Build-up of the *Global Grammar* feature structures

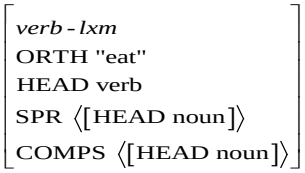
Lars Hellan, February 2015

In presenting this grammar, we start from the point of a very simple HPSG parser as outlined in the earlier parts of Sag et al. 2003, and implemented in the grammars *g1-g4* in Copestake 2002 (also known as the 'LKB ESSLLI grammars', from their use in ESSLLI courses around year 2000 and later used in classrooms for introduction to LKB). It functionally covers the kinds of operations described in section 5 of chapter 1, meaning that valence lists play a crucial role. It also has the lexeme-word distinction, simple inflectional operations suited for English phrased as 'lexeme-to-word' rules, and some 'lexeme-to-lexeme' operations for verbs covering frame alternations. We then add to this simple grammar, step by step, specifications for linking to semantic roles, for adding grammatical functions, for analyzing verbal extensions such as morphological causatives, applicatives and passives, and for analyzing multiverb constructions like serial verb constructions and con-verb constructions.

1. Resumé of simple HPSG grammar

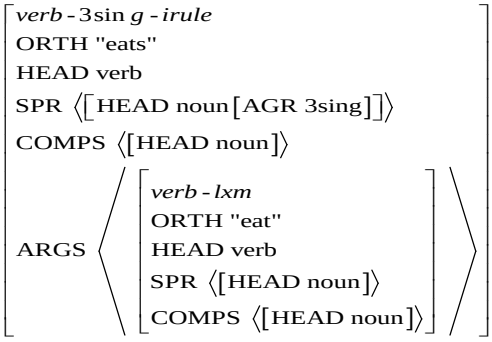
In this grammar, the AVM for a lexical entry of "eat" will be as follows, exemplifying the type *verb lexeme* (the type of the whole AVM is indicated in italics in the upper left corner):

(1)



This is the simplest structure encountered for a verbal sign. A more complex structure is encountered for *inflected verbs*, exemplified in (2):

(2)



This construct is analyzed as a *word* type item with a *verb lexeme* as daughter ('**ARGS**' stands for daughters), where the *word* item is of the subtype *verb-3sing-irule*, defined as indicated in the upper half of the AVM, more concisely as:

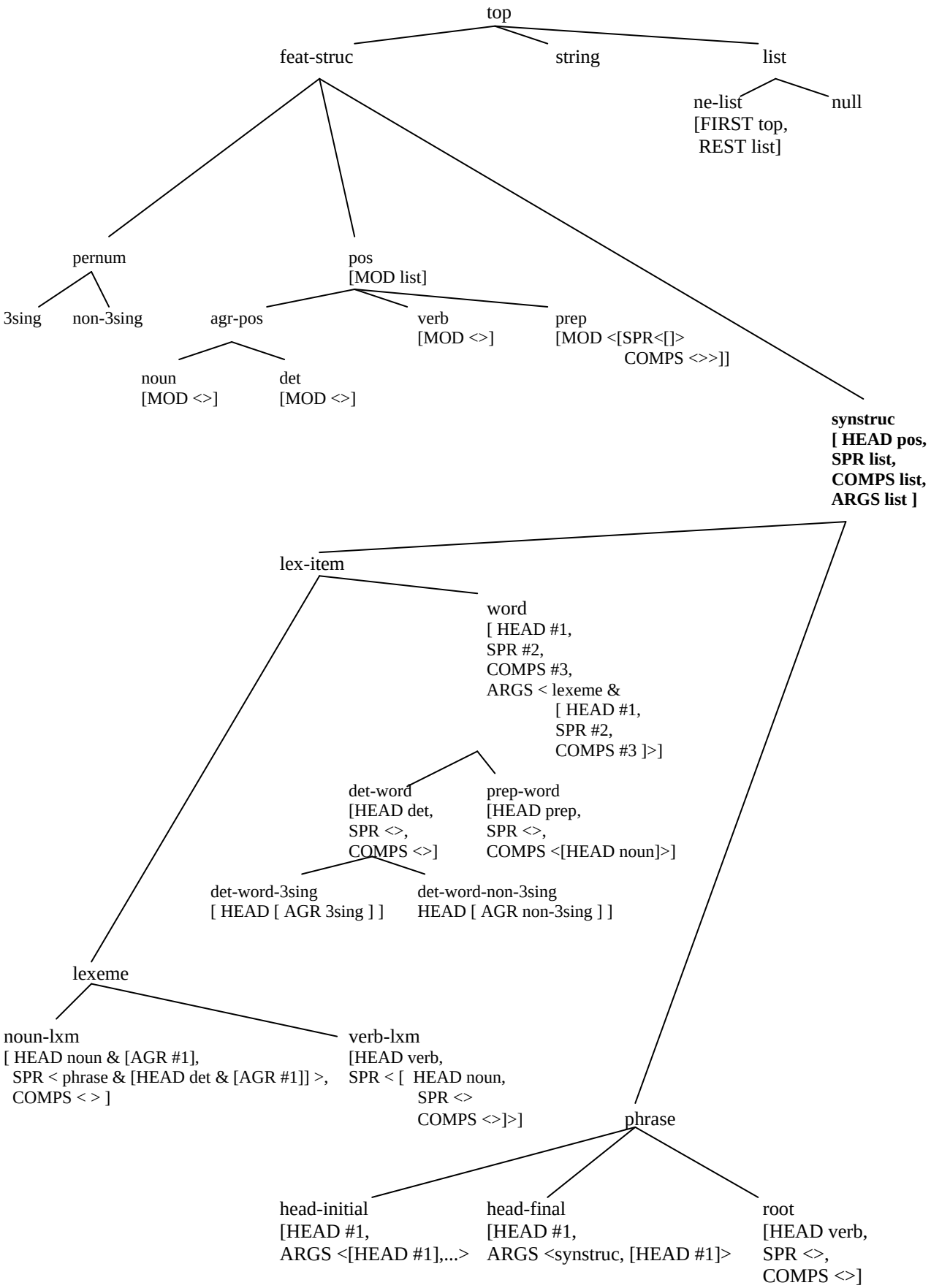
(3)

```
verb-3sing_irule :=
%suffix (!s !ss) (!ss !ssses) (ss sses)
word &
[ SPR < [ HEAD [ AGR 3sing ] ] >,
  ARGS < verb-lxm > ].
```

.

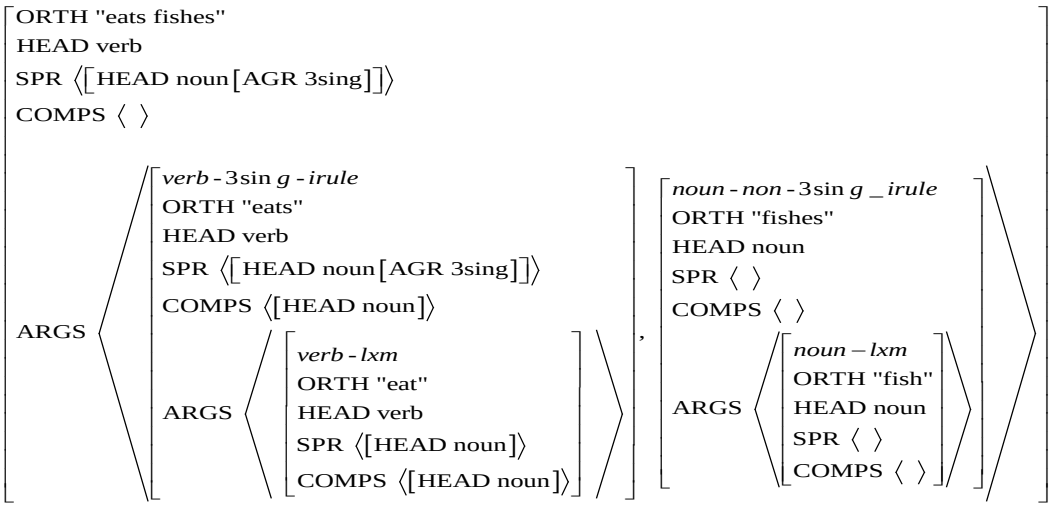
The definition of the type *word* can be seen in the overview of type definitions in Fig 1 below – as will be noted, the equal values of mother and daughter are represented by reentrancy. In this type hierarchy, it is the type *synstruc* (rendered in boldface) which declares the attributes used in the above AVMs.

Figure 1 Overview of types in the Simple HPSG grammar



A still more complex sign is that of a verbal *phrase*, exemplified by “eats fishes”, displayed in (4); here the two items combined by the phrasal rule are at *word* level – no phrasal combination applies to *lexemes* (but can of course apply to phrases).

(4)



The grammar has five phrasal rules, three of them rendered in their tdl [define] formulations:

(5) a. Combining V and Object (instantiated in (4)):

```
head-complement-rule-1 := head-initial &
[ SPR #spr,
  COMPS < >,
  ARGS < word &
       [ SPR #spr,
         COMPS < #cdtr & [ SPR < > ] > ],  #cdtr >  ].
```

b. Combining VP and Subject, and N and Specifier:

```
head-specifier-rule := head-final &
[ SPR < >,
  COMPS #comps,
  ARGS < phrase &
       #sdtr & [ SPR < > ],
       phrase &
       [ SPR < #sdtr >,
         COMPS #comps ] > ].
```

c. Combining a Modifier with its Head:

```
head-modifier-rule := head-initial &
[ SPR #spr,
  COMPS #comps,
  ARGS < phrase & #hdtr &
       [ SPR #spr,
         COMPS #comps ],
       phrase &
      [HEAD [MOD < #hdtr > ]] > ].
```

The lexicon of this grammar is as follows:

(6)

```
the := det-word &
[ ORTH "the" ].

that := det-word-3sing &
[ ORTH "that" ].

those := det-word-non-3sing &
[ ORTH "those" ].

aardvark := noun-lxm &
[ ORTH "aardvark" ].

dog := noun-lxm &
[ ORTH "dog" ].

cat := noun-lxm &
[ ORTH "cat" ].

bark := verb-lxm &
[ ORTH "bark",
  COMPS < > ].

chase := verb-lxm &
[ ORTH "chase",
  COMPS < phrase & [ HEAD noun ] > ].

give := verb-lxm &
[ ORTH "give",
  COMPS < phrase & [ HEAD noun ],
          phrase & [ HEAD noun ] > ].

to := prep-word &
[ ORTH "to" ].

near := prep-word &
[ ORTH "near" ].
```

Developments will improve the format of the lexical entries in:
- expressing valence in the lexical type rather than in a full AVM specification (so, saying that *chase* is 'transitive verb' rather than specifying it as 'COMPS < phrase & [HEAD noun] >')
- using entry identifiers beyond the orthographic form, to distinguish between homonyms and otherwise multiple entries involving the same form
- having a representation of the *meaning* of the lexical item.

2. Semantics

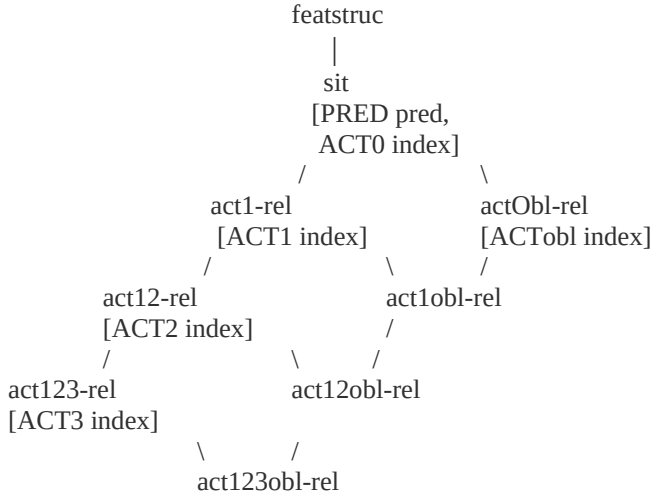
As will have been observed, the Simple grammar of section 1 includes no semantics. From the outline in chapter2, we envisage that at least part of a semantic specification should be introduced by a feature **ACTNTS**, for 'actants', taking as value the type *sit* (for 'situation'), which in turn declares an attribute **ACT0** for the 'index' of the situation, a number of 'participant' attributes **ACT1, ACT2, ACT3, ACTobl**, all with *index* as their type, signifying the pointer to an individual, and a feature **AKTRT** for 'Aktionsart'. Thus, for a ditransitive verb, the semantic part of the AVM will look as follows:

(7)

$$\textit{synstruc}\left[\begin{array}{l}\text{ACTNTS } \textit{sit}\left[\begin{array}{l}\text{PRED } \textit{pred}\\ \text{ACT0 } \textit{index}\\ \text{ACT1 } \textit{index}\\ \text{ACT2 } \textit{index}\\ \text{ACT3 } \textit{index}\\ \text{AKTRT } \textit{aktionsart}\end{array}\right]\end{array}\right]$$

To technically introduce these configurations in the grammar, we add a type *sit* under *featstruc*, with a declaration of **ACT**-features organized as follows:

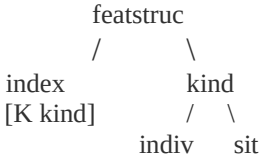
(8)



These attributes do not indicate specific roles, but reflect *role dependency* relations, in that whatever role **ACT2** turns out to have, it could not be unfolded in a situation expressed by the verb in question without there being also a participant having the role of **ACT1**, and analogously for **ACT3** vs **ACT2**. This follows the same logic as is used for the attributes 'ARG1', 'ARG2', etc. in the Matrix semantic system. **ACTobl** is whatever is expressed as an oblique argument.

We assume that all syntactic constituents have a *referent* of one kind or another. NPs are standardly assumed to refer to *individuals*, and sentences and VPs to *situations* (the latter is the view of 'Situation Semantics', the theory advocated by Barwise and Perry 1980). We may assume that prepositions and adjectives refer to situations as well, just of a less dynamic kind than many situations referred to by verbs. The type *index* in general declares an attribute **KIND**, abbreviated **K**, whose value *kind* has the subtypes *sit* and *indiv* – as displayed below, a further specification added to figure 1:

(9)



The type *aktionsart* has a limited number of subtypes, one of which is *accomplishment*. We return later to a discussion of these types.

The type *synstruc* will have to accommodate attributes reflecting the specifications now introduced. The type name itself is also becoming less plausible, since it seems to represent only syntactic structure. We therefore replace this type name by the type name *sign*. Thus amended, the type definition (10a) gives way to (10b) in the type tree in figure 1:

(10)

a. **synstruc**
[HEAD pos,
SPR list,
COMPS list,
ARGS list]

b. **sign**
[HEAD pos,
SPR list,
COMPS list,
INDX index,
ACTNTS sit,
ARGS list]

Now consider the linking between the participant attributes just introduced and the syntactic constituents representing the participants (attention now being on *expressed* participants). For any verb, the referent of its *subject* will be the value of its **ACT1**. We can display this as follows (omitting all other aspects of the structure):

(11)
$$\begin{bmatrix} \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{ACTNTS} \left[\text{ACT1 } \boxed{1} \right] \end{bmatrix}$$

Similarly, for any *object* in a single-object construction, its referent will be the value of **ACT2**:

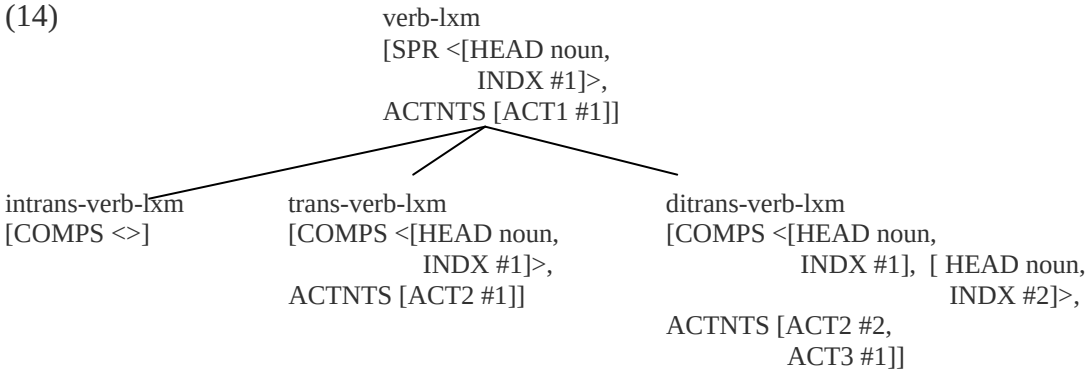
(12)
$$\begin{bmatrix} \text{COMPS} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{ACTNTS} \left[\text{ACT2 } \boxed{1} \right] \end{bmatrix}$$

For a ditransitive verb, in turn, the linkages to be defined could be as follows, if we have in mind the structure of a Germanic verb phrase – the indirect object precedes the direct object, but is logically dependent on it, and so serves as **ACT3** in the semantic structure:

(13)
$$\begin{bmatrix} \text{COMPS} \left\langle \left[\text{INDX } \boxed{1} \right], \left[\text{INDX } \boxed{2} \right] \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT2 } \boxed{2} \\ \text{ACT3 } \boxed{1} \end{bmatrix} \end{bmatrix}$$

Where should these specifications be introduced in the overall type system? It will be recalled that as *verb-lxm* was defined in figure 1, it has an obligatory subject. The required linkage displayed in (11) could then be added to this definition of verb-lxm. Moreover, if we assume that the linking in (12) is a standard property of transitive verbs, we can define a type *trans-verb-lxm* as a subtype of *verb-lxm*, with the specification in (12) as what defines the subtype, and correspondingly for a type *ditrans-verb-lxm* likewise inheriting from *verb-lxm* and with the specifications in (13) added, as suggested in (14):

(14)



With these linkages having been introduced, the lexical entries for the verbs *bark*, *chase* and *give* in (6) can be amended as follows (using entry identifiers reflecting the valence):

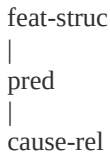
(15) bark_intr := intrans-verb-lxm &
[ORTH "bark"].

chase_tr := trans-verb-lxm &
[ORTH "chase].

give_ditr := ditrans-verb-lxm &
[ORTH "give"].

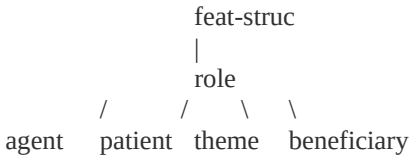
What is conceivably still missing in these entries is an indicator of the item-meaning of each lexical item, and therewith an attribute for item-specific meaning. In some of the examples in chapter 2, we indeed have an attribute **PRED** used for the value *cause-rel*, where the latter reflects a grammatically induced meaning rather than a basic lexical meaning. We leave open what to do about these for the moment, and so leave the entries in (15) without a **PRED** specification, but we nevertheless recognize the need for a **PRED** attribute, as is present in the AVM schema (7). The definition of its value *pred* we hypothesize as a type with *cause-rel* as one of its subtypes, but we otherwise say no more about the type *pred* here:

(16)



Given that the **ACT**-features do not represent *participant roles*, a next question is how participant roles should or could be represented. We outline one strategy, first illustrated by the following set of roles, analyzed as subtypes of a type *role*: *agent*, *patient*, *theme*, *beneficiary*:

(17)



The type *role* will be the value of an attribute **ROLE**, declared by *index*; thus, the amended declaration for *index* will be:

(18)
index
[K kind,
ROLE role]

We can now start modeling some semantically based verb types which have been much referred to in the literature. First, an *unaccusative* verb is often defined as an intransitive verb with the subject having the role *theme*. We can define this type as a subtype of *intrans-verb-lxm* by adding the appropriate specification:

(19)
intrans-verb-lxm
|
intrans-unacc-verb-lxm
[ACTNTS.ACT1.ROLE theme]

Likewise we can define an *agentive* intransitive verb, suitable, e.g., for *bark*:

(20)
intrans-verb-lxm
|
intrans-agentive-verb-lxm
[ACTNTS.ACT1.ROLE agent]

As for the other valence types, the verb *kick* may be said to involve an *agent* and a *patient*. The following would be an inheritance specification providing a lexical type reflecting these roles:

(21)
trans-verb-lxm
|
trans-affecting-verb-lxm
[ACTNTS [ACT1.ROLE agent,
ACT2.ROLE patient]]

For the type of ditransitive represented by *give*, the following lexical type would serve:

(22)
ditrans-verb-lxm
|
ditrans-transfer-verb-lxm
[ACTNTS [ACT1.ROLE agent,
ACT2.ROLE theme,
ACT3.ROLE benefactive]]

Introducing roles may thus allow one to represent some common, intuitive notions in a straightforward manner. However, the manner illustrated is not without problems, as can be exemplified as follows. ‘Kicking oneself’ may intuitively be conceived as something which should have index identity between the agent and patient, as in (23):

(23)
$$\begin{bmatrix} \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{COMPS} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{1} \end{bmatrix} \end{bmatrix}$$

But the structure identity of what follows the reentrancy symbols under **ACTNTS** will fail for the structures ‘**ROLE** *agent*’ (specifying **ACT1**) and ‘**ROLE** *patient*’ (specifying **ACT2**).

There are at least three ways of avoiding this problem:
a. Declaring the attribute **ROLE** by another type than *index*.
b. Avoiding using reentrancy to represent reflexivity, and rather introduce another **ACTNTS** relation (informally called '**ACTNTS-II'** in (24)) to explicitly state the identity, as in (24):

(24)
$$\begin{bmatrix} \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{COMPS} \left\langle \left[\text{INDX } \boxed{2} \right] \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \\ \text{ACTNTS} - \text{II} \begin{bmatrix} \text{PRED identical - rel} \\ \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \end{bmatrix}$$

c. Introducing a further attribute inside *index* to express referential 'this-ness', e.g. '**HAEC**' for 'haeccitas':

(25)
$$\begin{bmatrix} \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{COMPS} \left\langle \left[\text{INDX } \boxed{2} \right] \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \begin{bmatrix} \text{HAEC } \boxed{3} \\ \text{ROLE agent} \end{bmatrix} \\ \text{ACT2 } \boxed{2} \begin{bmatrix} \text{HAEC } \boxed{3} \\ \text{ROLE patient} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

On this alternative, the declaration of *index* thus gets three attributes (where the value type *id* is a further subtype of *featstruc*):

(26)
index
[HAEC id,
K kind,
ROLE role]

In the Matrix system, roles are not represented. In the grammar *NorSource*, built on the Matrix but using roles, strategy b. is the one used. In TypeGram we for the present assume this strategy as well.

Given these developments, the combinatorial rules in rules.tdl must state that the **ACTNTS** and **INDX** specification of the head daughter is identical to the **ACTNTS** and **INDX** specification of the mother. Likewise, in the definition of *word* in types.tdl, we need to reenter the **ACTNTS** and **INDX** values of the input lexeme with the **ACTNTS** and **INDX** values of the output word.

This completes the first step of introducing a simple semantics in the Simple grammar. We have introduced a feature space for participants and for participant roles, and linking of these to items previously present in the grammar. We have noted consequences that these moves have for the lexical specification of verbs. By now, the grammar will still have the same coverage as Simple grammar, but each parse will in addition have specifications for **ACTNTS** and **INDX**, making it in principle possible to see which participants link to which NPs. The next section will bring us to the point where these linkages can be represented in the AVM of the highest constituent in the parse tree.

3. Grammatical Functions

3.1. Modeling Grammatical Functions

Traditional Grammatical Function (GF) notions like 'subject', 'object', 'direct object' and so forth are not explicitly represented in standard HPSG, only indirectly through their placement in valence lists. This strategy has a couple of disadvantages:

(a) From the mere order in a COMPS list it is not always easy to decide which GFs are being represented, and:

(b) Since valence lists are cancelled through combination up through a tree, when one reaches the top S-node, there is not even an indirect GF representation left. That means that the feature structure of a full construction gives no information about which GFs are realized inside the construction.

The situation can be remedied in the following way. Attributes for GFs taking *signs*, not lists of signs, will be defined, in much the same way as LFG uses attributes for Grammatical Functions. These GF specifications will be correlated with valence lists of lexical items, but propagated unchanged up through the combination trees, formally in the same way as the ACTANTS attribute is propagated up, so that the full construction has a record of the GFs realized. We now present a procedure for building such a system.

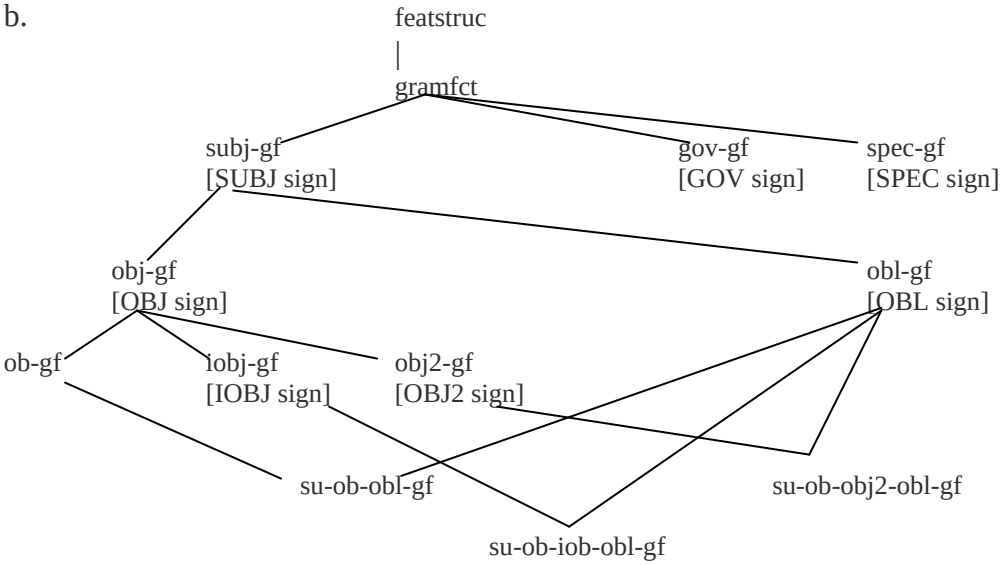
A new attribute will be introduced under the type we have re-labeled as *sign*, namely **GF**, with the value *gramfct*, a subtype of *featstruc*. The subtype system under *gramfct* is what declares the specific GF attributes – viz. **SUBJ, OBJ, GOV, IOBJ, OBJ2, OBL, SPEC**. This is done in an analogous way to how the *sit* hierarchy is defined (see (8) above). In rendering these definitions, and others below, we first use the tdl notation, where ':=' means 'is a subtype of' ((27a)), and then for visualization, a type tree ((27b)).

(27)

a.

```
gramfct := feat-struc.
subj-gf := gramfct & [ SUBJ sign ].
gov-gf := gramfct & [ GOV sign ].
obj-gf := subj-gf & [ OBJ sign ].
iobj-gf := obj-gf & [ IOBJ sign ].
obj2-gf := obj-gf & [ OBJ2 sign ].
obl-gf := subj-gf & [ OBL sign ].
spec-gf := gramfct & [ SPEC sign ].
su-ob-obl-gf := obj-gf & obl-gf.
su-ob-iob-obl-gf := iobj-gf & obl-gf.
su-ob-ob2-obl-gf := obj2-gf & obl-gf.
```

b.



For double object constructions, some traditions use the terms "direct object" and "indirect object", others use "first object" and "second object". In the present encoding, "direct object" and "first object" are represented by the attribute '**OBJ**', whereas '**IOBJ**' represents 'indirect object', and '**OBJ2**' represents 'second object'. Both of the latter are considered dependent on the presence of '**OBJ**', thus, they will not occur unless there is **OBJ** occurring. Moreover, '**IOBJ**' and '**OBJ2**' never co-occur in a feature matrix. As for the type declaring **GOV**, namely *gov-gf*, it does not presuppose the presence of a **SUBJ**, whereas *obj-gf* which declares **OBJ**, inherits from *subj-gf*; this is the type relevant for verbal specification, whereas *gov-gf* will be relevant for prepositions and adjectives.

Note that the types defined in (27) are not on *sign*-level – they are one path-slot embedded inside this type. At *sign* level, we define the following sub-types of *sign* according to which GFs they contain:

(28)
```
subj-sign := sign & [ GF subj-gf ].
gov-sign := sign & [ GF gov-gf ].
obj-sign := sign & [ GF ob-gf ].
iobj-sign := sign & [ GF iobj-gf ].
obj2-sign := sign & [ GF obj2-gf ].
obl-sign := sign & [ GF obl-gf ].
spec-sign := sign & [ GF spec-gf ].
```

The type *spec-sign* will be associated only with nouns, displaying the *Specifier* constituent of a noun phrase through the GF '**SPEC**'.

3.2. Linking Grammatical Functions to Participant Roles

The type system for GFs now will be linked to semantic specification. The following types inter-relate these levels of specification:

(29)
```
su-act1-link := subj-sign &
 [ GF.SUBJ.INDX #1,
   ACTNTS.ACT1 #1 ].
gov-act2-link := gov-sign &
 [ GF.GOV.INDX #1,
   ACTNTS.ACT2 #1 ].
obj-act2-link := obj-sign &
 [ GF.OBJ.INDX #1,
   ACTNTS.ACT2 #1 ].
iob-act3-link := iobj-sign &
 [ GF.IOBJ.INDX #1,
   ACTNTS.ACT3 #1 ].
ob2-act3-link := obj2-sign &
 [ GF.OBJ2.INDX #1,
   ACTNTS.ACT3 #1 ].
obl-actObl-link := obl-sign &
 [ GF.OBL.GF.GOV.INDX #1,
   ACTNTS.ACTobl #1 ].
spec-act2-link := spec-sign & phrase &
 [ INDX #2,
   GF.SPEC.INDX #1,
   ACTNTS.ACT2 #2,
   ACTNTS.ACT1 #1 ].
```

In terms of these linkages, construction types can now be defined in terms of both GFs and ACTANT values:

(30)
```
poss-sign := spec-act2-link.
intr-sign := su-act1-link.
intrObl-sign := obl-actObl-link.
tr-sign := intr-sign & ob-act2-link.
trObl-sign := intrObl-sign & tr-sign.
ditr-sign := tr-sign & iob-act3-link.
dbob-sign := tr-sign & ob2-act3-link.
ditrObl-sign := trObl-sign & ditr-sign.
dbobObl-sign := trObl-sign & dbob-sign.
gvmt-sign := gov-act2-link.
```

This linkage will provide the kind of linking shown in the early part of chapter 1, like in (3), repeated as (31), now exposed with the relevant type among those defined:

(31)

$$sign\left[\begin{array}{l} ditr-sign \\ \text{HEAD } verb \\ \text{GF } grmfct\left[\begin{array}{l}\text{SUBJ } sign\left[\text{INDX } \boxed{1} index\right] \\ \text{OBJ } sign\left[\text{INDX } \boxed{2} index\right] \\ \text{IOBJ } sign\left[\text{INDX } \boxed{3} index\right]\end{array}\right] \\ \text{ACTNTS } sit\left[\begin{array}{l}\text{ACT0 } index \\ \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \\ \text{ACT3 } \boxed{3} \\ \text{AKTRT } aktionsart\end{array}\right]\end{array}\right]$$

It may be noted that the label 'ditr' reflects the presence of the attributes **OBJ** and **IOBJ**, whereas 'dbob' (for 'double object') reflects the presence of the attributes **OBJ** and **OBJ2**. The type *gvmt-sign* (for 'government sign') is a type suited for prepositions, which have a governee, represented by the attribute **GOV**, but not in general a subject.

3.3. Linking Grammatical Functions to Valence lists

We next integrate this system of **GF**s correlated with **ACTNTS** into the parsing grammar. The parsing procedure hinges crucially on lexical valence specifications. For lexical correlation of valence lists and GFs, and in turn participant roles, the following types are therefore defined:

(32)
```
intr-lex := intr-sign &
 [ GF.SUBJ #1,
   SPR < #1 > ].
intrObl-lex := intr-lex & intrObl-sign &
 [ GF.OBL #1,
   COMPS < #1 > ].
tr-lex := intr-lex & tr-sign &
 [ GF.OBJ #1,
   COMPS < #1 > ].
trObl-lex := intr-lex & trObl-sign &
 [ GF.OBJ #2,
   GF.OBL #1,
   COMPS < #2, #1 > ].
ditr-lex := intr-lex & ditr-sign &
 [ GF.OBJ #2,
```

```
   GF.IOBJ #1,
   COMPS < #1, #2 > ].
dbob-lex := intr-lex & dbob-sign &
 [ GF.OBJ #1,
   GF.OBJ2 #2,
   COMPS < #1, #2 > ].
ditrObl-lex := intr-lex & ditrObl-sign &
 [ GF.OBJ #2,
   GF.IOBJ #1,
   GF.OBL #3,
   COMPS < #1, #2, #3 > ].
dbobObl-lex := intr-lex & dbobObl-sign &
 [ GF.OBJ #1,
   GF.OBJ2 #2,
   GF.OBL #3,
   COMPS < #1, #2, #3 > ].
gvmt-lex := gvmt-sign &
  [ GF.GOV #1,
    COMPS < #1 > ].
poss-lex := poss-sign &
  [ GF.SPEC #1,
       SPR < #1 > ].
```

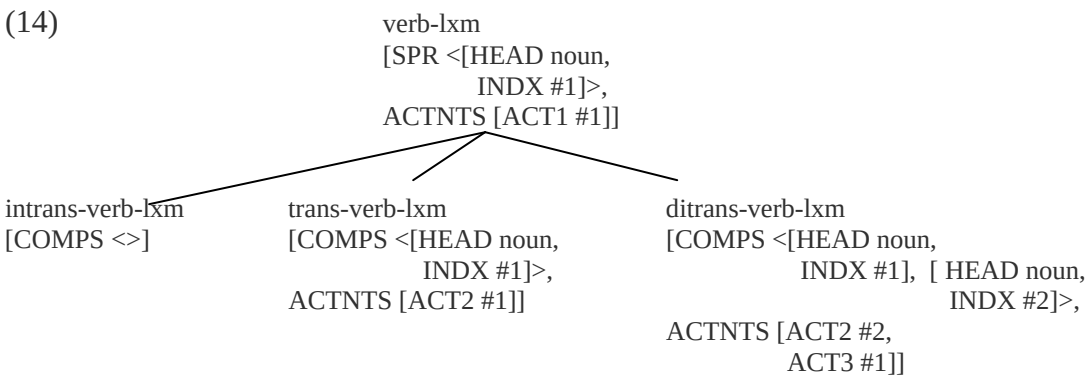
At this point, we have supplied the specifications that underlie an AVM like the lowest node in the analysis tree (38b) in chapter 1, repeated as (33), i.e., an AVM where GF, valence lists and participant specification are all interlinked:

(33)

$$\begin{bmatrix} \text{HEAD verb} \\ \text{GF} \begin{bmatrix} \text{SUBJ } \boxed{3}\left[\text{INDX } \boxed{1}\right] \\ \text{OBJ } \boxed{4}\left[\text{INDX } \boxed{2}\right] \end{bmatrix} \\ \text{SPR } \langle \boxed{3} \rangle \\ \text{COMPS } \langle \boxed{4} \rangle \\ \text{ACTNTS} \begin{bmatrix} \text{PRED fange-rel} \\ \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \end{bmatrix}$$

Compared to the *lexical types* defined in (14), repeated, based on **ACTNTS** specification and valence lists only,

(14)



the types *intr-lex*, *tr-lex* and *ditr-lex* from (32) above differ in that they also include GF specification, and so will take precedence over the types in (14). The latter are thus no more included in the grammar.

In ‘verb’ not being mentioned in the type names in (32), we disentangle frame specification from POS of the head. In coining verb type names, we rather add ‘v-‘ to the names already defined in (32), so that instead of (14) we have the verb type definitions (34), and as for lexical entries, we will have the lexical specifications (35) rather than those in (15) (*v-lex* being a type defined as having *verb* as head):

(34)
```
v-intr-lxm := intr-lex & v-lex.
v- intrObl-lxm := intrObl-lex & v-lex.
v-tr-lxm := tr-lex & v-lex.
v-trObl-lxm := trObl-lex & v-lex.
v-ditr-lxm := ditr-lex & v-lex.
v-dbob-lxm := dbob-lex & v-lex.
v-ditrObl-lxm := ditrObl-lex & v-lex.
v-dbobObl-lxm := dbobObl-lex & v-lex.
```

(35)
```
bark_intr := v-intr-lxm &
[ ORTH "bark" ].

chase_tr := v-tr-lxm &
[ ORTH "chase].

give_ditr := v-ditr-lxm &
[ ORTH "give” ].
```

To fully integrate the GFs in the parsing grammar, we have to propagate the value of **GF** from head to mother in all syntactic rules, lexical rules and in the type *word* (just as we did with **ACTNTS** and **INDX**). And rather than (10b), the declaration of *sign* is now (36):

(36)
```
sign
[ HEAD pos,
GF gramfct,
SPR list,
COMPS list,
INDX index,
ACTNTS sit,
ARGS list ]
```

Three analytic perspectives have now been combined. For valence, there are two perspectives, one following the ‘satisfaction’ aspect of valence as modeled in the parsing algorithm, and one for exposure of valence as the backbone of the sentence as a fully analyzed entity. The third perspective exposes the meaning of the sentence as a fully analyzed entity. The latter two perspectives will be of interest if one wants to classify construction types as such, independently of a parsing mechanism; on this, see chapter X (on ‘enumeration’ and ‘ontology’).

4. Some prepositional and nominal constructions

In Simple grammar, nouns require a specifier but take no complements. Prepositions require a complement but take no specifier, however they have a **MOD** feature by which they specify what they can modify (in Simple grammar that being either a noun or a verb). As GF categories, the specifier of a noun is **SPEC**, and the complement, or governee, of a preposition is **GOV**, by the extensions made above. Semantically, prepositions are a bit like transitive verbs, with an **ACT1** and an **ACT2**, where **ACT2** is the referent of the governee, as stated in (29). We now discuss three types of prepositions, widely recognized as different and here differentiated amongst others according to how their **ACT1** is specified.

4.1. Oblique arguments and *selected prepositions.*

An example of an *oblique argument* is *John* in *rely on John*. By general conception, an *oblique constituent* Y relative to some head X is an item required by X, or required in order for X to carry the meaning it has in the construction, where Y consists of a head Z, typically of the category *preposition*, and the governee G of the preposition, where G is perceived as having a role relative to X, and Z indicates what that role is.

rely	*on*	*John*
X	Z	G
	Y	

Thus, in *rely on John*, 'John' is seen as having a role relative to 'rely', but is syntactically governed by *on*, whereby it is 'shielded' from the verb and relates to it only 'obliquely'. Hence G, or 'John', is called an argument of X, or 'rely', and due to the way Z (*on*) mediates the argumenthood, an *oblique argument* of X. (Another term used is indirect argument.) The phrase *on John* may at the same time be called an 'oblique constituent'. Prepositions like *on* in the present capacity are commonly called *selected prepositions*.

In the lexical specification of *rely*, we accordingly need to specify the NP contained in the PP, since it is the index of this NP which represents a participant relative to the verb, not the index of the PP. With the GF **GOV** available for this NP, the lexical specification of *rely* will be as follows:

(37)
$$\begin{bmatrix} \text{HEAD verb} \\ \text{GF} \begin{bmatrix} \text{SUBJ } \boxed{3}\begin{bmatrix}\text{INDX } \boxed{1}\end{bmatrix} \\ \text{OBL } \boxed{4}\begin{bmatrix}\text{HEAD prep} \\ \text{GF}\begin{bmatrix}\text{GOV}\begin{bmatrix}\text{INDX } \boxed{2}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \text{SPR } \langle \boxed{3} \rangle \\ \text{COMPS } \langle \boxed{4} \rangle \\ \text{ACTNTS}\begin{bmatrix}\text{PRED rely-rel} \\ \text{ACT1 } \boxed{1} \\ \text{ACTobl } \boxed{2}\end{bmatrix} \end{bmatrix}$$

The preposition *on* itself will have the following lexical specification, when used in this configuration:

(38)
$$\begin{bmatrix} \text{HEAD prep} \\ \text{GF } \begin{bmatrix}\text{GOV } \boxed{4}\begin{bmatrix}\text{INDX } \boxed{2}\end{bmatrix}\end{bmatrix} \\ \text{SPR } \langle\ \rangle \\ \text{COMPS } \langle \boxed{4} \rangle \\ \\ \text{ACTNTS}\begin{bmatrix}\text{PRED on-rel} \\ \text{ACT1 index} \\ \text{ACT2 } \boxed{2}\end{bmatrix} \end{bmatrix}$$

As this is not a preposition that can be used predicatively or in a modifying function, the **ACT1** is not linked to any syntactic constituent. When (37) combines with (38), the value of the preposition's **ACT2** is unified with the verb's **ACTobl**, and in the latter capacity propagated further up in the construction specification.

4.2. Modifying PPs

The specification (38) may be compared to the specification of 'regular' modifying prepositions, such as *under* in *he sits under the table*, shown in (42), chapter 1, here repeated as (39). Here, in contrast to the preceding case, the **ACT1** has a value, namely *the index of the item modified*. Even in this case, the item modified will typically not be naturally associated with a GF notion like 'subjects' relative to the adjunct, so we will still not assume any GF attribute for this function. However, for the valence satisfaction aspect, we provide the adjunct with the attribute **MOD**, standing for 'item with which I combine'. As a change from Simple grammar, where the attribute **MOD** stands inside of **HEAD**, thus declared by *pos*, it is here declared directly by *sign*, as a valence feature on a par with **SPR** and **COMPS**. The preposition *under*, when used in constructions like the one mentioned, will thus have the specification (39):

(39)

$$\begin{bmatrix} \text{HEAD } \textit{prep} \\ \text{GF} \left[\text{GOV} \left[\text{INDX } \boxed{2}\right]\right] \\ \text{MOD} \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{INDX } \boxed{1} \end{bmatrix} \right\rangle \\ \text{COMPS} \left\langle \left[\text{INDX } \boxed{2}\right] \right\rangle \\ \\ \text{ACTNTS} \begin{bmatrix} \text{PRED } \textit{under} \\ \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \end{bmatrix}$$

For this kind of preposition, which is seen as the more general kind of preposition, the main rule of combination is (40), combining the Adjunct with either N or VP. This contrasts with the case of *rely on John*, where *on John* combines with *rely* by *Head-complement-rule1*. Making use of **MOD** as now stipulated, (40) replaces (5c) as *Head-Modifier rule* I.

(40) *Head-Modifier rule* I:

$$\begin{bmatrix} \text{HEAD } \boxed{5} \\ \text{GF } \boxed{1} \\ \text{SPR } \langle \boxed{3} \rangle \\ \text{COMPS } \langle\ \rangle \\ \text{INDX } \boxed{2} \\ \text{ACTNTS } \boxed{6} \\ \textit{HEAD-DTR} \quad \boxed{4} \begin{bmatrix} \textit{HEAD } \boxed{5} \\ \textit{GF } \boxed{1} \\ \textit{SPR } \langle \boxed{3} \rangle \\ \textit{COMPS } \langle\ \rangle \\ \textit{INDX } \boxed{2} \\ \textit{ACTNTS } \boxed{6} \end{bmatrix} \\ \textit{NONHEAD-DTR} \begin{bmatrix} \textit{MOD } \langle \boxed{4} \left[\textit{INDX } \boxed{2}\right] \rangle \\ \textit{COMPS } \langle\ \rangle \\ \textit{ACTNTS } \left[\textit{ACT1 } \boxed{2}\right] \end{bmatrix} \end{bmatrix}$$

Rather than (36), the declaration of *sign* is now (41):

(41) **sign**
[HEAD pos,
GF gramfct,
SPR list,
COMPS list,
MOD list,
INDX index,
ACTNTS sit,
ARGS list]

We will type-refer to the preposition type in (38) as *sel-prep-word* ('selected preposition') and the one in (39) as *mod-prep-word* ('modifying preposition').[1]

4.3. Directional and locative prepositions

In the representation of a directional expression, such as *runs to* in *the boy runs to the house* or *throws... through* in *the boy throws the ball through the window*, it has to be indicated which entity performs the directional function. As seen above, in the case of *mod-prep-word*, the ACT1 of the preposition, when modifying a verb, is the *event index* of the verb. In contrast, in *the boy throws the ball through the window*, what induces the path is 'the ball', that is, the direct object, while in *the boy runs to the house*, it is the subject, i.e., 'the boy'. For these reasons, when assigning the ACT1 of a directional preposition the correct value, we need to equate the ACT1 of the preposition with the relevant *participant* of the verbal event, rather than with the event as such. We will model the present

[1]For convenience, the following type definitions from above are relevant for the composition of the sign (37):

(i)
```
subj-gf := gramfct & [ SUBJ sign ].
obl-gf := subj-gf & [ OBL sign ].
subj-sign := sign & [ GF subj-gf ].
obl-sign := sign & [ GF obl-gf ].
su-act1-link := subj-sign &
 [ GF.SUBJ.INDX #1,
   ACTNTS.ACT1 #1 ].
obl-actObl-link := obl-sign &
 [ GF.OBL.GF.GOV.INDX #1,
   ACTNTS.ACTobl #1 ].
intr-sign := su-act1-link
intrObl-sign := obl-actObl-link.
intr-lex := intr-sign &
 [ GF.SUBJ #1,
   SPR < #1 > ].
intrObl-lex := intr-lex & intrObl-sign &
 [ GF.OBL #1,
   COMPS < #1 > ].
v- intrObl-lxm := intrObl-lex & v-lex.
```

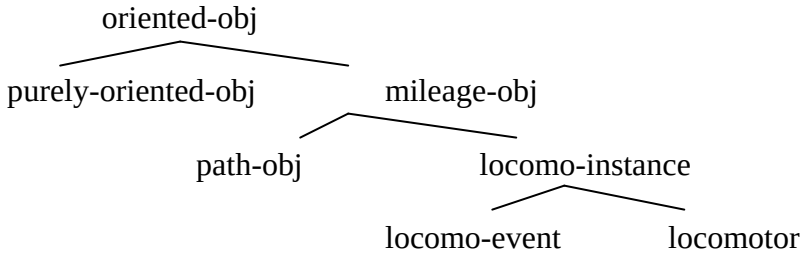
Similarly, the following type definitions from above are relevant for the composition of the sign (38) ((*p-lex* being a type defined as having *prep* as head):

(ii)
```
gov-gf := gramfct & [ GOV sign ].
gov-sign := sign & [ GF gov-gf ].
gov-act2-link := gov-sign &
 [ GF.GOV.INDX #1,
   ACTNTS.ACT2 #1 ].
gvmt-sign := gov-act2-link.
gvmt-lex := gvmt-sign &
  [ GF.GOV #1,
    COMPS < #1 > ].
p-gov-lex := gvmt-lex & p-lex.
```

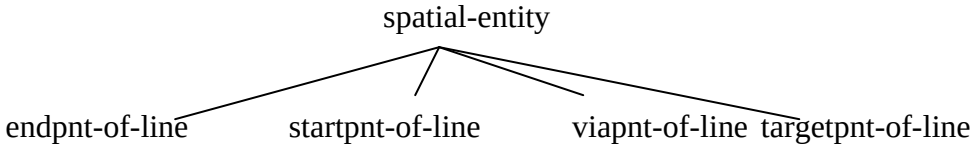
approach on the analysis of directional expressions proposed in (Jackendoff 1987), where syntactically, the directional PP is treated as an argument of the verb, and semantically, it introduces a structure where the governed NP is represented as an end-point, start-point or via-point (possibly embedded in a place-structure) of a relation 'GO' or 'ORIENT' connecting the directed participant to the end-/start-/via-point. Jackendoff's function-argument structures will not be directly reproduced here, but with typed feature structures we will model the same content.

As a subtype of *actants* we introduce *dir-rel*, which declares an attribute **DIR**. The value of **DIR** is *act12-rel*, where **ACT1** is the directed entity and **ACT2** is the reference point, such as the end-point, start-point or via-point. The exact status as directed entity for **ACT1** or end-, start- or via-point for **ACT2** is defined as a (subtype of) *role*. We model the following roles in the type hierarchy under *role*:

(42)
a. Roles for **ACT1**:



b. Roles for **ACT2**:



Illustrations of the roles of **ACT1**s in (42a) are given in (43) (leaf nodes from right to left):

(43)
a. - as a 'mover' along a path, to be called a *locomotor* (e.g., *Ernst ran to Hamburg*)
b. - as an event along a path, to be called a *locomo-event* (e.g., *the tour went to Hamburg*)
c. - as an extended object, to be called a *path-obj* (e.g., *the road went to Hamburg*)
d. - as a purely oriented object, to be called a *purely-oriented-obj* (e.g., *the sign points to Hamburg*)

As seen from the examples, a preposition like *to* can serve in all of these **ACT1** roles; however, relative to the types in (42b) reflecting possible **ACT2** roles, only *endpnt-of-line* is a possible role for *to*. Regarding verbs which can interact with the options in (42a), *point* goes only with a subject having the role *purely-oriented-obj*, *go* can be used in any of the other three options, and a verb like *stroll* probably only as a *locomotor*. The object of the transitive verb *throw* is necessarily a *locomotor*, whereas the object of *draw* (like in *draw the line along the edge*) is a *path-obj*.

Reflecting these specifications, we illustrate how the salient aspects of the feature structures of *stroll* and *to*, as in *the cat strolled to the dog* should look, and *throw* and *to*, as in *I threw the ball to John*:

(44)

a.
$$\begin{bmatrix} \text{ORTH "stroll"} \\ \\ \text{HEAD verb} \\ \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{COMPS} \left\langle \begin{bmatrix} \text{HEAD prep} \\ \text{INDX } \boxed{2} \end{bmatrix} \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{DIR.K } \boxed{2} \left[\text{ACT1 } \boxed{1} \left[\text{ROLE locomotor} \right] \right] \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \text{ORTH "throw"} \\ \\ \text{HEAD verb} \\ \text{SPR} \left\langle \left[\text{INDX } \boxed{1} \right] \right\rangle \\ \text{COMPS} \left\langle \begin{bmatrix} \text{HEAD noun} \\ \text{INDX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{HEAD prep} \\ \text{INDX } \boxed{3} \end{bmatrix} \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \\ \text{DIR.K } \boxed{3} \left[\text{ACT1 } \boxed{2} \left[\text{ROLE locomotor} \right] \right] \end{bmatrix} \end{bmatrix}$$

c.
$$\begin{bmatrix} \text{ORTH "to"} \\ \\ \text{HEAD prep} \\ \text{INDX } \boxed{1} \\ \text{COMPS} \left\langle \begin{bmatrix} \text{HEAD noun} \\ \text{INDX } \boxed{1} \end{bmatrix} \right\rangle \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1} \left[\text{ROLE oriented} - \text{obj} \right] \\ \text{ACT2 } \boxed{1} \left[\text{ROLE endpnt} - \text{of} - \text{line} \right] \end{bmatrix} \end{bmatrix}$$

ACT1 of *to* has the role *oriented-obj*, which is the highest type in the hierarchy (42a), reflecting the fact that it can be used in all the directional types illustrated in (43). When *to* is combined with *stroll* or *throw*, the specifications of these verbs force the role of the **ACT1** of *to* be *locomotor*, which is a subtype of *oriented-obj*, and thus compatible with the general specification of *to* in (c).

As the **ACT1** of a directional preposition is in all cases identical to one of the arguments of the verb (and carries the role *oriented-obj*), it contrasts with the **ACT1** of a *modifying locative* preposition, which has as its **ACT1** the index of the verb itself (cf. (39)), and with the **ACT1** of a selected preposition, which is left undefined (cf. (38)).[2]

While (uses of) directional prepositions can be classified in terms of roles as indicated, the main classificatory notions relevant for modifying locative prepositions are topological, like in the following table, where 'FIG' is the **ACT1** and 'GROUND' the **ACT2**:[3]

[2] When qualifying a *noun*, the **ACT1** of both directional and modificational prepositions will be the index of that noun, but still with distinct types of roles - the **ACT1** of the directional *to* in *a bus to Beijing* will be *oriented-obj*, as opposed to the **ACT1** of *under* in *the cat under the table*.

[3] From Hellan and Beermann 2009, building to a large extent on Trujillo 1994.

Topological features (all boolean):	Definitions:
FRONT	FIG is in front of GRND
BACK	FIG is behind GRND
EMBEDDED	FIG is embedded in GRND
CONTAINED	FIG is contained in GRND
SCALAR	Relation between FIG and GRND can be quantified (like in "2 cm behind)
TRANSITIVE	If R(A,B) and R(B,C), then R(A,C)
UPSIDE-OF	FIG is upside in a vertical relation to GRND
DOWNSIDE-OF	FIG is downside in a vertical relation to GRND
INTEGRATED	FIG is integrated into GRND

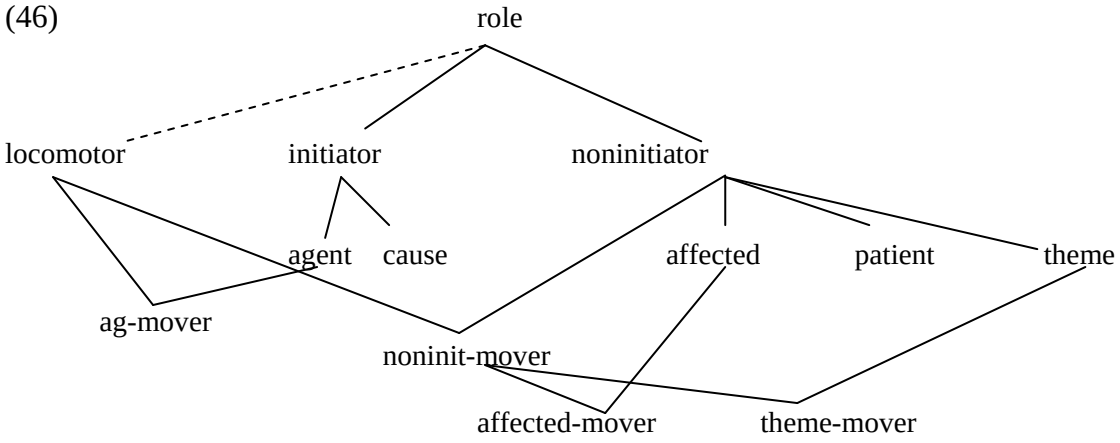
Table 1: Main topological features

Features like these qualify the *relation* between the **ACT1**and the **ACT2** rather than the role of any one of them in particular, and so the appropriate locus for these specifications will be under the attribute **ACT0** rather than under **ACT1** or **ACT2**. It will be introduced by a feature **CLASS** whose value *topology* introduces these binary features. (45) illustrates this with the semantic part of the lexical specification of *behind*, as in *the cat sits behind the chimney*:

(45)
$$\begin{bmatrix} \text{PRED 'behind}-\text{rel'} \\ \text{ACT0}\begin{bmatrix}\text{CLASS}\begin{bmatrix}\text{FRONT}- \\ \text{BACK}+ \\ \text{TRANSITIVE}+ \\ \text{SCALAR}+ \\ \text{INTEGRATED}-\end{bmatrix}\end{bmatrix} \\ \text{ACT1}\left[\text{ROLE fig}\right] \\ \text{ACT2}\left[\text{ROLE grnd}\right] \end{bmatrix}$$

Returning to roles, those that have now been introduced in connection with prepositions will sit in the same over-all hierarchy as 'agent' etc. as discussed previously, and these hierarchies can be merged, as partly indicated below (the dotted line down to 'locomotor' points to the hierarchy in (42a)):[4]

(46)



[4] The roles presented in (42) constitute what we may call a 'directional space', whereas those introduced in the previous section, such as *agent, patient*, etc., may be seen to constitute a 'force' space. Jackendoff 1987, 1990 devised a representation of distinct 'tiers' for these spaces. In the present system, through multiple inheritance, these role types can be combined in unified hierarchies.

To implement the structures now considered for directional prepositions and verbs of direction, we define a type *v-intrPath* for verbs which have a directional complement, with the subtypes *v-intrPath-suMover*, *v-intrPath-suEvent*, *v-intrPath-suPath*, and *v-intrPath-suOrient*, according to its **ACT1**-role (from right to left in (42a)). *V-intrPath* will be a subtype of *verb-lxm*, defined like *v-tr* except that its **COMPS** item is a PP and its **ACTANTS** has an attribute **DIR** rather than **ACT2**; (47a) induces the structure instantiated for *stroll* in (44a). For transitives like *throw*, similarly, (47b) induces the structure instantiated in (44b), and (47c) induces the type of preposition instantiated in (44c):

(47)

a.

```
v-intrPath := verb-lxm &
  [ COMPS < [ HEAD prep-or-adv,
                INDX #2,
                ACTANTS [ ACT1 #1 & [ ROLE oriented-obj] ]>,
    ACTANTS act1dir-rel & [ ACT1 #1,
                            DIR.K #2 act12-rel  & [ ACT1 #1 ]]].
```

b.

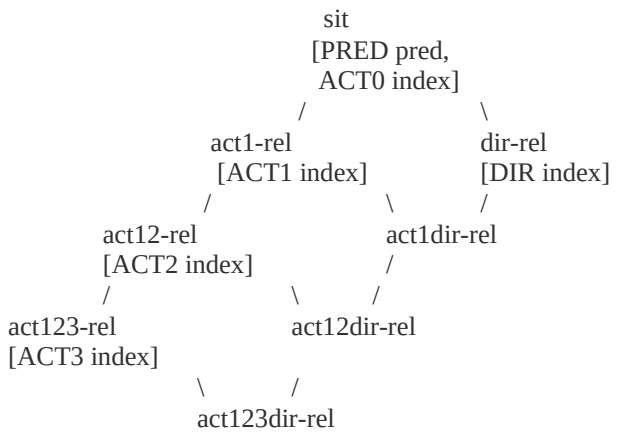
```
v-trPath := verb-lxm &
  [ COMPS < [ HEAD noun,
                INDX #1     ],    [ HEAD prep-or-adv,
                                    INDX #2,
                                    ACTANTS [ ACT1 #1 & [ ROLE oriented-obj] ]>,
    ACTANTS act12dir-rel & [ ACT2 #1,
                             DIR.K #2 & act12-rel  & [ ACT1 #1 ]]].
```

c.

```
dir-prep-word := p-gov-lex &
 [ ACTNTS [ ACT11 oriented-obj ] ].
```

The **HEAD** value *prep-or-adv* in (a) and (b) reflects the fact that the directional constituent with such verbs combine can also be an adverb, as in *run away*. To induce the types *act1dir-rel* and *act12dir-rel*, we expand the *-rel* –hierarchy in the same way as for the *–obl* extension of the *–rel* types (cf. (8) above):

(48)



For the sake of comparison with (47c), (49a,b) state the specifications of selected prepositions and modifying prepositions, respectively:

(49)

a. mod-prep-word := p-gov-lex &
[MOD < [INDX #1] >,
ACTNTS [ACT11 #1]].

b. sel-prep-word := p-gov-lex.

4.4. Relational nouns and possessive constructions

We here model combination of a noun with a possessive noun. We focus on the class of cases where the head noun is in some sense 'predisposed' for combining with another noun, namely *relational nouns*. The relevant types introduced so far are (50a-d), and we want to define (50e), a lexical type instantiated in the lexical entry (50f):

(50)

a. (from (29)):
spec-act2-link := spec-sign & phrase &
[INDX #2,
GF.SPEC.INDX #1,
ACTNTS.ACT2 #2,
ACTNTS.ACT1 #1].

b. (from (30)):
poss-sign := spec-act2-link.

c. (from (32)):
poss-lex := poss-sign &
[GF.SPEC #1,
SPR < #1 >].

d. (standard HPSG):
noun-lxm := lexeme &
[HEAD noun,
SPR < [ ] >,
COMPS < >].

e. rel-noun-lxm := noun-lxm & poss-lex.

f. leg_relnoun := rel-noun-lxm &
[ORTH « leg »,
PRED leg-rel].

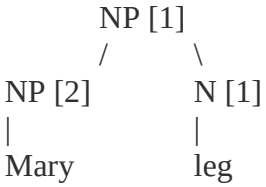
With these types, an NP like *Mary's leg* will have the AVM structure (51), which reads something like "leg x such that Mary 'leg-relates' to x":

(51)

$$\begin{bmatrix} \text{HEAD noun} \\ \text{SPEC} \begin{bmatrix} \text{HEAD noun} \\ \text{INDX } \boxed{2} \\ \text{ACTNTS} \begin{bmatrix} \text{PRED 'Mary'} \end{bmatrix} \end{bmatrix} \\ \text{INDX } \boxed{1} \\ \text{ACTNTS} \begin{bmatrix} \text{PRED 'leg} - \text{rel'} \\ \text{ACT0 } \boxed{1} \\ \text{ACT1 } \boxed{2} \\ \text{ACT2 } \boxed{1} \end{bmatrix} \end{bmatrix}$$

(52) renders the gross outline of a constituent structure for this phrase, with the index on each node corresponding to the indices in (51):

(52)



Thus, in the **ACTNTS** specification of the top NP, the specifier (‘Mary’) is the **ACT1** and the head (‘leg’) the **ACT2**, and a relation (like 'whole-part', 'family relation', 'X's surface' etc., named in the same way as the head) is declared as holding between **ACT1** and **ACT2**.

This semantic analysis logically works only for those head nouns which precisely indicate a relation that can hold between the NP referent and the specifier phrase. When this is not the case, like in *John’s house*, the most one can do in the semantics is to indicate that there is *some* relation, but the way to technically introduce it will be not via the lexical specification of the noun itself, but in the combinatorial rule for the configuration. This rule will have the form (53a), distinct from and added to (5b) (repeated for convenience of comparison); the indices are distributed on the branches as they are in (52), but the ACTNTS specification introduced is not rooted in any of the lexical items occurring – we may call this a *combinatorially introduced semantic specification*:

(53)

a.
```
head-specifier-rule := head-final &
[ SPR < >,
  COMPS #comps,
  ACTNTS [PRED poss-rel,
          ACT1 #2,
          ACT2 #1 ],
  ARGS < phrase &
      #sdtr & [ HEAD noun,
                SPR < >
                INDX #2 ],
      phrase &
      [ SPR < #sdtr >,
        COMPS #comps
        INDX #1 ] > ].
```

b.
```
head-specifier-rule := head-final &                    (= (5b))
[ SPR < >,
  COMPS #comps,
  ARGS < phrase &
      #sdtr & [ SPR < > ],
      phrase &
      [ SPR < #sdtr >,
        COMPS #comps ] > ].
```

(53a) retains from (53b) the assumption that the specifier NP is valence-bound by the head, which of course makes little sense empirically.[5] This is a technicality, however, in that ‘normal’ nouns will have a *list* as **SPEC** unspecified as to whether it is empty or non-empty, thus accepting also cases where there is no specifier.[6] To make (53b) less anglo-idiosyncratic, it too should be read with this

[5]Here one issue will be to distinguish between NPs acting as specifiers and determiners acting as specifiers – for English, the latter may still be required except for bare plurals, while in a general grammar design, no spedifier requirement is motivated. Another issue is that in English, a possessor and a head are connected by the *'s* attached to the possessor NP, while many languages just combine the NP and the N directly, and still others have a separate word between. What the above analysis models is only the direct combination strategy. Note that (53a) is for specifier NPs only, while (53b) covers both NPs and determiners.

[6] But nevertheless so that it unifies with the pattern described in (53).

interpretation, so that an obligatory specifier NP is assumed only when the head noun so dictates, as in *rel-noun-lxm*.

A particular case of *rel-noun-lxm* is arguably constituted by a large group of so-called *postpositions* in many languages. These are items traditionally seen as being of the same type as prepositions, only following rather than preceding the governee, however, in many cases they can be shown to behave like nouns with meanings like 'upside', 'underside', etc., and thus be analyzable *rel-noun-lxm* by the pattern in (51)/(52). Since these items display the same range of meanings as locative prepositions with modifying function, they should be semantically analyzable using the distinctions in Table 1, so that for at least these nouns, the attribute **CLASS** will be declared by *index*, like for locative prepositions (cf. (45) above).

As for so-called 'determiners', these divide into classes such as articles, quantifiers, demonstratives, and numerals, and with co-occurrence restrictions varying very much cross-linguistically. Semantically, the analysis of *quantifiers* should most likely follow the lead of MRS analysis (cf. chapter 3), whereas for *articles* a propagation up to the NP-node of a specification +/- of a feature **DEF** (as done in LFG) may well be the more natural analysis. As our main focus here is the analysis of verbs and verb constructions, we do not go further into the treatment of determiners in TypeGram.

5. Morphological Causative and Applicative Formation

Among the most common valence-increasing operations across languages are *Morphological Causativization* and *Applicative Formation*. TypeGram addresses both. The linguistic literature on causation commonly assumes that the notion 'cause' has two arguments, the second of which is necessarily a proposition, and the first of which can also be a proposition, as in 'John's singing caused the room to become empty'. Our type representing the notion 'proposition' is here *sit*,[7] hence we may define the construal of causation mentioned as follows:

(54)
```
causation-with-causingevent := act12-rel &
 [ PRED cause-rel,
  ACT1.K sit,
  ACT2.K sit ].
```

For a construction like 'John caused the room to become empty', on the other hand, it may be proposed that the ACT1 is an *individual* rather than a *proposition*. We take this option into account by defining a further type:

(55)
```
causation-with-causer := act12-rel &
 [ PRED cause-rel,
   ACT1.K indiv,
   ACT2.K sit ].
```

In morphological causative constructions, it is the latter construal which is relevant. In these constructions, syntactic and semantic form generally are not 'isomorphic', since what in the semantic structure will be the embedding of one proposition as argument of the predicate 'cause', is reflected at syntactic level simply through the addition of one more object or oblique to the same set of GFs.

5.1. Parameters and derivation of morphological causatives

One of the main parameters of morphological causatives resides in whether the 'caused' event is expressed by an intransitive or transitive verb. In the former case, the 'subject' of the caused event - often referred to as the *causee* of the construction - is realized as an *object* of the verb in its derived

[7] Cf. (9) above.

form. If the 'caused' event is represented by a transitive verb, the causee may in principle end up either as object or oblique of the derived verb, and the 'underlying' object also has these options, although there will typically be at least one object.

In either case, the analysis of the morphological causative will be phrased as a *derivational* process, which is technically a unary branching structure where the stem of the causative form is represented as the daughter constituent and the causative form itself as the mother constituent; such a constellation is generally referred to as a *lexical rule*, similar to an inflectional rule except that the rules now in question derive lexemes from lexemes, while inflectional rules derive words.

The rule deriving a morphological causative from an intransitive verb stem can be given the definition (56), deriving a transitive construction with an *individual* as *causer* (stated through the specification '**ACTANTS** causation-with-causer') and the *causee* expressed as *object* ('*trCs*' standing for 'transitive formed by Causativization'); the **CAUS-DERIVED** feature is for forestalling possible iteration of the rule. For the affixation itself, suppose that we are for now working with a 'bantoid' version of English, where verb stems and tense inflections are as in English, but verb extension affixes are used for Causativization, Applicatives and Passive, string-realized as *CAUS, APPL* and *PASS*, respectively, so that the grammar will produce instances of the string *chaseCAUS*, where *CAUS* is added to the stem *chase* by a lexical rule.

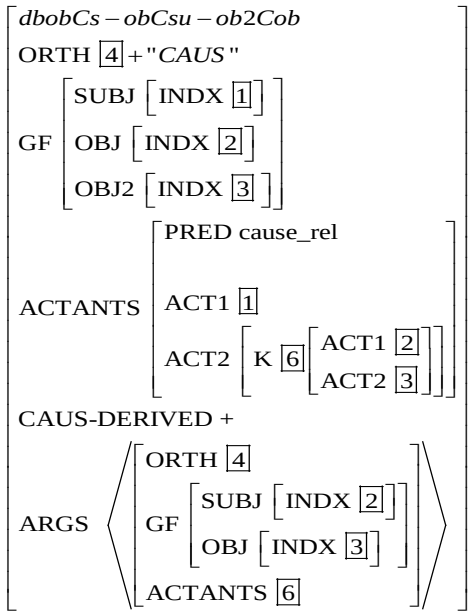
(56)

```
v-trCs_lrule := obj-sign &
 [ ORTH #4+"CAUS"
   GF.SUBJ.INDX #1,
   GF.OBJ.INDX #2,
   ACTANTS causation-with-causer &
           [ACT1 #1,
           ACT2.K act1-rel & #3 & [ ACT1 #2 ]],
   LEXEME +,
   CAUS-DERIVED +,
   ARGS < [ ORTH #4
              HEAD verb,
              COMPS <>,
              GF.SUBJ.INDX #2,
              CAUS-DERIVED -,
              ACTANTS #3 ] > ].
```

When the caused event has more than one participant, the construction has to provide a GF both for the causee and for the other participant(s), and here languages will differ as to whether both/all become objects, or one of them becomes object and the other(s) oblique, and in the latter case, which of them becomes what. (See Kroeger 2004 for a presentation of some principles and strategies at work here.) Below is a sign for a two-participant caused event where the causee is realized as **OBJ** and the other participant as **OBJ2**. This we refer to as a 'double object construction', referring to the two objects as 'first object' and 'second object'. The sign type here in question may be called *dbobCs-sign*, and to precisely indicate which 'underlying' GFs become realized as what, we add the specification *obCsu-ob2Cob*, which means: *object is derived by Causativization from subject, and object2 is derived by Causativization from object*: [8]

[8] The two other conceivable realizations will correspondingly receive the types *oblCsu-obCob* for when the causee is expressed as oblique and the other participant of the caused event as (direct) object, and *obCsu-oblCob* for when the causee is expressed as object and the other participant of the caused event as oblique.

(57)



Also here the derivational rule attaches the causative morpheme. An alternative is to let only inflectional rules perform affixation, and thus use an inflection rule to integrate the addition of 'CAUS'. With a specification like CAUS-DERIVED + available in the output, this could be used as the defining licenser, and the inflectional rule would be (58), parsing, in the mock-style language, *The woman barkCAUS the dog*.

(58) verb-CAUS_irule :=
%suffix (* CAUS)
word &
[ARGS < verb-lxm & [CAUS-DERIVED +]>].

While the approach in (56) and (57) would carry the standard predictions of the 'mirror principle' (see Kroeger op. cit.), by which the affix attached by a 'later' rule comes farther from the root than the affix attached by an 'earlier' rule (if they both attach their affixes on the same side of the root), the approach using (58) is not constrained by this prediction. We will not go into assessing this issue here, but move on to the next type of derivational process.

5.2. Applicatives and chaining of derivations

Below is an example of an Applicative, showing a transitive construction formed (from 'intransitive oblique') through Applicative formation. We label the construction *trAp-obAobl*, in accordance with the conventions just introduced, where obAobl indicates that the object is 'promoted' from oblique.

(59) (Ex. Citumbuka, by Jean Chavula, pc)

Temwani	wa-gon-**er**-a	mphasa
Temwani	1SM-sleep-Ap-Fv	9mat

'Temwani has slept on a mat'

For similar reasons as for Causatives, we may want to construe this as produced by a derivation. Unlike the case of Causative, the process here involved may be assumed to be meaning preserving. Analogously to above, we introduce a feature '**APPL-DERIVED** *bool*', and let an inflectional rule affixing *APPL* be dependent on a positive specification, called *verb-APPL_irule*, analogous to *verb-CAUS_irule*.

To illustrate the derivation of a double object construction in our mock-English universe, a natural English verb of type *trObl* might be *tell* (as in *tell John about Susan*), and an 'applied' version could be '*Mary tellAPPL Susan John*', illustrating the type *dbobAp-obAobl.*

Consider then an interaction between Causativization and Applicative, again using an example from Citumbuka, illustrating a case where a sequence of applications of first the Applicative rule *dbobAp-obAobl_lrule*, and then an application of Causativization, produce a verb construction of type *ditrOblCsAp-oblCsu_obCob2Aobl_ob2Cob* (= 'ditransitive-plus-oblique construction produced by Causativization preceded by Applicative' - the label reflecting the order of 'peeling off' one process after the other, starting from the result – and mentioning only the GF-changing mappings). The constituent label *oblCsu* here says that the oblique is derived by Causativization from underlying subject, *obCob2Aobl* says that the first object is 'promoted' by Causativization from obj2, before that promoted by Applicative from underlying oblique, and *ob2Cob* says that the second object is derived by Causativization from underlying object:

(60) (Ex.Citumbuka, by Jean Chavula, pc)

Tumbikani wakamuphik**iskir**a Temwa nchunga kwa Mary

Tumbikani	wa-ka-mu-phik-**isk-ir**-a	Temwa	nchunga kwa Mary
Tumbikani	1SM-pst-1OM-cook-Caus-Appl-fV	Temwa	beans 'to' Mary

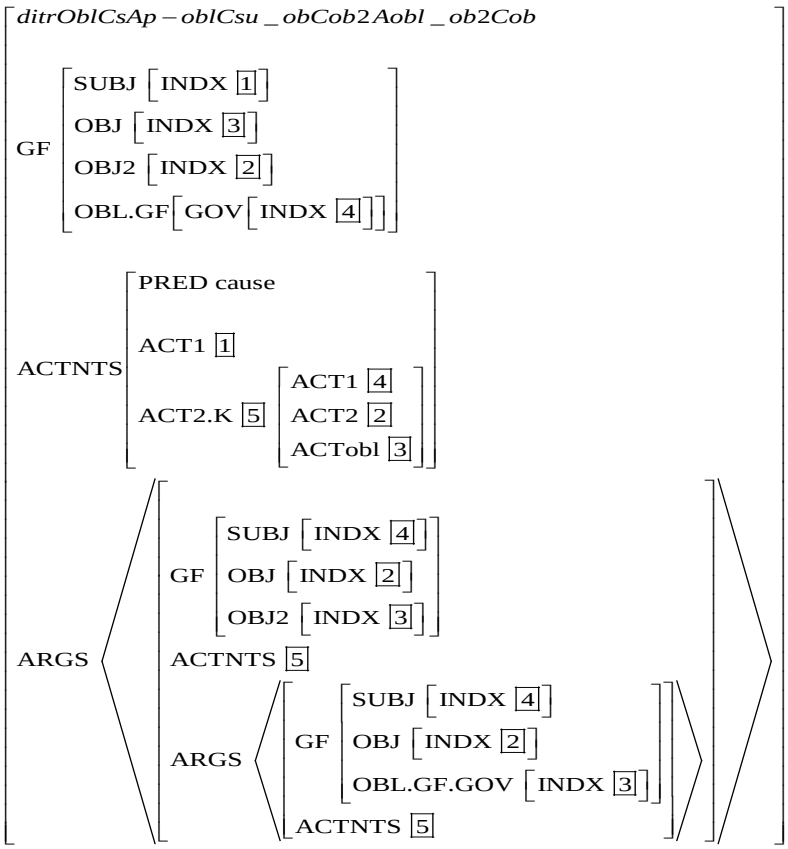
'Tumbikani made Mary cook beans for Temwa'

The resulting pattern expresses a causation with a person-causer, a three-actant caused event, and the **ACT1** of the caused event (the causee) expressed as oblique and the **ACT2** of the caused event as second object, whereas **ACTobl** of the caused event (labelled according to its 'pre-applicative' status), takes the position of first object. This is represented in (61), and (62) adds the derivational history:

(61)

$$\begin{bmatrix} \text{GF} & \begin{bmatrix} \text{SUBJ} & [\text{INDX } \boxed{1}] \\ \text{OBJ} & [\text{INDX } \boxed{3}] \\ \text{OBJ2} & [\text{INDX } \boxed{2}] \\ \text{OBL} & [\text{GOV}[\text{INDX } \boxed{4}]] \end{bmatrix} \\ \text{ACTNTS} & \begin{bmatrix} \text{PRED cause} \\ \text{ACT1 } \boxed{1} \\ \text{ACT2} & \begin{bmatrix} \text{ACT1 } \boxed{4} \\ \text{ACT2 } \boxed{2} \\ \text{ACTobl } \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(62)



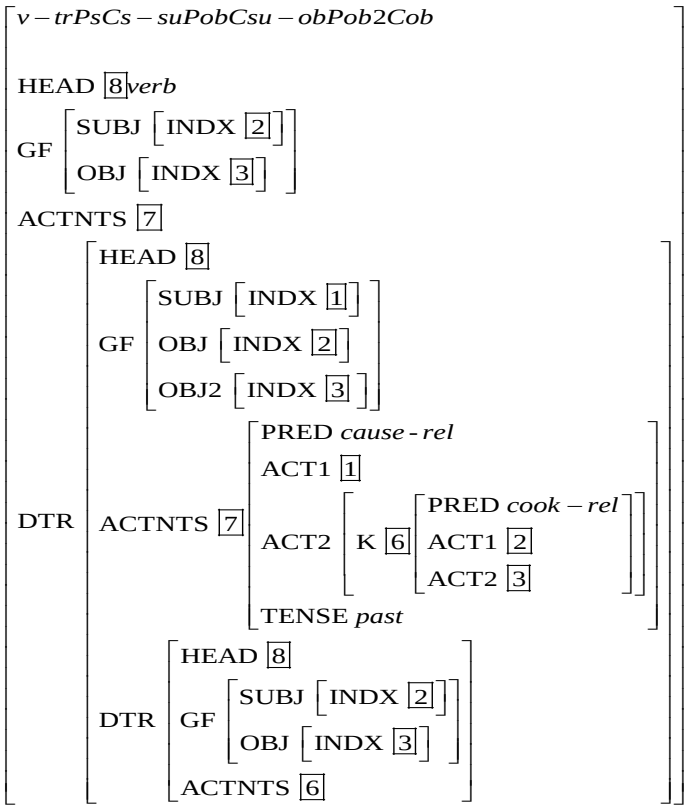
In this example, the causative morpheme precedes the applicative morpheme, both affixes being to the right of the root, which by the 'mirror principle' would suggest that Causativization has applied first, contrary to the analysis given. If the latter is correct, this case would thus count in favour of using inflectional rules for attaching the affixes, as suggested above – cf. the inflectional rule (58) – but there is probably room for exploring the opposite rule ordering, so we make no conclusion here. But we may note that in another example of chaining of derivations discussed in chapter 1, section 2, the proposed order of rule applications depicted in (9), chapter 1, does conform to the mirror principle – the example (10) from chapter 2 is repeated as (63),

(63) (example of (10), from Kiswahili, based on Vitale, 1981:165, quoted in Kroeger, 2004, p.196, ex. (11a))

mke wake a-li-pi-**ish**-**w**-a uji na Sudi
wife his S.agr-PAST-cook-CAUS-Pass-Ind gruel by Sudi
'His wife was made to cook gruel by Sudi'

with the analysis largely repeated as (64) (note that the 'input' is here introduced by the attribute **DTR** taking a sign as value rather than **ARGS** taking as value a list consisting of one sign – these are equivalent representations):

(64) Verb derived by *morphological causative* and subsequently *passive*:



The type of the verb in this case is:

(65) v-trPsCs-suPobCsu-obPob2Cob

where *trPsCs* labels a transitive argument structure derived through morphological causativization followed by passive formation, and the sequence *suPobCsu-obPob2Cob* means that Subj is derived by passive from underlying Obj, in turn derived from underlying Subj, and that Obj is derived by passive from underlying Obj2, in turn derived from underlying Obj. The sequence *suPobCsu* thus corresponds to the AVM segment

GF.SUBJ.INDX 2
DTR.GF.OBJ.INDX 2
DTR.DTR.GF.SUBJ.INDX 2

and could in principle be derived from it, as a compact declaration of this part of the overall AVM. A large set of possible combinations of 'extensions' of this kind can be defined in this code, as laid out in chapter X, section Y.

6. Serial Verb Constructions.

6.1. SVCs as adjunction structures

In our comments on SVCs in chapter 2, sections 4.3 and 5, we mentioned that one proposal is to treat it as a kind of adjunction, with the first VP as head and the second as an adjunct. One argument for this analysis is that the second verb fills no valence requirement of the first verb – in general in SVCs, both verbs are free to occur by themselves, i.e., as heads in non-serial verbal constructions. That a head-complement relation can thereby be argued against, of course does not mean that a head-adjunct analysis is by the same token vindicated – whether the 'standard' functional relations from languages lacking multiverb constructions are sufficient to accommodate, e.g., SVCs, is indeed a crucial question. Still, this is the basis on which we frame our current analyses.

First of all, to technically allow for a head-adjunct analysis of SVCs, we need to give verbs in general the possibility of modifying other verbs; while unexpected from the viewpoint of European languages, there is no problem connected to this move. Otherwise the following remarks apply.

As noted at the end of chapter 2, section 5 for cases of *object sharing* and *switch sharing*, since the relevant **COMPS** lists of the daughter VPs will be empty at that stage of combination where identities must be be assessed, the object-related requirements have to be imposed on the path **GF.OBJ**, since only the GF information is propagated unchanged up through the combination tree. In cases of object sharing, moreover, the second VP will typically be one where a transitive verb occurs without its object. Thus, the analysis of these constructions not only requires a way of assigning identities, but also a way of accounting for 'missing' objects. The most obvious strategy for the latter will be a 'de-transitivizing' rule, which applies only inside of the second VP in an SVC. The operation stated in (66b) below does that, as an operation turning a transitive lexeme into an intransitive one, and the way in which this operation gets related to the VP's participation in an SVC is through its **MOD** specification, which stipulates that this happens only when the VP in question follows another VP whose object is identical to the object of the VP in question.

This strategy of referencing a preceding VP through the **MOD** attribute is used also in (66a) and (66c), then 'deleting' the *subject* under a stipulated identity with the appropriate item in the preceding VP. This item is the *subject* in (66a) ('*subject sharing'*), and the *object* ('*switch sharing'*), in (66c). When an SVC displays both subject sharing and object sharing, both (a) and (c) will apply.

(66)

a.

$$\begin{bmatrix} subj-sharing-rule \\ \text{HEAD verb} \\ \text{SPR } \langle\ \rangle \\ \text{COMPS } \boxed{2} \\ \text{MOD } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{SPR } \langle \boxed{1} \rangle \\ \text{GF } [\text{SUBJ } \boxed{1}] \end{bmatrix} \right\rangle \\ \text{GF } \boxed{5}[\text{SUBJ } \boxed{1}] \\ \text{ACTNTS } \boxed{4} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{SPR } \langle \boxed{1} \rangle \\ \text{COMPS } \boxed{2} \\ \text{GF } \boxed{5} \\ \text{ACTNTS } \boxed{4} \end{bmatrix} \right\rangle \end{bmatrix}$$

b.

$$\begin{bmatrix} obj-sharing-rule \\ \text{HEAD verb} \\ \text{SPR } \boxed{2} \\ \text{COMPS } \langle\ \rangle \\ \text{MOD } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{GF } [\text{OBJ } \boxed{1}] \end{bmatrix} \right\rangle \\ \text{GF } \boxed{5}[\text{OBJ } \boxed{1}] \\ \text{ACTNTS } \boxed{4} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{SPR } \boxed{2} \\ \text{COMPS } \langle \boxed{1} \rangle \\ \text{GF } \boxed{5} \\ \text{ACTNTS } \boxed{4} \end{bmatrix} \right\rangle \end{bmatrix}$$

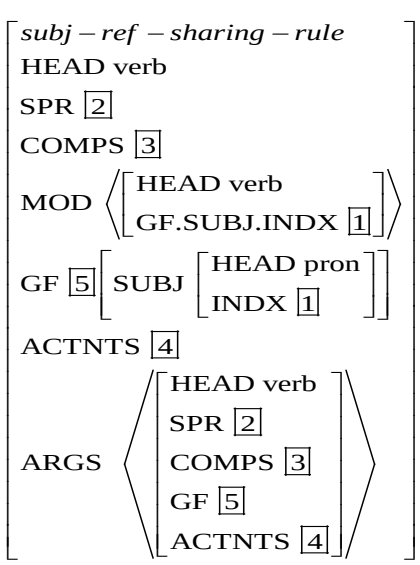
c.

$$\begin{bmatrix} switch-sharing-rule \\ \text{HEAD verb} \\ \text{SPR } \langle\ \rangle \\ \text{COMPS } \boxed{2} \\ \text{MOD } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{GF } [\text{OBJ } \boxed{1}] \end{bmatrix} \right\rangle \\ \text{GF } \boxed{5}[\text{SUBJ } \boxed{1}] \\ \text{ACTNTS } \boxed{4} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{SPR } \langle \boxed{1} \rangle \\ \text{COMPS } \boxed{2} \\ \text{GF } \boxed{5} \\ \text{ACTNTS } \boxed{4} \end{bmatrix} \right\rangle \end{bmatrix}$$

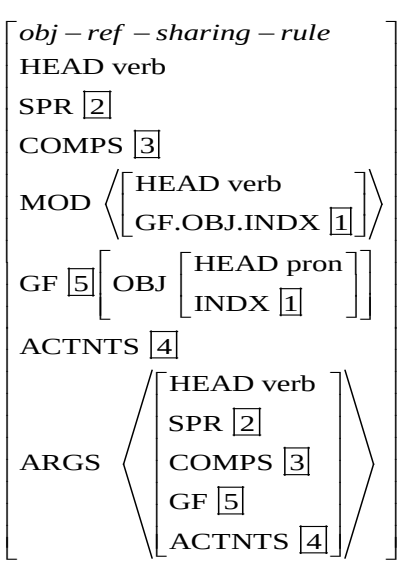
In (67) below, the same patterns are repeated for cases where the relevant items are not deleted, but appear as *pronouns* (cliticized or not) – in such cases we will speak of *reference sharing*.

(67)

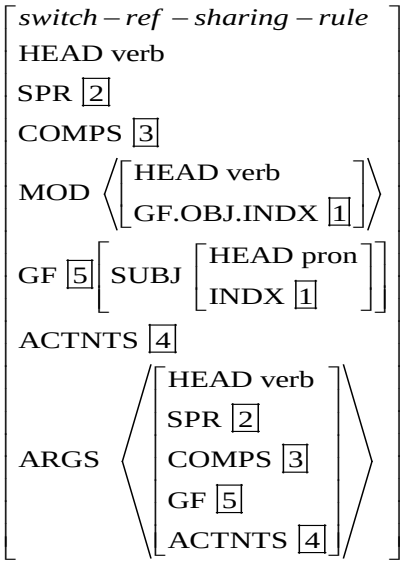
a.



b.



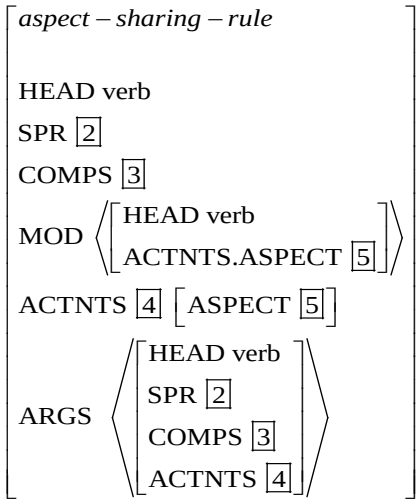
c.



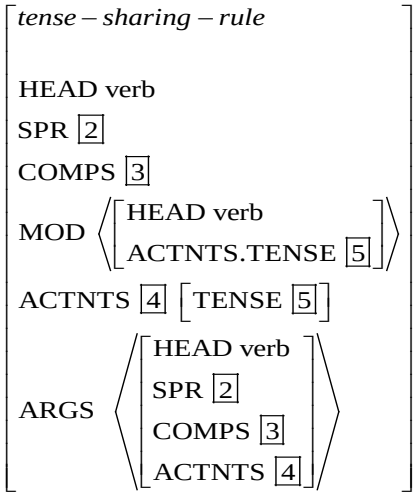
A similar set of rules can be made to constitute what we may call the *TAM-sharing* cluster, where again more than one of the rules can apply ((68)). (Most of the material in these specifications can be stated in a super-type of these rules, but for convenience of inspection we assemble the all information in one AVM in each case.)

(68)

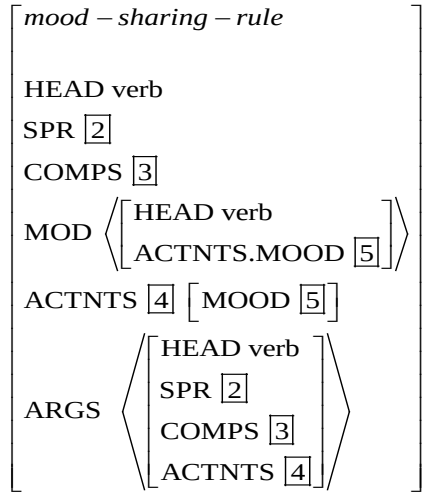
a.



b.



c.



These rules, combined with the general *Head-Modifier Rule* for the point where one VP combines with another, will be adequate for parsing most known patterns of SVCs, and of any length, since what is the head of one adjunct can be an adjunct of another VP. Assuming that SVCs are composed of binary branching structures, thus, the rules given constitute recursive mechanisms for verb sequences of any length.[9]

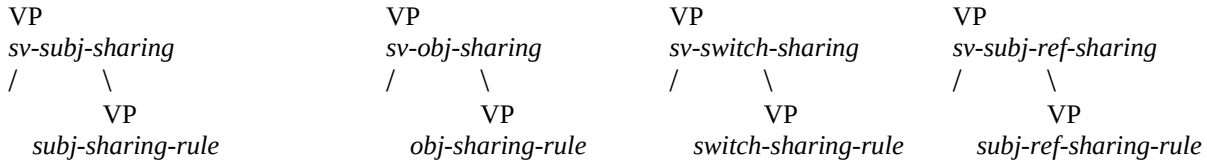
[9] Stipulations in the combination mechanism can regulate whether in such structures of iterated adjunctions, the over-all branching will be leftwards or rightwards. Empirical evidence for what is to be preferred may come from 'movement' or

6.2. Formal type-assignment to SVCs

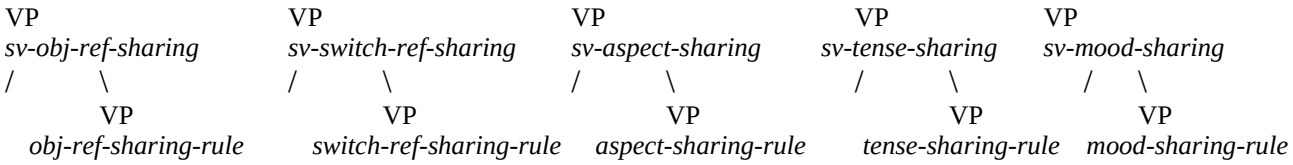
While actual *parsing* of the various types of SVCs is essential to the present grammar, *classification* is also essential. In the previous section we mentioned *type* labels like the one entered in (65) for derived verb forms; such labels are useful as terms for classification of derived verb types, and we will want a similar system for classifying SVCs. Grammar-technically, SVCs are sentence level entities, and in classifications like tree-banking, for instance, the basis for classification is the set, and order, of *rules* taking part in the derivation of a sentence. What will be relevant here are only those rules directly relevant to SVCs, and to make them 'visible' for classification purposes they should preferably be reflected at the top AVM of the derivational tree, just like GFs and verb types (verbs being heads, their types are in general propagated to this level). The operations of the rules in (66), (67) and (68) are restricted to adjuncts, from which their type labels will not in the standard case propagate up the head projection. However, in the reference made by each rule to the head through the **MOD** feature, a specification can be induced to that head VP, so that in each VP+VP combination, the head VP, and thus also the mother VP, can carry the specification. This specification will have the rule name of the relevant rule in (66)-(68) with *sv-* prefixed. Thus, rule (66a) will have the fuller formulation (69), and correspondingly for the other rules, as summarized in (70):

(69)
$$\begin{bmatrix} subj-sharing-rule \\ \text{HEAD verb} \\ \text{SPR } \langle\ \rangle \\ \text{COMPS } \boxed{2} \\ \text{MOD } \left\langle \begin{bmatrix} sv-subj-sharing \\ \text{HEAD verb} \\ \text{SPR } \langle \boxed{1} \rangle \\ \text{GF } [\text{SUBJ } \boxed{1}] \end{bmatrix} \right\rangle \\ \text{GF } \boxed{5}[\text{SUBJ } \boxed{1}] \\ \text{ACTNTS } \boxed{4} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{HEAD verb} \\ \text{SPR } \langle \boxed{1} \rangle \\ \text{COMPS } \boxed{2} \\ \text{GF } \boxed{5} \\ \text{ACTNTS } \boxed{4} \end{bmatrix} \right\rangle \end{bmatrix}$$

(70) Types assigned to mother- and adjunct VP-nodes in SVC syntactic combination trees:

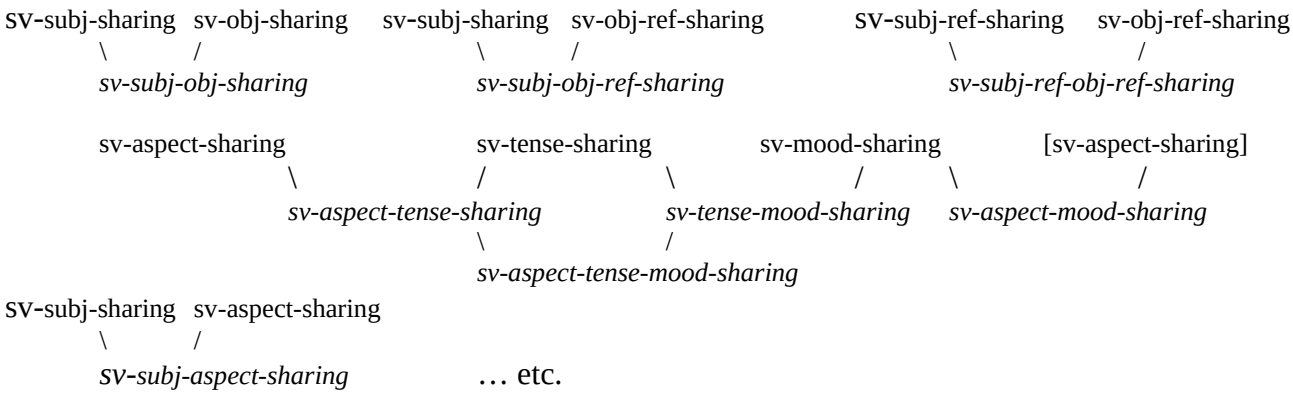


'gapping' possibilities involving sequences of VPs while leaving other VPs unaffected, but we will not go into facts from any single language here. (Intuitively, either construal is probably acceptable.)



To accommodate multiple operations, the following type unifications are in turn declared, meaning that if the last VP in a VP+VP combination for instance has undergone both *subj-sharing-rule* and *obj-sharing-rule*, the mother VP will have the type *sv-subj-obj-sharing*, whereby both rule applications are recorded in the AVM:

(71) Some type hierarchies:



For binary SVCs, these type specifications will ensure that the top S-node of the structure carries a type among those defined in (71) (or the indicated extensions of it), thereby providing a classificatory specification for the SVC.

For SVCs with three or four verbs, these classifications are not relevant, since the types in (71) only indicate the relationship between two adjacent VPs. If V1 and V2 in an SVC have a certain sharing relationship R1 but V2 and V3 have another – R2 -, then we do not yet have a way of counting R1 and R2 together, if we wish to have a top level display of the whole sequence in something like a single specification. Since a common way of analytically navigating inside an SVC is indeed by reference to 'V1', 'V2', 'V3', etc., it will be desirable to bring these labels out as equal-level attributes in the AVM of an SVC, and from this vantage point also summarize in one sequence the salient relations between the various VPs.

No verb by itself could reasonably be lexically classified as 'first svc-verb', 'second svc-verb', etc., since any verb can occur at any place in a sequence, and by itself. Hence, if we want attributes like 'V1', 'V2', etc. to appear in the feature structure of an SVC, they have to be introduced in the course of the combinations. To this end, a rule like (69) can be recast as (72a), where the labels 'V1' and 'V2' are brought into play for subject-sharing, and in (72b) correspondingly for switch-sharing. The rules in (72) both inherit from *head-modifier-phrase* and *sv* (see (73b)). An instantiation of (72b) is shown in (46) in chapter 1.[10]

[10]The effect described in addition needs to be propagated from the highest VP node to the S node, by the following addition to the *Head-Specifier Rule* already established (in (5)):

(72) a.

$$\begin{bmatrix} sv-subj-sharing-phrase \\ \text{V1}\ \boxed{1} \\ \text{V2}\ \boxed{2} \\ \text{ARGS}\ \left\langle \boxed{1}\left[sv-subj-sharing\right], \boxed{2}\left[subj-sharing-rule\right] \right\rangle \end{bmatrix}$$

b.

$$\begin{bmatrix} sv-switch-sharing-phrase \\ \text{V1}\ \boxed{1} \\ \text{V2}\ \boxed{2} \\ \text{ARGS}\ \left\langle \boxed{1}\left[sv-switch-sharing\right], \boxed{2}\left[switch-sharing-rule\right] \right\rangle \end{bmatrix}$$

In (73) below is defined a set of types for characterizing SVCs. (73a) is a set of verb types that can be added to what we have already developed in section 3, to be used further in (73b,c). The types in (b) are those that introduce the labels 'V1' and 'V2' explicitly, and make characterizations of identities across the VPs in an SVC. The types in (c) are for characterizing the valence of the component VPs of an SVC. These types are all purely classificatory, in that they reflect information expressed in an AVM, but do not 'drive' the parsing process or add to the coverage of the grammar.

(73)
a.

```
vp := phrase & [ HEAD verb,
                 COMPS < > ].
v-sign := syn-struc & [ HEAD verb ].
v-word := word & v-sign.
v-intr-sign := intr-sign & v-sign.
v-intrObl-sign := intrObl-sign & v-sign.
v-tr-sign := tr-sign & v-sign.
v-trObl-sign := trObl-sign & v-sign.
v-ditr-sign := ditr-sign & v-sign.
v-dbob-sign := dbob-sign & v-sign.
```

(i)

$$\begin{bmatrix} svc-subj-sharing-phrase \\ \text{V1}\ \boxed{1} \\ \text{V2}\ \boxed{2} \\ \text{ARGS}\ \left\langle \left[\text{HEAD noun}\right], \begin{bmatrix} subj-sharing-rule \\ \text{V1}\ \boxed{1} \\ \text{V2}\ \boxed{2} \end{bmatrix} \right\rangle \end{bmatrix}$$

Note that rules like (i) and (72) display only **V1** and **V2**, not **V3** or **V4** or higher. Technically, however, in the constellation (ii)

(ii)

$$\begin{bmatrix} \text{V1} \\ \text{V2}\begin{bmatrix} \text{V1} \\ \text{V2} \end{bmatrix} \end{bmatrix}$$

the path **V2.V2** can be interpreted as **V3**, and similarly for the higher numbers.

```
v-ditrObl-sign := ditrObl-sign & v-sign.
v-dbobObl-sign := dbobObl-sign & v-sign.
v-trCs-sign := trCs-sign & v-sign.
v-dbobCs-sign := dbobCs-sign & v-sign.
v-trOblCs-oblCsu-sign := trOblCs-oblCsu-sign & v-sign.
v-trOblCs-obCsu-sign := trOblCs-obCsu-sign & v-sign.

b.
sv := syn-struc &
 [ V1 v-sign,
   V2 v-sign ].
sv3 := sv &
 [ V3 v-sign ].
sv4 := sv3 &
 [ V4 v-sign ].
sv_suIDALL := sv &
 [ V1.GF.SUBJ #1,
   V2.GF.SUBJ #1 ].
sv3_suIDALL := sv3 & sv_suID &
 [ V2.GF.SUBJ #1,
   V3.GF.SUBJ #1 ].
sv4_suIDALL := sv4 & sv3_suID &
 [ V3.GF.SUBJ #1,
   V4.GF.SUBJ #1 ].
sv_obIDALL := sv &
 [ V1.GF.OBJ #1,
   V2.GF.OBJ #1 ].
sv3_obIDALL := sv3 & sv_obID &
 [ V2.GF.OBJ #1,
   V3.GF.OBJ #1 ].
sv4_obIDALL := sv4 & sv3_obID &
 [ V3.GF.OBJ #1,
   V4.GF.OBJ #1 ].
sv_aspIDALL := sv &
 [ V1.ASPECT #1,
   V2.ASPECT #1 ].
sv3_aspIDALL := sv3 & sv_aspID &
 [ V2.ASPECT #1,
   V3.ASPECT #1 ].
sv4_aspIDALL := sv4 & sv3_aspID &
 [ V3.ASPECT #1,
   V4.ASPECT #1 ].
sv_suObIDALL := sv_suID & sv_obID.
sv3_suObIDALL := sv3_suID & sv3_obID.
sv4_suObIDALL := sv4_suID & sv4_obID.
sv_suAspIDALL := sv_suID & sv_aspID.
sv3_suAspIDALL := sv3_suID & sv3_aspID.
sv4_suAspIDALL := sv4_suID & sv4_aspID.
sv_suObAspIDALL := sv_suObID & sv_aspID.
sv3_suObAspIDALL := sv3_suObID & sv3_aspID.
```

sv4_suObAspIDALL := sv4_suObID & sv4_aspID.

c.
v1intr := sv &
 [V1 v-intr-sign].
v2intr := sv &
 [V2 v-intr-sign].
v3intr := sv3 &
 [V3 v-intr-sign].
v4intr := sv4 &
 [V4 v-intr-sign].
v1tr := sv &
 [V1 v-tr-sign].
v2tr := sv &
 [V2 v-tr-sign].
v3tr := sv3 &
 [V3 v-tr-sign].
v4tr := sv4 &
 [V4 v-tr-sign].
v1ditr := sv &
 [V1 v-ditr-sign].
v2ditr := sv &
 [V2 v-ditr-sign].
v3ditr := sv3 &
 [V3 v-ditr-sign].
v4ditr := sv4 &
 [V4 v-ditr-sign].

This type apparatus will allow for type characterizations like (74), (a) for a three-VPs SVC with subject and aspect sharing and V1 and V2 being transitive and V3 intransitive, (b) for a two-VPs SVC with switch sharing (‘v1obIDv2Su ‘ meaning that **V1**’s object is identical to **V2**’s subject) and aspect sharing, and (c) for a two-VPs SVC with subject-, object- and aspect sharing:

(74) a. sv3SuAspIDALL-v1tr-v2tr-v3intr
b. svAspIDALL- v1tr-v2tr-v1obIDv2Su
c svSuObAspIDALL

Instances of these are the following:

(75) For (74a):

Á-gbele	gbɛ	á-ha	bo
3.PRF-open	road	3.PRF-give	2S
V	N	V	Pron

‘You have been granted permission.’

For (74b):

Kofi	to-o	ne	nan	wɔ-ɔ	Kwame
Kofi	throw-PERF	3Poss	leg	pierce-PERF	Kwame
N	V	Pron	N	V	N

‘Kofi kicked Kwame’

For (74c):

	Ama	tu-u		bayerɛ	twitwa	noa	di-i
Ama	uproot-PERF		yam	cut	cook	eat-PERF	
N	V		N	V	V	V	

‘Ama uprooted (tuber of) yam, cut it in pieces, boiled them (and) ate’

Contrary to types like (66) which reflect valence frames of verbs and thereby determine the parsing process of sentences containing the verbs, types like those in (74) cannot be associated with any particular verb of a given sentence, and thereby will not play a role in a parser, except as a type which can be associated with the parse result for a given sentence for classificatory purposes. This theme will be developed in chapter XX.

Our inventory of top level attributes in the type *sign* (i.e., attributes declared by *sign*) is now (76), compared to (10a) at the start of the chapter, and the types *sign*, *gramfct*, *semarg*, *aspect* and *sit*, together with their hierarchies and attributes declared by them, and so on, have in turn been added, via the stages recorded in (10b), (36) and (41).

(76) **sign**
[**V1 sign,**
V2 sign,
HEAD pos,
SPR *list*,
COMPS *list*,
MOD *list*,
GF gramfct,
INDX semarg,
ASPECT aspect,
ACTNTS sit,
ARGS *list*,
LEXEME bool,
CAUS-DERIVED bool,
APP-DERIVED bool].