From Grammar-Independent Construction Enumeration to Lexical Types in Computational Grammars

Lars Hellan
NTNU
N-7491 Trondheim
Norway
lars.hellan@hf.ntnu.no

Abstract

The paper presents a code for enumerating verb-construction templates, from which lexical type inventories of computational grammars can be derived, and test suites can be systematically developed. The templates also serve for descriptive and typological research. The code is string-based, with divisions into slots providing modularity and flexibility of specification.

1 Introduction

This paper presents a code for enumerating verb-construction templates. The code is string-based, with divisions into slots providing modularity and flexibility of specification. The templates provide slots for specifying, relative to a construction

- part of speech (POS) of the head
- grammatical relations exposed
- valence bound items
- thematic roles expressed
- situation type
- aspect (Aktionsart)
- part of speech of valence-bound items.

(These parameters altogether cover what is commonly referred to as 'argument structure' relative to the main predicate.) The code is outlined in sections 2-5, and 8.

From the verb construction templates, lexical type inventories of computational grammars can be derived (section 6). The design offers a systematic way of organizing test suites, and herewith improved means of defining intra- and cross-framework reference points of coverage and depth of analysis. The template code also lends itself for descriptive and typological research (section 7).

The design is not geared to any particular framework of computational grammar or linguistics. Examples will be offered relative to HPSG- and LFG- grammars, and the actual conversions from templates to lexical types so far developed relate to HPSG grammars using the LKB platform (cf. (Copestake 2002)), based on the 'HPSG Grammar Matrix' design ((Bender et al. 2002)). Our exposition will be based on the design as it relates to the LKB-grammar *NorSource* (cf. (Beermann and Hellan 2004)) and a Verb-Construction enumeration for Norwegian.

The enterprise here presented has lines going back at least to the mid and late 80ies, both regarding test suite development (e.g., (Flickinger et al. 1987), (Lehmann et al. 1996)) and argument frame inventories ((Hellan et al. 1889)).

2 Code for Template Enumeration

By a *template* for a verb construction we understand a standardized way of exposing selected features of the construction. Exposed features are *classificatory* features, and in this respect, a template may be regarded as a *type*.

A system for enumerating templates should be designed such that they are, internal to a given language, complete and transparent, and across languages, comparable both in templates shared and in templates distinct. Technologically they should be as low level as possible, and platform independent, and be equally accessible to practising field linguists as to NLP researchers in computational settings. With such desiderata in mind,

we develop a code residing simply in *strings* of symbols with a minimum of internal syntax.

The basic structural parts of such strings are referred to as *slots*. In the slot specification, the following conventions are observed:

* Slots are interconnected by '-' (hyphen).
* Distinct items inside a slot are interconnected by '_' (underline).
* An item label containing neither '-' nor '_' is an uninterrupted string of letters. Internal composition is indicated by alternation between small and capital letters.

The format can be applied also to non-verbal constructions, but we here focus exclusively on verbal ones. These have a template structure with *five* slots:

Slot 1: POS of the head, and diathesis information.
Slot 2: Valency, or transitivity specification (e.g., `intr`, `tr`, `ditr`,...).
Slot 3: Dependents' specification (syntactic and referential properties of arguments).
Slot 4: Participant roles.
Slot 5: Situation type (written in SMALL CAPS).

Slots 1 and 2 are always filled, the others need not be. A slot not specified is not displayed as empty; however, the sets of labels defined for the various slots are disjoint. Likewise, no labels are distinguished in terms of capital letter vs. not. (1) illustrates the composition, for a type instantiated by the clause *Mary throws the ball*:

(1) v-tr-obDir-suAg_obEjct-EJECTION

Slot 1 here says the head is Verb, slot 2 says that the construction is transitive, slot 3 says that the object har a directional function, slot 4 says that the thematic roles are 'agent', expressed by the subject, and 'ejected', expressed by the object, and slot 5 says that the situation type is one characterizable as 'ejection'.

We start with a survey of the labels used for slot 2, valence information. First, for the use of the notions `intr`, `tr`, `ditr`, the following definitions apply. By a *direct syntactic argument* of a verb, we understand a nominal constituent syntactically related to the verb as *subject-of*, *direct object-of*, or *indirect object-of*, and any clausal constituent with either of these functions. (This *in*cludes expletive subjects and objects, and *ex*cludes clausal constituents in extraposed position; it also excludes any NP or clause governed by a preposition or another relation-item.) An *intransitive* construction is then one with only SUBJECT as a direct syntactic argument, a *transitive* construction has a SUBJECT and one OBJECT as direct syntactic arguments, and a *ditransitive* construction has a SUBJECT and two OBJECTs as direct syntactic arguments. Any valence-bound item other than those now described is represented by an extension of the above transitivity-strings, for instance, in the strings `intrObl` and `trObl`, `Obl` means 'oblique', that is, in addition to the number of arguments represented by `intr/tr`, there is a PP with 'selected' status.

The valence slot includes information as to referential status of the arguments. We say that a direct syntactic argument is *standardly linked* when it has referential content and has a semantic argument function relative to the verb. This *ex*cludes expletive subjects and expletive objects, as well as 'raised' full NPs. The following substrings in slot 2 indicate the presence of items that are *not standardly linked*:

`Impers` ('impersonal'), `Presentational`, `Epon` ('extraposition'), `Nrf` ('non-referential'), `Rais` ('item raised, i.e., non-argument'), `Nrg` ('non-argument' - used in slot 3)).

Specifications are sought to be non-redundant. For instance, the string `intrEpon` occurring in slot 2 entails that there is an expletive subject, and when used for a langauge like English, there is no need to say elsewhere that the subject is expletive. Since what is redundant relative to one language may not be so relative to another, questions of language-parametrized interpretation of code may arise; however, we do not have a basis yet for suggesting whether and how this would be accommodated.

3 Valency labels

The slot for valency, slot 2, has around 45 possible specifications relevant to Norwegian, and we state those in full, to give an impression of what may be the expected scope of this slot; an example illustrates each type:

`intr` = intransitive, i.e., with only SUBJECT as direct syntactic argument.
`intrImpers` = impersonal intransitive, i.e., SUBJECT is an expletive not linked to any other item in the clause. (Ex.: *det regner* 'it rains')
`intrImpersPrtcl` = impersonal intransitive with an aspectual particle. (Ex.: *det klarner opp* 'it clears up')
`intrImpersObl` = impersonal intransitive with an oblique argument. (Ex.: *det synger i fjellene*

'it sings in the mountains')

`intrPresentational` = intransitive with a presentational structure, i.e., an expletive subject and an indefinite NP.

`intrDirPresentational` = `intrPresentational` where the presented NP has a directional function**.** **(**Ex.: *det springer en mann* 'there runs a man')

`intrPresentationalLoc` = `intrPresentational` with a locative constituent. (Ex.: *det sitter en mann i stolen* 'there sits a man in the chair')

`intrDir` = intransitive where the subject has a directional function. (Ex.: *gutten løper* 'the boy runs')

`intrAdv` = intransitive with an obligatory adverb. (Ex.: *han fungerer godt* 'he functions well')

`intrPrtcl` = intransitive with an aspectual particle. (Ex.: *regnet varer ved* 'the rain lasts')

`intrObl` = intransitive with an oblique argument. (Ex.: *jeg snakker om Ola* 'I talk about Ola')

`intrOblRais` = intransitive with an oblique argument from which an NP has been 'raised'. (Ex.: *han later til å komme* 'he appears [to] to come')

`intrScpr` = intransitive with a secondary predicate (Small Clause predicate). (Ex.: *gutten virker syk* 'the boy seems sick')

`intrEpon` = intransitive with an 'extraposed' clause. (Ex.: *det hender at han kommer* 'it happens that he comes')

`intrPrtclEpon` = intransitive with an 'extraposed' clause and an advparticle. (Ex.: *det hører med at han kommer* Mock Eng: "it hears along that he comes")

`intrOblEpon` = intransitive with an 'extraposed' clause and an oblique argument. (Ex.: *det haster med å rydde* Mock Eng: "it urges with to tiden up")

`intrPrtclOblEpon` = intransitive with an 'extraposed' clause, an oblique argument, and an advparticle. (Ex.: *det ser ut til at han kommer* Mock Eng: "it looks out to that he comes")

`intrPrtclOblRais` = intransitive with an oblique argument from which an NP has been 'raised', and an advparticle. (Ex.: *han ser ut til å komme* Mock Eng: "he looks out to to come")

`intrImplicobjObl` = intransitive with implicit object, followed by PP (Ex.: *han sølte på seg* 'he spilled on himself')

`tr` = transitive, i.e., with SUBJECT and one OBJECT.

`trDir` = transitive, where the subject is understood in a directional function. (Ex.: *Kari når toppen* 'Kari reaches the top')

`trPrtcl` = transitive with an advparticle. (Ex.: *Kari fant ut svaret* 'Kari found out the answer')

`trPresentational` = presentational structure with an NP preceding the 'presented' NP. (Ex.: *det venter ham en ulykke* 'there awaits him an accident')

`trObl` = transitive with an oblique. (Ex.: *jeg sparker Ola i baken* 'I kick Ola in the butt')

`trEponSu` = transitive with an extraposed clause correlated with the subject, and an argument object. (Ex.: *det bekymrer meg at han kommer* 'it worries me that he comes')

`trEponOb` = transitive with an extraposed clause correlated with the object, and an argument subject.. (Ex.: *vi muliggjorde det at han fikk innreisetillatelse* 'we possible-made it that he got entrance visa')

`trScpr` = transitive with a secondary predicate (Small Clause predicate). (Ex.: *han sparket ballen flat* 'he kicked the ball flat')

`trNrf` = transitive whose object is non-referential. (Ex.: *Kari skammer seg* "Kari shamess herself" - 'Kari is ashamed')

`ditr` = ditransitive, i.e., with SUBJECT and two OBJECTs (here referred to by the traditional terms 'indirect' ('iob') and 'direct' object, when distinction is necessary).

`ditrObl` = ditransitive with oblique. (Ex.: *jeg kaster Ola kakestykker i ansiktet* "I throw Ola cakes in the face" - 'I throw cakes in the face of Ola')

`ditrNrf` = ditransitive whose indirect object is non-referential. (Ex.: *han foresetter seg å komme* 'he [foresetter] himself to come)

`copAdj` = predicative copular construction with adjectival predicative. (Ex.: *han er snill* 'he is kind'). (Similarly: `copN`. (Ex.: *han er bonde* 'he is peasant'); `copPP` (Ex.: *forestillingen var under enhver kritikk* 'the performance was below critique'); `copPredprtcl` (Ex.: *Ola er som en gud* 'Ola is like a god'))

`copIdN` = identity copular construction with nominal predicative. (Ex.: *dette er mannen* 'this is the man'.) (Similarly: `copIdAbsinf` (Ex.: *oppgaven er å spise silden* 'the task is to eat the herring'.); `copIdDECL` (Ex.: *problemet er at han spiser silden* 'the problem is that he eats the herring'.); `copIdYN` (Ex.: *problemet er om han spiser silden* 'the problem is whether he eats the herring'.); `copIdWH` (Ex.: *spørsmålet er hvem som spiser silden* 'the question is who eats the herring'.))

`copEponAdj` = predicative copular construction with adjectival predicative and the 'logical subject' extraposed. (Ex.: *det er uvisst hvem som kommer* 'it is uncertain who that comes'.) Similarly: `copEponN` (Ex.: *det er et spørsmål hvem som kommer* 'it is a question who comes'.); `copEponPP` (Ex.: *det er hinsides diskusjon at han kommer* 'it is beyond discussion that he comes'.); `copEponPredprtcl`

(Ex.: *det var som bestilt at han tapte igjen* 'it was like booked that he lost again'.))

4 Dependents' labels

The slot for dependents, slot 3, has around 40 labels relevant for Norwegian. Each is built up with a first part indicating the grammatical function of the item specified (`su, ob, iob, obl, sc, epon`), and the remainder providing the specification, possibly also with some internal structure. The following lists most of them:

`suExpl` = subject is an expletive.

`suDECL` = subject is a declarative clause. (Similarly: `suYN` = subject is a yes-no-interrogative clause; `suWH` = subject is a wh-interrogative clause; `suAbsinf` = subject is an infinitival clause with non-controlled interpretation.)

`suNrg` = subject is a non-argument.

`obDir` = object is understood in a directional capacity.

`obRefl` = object is a reflexive.

`obReflExpl` = object is an expletive reflexive.

`obDECL` = object is a declarative clause. (Similarly: `obYN`, `obWH`, `obAbsInf`)

`obEqInf` = object is an infinitive equi-controlled by the only available possible controller.

`obEqSuInf` = object is an infinitive equi-controlled by subject.

`obEqIobInf` = object is an infinitive equi-controlled by indirect object.

`obEqSuBareinf` = object is a bare infinitive equi-controlled by subject.

`obEqIobBareinf` = object is a bare infinitive equi-controlled by indirect object.

`iobRefl` = indirect object is a reflexive.

`iobReflExpl` = indirect object is an expletive reflexive.

`oblEqSuInf` = the governee of the oblique is an infinitive equi-controlled by subject.

`oblEqObInf` = the governee of the oblique is an infinitive equi-controlled by object.

`oblRaisInf` = the governee of the oblique is an infinitive which is raising-controlled by the subject.

`oblRefl` = the governee of the oblique is a reflexive.

`oblDECL` = the governee of the oblique is a declarative clause. (Similarly: `oblYN`, `oblWH`, `oblAbsinf`)

`oblPRTOFsubj` = the referent of the governee of the oblique is interpreted as part-of the referent of the subject. (Ex.: *jeg fryser på ryggen* 'I freeze on the back' - I'm cold on my back')

`oblPRTOFobj` = the referent of the governee of the oblique is interpreted as part-of the referent of the object. . (Ex.: *jeg sparker Ola i baken* 'I kick Ola in the butt')

`oblPRTOFiobj` = the referent of the governee of the oblique is interpreted as part-of the referent of the indirect object. (Ex.: *jeg kaster Ola kakestykker i ansiktet* "I throw Ola cakes in the face" - 'I throw cakes in the face of Ola')

`oblEponAbsinf` = extraposed is a non-controlled infinitive occurring as governee of an oblique.

`oblEponDECL` = extraposed is a declarative clause occurring as governee of an oblique.

`scSuNrg` = the secondary predicate is predicated of a non-argument subject (i.e., a subject not serving as semantic argument of the matrix verb - i.e., a 'raising to subject' subject).

`scObNrg` = the secondary predicate is predicated of a non-argument object (i.e., an object not serving as semantic argument of the matrix verb - i.e., a 'raising to object' object).

`scAdj` = the secondary predicate is headed by an adjective. (Similarly: `scN`, `scPP`, `scPredprtcl`, `scInf`, `scBareinf`)

`eponDECL` = extraposed is a declarative clause. (Similarly: `eponYN`, `eponWH`, `eponCOND`, `eponEqInf`, `eponAbsinf`)

We illustrate with a use of the 'small clause' specification `scSuNrg`. One of the construction types it serves to qualify is exemplified by

han synes syk 'he seems sick',

which is a raising construction of the logical form 'seem (he sick)', where the subject *han* does not have a semantic argument function relative to the verb. The label specifying this type is

`v-intrScpr-scSuNrg_scAdj`

where `intrScpr` states that the only constituents syntactically present are a subject and a secondary predicate, `scSuNrg` states that the subject lacks semantic argument status relative to the verb, and `scAdj` states that the secondary predicate is headed by an adjective. The circumstance that the latter two specifications concern dependents rather than over-all valence, is marked by an underscore interrelating them.

A transitive counterpart to this type is exemplified by *han synes meg syk* 'he seems me sick', of the logical form 'seem-to-me (he sick)', where the subject *han* still does not have a semantic

argument function relative to the verb. The label specifying this type is

v-trScpr-scSuNrg_scAdj

where trScpr states that the constituents syntactically present are a subject, an object and a secondary predicate, and the other specifications serve as in the previous example.

With utilization of the slot 2 and slot 3 determinants, around 200 templates have been defined for Norwegian (these can be viewed at the site typecraft.org (research/research projects).

Deciding what is to go in slot 2 and what in slot 3 is in most cases starightforward, but not always. For instance, it will be noted that in the copula valence labels entered at the end of the list in section 3, specifications like 'YN'. 'DECL' etc are used which are otherwise used mainly in dependents' specifications. For one thing, in a case where an adverb or a PP is obligatory, and there is no relational 'super-term' available for specifying its presence, one will refer to the constituent by head property directly, as in inTrAdv. In the case of the copulas, one might have entered 'YN' etc tied to a grammatical relation 'identifier' for the identity copula, and 'predicative' for predicative copula, giving, e.g., v-copPred-predAdj instead of v-copAdj for the predicative adjectival copula construction, and v-copID-idN instead of v-copIdN for the identity construction. Here it is essentially length of labels, and sparsity concerns concerning grammatical relations notions, which have counted in favor of the latter options - either option is in principle possible.

Conversely, instead of writing 'trScpr-scSuNrg_scAdj' in the example discussed, one could have written 'trScprAdj-scSuNrg', or 'trScpr-scSuNrgAdj' - against the former is a wish to generally have POS of dependents being stated in the dependents' slot, and against the latter counts the circumstance that the secondary predicate specifications are in general rather complex already; this point will be further illustrated in section 8 below.

5 Thematic roles and situation types

In specifying semantic roles and situation types, classifications available are less robust than they are for the factors covered above, and for that reason, the notational system will not insist that they should be provided. Closest to practical consensus are core semantic role labels such as 'agent', 'patient' and the like, and aspectual specifications; much less established is a set of situation types covering the full range of constructions in a language. In this section we do not provide any tentative list of values to be used, but comment only on how they are expressed.

As exemplified in (1), each semantic role label is built up with a first part indicating the grammatical function of the item specified, and the remainder providing the specification - thus, suAg, obEjct.. Unlike the case with dependents' labels, the remaining part has no internal structure.

Situation types may in principle cover anything between Aktionsart and detailed situational specification, like in a FrameNet label (cf. http://framenet.icsi.berkeley.edu/). In the system currently implemented, the level of specification is somewhere between these two: Sitaution type labels can be decomposed into standard aspectual notions (like those proposed in Smith 1991, 1997) and specifications uniquely identifying each type. An example is the possible situation label CAUSATION_WITH_CAUSINGEVENT, which means "causation where the cause is itself an event and its event type is linguistically identified", and which implies certain aspectual notions, such as 'dynamic' and 'telic'.

We illustrate the full specification of the example *han synes meg syk* 'he seems me sick' discussed above, which is:

(2)
v-trScpr-scSuNrg_scAdj-
obCog_scThSit-PROPOSITIONALATTITUDE

'obCog' here means that the object expresses a 'cognizer', and 'scThSit' that the secondary predication expresses a 'situational theme'. It will be noted that, consistent with the non-argument status of the subject, there is no thematic role tied to the subject.

With utilization of the slot 4 and slot 5 determinants, around 280 templates are currently defined for Norwegian.

Slots 3 and 4 are both 'constituent oriented', and may provide specifications of one and the same item. For instance, in (2) all of scSuNrg, scAdj (slot 3), and scThSit (slot 4) define the secondary predicate. In principle it would be possible to draw these different specifications together into a unitary, but more complex, specification. This was done, e.g., in the TROLL system (cf. (Hellan et al. 1989)), where arguments were specified as triples of (i) head's POS, (ii)

grammatical function, and (iii) thematic role (including possible non-argument status). Among possible advantages of the current system are that it better profiles 'global' properties of the construction, that it better displays the profile of participant roles, when entered, and makes omission of them practically more easy. Cf. (Lehmann et al. 1996) for further discussion.

6 From Templates to Grammars

The information encoded in the first three slots attains the same depth of argument structure description as is modeled in standard Matrix-HPSG grammars, and approximately as in standard LFG-Pargram grammars (cf. (Butt et al. 1999)). Argument structure being what is generally encoded in *lexical entries for verbs* in such grammars, we now address how the template system can be used as lexical types or macros.

Minimally, templates could be imported as 'en bloc' type- or macro labels into computational grammars. However, the hyphenation and underscore structure of the templates suggest more modular strategies, as we will now show for a typed feature structure design.

For instance, for the template in (2) - `v-trScpr-scSuNrg_scAdj-obCog_scThSit-PROPOSITIONALATTITUDE` one could see this as equivalent to a unification of syntactic types representing, resp., 'verb-headed', 'transitive with a secondary predicate', 'secondary predicate predicated of raised subject', and 'secondary predicate headed by an adjective', and the semantic types 'cognizer, as role of object', and 'situational theme', as role of secondary predicate. In the tdl notation used in LKB grammars, this would suggest (3) as one of its type definitions (ignoring the situation type label for now):

(3)
v-trScpr-scSuNrg_scAdj-obCog_scThSit := v & trScpr & scSuNrg & scAdj & obCog & scThSit.

Here, the type in line 1 is defined as the unification of the types inter-connected with '&'. Mechanically speaking, in going from template to grammatical type, one simply replaces each hyphen or underline in the template label by a type unification symbol. As individual types (as is customary, mention of such types is done with italics) *v*, *trScpr*, *scSuNrg*, *scAdj, obCog* and *scThSit* will all be at 'sign' level. That is: when, in an LKB-like grammar, these types are to unify with each other, they must belong to a common supertype, and given that what they are composing together is the type of a verb lexeme, this is, in a Matrix-type grammar, an instance of the type *sign*. For instance, the type definition for *scAdj*, relative to NorSource, is (with PREDIC being an attribute introducing secondary predicates, and QVAL introducing grammatical relations in a non-list fashion, à la LFG):

(4) scAdj := sign &
[SYNSEM | LOCAL | CAT | QVAL | PREDIC | LOCAL | CAT | HEAD adj].

In what part of the over-all grammar will these types be introduced? A first question is if 'construction' is a type of entity to be assumed among the building blocks of the grammar. In standard HPSG and LFG design, the tendency is to project construction types into the inventory of lexical types, so that verb-construction types enter the grammar through the subcategorization frames associated with verbs. On this view, a definition like (3) will be in an inventory of *lexical type definitions*.

How do lexical items, in this case verbs, relate to these types? If we consider the more standard design in HPSG and LFG grammars, where a verb has as many entries as there are construction frames in which it can enter, most verbs can enter more than one constructional environment.[1] Thus, in the typical case, a verb will be associated with a subset of the totality of types derivable from the template collection, and thus have entries each one defined by one of these types.

Some points of useful flexibility in this mapping may be noted, illustrated with the choice of head in secondary predicate constructions (cf. (4)): in constructions like those discussed, eligible such heads are in principle adjectives, adverbs, prepositions, predicative particles and infinitivals. For a given verb, the full range of options need not be open, hence in defining the general verb type corresponding to the template `v-trScpr-scSuNrg_scAdj-obCog_scThSit` one may want to leave the sc-head open, and rather have a way of appending that information for each individual verb. By separating out the

[1] We here ignore possible designs which might, for each verb, represent it with one single entry, and account for its many frames of occurrence either through a network of lexical rules, or through underspecifying each entry to yield, for each verb, exactly the range of environments it can occur in.

relevant part (*_scAdj, _scAdv...,)*, and defining *v-trScpr-sSubNrg_scAdj*, *v-trScpr-scSuNrg_scAdv*, etc. as subtypes of *v-trScpr-sSubNrg*, one can in an LKB grammar enter each verb in the lexicon with the appropriate last part provided (and leave them out when the verb actually can combine with all options). In such an approach one has to define all constellations in the relevant type file, the gain lies in the flexibility one has in the lexical entry specifications. The same advantages apply with regard to specification of semantic roles.

7 Uses of the template inventories

A first possible usage of a template inventory is that one can employ a set of example sentences illustrating the various templates as a *test suite* for the grammar. Given the systematic design of the template list, one is assured to have a systematic test suite in the respects covered by the templates.

A second benefit of the design is as a basis for building cross-linguistically matching test-suites, to the extent that templates coincide cross-linguistically.

For linguistic typology, once one has template lists developed for many languages, comparison and systematization of differences can be facilitated.

For linguistic description and grammar creation, having established template lists for related languages may enhance efficiency, in providing 'check-list' starting and reference points.

All of these points will presuppose that one can reach a commonly approved standard of notation. (In principle, with different types of notation, but a one-to-one correlation between notations, similar effects may be gained, although there is then an extra step of identifying correlations.)

Currently, such a combined initiative of notation development and typological investigation is being pursued for a group of Ghanaian languages in consonance with the Norwegian system; cf. (Dakubu, 2008). (For both systems, full template and example lists can be viewed at the site typecraft.org mentioned above.)

As still another enterprise connected to the present template inventory may be mentioned a partial *ontology of verb construction types* developed with the LKB platform (in principle exportable also to OWL), representing all of the templates in the Norwegian inventory and some more. For a partial description, see (Hellan 2007).

Relative to the present system, a *verb class* can be identified as a set of verbs which are accommodated by the same set of construction types. (This notion of 'verb class' is related to that employed in (Levin 1993), which is based on *alternations* between construction types. An alternation, such as the *'spray-load* alternation', can be viewed as a *pair* of construction types in which a number of verbs can participate, typically with rather similar semantics, highlighting – by a 'minimal pair' technique - semantic properties of the constructions chosen.)

8 More complex types

In its current version, the system does not include 'derived' constructions, of which in Norwegian *passive* constructions would be the main instance. As a prerequisite for a notational system for derivation, systems will first be made for selected Bantu and Ethio-Semitic languages (representing future development)

Possibly also of a derivational nature, but here treated as basic patterns, are 'Secondary predicate' constructions, a few of which were discussed above. To indicate where the Norwegian label inventory probably reaches its peak of complexity, we give a brief resymé of the parameters involved in these constructions, and the more complex labels employed.

The secondary predicate (henceforth: secpred) can relate to the main predicate either as the content of a propositional attitude or perception, or as concurring in time, or as the causee of a causation. In the latter case, either an event is portrayed as the cause (indicated by the substring `….Cse`), or an entity. In the former case, the causing event can have from zero to two participants, and when one or two, one can be implicit. What can never be implicit is the entity of which the secpred is predicated: it may occur as subject or object, and in either case either realizing this grammatical function by itself (in which case the function is 'non-argument'), or sharing it with a participant of the causing event (in which case the function has 'argument' status). The following slot 3 labels serve to encode the various possibilities:

`scObArgConcurr` (*he drank the coffee warm*)

`scObNrgRes` (*he made me sick*): Of the causing event, only the participant denoted by the subject is specified.

`scSuArgCse` **(***kaffen koker bort* 'the coffee boils away'): The matrix verb (together with its argument subject) expresses part of the description of the causing event.

`scObArgCse` **(***han sparket ballen flat* 'he kicked the ball flat'): The secondary predicate is predicated of an argument object, and the matrix verb (together with its object) expresses part of the causing event.

`scSuNrgCse` (*landsbyen snør ned* 'the village snows down'): The secondary predicate is predicated of a non-argument subject, and the matrix verb expresses part of the causing event.

`scObNrgCse` (*han sang rommet tomt* 'he sang the room empty'): The secondary predicate is predicated of a non-argument object, and the matrix verb (together with its subject) expresses part of the causing event.

In dealing with typologically different languages, it is not a priori given what constructional template options may present themselves (see Dakubu op.cit. for discussion of some Volta Basin languages). Whatever these additional types may be, in designing labels, one probably should not exceed the complexity of the labels just presented.

9 Conclusion

With an encoding of a construction type's argument structure and semantics which is probably representative of what one may want to expose, each template in the system presented here is by itself as compressed as can be, which gives the template structure some interest by itself. However, it is through the totality of templates, and through the design by which they can be easily enumerated, compared and computed, that the system presented may be a contribution to grammar engineering and language typology alike. While the system reflects such ambitions, it is still in an initial state of deployment both in grammar engineering and typology, and its potential value will reside in the extent to which it will be used, and receive feedback for usability.

References

Beermann, Dorothee and Lars Hellan. 2004. A treatment of directionals in two implemented HPSG grammars. In St. Müller (ed) *Proceedings of the HPSG04 Conference,* CSLI Publications */http://csli-publications.stanford.edu/*

Bender, Emily M., Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starter kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proceedings of the Workshop on Grammar Engineering and Evaluation*, Coling 2002, Taipei.

Butt, Miriam, Tracy Holloway King, Maria-Eugenia Nini and Frederique Segond. 1999. *A Grammar-writer's Cookbook*. Stanford: CSLI Publications.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars.* CSLI Publications, Stanford.

Dakubu, Mary E. K. 2008. The Construction label project: a tool for typological study. Presented at West African Languages Congress (WALC), Winneba, July 2008.

Flickinger, Daniel, John Nerbonne, Ivan A. Sag, and Thomas Wassow. 1987. Toward Evaluation of NLP Systems. Technical report. Hewlett-Packard Laboratories. Distributed at the 24th Annual Meeting of the Association for Computational Linguistics (ACL).

Hellan, Lars. 2007. On 'Deep Evaluation' for Individual Computational Grammars and for Cross-Framework Comparison. In: T.H. King and E. M. Bender (eds) *Proceedings of the GEAF 2007 Workshop*. CSLI Studies in Computational Linguistics ONLINE. CSLI Publications. *http://csli-publications.stanford.edu/*

Hellan, Lars., Lars Johnsen and Anneliese Pitz. 1989. TROLL. Ms., NTNU

Lehmann, Sabine., S. Oepen, S. Regier-Prost, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion ,J. Baur, L. Balkan, D. Arnold. 1996. Test Suites for Natural Language Processing. *Proceedings of COLING* 16, p. 711-16.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Smith, Carlota. 1991, 1997. *The Parameter of Aspect*. Kluwer Publishers, Dordrecht.