syntax together with its associated Montague-like semantics:

$$
\begin{array}{rlrl}
\textsc{Alice} : & np & [\![\textsc{Alice}]\!] = & \lambda P.P\ \textbf{alice} \\
\textsc{someone} : & np & [\![\textsc{someone}]\!] = & \lambda P.\exists_e x.P\ x \\
\textsc{left} : & np \to s & [\![\textsc{left}]\!] = & \lambda S.S(\lambda x.\textbf{left}\ x) \\
\textsc{think} : & cc \to np \to s & [\![\textsc{think}]\!] = & \lambda C.\lambda s.s(\lambda x.\textbf{think}\ x\ (C\ (\lambda t.t))) \\
\textsc{that} : & (np \to s) \to (np \to cc) & [\![\textsc{that}]\!] = & \lambda P.\lambda s.\lambda r.s\ (\lambda x.P(\lambda q.r\ (q\ x)))
\end{array}
$$

where $[\![\cdot]\!]$ denotes both the interpretation of the syntactic types into the semantic types and the interpretation of the abstract terms into the semantic terms. Moreover:

$$
\begin{array}{rlrl}
\textbf{alice} : & e & \textbf{human} : & e \to t \\
\textbf{left} : & e \to t & \textbf{think} : & e \to t \to t
\end{array}
$$

1. Check that the following term t is well-typed. What is its type?

$$t = \textsc{think}\ (\textsc{that left someone})\ \textsc{Alice}$$

Let $\Pi_1 = \dfrac{\textsc{that} : (np \to s) \to np \to cc \vdash \textsc{that} : (np \to s) \to np \to cc \quad \textsc{left} : np \to s \vdash \textsc{left} : np \to s}{\textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s \vdash \textsc{that left} : np \to cc}$

Let $\Pi_2 = \dfrac{\overset{\vdots\,\Pi_1}{\textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s \vdash \textsc{that left} : np \to cc} \quad \textsc{someone} : np \vdash \textsc{someone} : np}{\textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s, \textsc{someone} : np \vdash \textsc{that left someone} : cc}$

Π_2 *proves that* THAT LEFT SOMEONE *is correctly typed of type cc.*

Let $\Pi_3 = \dfrac{\textsc{think} : cc \to np \to s \vdash \textsc{think} : cc \to np \to s \quad \overset{\vdots\,\Pi_2}{\textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s, \textsc{someone} : np \vdash \textsc{that left someone} : cc}}{\textsc{think} : cc \to np \to s, \textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s, \textsc{someone} : np \vdash \textsc{think}\ (\textsc{that left someone}) : np \to s}$

The following derivation using Π_3 *proves that* THINK (THAT LEFT SOMEONE) ALICE *is well typed of type s*

$$\dfrac{\overset{\vdots\,\Pi_3}{\textsc{think} : cc \to np \to s, \textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s, \textsc{someone} : np \vdash \textsc{think}\ (\textsc{that left someone}) : np \to s} \quad \textsc{Alice} : np \vdash \textsc{Alice} : np}{\textsc{think} : cc \to np \to s, \textsc{that} : (np \to s) \to np \to cc, \textsc{left} : np \to s, \textsc{someone} : np, \textsc{Alice} : np \vdash \textsc{think}\ (\textsc{that left someone})\ \textsc{Alice} : s}$$

2. The syntactic category assigned to the complement clause *that someone left* is *cc*.

(a) Express the type of $[\![\textsc{think}]\!]$ in terms of $[\![cc]\!]$, $[\![np]\!]$, and $[\![s]\!]$.

According to the homomorphism requirement between syntactic types and semantics types the type of THINK *is:*

$$[\![cc \to np \to s]\!] = [\![cc]\!] \to [\![np]\!] \to [\![s]\!]$$

(b) According to the present hypothesis (given in the above mentioned lexicon), what are the semantic categories (semantic types) corresponding to *np*, *s*, and *cc*? Justify your answers for each of these types.

- *Because* ***alice*** *is of type* e *and* ALICE *is of type np,* $[\![np]\!] = (e \to \alpha) \to \alpha$. *Moreover, this is also the type of* $[\![\textsc{someone}]\!]$ *hence* α *is the type of* $\exists x.P\ x$ *which is* t*. So* $[\![np]\!] = (e \to t) \to t$.
- *Then because the type of* $[\![\textsc{left}]\!] = \lambda S.S(\textbf{\textit{left}})$[1] *is* $[\![np]\!] \to [\![s]\!]$ *and we know* S *in this term is of type* $(e \to t) \to t$*, we have that* $S(\textbf{\textit{left}}) : t = [\![s]\!]$.
- *Looking at* $[\![\textsc{think}]\!]$*,* C *is of type* $[\![cc]\!] = (\beta \to \beta) \to t$ *(because* $C(\lambda t.t)$ *is the second argument of* ***think****, hence of type* t*, and because the argument of* C *is the identity).* $[\![cc]\!]$ *is also the type of* $\lambda r.s\ (\lambda x.P\ (\lambda q.r\ (q\ x)))$ *in* $[\![\textsc{that}]\!]$*. But* P *here is of type* $[\![np]\!] \to [\![s]\!]$*, so* $\lambda q.r\ (q\ x)$ *is of type* $[\![np]\!] = (e \to t) \to t$ *so* q *is of type* $e \to t$ *and* $r\ (q\ x)$ *of type* t*. So* r *is of type* $t \to t$ *and* $\lambda r.s\ (\lambda x.P\ (\lambda q.r\ (q\ x)))$ *is of type* $(t \to t) \to t = [\![cc]\!]$.

3. Compute the semantic representation m of the sentence *Alice thinks that someone left*, whose abstract syntax is given by the term t.

$$\begin{aligned}[\![t]\!] &= [\![\textsc{think}\ (\textsc{that}\ \textsc{left}\ \textsc{someone})\ \textsc{alice}]\!]\\ &= [\![\textsc{think}]\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!])\ [\![\textsc{alice}]\!]\end{aligned}$$

$$\begin{aligned}[\![\textsc{that}]\!]\ [\![\textsc{left}]\!][\![\textsc{someone}]\!] &= (\lambda P.\lambda s.\lambda r.s\ (\lambda x.P(\lambda q.r\ (q\ x))))\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!]\\ &= \lambda r.[\![\textsc{someone}]\!](\lambda x.[\![\textsc{left}]\!](\lambda q.r\ (q\ x)))\\ &= \lambda r.[\![\textsc{someone}]\!](\lambda x.(\lambda S.S(\textbf{\textit{left}}))\ (\lambda q.r\ (q\ x)))\\ &= \lambda r.[\![\textsc{someone}]\!](\lambda x.(\lambda q.r\ (q\ x))\ (\textbf{\textit{left}}))\\ &= \lambda r.[\![\textsc{someone}]\!](\lambda x.r\ (\textbf{\textit{left}}\ x))\\ &= \lambda r.(\lambda P.\exists x.P\ x)(\lambda x.r\ (\textbf{\textit{left}}\ x))\\ &= \lambda r.\exists x.(\lambda x.r\ (\textbf{\textit{left}}\ x))\ x\\ &= \lambda r.\exists x.r\ (\textbf{\textit{left}}\ x)\end{aligned}$$

$$\begin{aligned}[\![\textsc{think}]\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!]) &= (\lambda C.\lambda s.s(\lambda x.\textbf{\textit{think}}\ x\ (C\ (\lambda t.t))))(\lambda r.\exists x.r\ (\textbf{\textit{left}}\ x))\\ &= \lambda s.s(\lambda x.\textbf{\textit{think}}\ x\ ((\lambda r.\exists x.r\ (\textbf{\textit{left}}\ x))\ (\lambda t.t)))\\ &= \lambda s.s(\lambda x.\textbf{\textit{think}}\ x\ (\exists x.(\lambda t.t)\ (\textbf{\textit{left}}\ x)))\\ &= \lambda s.s(\lambda x.\textbf{\textit{think}}\ x\ (\exists x.\textbf{\textit{left}}\ x))\end{aligned}$$

[1] Remember that $\lambda x.\textbf{left}\ x$ is η-equivalent to **left**.

$$\begin{aligned}
[\![\textsc{think}]\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!])\ [\![\textsc{Alice}]\!] &= (\lambda s.s(\lambda x.\textbf{\textit{think}}\ x\ (\exists x.\textbf{\textit{left}}\ x)))(\lambda P.P\ \textbf{\textit{alice}})\\
&= (\lambda P.P\ \textbf{\textit{alice}})(\lambda x.\textbf{\textit{think}}\ x\ (\exists x.\textbf{\textit{left}}\ x))\\
&= (\lambda x.\textbf{\textit{think}}\ x\ (\exists x.\textbf{\textit{left}}\ x))\ \textbf{\textit{alice}}\\
&= \textbf{\textit{think alice}}\ (\exists x.\textbf{\textit{left}}\ x)
\end{aligned}$$

(a) According to m, does Alice necessarily have an idea of who left?
No. She could have some clue that someone left without knowing precisely who is that person.

(b) Given to expressions s_1 and s_2, when do we say that s_1 intuitively entails s_2?
s_1 intuitively entails s_2 if and only if for all models M in T $[\![S_1]\!]^M \leq [\![S_2]\!]^M$.

4. Give a semantic recipe to THINK such that the semantic reading associated to *Alice thinks that someone left* is now *There's a precise person such that Alice thinks this person left.*

$$[\![\textsc{think}']\!] = \lambda C.\lambda s.s(\lambda x.C(\lambda t.\textbf{\textit{think}}\ x\ t))$$

is such that

$$[\![\textsc{think}']\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!])\ [\![\textsc{Alice}]\!] = \exists x.\textbf{\textit{think alice}}\ (\textbf{\textit{left}}\ x)$$

because

$$\begin{aligned}
[\![\textsc{think}']\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!]) &= (\lambda C.\lambda s.s(\lambda x.C(\lambda t.\textbf{\textit{think}}\ x\ t)))(\lambda r.\exists x.r\ (\textbf{\textit{left}}\ x))\\
&= \lambda s.s(\lambda x.(\lambda r.\exists x.r\ (\textbf{\textit{left}}\ x))(\lambda t.\textbf{\textit{think}}\ x\ t))\\
&= \lambda s.s(\lambda y.(\lambda r.\exists x.r\ (\textbf{\textit{left}}\ x))(\lambda t.\textbf{\textit{think}}\ y\ t))\\
&= \lambda s.s(\lambda y.\exists x.(\lambda t.\textbf{\textit{think}}\ y\ t)\ (\textbf{\textit{left}}\ x))\\
&= \lambda s.s(\lambda y.\exists x.\textbf{\textit{think}}\ y\ (\textbf{\textit{left}}\ x))
\end{aligned}$$

Hence

$$\begin{aligned}
[\![\textsc{think}]\!]\ ([\![\textsc{that}]\!]\ [\![\textsc{left}]\!]\ [\![\textsc{someone}]\!])\ [\![\textsc{Alice}]\!] &= (\lambda s.s(\lambda y.\exists x.\textbf{\textit{think}}\ y\ (\textbf{\textit{left}}\ x)))(\lambda P.P\ \textbf{\textit{alice}})\\
&= (\lambda P.P\ \textbf{\textit{alice}})(\lambda y.\exists x.\textbf{\textit{think}}\ y\ (\textbf{\textit{left}}\ x)))\\
&= (\lambda y.\exists x.\textbf{\textit{think}}\ y\ (\textbf{\textit{left}}\ x))\ \textbf{\textit{alice}}\\
&= \exists x.\textbf{\textit{think alice}}\ (\textbf{\textit{left}}\ x))
\end{aligned}$$

5. Check that $[\![\textsc{that}]\!]$ is well typed.

It amounts to check that $[\![\textsc{that}]\!]$ is of type

$$\begin{aligned}
&[\![(np \to s) \to (np \to cc)]\!]\\
=\ &[\![(np \to s) \to (np \to cc)]\!]\\
=\ &([\![np]\!] \to [\![s]\!]) \to [\![np]\!] \to [\![cc]\!]\\
=\ &(((e \to t) \to t) \to t) \to ((e \to t) \to t) \to (t \to t) \to t
\end{aligned}$$

$$\textit{Let } \Pi = \dfrac{\dfrac{r : t \to t \vdash r : t \to t \qquad \dfrac{q : e \to t \vdash q : e \to t \qquad x : e \vdash x : e}{q : e \to t, x : e \vdash q\ x : t}}{r : t \to t, q : e \to t, x : e \vdash r\ (q\ x) : t}}{r : t \to t, x : e \vdash \lambda q.r\ (q\ x) : (e \to t) \to t}$$

Then the following derivation proves that $[\![\text{THAT}]\!]$ is well typed.

$$\dfrac{\dfrac{\dfrac{s : [\![np]\!] \vdash s : [\![np]\!] \qquad \dfrac{\dfrac{P : [\![np]\!] \to [\![s]\!] \vdash P : [\![np]\!] \to [\![s]\!] \qquad \begin{array}{c}\vdots\,\Pi \\ r : t \to t, x : e \vdash \lambda q.r\ (q\ x) : [\![np]\!]\end{array}}{P : [\![np]\!] \to [\![s]\!], r : t \to t, x : e \vdash P(\lambda q.r\ (q\ x)) : t}}{P : [\![np]\!] \to [\![s]\!], r : t \to t \vdash \lambda x.P(\lambda q.r\ (q\ x)) : e \to t}}{s : [\![np]\!], P : [\![np]\!] \to [\![s]\!], r : t \to t \vdash s(\lambda x.P(\lambda q.r\ (q\ x))) : t}}{s : [\![np]\!], P : [\![np]\!] \to [\![s]\!] \vdash \lambda r.s(\lambda x.P(\lambda q.r\ (q\ x))) : (t \to t) \to t}}{\dfrac{P : [\![np]\!] \to [\![s]\!] \vdash \lambda s\ r.s(\lambda x.P(\lambda q.r\ (q\ x))) : [\![np]\!] \to (t \to t) \to t}{\vdash \lambda P\ s\ r.s(\lambda x.P(\lambda q.r\ (q\ x))) : ([\![np]\!] \to [\![s]\!]) \to [\![np]\!] \to (t \to t) \to t}}$$