Practical Grammar Engineering using HPSG
Extra: Relative clauses

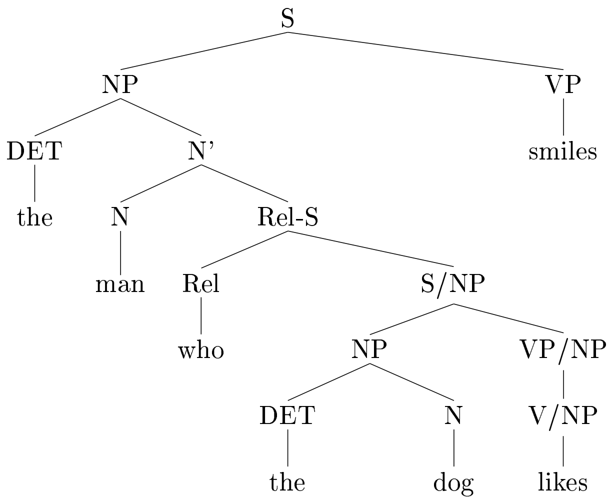
Goals:

1. Introduce relative clauses.
2. Learn about the SLASH list.

Exercises:

1. We will first look at how one can make the grammar parse sentences like "The dog likes the man who smiles" and "The man who the dog likes, smiles". In the first sentence, the noun modified by the relative clause shares reference with what the verb in the relative clause wants as its subject. In the second sentence, the modified noun shares reference with what the verb in the relative clause wants as its object. In order to account for these facts, we will use the SLASH mechanism to store extracted elements on a difference list. (The SLASH list is situated in NON-LOCAL.) The SLASH mechanism can be viewed as three operations. First, an extraction rule removes an element from a valence list and inserts the object's *local* into the SLASH list. Second, the SLASH feature is unified with all signs in the head projection. (See the *head-valence-phrase*.) Third, the element on the SLASH list is removed from the SLASH list and unified with the LOCAL value of the first daughter by a head filler rule. An analysis may look like this:

   Figure 1: The man who the dog likes, smiles

   In the tree above the "VP/NP" node is an *extracted-comps-phrase* and the "Rel-S" node is a *head-filler-phrase*.

   (a) Introduce the type *extracted-subj-phrase* and let it inherit from *basic-extracted-subj-phrase*. Introduce a message relation on the RELS list in C-CONT and a handle constraint on the HCONS list in C-CONT. Make the MARG of the introduced message qeq the LTOP of the daughter, and unify the handle of the message relation with the LTOP of the mother. Also unify the index of the daughter with the index of the mother.

   (b) Introduce the type *extracted-comp-phrase* as a subtype of *basic-extracted-comp-phrase*. Make sure that it doesn't introduce any relations or handle constraints in C-CONT. Introduce the rules for the new types.

(c) Introduce a type *rel-pron* under *word*. Constrain its synsem type to be *non-canonical*, the head value to be *noun* and the valence and CONT lists to be empty. Let your relative pronoun word inherit from this type.

(d) Introduce a type *rel-head-filler-phrase* under *basic-head-filler-phrase*. Constrain the first daughter to be a relative pronoun. Let the INDEX of the first daughter be the index of the element on the MOD list. Constrain the element on the MOD list to have the head value *noun* and link the LTOP of the element on the MOD list to the LTOP of the second daughter. Introduce the rule for this new phrase type. The expanded type of the new rule should contain the information in figure 2:

Figure 2: rel-head-filler-rule

$$
\begin{bmatrix}
\text{SYNSEM} & \begin{bmatrix}
\text{LOCAL}\,|\,\text{CAT} & \begin{bmatrix}
\text{HEAD}\,|\,\text{MOD} & \left\langle \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD noun} \\ \text{VAL}\,|\,\text{SPR} \left\langle \text{synsem} \right\rangle \end{bmatrix} \\ \text{CONT}\,|\,\text{HOOK} & \begin{bmatrix} \text{LTOP } \boxed{0} \\ \text{INDEX } \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle \\
\text{VAL} & \begin{bmatrix} \text{SUBJ } \langle\rangle \\ \text{SPR} \left\langle \text{anti-synsem} \right\rangle \\ \text{COMPS } \langle\rangle \end{bmatrix}
\end{bmatrix} \\
\text{NON-LOCAL}\,|\,\text{SLASH} \left\langle !\ ! \right\rangle &
\end{bmatrix} \\
\text{ARGS} & \left\langle \begin{bmatrix} \text{rel-pron} & \\ \text{SS} & \begin{bmatrix} \text{LOC } \boxed{2} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD noun} & \\ \text{VAL} & \begin{bmatrix} \text{SUBJ onull} \\ \text{SPR onull} \\ \text{COMPS onull} \end{bmatrix} \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{HOOK}\,|\,\text{INDEX } \boxed{1} \\ \text{RELS} \left\langle !\ ! \right\rangle \\ \text{HCONS} \left\langle !\ ! \right\rangle \end{bmatrix} \end{bmatrix} \\ \text{NON-LOCAL}\,|\,\text{SLASH} \left\langle !\ ! \right\rangle & \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD verb} & \\ \text{VAL} & \begin{bmatrix} \text{SUBJ olist} \\ \text{COMPS olist} \end{bmatrix} \end{bmatrix} \\ \text{CONT}\,|\,\text{HOOK}\,|\,\text{LTOP } \boxed{0} & \end{bmatrix} \\ \text{NON-LOCAL}\,|\,\text{SLASH} \left\langle !\ \boxed{2}\ ! \right\rangle & \end{bmatrix} \end{bmatrix} \right\rangle \\
\text{C-CONT} & \begin{bmatrix} \text{RELS} \left\langle !\ ! \right\rangle \\ \text{HCONS} \left\langle !\ ! \right\rangle \end{bmatrix}
\end{bmatrix}
$$

(e) Try to parse a sentence with a relative clause. If you are lucky, you have already introduced the modifier rule you need. You should decide whether you can use/modify an already existing modifier rule or whether you want to make a completely new one. You should also decide a proper HEAD value for the *rel-head-filler-phrase*. When you have made the sentence parse, you may find out that in *rel-head-filler-phrase* you want to constrain the SUBJ list in the second daughter and also the MOD list.

2. In addition to sentences where the the modified noun shares reference with what the verb in the relative clause wants as its subject or object, one can also think of sentences where the modified noun shares reference with an argument from a subordinate clause inside the relative clause and so on. Now we

will add some constraints that will make it possible to let the index of an extracted argument from a clause embedded in the relative clause be shared with the index of the noun that is modified by the relative clause. This is the case in a sentence like "The man who the cat claims that the dog admires, smiles". The way this is done in ERG is to let the lexemes having elements on their SUBJ and COMPS lists collect the slashes these elements may contain. A synsem that is extracted (*gap*) has its own local on its NON-LOCAL.SLASH list. Synsems that are realized in the canonical position (*canonical-synsem*) will have empty slash lists. This makes it possible to accumulate the slashes from the complements and the subjects with difference lists. There will never be more than one element on the SLASH list (see *non-local*). In order to build in this machinery, do the following:

(a) Introduce the following types into your grammar:

```
slash-subj := lex-item &
  [ SYNSEM [ LOCAL.CAT.VAL [ SUBJ < [ NON-LOCAL.SLASH #slash ] >,
                             COMPS <> ],
             NON-LOCAL.SLASH #slash ] ].

slash-1comp := lex-item &
  [ SYNSEM [ LOCAL.CAT.VAL [ SUBJ <>,
                             COMPS < [ NON-LOCAL.SLASH #slash ] > ],
             NON-LOCAL.SLASH #slash ] ].

slash-subj-1comp := lex-item &
  [ SYNSEM [ LOCAL.CAT.VAL [ SUBJ < [ NON-LOCAL.SLASH [ LIST #middle,
                                                        LAST #last ] ] >,
                             COMPS < [ NON-LOCAL.SLASH [ LIST #list,
                                                         LAST #middle ] ] > ],
             NON-LOCAL.SLASH [ LIST #list,
                               LAST #last ] ] ].

slash-subj-2comp := lex-item &
  [ SYNSEM [ LOCAL.CAT.VAL [ SUBJ < [ NON-LOCAL.SLASH [ LIST #middle2,
                                                        LAST #last ] ] >,
                             COMPS < [ NON-LOCAL.SLASH [ LIST #list,
                                                         LAST #middle1 ] ],
                                     [ NON-LOCAL.SLASH [ LIST #middle1,
                                                         LAST #middle2 ] ] > ],
             NON-LOCAL.SLASH [ LIST #list,
                               LAST #last ] ] ].
```

Make sure that your argument taking lexemes inherit from one of these types. Intransitive verbs should inherit from the *slash-subj* type. Transitive verbs should inherit from the *slash-subj-1comp* type, while verbs with two complements should inherit from the *slash-subj-2comp* type. Let your complementizers *inf-comp* and *subord-comp* inherit from the type *slash-1comp*. Also prepositions should inherit from this type.

(b) Parse a short sentence with a relative clause and try to generate from the MRS. Use the generated sentences and the generator chart to make your grammar more precise.

(c) Parse a sentence like "The man who the cat claims that the dog admires, smiles". Check that the semantic representation is correct.