Encoding glue premises in the f-structure
Glue meeting, November 2019

1 Representing glue premises

Proper name:

(1) a. *David*: d_e

b. $$d : \left[\begin{array}{ll} \text{PRED} & \text{David} \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{REL} & \text{David} \\ \text{SEMSTR} & d \\ \text{TYPE} & e \end{array}\right] \right\} \end{array}\right]$$

Intransitive verb:

(2) a. *yawn*: $d_e \multimap y_t$

b. $$y : \left[\begin{array}{ll} \text{PRED} & \text{`YAWN'<SUBJ>'} \\ \text{SUBJ} & d : [\ ] \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{REL} & \text{YAWN} \\ \text{ARG1} & \left[\begin{array}{ll} \text{SEMSTR} & d \\ \text{TYPE} & e \end{array}\right] \\ \text{SEMSTR} & y \\ \text{TYPE} & t \end{array}\right] \right\} \end{array}\right]$$

Transitive verb:

(3) a. *see*: $d_e \multimap (c_e \multimap s_t)$

b. $$s : \left[\begin{array}{ll} \text{PRED} & \text{`SEE<SUBJ,OBJ>'} \\ \text{SUBJ} & d : [\ ] \\ \text{OBJ} & c : [\ ] \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{REL} & \text{SEE} \\ \text{ARG1} & \left[\begin{array}{ll} \text{SEMSTR} & d \\ \text{TYPE} & e \end{array}\right] \\ \text{ARG2} & \left[\begin{array}{ll} \text{SEMSTR} & c \\ \text{TYPE} & e \end{array}\right] \\ \text{SEMSTR} & s \\ \text{TYPE} & t \end{array}\right] \right\} \end{array}\right]$$

Common noun:

(4) a. *man*: $p_e \multimap p_t$

b. $$m : \left[\begin{array}{ll} \text{PRED} & p : \text{‘MAN’} \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{REL} & \text{MAN} \\ \text{ARG1} & \left[\begin{array}{ll} \text{SEMSTR} & p \\ \text{TYPE} & e \end{array}\right] \\ \text{SEMSTR} & p \\ \text{TYPE} & t \end{array}\right] \right\} \end{array}\right]$$

Determiner:

(5) a. *a*: $\forall F.(p_e \multimap p_t) \multimap (m_e \multimap F_t) \multimap F_t$

b. $$m : \left[\begin{array}{ll} \text{PRED} & p \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{REL} & a \\ \text{ARG1} & \left[\begin{array}{ll} \text{ARG1} & \left[\begin{array}{ll} \text{SEMSTR} & p \\ \text{TYPE} & e \end{array}\right] \\ \text{SEMSTR} & p \\ \text{TYPE} & t \end{array}\right] \\ \text{ARG2} & \left[\begin{array}{ll} \text{ARG1} & \left[\begin{array}{ll} \text{SEMSTR} & m \\ \text{TYPE} & e \end{array}\right] \\ \text{SEMSTR} & F \\ \text{TYPE} & t \end{array}\right] \\ \text{FORALL} & F \\ \text{SEMSTR} & F \\ \text{TYPE} & t \end{array}\right] \right\} \end{array}\right]$$

NB: For some reason the “FORALL” attribute does not show up in the f-structure display, but it is there in the Prolog file.

Modifier, 2 premises:

(6) a. *interesting*: $(ip_e \multimap i_t)$,
$\lambda P.\lambda Q.\lambda x.and(P(x),Q(x)) : (ip_e \multimap i_t) \multimap (p_e \multimap p_t) \multimap (p_e \multimap p_t)$

b.
$$m : \left[\begin{array}{ll}
\text{PRED} & p \\
\text{ADJ} & \left\{ i : \left[\begin{array}{ll} \text{PRED} & ip\text{: `INTERESTING'} \end{array}\right] \right\} \\
\text{GLUE} & \left\{ \begin{array}{l}
\left[\begin{array}{ll}
\text{ARG1} & \left[\begin{array}{ll}\text{SEMSTR} & ip \\ \text{TYPE} & e\end{array}\right] \\
\text{REL} & \text{INTERESTING} \\
\text{SEMSTR} & i \\
\text{TYPE} & t
\end{array}\right] \\
\left[\begin{array}{ll}
\text{ARG1} & \left[\begin{array}{ll} \text{ARG1} & \left[\begin{array}{ll}\text{SEMSTR} & ip \\ \text{TYPE} & e\end{array}\right] \\ \text{SEMSTR} & i \\ \text{TYPE} & t \end{array}\right] \\
\text{ARG2} & \left[\begin{array}{ll} \text{ARG1} & \left[\begin{array}{ll}\text{SEMSTR} & p \\ \text{TYPE} & e\end{array}\right] \\ \text{SEMSTR} & p \\ \text{TYPE} & t \end{array}\right] \\
\text{ARG3} & \left[\begin{array}{ll}\text{SEMSTR} & p \\ \text{TYPE} & e\end{array}\right] \\
\text{REL} & \text{/P./Q./x.AND(P(x),Q(x))} \\
\text{SEMSTR} & p \\
\text{TYPE} & t
\end{array}\right]
\end{array} \right\}
\end{array}\right]$$

2 The premise rewriting component

The component that transfers f-structures to premises operates as follows:

- Read in the Prolog representation of the parsed sentence. Throw away the c-structure, and unpack the f-structure.
- For each f-structure, gather up the values of the GLUE attributes – these are the f-structure encodings of the glue premises. The value of the GLUE attribute can be a set of premises (this has been tested) or a single premise (this has not been tested). Throw the rest of the f-structure away.
- Transfer each f-structure premise to the prover format.
- Collect up the output premise sets for each f-structure, and get rid of duplicate premise sets.
- Pass all of the premise sets to the prover.

3 F-structure attributes and values in premises

Format for f-structure premises:

- Each premise is expected to have three attributes:
  - REL, whose value is the meaning side of the premise.
  - SEMSTR, whose value is an f-structure (standing in for the semantic structure)
  - TYPE, the type of SEMSTR (generally e or t, though this is not checked by the transfer component)
- Premises can also have arguments. Each argument should contain a SEMSTR and a TYPE. It can also contain argument attributes and values, for embedded implications.
- Attribute names for arguments: I have used ARG1-ARGn in the sample grammar. Actually, the transfer component does not care what the names are. It assumes that there are 4 special attributes (REL, SEMSTR, TYPE, and FORALL), and any other attributes are expected to encode arguments, with alphabetical order determining the order of arguments. So ARG1-ARGn should work, or A1-An, or A,B,C,... (but this has not been tested).
- FORALL is a special attribute with an f-structure value. Assume that its value is the f-structure that is labeled 18 in the Prolog representation. Then the resulting glue formula will have a universal quantifier binding f-structure 18 added at the beginning (`AF18`) and all occurrences of 18 in the rest of the formula will be prefixed by F, indicating that it is a variable. The value of FORALL can also be a set of f-structures, and then each of the members of the set will be quantified at the beginning of the formula, and occurrences of each member of the set in the body of the formula will be prefixed by F.

4 Input format to prover

The prover accepts premises that look like this, with meanings like `/u.dog(u)`:

```
/u.dog(u) : (g_e -o g_t)
/v.bone(v) : (h_e -o h_t)
/P./Q./z.every(z,P(z),Q(z)) : ((g_e -o g_t) -o AX_t.(i_e -o X_t) -o X_t)
/R./S./z.a(z,P(z),Q(z)) : ((h_e -o h_t) -o AY_t.(j_e -o Y_t) -o Y_t)
/x./y.eat(x,y) : (i_e -o (j_e -o f_t))
```

Characters like the forward slash/lambda, comma, etc. in values of REL in the grammar must be escaped with backquote:

```
(^ REL) = `/P`.`/Q`.`/x`.and`(P`(x`)`,Q`(x`)`)
```