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

1 Representing glue premises

Proper name:

(1) a. $David: d_e$ b. $d:\begin{bmatrix} PRED & DAVID \\ GLUE & \left\{\begin{bmatrix} REL & DAVID \\ SEMSTR & d \\ TYPE & e \end{bmatrix}\right\} \end{bmatrix}$

Intransitive verb:

 $(2) \quad \text{a.} \quad yawn: \ d_e \multimap y_t$ $\text{b.} \quad y: \begin{bmatrix} \text{PRED 'YAWN'<SUBJ>'} \\ \text{SUBJ } d:[\] \\ \\ \text{GLUE } \left\{ \begin{bmatrix} \text{REL YAWN} \\ \text{ARG1 } \begin{bmatrix} \text{SEMSTR } d \\ \text{TYPE } e \end{bmatrix} \right] \\ \\ \text{SEMSTR } y \\ \\ \text{TYPE } t \end{bmatrix}$

Transitive verb:

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

$$\begin{bmatrix} \text{PRED 'SEE} < \text{SUBJ,OBJ} > ' \\ \text{SUBJ } d : [\] \\ \text{OBJ } c : [\] \end{bmatrix}$$
b. $s: \begin{bmatrix} \text{REL SEE} \\ \text{ARG1} \begin{bmatrix} \text{SEMSTR } d \\ \text{TYPE } e \end{bmatrix} \\ \text{ARG2} \begin{bmatrix} \text{SEMSTR } c \\ \text{TYPE } e \end{bmatrix} \end{bmatrix}$

$$\begin{bmatrix} \text{SEMSTR } s \\ \text{TYPE } t \end{bmatrix}$$

Common noun:

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

b.
$$m: \begin{bmatrix} \text{PRED} & p: \text{`MAN'} \\ \\ \text{GLUE} & \begin{bmatrix} \text{REL MAN} \\ \\ \text{ARG1} & \begin{bmatrix} \text{SEMSTR} & p \\ \\ \text{TYPE} & e \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Determiner:

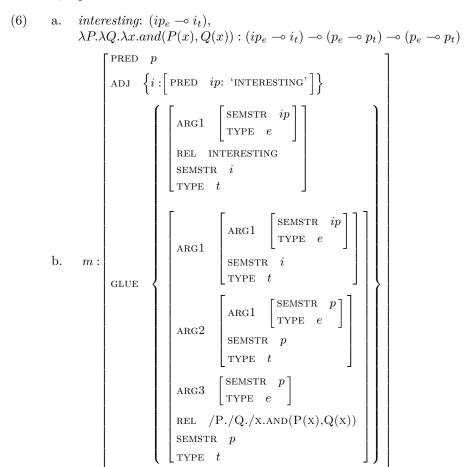
(5) a. a:
$$\forall F.(p_e \multimap p_t) \multimap (m_e \multimap F_t) \multimap F_t$$

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

$$\begin{bmatrix} & & & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\$$

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:



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.

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')')
```