**Practical Grammar Engineering using HPSG**
Exercises day5: Valence

**Goals:**

1. Learn about argument frames.
2. Introduce transitive and ditransitive verbs.
3. Introduce the OPT mechanism.
4. Introduce infinitival complements and subordinate clause complements.

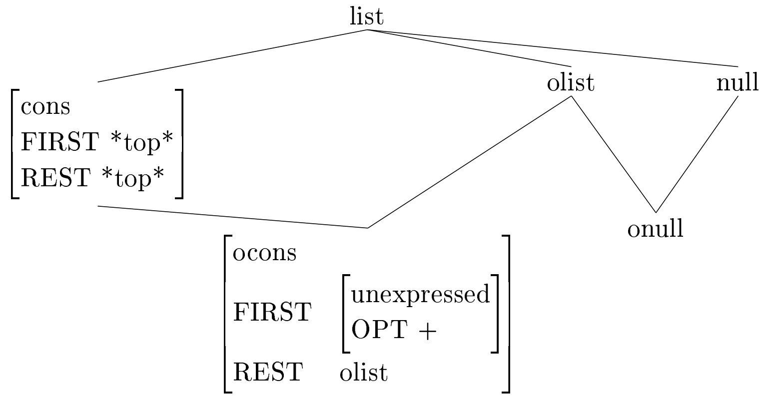
**Exercises:**

1. Introduce transitive and ditransitive verbs.
   (a) If you don't have a lexeme type for transitive verbs in your grammar yet, introduce a type *trans-verb-lexeme* which has *arg2-comps-lex-item* as one of its supertypes, and which has one element of type *phr-synsem* with head value *noun* on the COMPS list.
   (b) Introduce a lex-item type that links the ARG3 of the key relation to the first complement, and another type that links the ARG2 to the second complement. Create a lexeme type *ditrans-verb-lexeme* which inherits from all the relevant types. Try to make the type link the subject to ARG1, the direkt object to ARG2 and the indirect object to ARG3. The COMPS list should be constrained to contain two NPs.
   (c) Parse a couple of sentences containing transitive and ditransitive verbs. Check that the generator works. If the generator is a bit slow, you may have a look at the generator chart by selecting 'options'-'expand menu' and then 'generate'-'show gen chart'. In the genarator chart you may click on nodes you suspect shouldn't be there, and then select the tree number. Then a tree structure pops up, which is generated by your grammar. If you find a tree structure you don't want, you may consider putting more constraints on your lexical entries and/or rules.

2. In order to reduce the amount of lexical entries for verbs with many argument frames, the ERG makes use of a feature OPT *bool*. This feature is introduced by the type *synsem-min*, which means that it is present on all kinds of synsems. The elements on a valence list that are optional are marked as OPT +, while the elements that are not optional are marked as OPT -. So a lexical entry may look like the one below:

```
email_v := trans-verb-lxm &
  [ STEM < ``email'',
    SYNSEM [ LOCAL.CAT.VAL [ SUBJ < [ OPT - ] >,
                             COMPS < [ OPT + ] > ],
             LKEYS.KEYREL.PRED ``email_rel'' ] ].
```

   Here the subject must be realized, while the object is optional. You may have noticed the type *olist* earlier, as a value of SUBJ, COMPS, and SPR (see *basic-head-subj-phrase* in 'matrix.tdl'). These lists are lists which either are empty or which contain optional synsems. If you study the (partial) hierarchy below, you may get an idea of how it works:




Here you see that the type *olist* has two subtypes, namely *ocons* and *onull*. *ocons* has an optional element as its first element, and another *olist* as value of REST. In this way there can be more than one optional element in an *olist*. The *onull* is an empty list.

(a) Enter a couple of transitive verbs into your lexicon and mark them with regard to optionality of the complements. One of the verbs should have an optional complement and the other an obligatory complement. Also remember to update the verb lexemes that you already have with regard to optionality.

(b) You can now remove the *phr-synsem* constraint on the complements of types like *trans-verb-lxm* and *ditrans-verb-lxm*, and see whether the OPT feature works as it is supposed to.

3. The machanisms we have so far works fine as long as there are no obligatory complements following the optional complements like in "The dog showed the cat the letter", where "the cat" is optional, and "the letter" is obligatory. In order to parse the sentence "The dog showed the letter", we need a rule which removes the optional complement before the head complement rule can work on the last complement.

   (a) Introduce a type for verbs like "show", where the ARG2 is linked to an obligatory NP and ARG3 is linked to an optional NP. Enter one such verb in the lexicon. The linking and optionality constraints should look like this:

$$\begin{bmatrix} \text{ditrans-verb-lxm} \\ \text{SS} \begin{bmatrix} \text{LOC}\,|\,\text{CAT}\,|\,\text{VAL} \begin{bmatrix} \text{SUBJ} \left\langle \begin{bmatrix} \text{OPT -} \\ \text{LOCAL}\,|\,\text{CONT}\,|\,\text{HOOK}\,|\,\text{INDEX}\ \boxed{0} \end{bmatrix} \right\rangle \\ \text{COMPS} \left\langle \begin{bmatrix} \text{OPT +} \\ \text{LOCAL ... INDEX}\ \boxed{1} \end{bmatrix}, \begin{bmatrix} \text{OPT -} \\ \text{LOCAL ... INDEX}\ \boxed{2} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{LKEYS}\,|\,\text{KEYREL} \begin{bmatrix} \text{ARG1}\ \boxed{0} \\ \text{ARG2}\ \boxed{2} \\ \text{ARG3}\ \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

   (b) Introduce a new rule which inherits from *basic-head-opt-one-comp* in 'matrix.tdl'. Check that you can parse sentences both with and without the optional complement. Also use the generator to confirm that your grammar is in good shape. You may want to change some constraints in:

      i. *mod-lex-item* (SYNSEM.LOCAL.CAT.HEAD.MOD.FIRST.LOCAL.CAT.VAL.COMPS *olist*) and
      ii. *basic-comp* (SYNSEM.LOCAL.CAT.HEAD.VAL.COMPS.FIRST.LOCAL.CAT.VAL.COMPS *olist*)

4. Infinitival complements.

Figure 1: Lexical entry for the infinitival complementizer

$$
\begin{bmatrix}
\textbf{inf-comp} & \\
\text{SYNSEM}\,|\,\text{LOCAL} &
\begin{bmatrix}
\text{CAT} &
\begin{bmatrix}
\text{HEAD comp} & \\
\text{VAL} &
\begin{bmatrix}
\text{COMPS} & \left\langle
\begin{bmatrix}
\text{LOCAL} &
\begin{bmatrix}
\text{CAT} &
\begin{bmatrix}
\text{HEAD verb,} & \\
\text{VAL} &
\begin{bmatrix}
\text{SUBJ} \left\langle \text{synsem} \right\rangle \\
\text{COMPS} \left\langle \right\rangle
\end{bmatrix}
\end{bmatrix} \\
\text{CONT}\,|\,\text{HOOK} &
\begin{bmatrix}
\text{LTOP } \boxed{0} \\
\text{INDEX}\,|\,\text{E}\,|\,\text{TENSE infin} \\
\text{XARG } \boxed{1}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\right\rangle
\end{bmatrix}
\end{bmatrix} \\
\text{CONT} &
\begin{bmatrix}
\text{HOOK} &
\begin{bmatrix}
\text{LTOP} & \boxed{2} \\
\text{XARG} & \boxed{1}
\end{bmatrix} \\
\text{RELS} & \left\langle !
\begin{bmatrix}
\text{message} \\
\text{PRED “proposition\_m\_rel”} \\
\text{LBL } \boxed{2} \\
\text{MARG } \boxed{3}
\end{bmatrix}
! \right\rangle \\
\text{HCONS} & \left\langle !
\begin{bmatrix}
\text{HARG} & \boxed{3} \\
\text{LARG} & \boxed{0}
\end{bmatrix}
! \right\rangle
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

(a) If your language has infinitival complementizers, introduce a type *inf-comp* under *word* which takes an infinitival VP as its complement. You will need to introduce tense types. Let the HEAD value be of type *comp*. Link the XARG of the complement to its own XARG. Insert a message-relation of type *message* with PRED value *proposition_m_rel* on the RELS list and a qeq on the HCONS list. Let the MARG of the message-relation qeq the LTOP of the complement, and link the handle of the message to the LTOP of the complementizer.

(b) Introduce a lexical entry for an infinitival complementizer. Check that it has the constraints shown in figure 1.

(c) If your language doesn't have complementizers, consider how the relation added by the word above can be inserted by using C-CONT in a unary rule.

(d) Introduce a linking type *subj-equi-lex-item* for verbs like English “try”, which links the index of the subject to the XARG of the infinitival clause complement. Make it link its ARG2 to the LTOP of its complement. The type should have at least the constraints in figure 2.

(e) Before you try to parse a sentence, constrain all verbs to link the index of their subject to their XARG.

(f) Now you can parse a sentence like “The dog likes to bark”. You can also try to parse the sentence “The dog likes to like to bark”. Use the generator to control that your grammar doesn't overgenerate.

(g) Add a new lex-item type for object equi verbs. Make your grammar parse sentences like “The dog allows the cat to drink the milk”.

5. Introduce subordinate clause complements.

   (a) Introduce a type *subord-comp* for complementizers that introduce subordinate clauses. It will have some similarities with the type *inf-comp*. The differences between them are:

      i. In *subord-comp* the complement has *olist* as value of SUBJ instead of a list with a *synsem*.

Figure 2: Subject equi linking type

$$
\begin{bmatrix}
\textbf{subj-equi-lex-item} & \\
\text{SYNSEM} & \begin{bmatrix}
\text{LOCAL} \mid \text{CAT} & \begin{bmatrix}
\text{HEAD verb} & \\
\text{VAL} & \begin{bmatrix}
\text{SUBJ} & \left\langle \begin{bmatrix}
\text{phr-synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\text{CAT} \mid \text{HEAD noun} \\
\text{CONT} \mid \text{HOOK} \mid \text{INDEX } \boxed{1}
\end{bmatrix}
\end{bmatrix} \right\rangle \\
\text{COMPS} & \left\langle \begin{bmatrix}
\text{phr-synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\text{CAT} \mid \text{HEAD comp} & \\
\text{CONT} \mid \text{HOOK} & \begin{bmatrix}
\text{LTOP } \boxed{0} \\
\text{XARG } \boxed{1}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \right\rangle
\end{bmatrix}
\end{bmatrix} \\
\text{LKEYS} \mid \text{KEYREL} & \begin{bmatrix}
\text{ARG1 } \boxed{1} \\
\text{ARG2 } \boxed{0}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

ii. In *subord-comp* the tense of the complement is *finite* instead of *infin*. (You may have to enrich the hierarchy below *tense*).

iii. The *subord-comp* has empty RELS and HCONS lists, and it links its LTOP to the LTOP of the complement.

Introduce a type *basic-compl* that has the constraints the two complementizer types have in common, and let it be their supertype. Enter a lexical entry of a complementizer for subordinate clauses into your grammar.

(b) Build a type *trans-subord-verb-lxm* for verbs like "know" in "The cat knows that the dog barks". Link the LTOP of the complement subordinate clause to the ARG2 of the verb relation. Introduce a lexical entry for a word like "know". Try to parse a sentence with your new verb. Make sure that you don't parse "*The dog knows to bark". One way to prevent this is to link the index of the complementizers to the index of their complements, and to make sure that the verbs only select complements with the right tense.

(c) Now you should be able to parse a long sentence like "The cat knows that the big dog likes to bark".