

On 'Deep Evaluation' for Individual
Computational Grammars and for
Cross-Framework Comparison

Lars Hellan

Norwegian University of Technology,
Trondheim

Proceedings of the GEAF 2007 Workshop

Tracy Holloway King and Emily M. Bender
(Editors)

CSLI Studies in Computational Linguistics
ONLINE

Ann Copestake (Series Editor)

2007

CSLI Publications

<http://csli-publications.stanford.edu/>

Abstract

A rather difficult point in grammar engineering evaluation is how to test and compare for analytic adequacy. A test design for 'deep' grammars is here proposed, where a parse is considered valid only if the assignment of syntactic and semantic structures that it displays obey certain conditions. The set of grammatical sentences in the test suite is construed as leaf types in a construction ontology, where the top types introduce the discriminants according to which constructions are categorized. These discriminants conform to notions shared across linguistic frameworks, and the validity conditions are defined within a well-known space of analytic parameters. One may envisage that with such a design, a meeting point can emerge for comparing frameworks with regard to agreed-upon aspects of linguistic content, and individual grammars with regard to their analytic aims and actual achievements relative to the aims.

In Phillip Pullman's *His dark materials*, humans in one of the worlds have their soul partly realized as a little animal always accompanying them, sharing their thinking and emotions, but still behaving partly as independent agents; they are called *daemons*.

1 Introduction

One way in which to improve quality and coverage of a grammar is to systematize its test suites by assembling construction types according to a fixed set of theoretically grounded parameters. Once the parameters are decided upon, such a systematicization can provide one dimension of desirable independence from the actual day-to-day development and testing. To be presented here is the composition of a test suite for verb constructions, developed in connection with the Norwegian HPSG-based computational grammar NorSource. The composition of the test suite reflects parameters of verb construction analysis which the grammar aims at implementing.

In this approach, consistency and perspicuity of the test suite is induced by the construction of a formal ontology of construction types, whose classifying properties match those reflected in the composition of the test

suite. Moreover, the ontology is defined in terms of attributes and values, which makes it possible to assign to any one (grammatical) sentence in the test suite a feature structure reflecting those properties held to distinguish the construction type represented by the sentence in question from other construction types. An example of such a feature structure, reflecting the grammatical concerns of NorSource but distinct from the feature structure NorSource itself produces for the sentence, is discussed below. In its capacity as a small system in its own right, we call the ontology, with the test sentences as its leaf types, a *Test Daemon*.¹

A further perspective on such a system is the following:

For computational grammars aiming at exposing the principles of morpho-syntactic and semantic composition of a language, one would welcome a mechanism which at least semi-automatically could allow one to test for the *adequacy* of the parses produced.²

Analytic adequacy is not an absolute measure. Linguistic frameworks differ as to what they recognize as adequate analyses, both for construction types and for over-all analytic design, and such diversities are reflected in computational grammars, reflecting the assumptions of the linguistic framework of which they are part, and within each framework, and even for the same constructions in the same language, differing in their analyses. So, if we want to develop a mechanism of adequacy testing, it cannot be a mechanism of *judgement*, but rather one where each grammar makes (i) a *declaration of what it wants to achieve*, and (ii) a way of displaying, for any parse, how that parse *lives up to the ambitions* of the grammar. Let us call these the *declaration part* and the *fulfillment part* of the mechanism,

¹ *NorSource* is an LKB-based grammar of Norwegian, currently developed by L. Hellan, B. Waldron and D. Beermann at NTNU, Trondheim (<http://www.ling.hf.ntnu.no/forskning/norsource/>). It was initiated in 2002, in the EU-project *DeepThought*, and is by now rather large; one of its features is a fairly detailed semantics (cf. Hellan and Beermann 2005). *NorSource* is part of the DELPH-IN consortium (<http://www.delph-in.net/>).

This work started in the TROLL project in the late '80s, as a construction inventory enterprise, much prior to the construction of a parsing grammar. (Both then and now, the notion *construction* is used in its standard meaning, and not necessarily adopting criteria of the Construction Grammar framework (Goldberg 1995).)

The Daemon version described here is still in its infancy. For comments on many aspects of the work I thank Dorothee Beermann, and for the design of the Ga system, also Mary Esther Kropp Dakubu and Felix Ameka. The system itself can most easily be obtained from the author at lars.hellan@hf.ntnu.no.

² We take for granted the availability of standard test suites and the apparatus developed in each framework based on whether sentences parse at all or not.

respectively. The test suite design described above - the Test Daemon - would be one form of a *declaration* part of such a system. We now comment on these parts.

2 The *declaration* part

The declaration part will require that the analytic specifications employed are perspicuous, and that their space of notions and distinctions is predictable. In the present connection, the latter means that the factors dealt with are those that the linguistic community at large may expect to be considered in connection with verbal constructions, such as aspect, tense, grammatical functions, thematic roles and their linking, and control patterns. Perspicuity will mean that the terms used are sufficiently rooted in at least one tradition of research to allow the identification of equivalents or correspondents in other frameworks. Below is an example of a type display aimed at meeting these requirements:

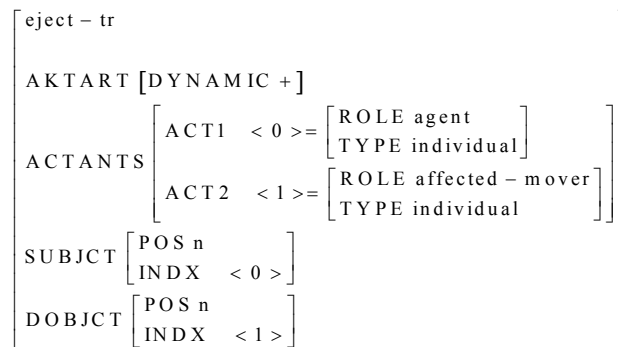


Fig. 1. The type *eject-tr*, as exemplified by *She throws the ball*

This feature structure displays grammatical functions by (slightly abbreviated) labels generally understood, and their linking to semantic 'participants' through perspicuous reentrancy. Thematic roles and ontological type are indicated by recognized labels, and the partial Aktionart specification is also standard. Such a feature structure, thus, illustrates what one might want as a perspicuous display of the factors mentioned. (In effect, it is also what one of the systems mentioned below provides.)

Intellectually, as said, the declarations part, or the *Test Daemon*, may have to live slightly outside of the grammar itself, since it must attain a level of intelligibility and perhaps quality which the parses produced by the grammar cannot always be expected to provide. If the test Daemon sits outside the grammar, then its most trivial materialization would be as a hand-provided recipe accompanying each sentence token in the test suite. A

more interesting design is to organize the set of grammatical sentences as leaf types in a construction ontology, where salient specifications are induced through inheritance, and the top types introduce the discriminants according to which constructions are categorized. These discriminants conform to notions shared across linguistic frameworks. The choice of ontology is in principle open, and the Test Daemons should be developed for all kinds of constructions.

The present design has been implemented at NTNU for two computational grammars, the Norwegian medium-size grammar *NorSource*, as mentioned above, and the much smaller grammar *Ga-gramm* for Ga (a language spoken in the Volta Basin area of West Africa), for both only in the verbal domain. Currently, for *NorSource*, there are two such Daemons, one representing syntactic valence and control/coreference properties, and the other in addition covering thematic roles and aspectual/Aktionsart properties (such as illustrated in fig. 1). The richer one comprises about 230 construction types, the leaner one about 170. Both inventories subsume only 'basic' patterns, that is, not passive constructions, and productive syntactic patterns of modification, wh-movement, subject-verb inversion and more are also abstracted away from. (In this respect, the construction inventory therefore has a straightforward connection to the verb lexicon, as this is standardly conceived in an HPSG or LFG grammar.) For Ga, only a set of 40 construction types has so far been encoded, based on the richer structure of the larger *NorSource* Daemon.

Each construction type is, apart from its type notion in the ontology, represented by one example sentence, serving as a leaf type in the ontology. The present ontology is stated in an LKB hierarchy (cf. Copestake 2002), as this system readily lends itself to the kinds of attribute paths often preferred by linguists.

This LKB hierarchy has been modelled as a small LKB-grammar, consisting of only one type of constructs, namely sentences formally modelled as multi-word lexical items. These sentences are identical to those entered as leaf types in the ontology. In this way, one is able to 'parse' the construction token, as a mock-sentence, or really, as a single constructional item. For example, for the sentence *she throws the ball* (now exemplifying with English), which is entered as a token of the type *ejct-tr* (cf. fig. 1), the 'construction entry' will have an identifier such as 'she_throws_the_ball', and in the orthographic specification, the string '<"she", "throws", "the", "ball">'; this string, thus, is defined as if it were a multi-word lexical item, from an LKB point of view. The Test Daemon can then, as a one-lexical-item parse, produce that string, with a feature structure being exactly the syntactic-semantic structure defined by the ontology for that type. Using the interface possibilities of an LKB grammar, one can thereby obtain a view of the properties of a given construction type as identified by its *construction name*

(through 'View Expanded Type') and through the feature structure exposed by the parse for the example sentence.

3 The *fulfillment* part

Technically the fulfillment part can be done in three ways. Common to all is that one defines a test suite for the parse grammar identical to (or overlapping with) the test suite designed for the Daemon.

Mode 1) Independent mode:

For each sentence, one verifies that the parse grammar and the Daemon separately produce feature structures (FSs) that correspond to each other in the respects focussed on. (Essentially, the Daemon FS will be a subpart of the FS produced by the parse grammar.)

Mode 2) Gently dependent mode:

The Daemon is integrated in the parse grammar, so that for each sentence parsed in the standard way, a parse is also displayed of the sentence-qua-construction, in the manner produced by the Daemon. (Thus, they can be viewed in parallel, in the same 'parse-forest'.)

For each sentence, one still needs to verify by hand that the systems separately produce feature structures that match in the respects focussed on.

Mode 3) Strongly dependent mode:

Again, the Daemon is integrated in the parse grammar, so that for each sentence parsed in the standard way, a parse is also displayed of the sentence-qua-construction, in the manner produced by the Daemon. However, in the FS produced by the parse grammar, the Daemon FS is replicated, with explicit declarations of how the information provided in the Daemon FS is reflected in the FS of the parse grammar.

We illustrate this mode with an edited excerpt from an FS of the parse grammar for Ga:

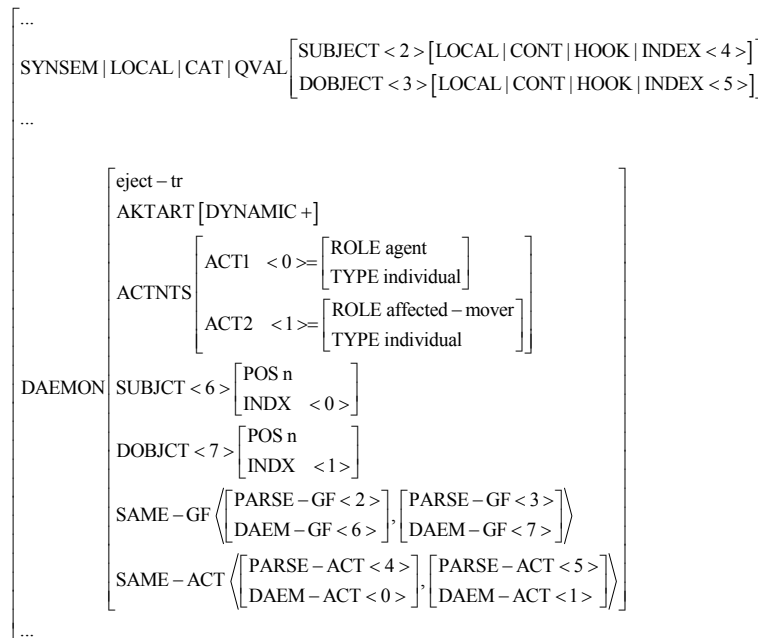


Fig.2 A view of a Mode 3 integration of a Daemon specification inside of a parse-grammar feature structure.

Its general feature architecture (derived from the HPSG Grammar Matrix, cf. Bender et al. 2002) includes the feature path `SYNSEM | LOCAL | CONT | HOOK | INDEX`, and in addition exposes some grammatical functions with dedicated attributes such as `SUBJECT` and `DOBJECT`, introduced by a feature `QVAL`. The counterparts of these features in the Daemon are `INDX`, `SUBJCT`, and `DOBJCT`; as attribute names in LKB can only be introduced with one unique type, and the types employed in the Daemon involve much less feature structure than those in a Matrix LKB grammar, both types and attributes must be distinct between the two systems. Hence the *correspondences* that one aims to expose can only be stated as relations, not by reentrancy, and these correspondences are given in the lists `SAME-GF`s and `SAME-ACT`s ('Same Grammatical Functions' and 'Same Actants', respectively).

The Norwegian LKB grammar so far uses only mode 1; the grammar for Ga is smaller, and thus allows more easily for the exercise, but its architecture of a mode 3 integration can in principle be generalized to all Matrix-based grammars. A technical description of the integration between the parse grammar and the Daemon, combining general features and some specifics about the grammars employed, is given in the Appendix.

How the modes 2 and 3 might be implemented in other frameworks, or with other platforms, has so far not been explored.

In the next section we describe how the mode 3 version can be put to use as a plug-in-like mechanism in an LKB grammar, and after that we return to the structure of the Daemon and issues that presuppose no more than mode 1 of integration.

4 A plug-in possibility derived from mode 3

From the apparatus used in mode 3, a certain plug-in effect for, e.g., semantic specification can be derived. One can make the specification in the Daemon richer than the specification in the parse grammar, in such a way that when including the Daemon in the parse grammar, through unification, it makes the extra specification from the Daemon also part of the parse grammar FS. An example is the following: In the Ga grammar, the attribute `INDEX|SORT` is unspecified for value. In the integration file, the value of `INDX` from the Daemon specification, which is declared for `ROLE` and `TYPE`, can be reentrant with the value of `SORT` in the parse grammar. Thereby also the `INDEX|SORT` specification of the parse grammar will provide semantic `ROLE` and `TYPE` information. Fig. 3 illustrates the effect, superimposed on the constellation already shown in fig. 2, with the plug-in effect shown in the `SORT` values on the uppermost lines:

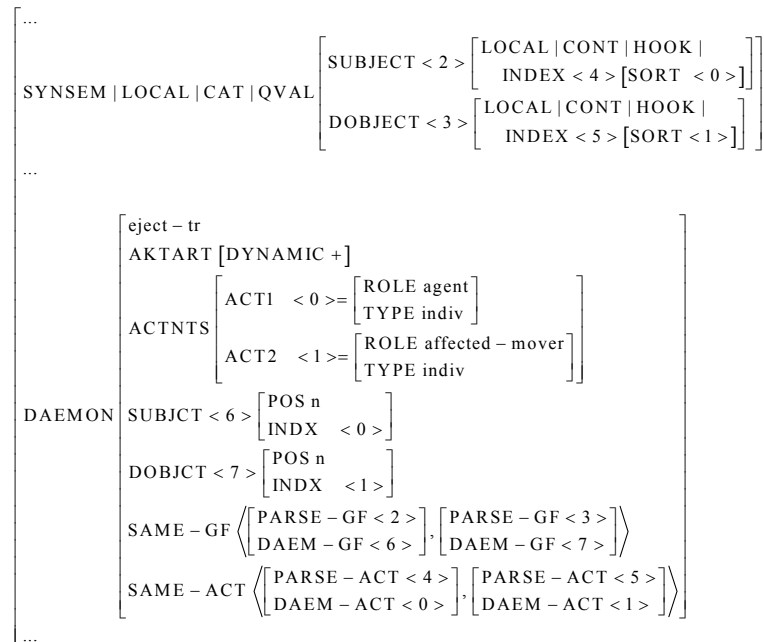


Fig.3 A view of a Mode 3 + plug-in integration of a Daemon specification inside of a parse-grammar feature structure, with the plug-in effect shown in the SORT values on the uppermost lines.

This way of inducing a richer semantic specification may be compared with, e.g., rewriting MRS representations, or connecting MRS representations with semantic ontologies. It is to be noted that this is a configuration the grammar writer can create solely inside the LKB, with no need for external components. (The exact steps needed are described in the Appendix.)

5 Properties of the Daemon

5.1 Attributes for grammatical functions

For the Norwegian Daemon, the following syntactic attributes representing grammatical functions are used:

SUBJCT - headed by a nominal constituent or consisting of a clause, and with either argument or non-argument status relative to the verb (see below)

DOBJCT ['direct object'] - headed by a nominal constituent or consisting of a clause, and with either argument or non-argument status relative to the verb

IOBJCT ['indirect object'] - headed by a nominal constituent, and with argument status relative to the verb

OBLIQUE - either headed by a pre- or postposition, or marked by a relevant NP-case, and with argument status relative to the verb; moreover, the governee of the adposition/case can in some cases be seen as having an indirect argument relation to the verb.

SECPRED ['predicative', in Jespersen's sense] - a 'secondary predicative' constituent, headed by any open class part of speech

EPON ['extraposition'] - a so-called 'extraposed' clause

IDENT ['identity term'] - the second argument in an identity predication

PRESENTED - the NP 'presented' in a presentational construction

PARTIKEL - an adverb- or particle-like element

The functions SUBJCT and DOBJCT are *unrestricted* in their values in the following ways: a) Their value can have either *argument* status relative to the verb they are functionally related to, or to another item; the latter applies to 'raised' subjects and objects, and is marked by the specification 'SUBJCT *nonarg*'/'DOBJCT *nonarg*'; this corresponds to the situation where in an LFG f-structure, an item is entered outside the angled brackets of a PRED value (as in *PRED = 'seem < XCOMP > SUBJ'*) (cf., e.g., Butt et al.). Otherwise the specification is 'SUBJCT *arg*'/'DOBJCT *arg*'. b) Their value can be either a 'full' item or an expletive noun, represented, resp., as 'SUBJCT *full*'/'DOBJCT *full*' or 'SUBJCT *expletive*'/'DOBJCT *expletive*'.

These dimensions cross-classify, in that a nonarg ('raised') item may be either full or expletive; specifying a position as 'expletive' in turn entails that this position does not carry a semantic role, but it does not entail 'nonarg', since it is not given that it has an argument role relative to another item. (The notion 'expletive' actually covers three cases, all encoded: *expl-pres* is the item introducing a presentational construction, *expl-epon* is the item correlated with an 'extraposed' clause, and *expl-absolute* is the item introducing an impersonal.)

For cases where the function OBLIQUE introduces a PP whose NP constituent has a specific relation to the verb or other constituents, the NP is exposed by the feature D-ARG (for 'dependent argument') and D-POS ('part-of-speech of dependent argument'); moreover, the NP is exposed in the semantic specification by the attribute ARG-OF-OBL, linked to the D-ARG of the functional feature (since the INDX of OBLIQUE is then not linked, this means that the PP as such is not a semantic argument, only its contained

NP). For those cases where the oblique PP has its whole specification contributing as an argument (as in the type *presentational-loc*, exemplified by *det sitter en katt i trappen* 'there sits a cat in the stairs'), the semantic argument role contributed by the full PP *i trappen* is represented as LOC, and linked to INDX of OBLIQUE; the construction at the same time illustrates the feature PRESENTED (see fig4, a and b):³

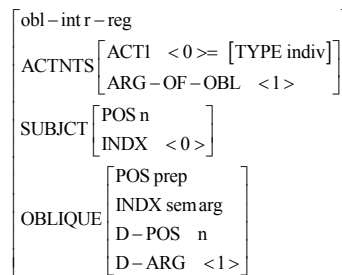


Fig 4a

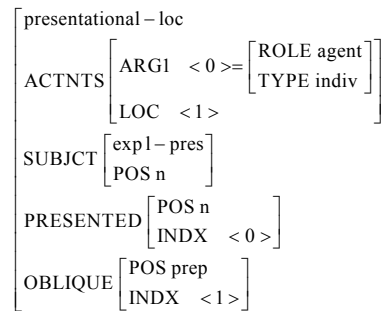


Fig 4b

The grammatical function SECPRED (for constituents carrying the predicational content in a secondary predication construction) is illustrated here by a construction with a causative semantic structure, *bøtta regner full* 'the bucket rains full' (= 'it rains (a situational ACT1) such that the bucket gets full (a situational ACT2)'), where the constituent *full* is the secondary predicate, and the reentrancy of '1' reflects the 'raising' structure of this construction:

³ In the literature on Norwegian presentationals, the NP 'presented' is not uncommonly analyzed as a direct object, since it occupies a position much like that of direct objects (e.g., following the indirect object); due to its logical status, it is often also counted as a subject. Using the attribute PRESENTED is for descriptive convenience, and a freedom allowed by the Daemon purpose. A similar expedient is the use of the attribute EPON - that of 'extraposed' clauses and infinitives.

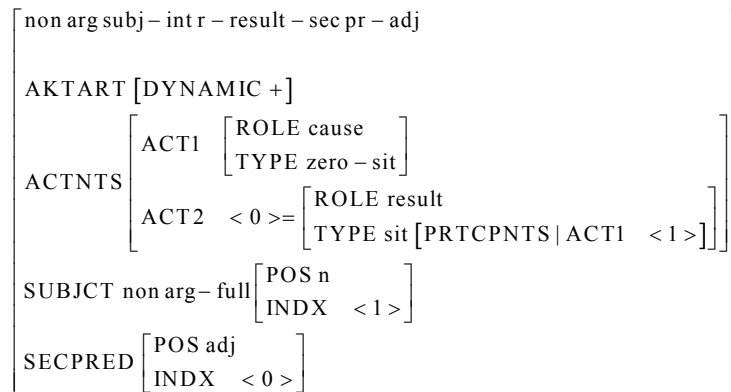


Fig. 5

This brief survey of some of the attributes used illustrates the following:

i) The analytic reasoning behind the inventory and value assignment in the FSs of a Daemon is no different from what applies in the building of a normal parse grammar.

ii) Attributes are not necessarily universal, but on the contrary can sometimes highlight properties particular to a language or family of languages; for instance, the use of SECPRED and PRESENTED reflect properties typical of Germanic languages.

iii) As a Daemon of NorSource, the attributes used do not necessarily match, attribute by attribute, those used in NorSource. For instance, although PREDIC in the NorSource grammar is equivalent to SECPRED as used in the Daemon, the feature PRESENTED corresponds to the specification

. . . QVAL | DOBJECT | LOCAL | CAT | HEAD | PRESENTED +

in NorSource. With the 'mode 3' of linking described in the previous section, such relationships are quite explicit, whereas with the modes 1 and 2, they have to be known. Still, in the latter case, the correspondences are not very many.

5.2 Principles of labelling construction types

A constructional type label should reflect the composition of the construction in as much detail as can conveniently be perceived when reading the label, and recalled when writing it. The more consistent and systematic the composition of the labels, the easier it is to reflect more content in them. In the three Daemons created so far - the syntactic and the syntactic-semantic Daemons for Norwegian, and the syntactic-semantic Daemon for Ga, the strategy of labelling differs among the three.

First, the *syntactic* labels for the first Norwegian Daemon are exemplified by the following samples: first in the label, *intr* or *tr* indicates degree of

transitivity, and it is then indicated if subject or object lacks argument status; then the presence of further constituents such as obliques or secondary predicates is signalled, and in the latter case its head category; and if the subject or object or the governee of a preposition is anything other than a regular NP, the category is indicated (like *obl-decl* meaning that the governee of a preposition is a declarative clause, or *tr-absinf-n* meaning that the subject is an absolute (arbitrary control) infinitive (and the object a normal NP), or *tr-raistoobj-bareinf* meaning that the object is 'raised' out of a succeeding bare infinitive); all these pieces of information are connected by hyphens, and the resulting system is presumably within the limits of user-friendliness:

intr-n	gutten sover 'the boy sleeps'
intr-obl-decl	de snakker om at det er for sent 'they talk about that it is too late'
intr-nonargsubj-secpr-adj	kjelen koker varm 'the kettle boils hot'
tr-n-n	Kari sparker ballen 'Kari kicks the ball'
tr-equi-inf	Kari prøver å komme 'Kari tries to come'
tr-absinf-n	Å bygge høyhus interesserer Kari 'to build highrises interests Kari'
tr-raistoobj-bareinf	jeg hørte ham synge 'I heard him sing'
tr-nonargsubj-secpr-adj	han synes meg syk 'he seems me sick'
tr-nonargobj-secpr-adv	han sang sorgene bort 'he sang the sorrows away'
tr-epon-decl	det bekymrer meg at han kommer 'it worries me that he comes'

Table 1. *Some labels for syntactically identified construction types in Norwegian*

If one wants to mark semantic information in addition to syntactic specifications, the maneuvering space gets more strained. One has to choose whether to use 'global' labels to indicate something like *situation type*, paired with the same syntactic specification as before, or *role* indicators tied to each constituent (and possible aspectual specification in addition). In the Norwegian system, global situation labels are used, but with much inconsistency as to how much syntactic information is also supplied, and in which order syntactic and semantic information is given; the assembly below is representative (in parenthesis behind the construction label is given a

situation type label, reflecting the semantic type intersecting with the syntactic type - cf. next sub-section):

nonargsubj-intr-result-secpr-adj (zerocause-event-causation)
 koppen renner full - 'the cup runs full' - "the cup fills up"
 raistosubj-intr-ascrpt-secpr-adj (unary-sit-ascription)
 gutten virker syk - 'the boy seems sick'
 path-endpnt-tr (path-endpnt)
 stien når toppen - 'the path reaches the top'
 weight-tr (weight-sit)
 stenen veier 5 kg - 'the stone weighs 5 kg'
 tr-exp-raistosubj-inf (sit-exper-sit)
 han synes meg å komme - 'he seems me to come'

Table 2. *Some labels for syntactically and semantically identified construction types in Norwegian*

In the Ga system, more consistency is achieved, as is seen below; here global syntactic information comes first, then role specification with the roles in the order of the constituents, and then with capital letters a global semantic characterization. While the latter might seem redundant given the role specifications, for some readers they may be informative, since the compositionality in many of the Ga constructions is not what most directly comes to mind from a European language perspective. The syntactic specification includes information as to whether the 'logical' actant is embedded in a postpositional NP, a counterpart to 'oblique', and possible identity with other constituents; *unifobj* stands for 'inherent complement' of the type discussed in Essegbey 1999. The last example is a serial verb construction, indicated by *sv-*, and by a succession of two verbal constructions, initiated as *vtr-* and *vditr-*, respectively, each having its internal roles indicated; the * attached to a role means that the role-bearing actant is repeated in the second VP, either just understood, or as a pronominal prefix, specified as **PRONPREF*. (In the feature structures for these constructions, some specifications will be particular to West African languages, as some of those discussed in 5.1 are to Germanic languages.) Here are some examples:

v-intr-postpsubj-locus-PROPTY	Nsɛɔ lɛ mli jɔɔ sea the inside cool-HAB 'The sea is cool'
v-tr-mover_endpt-MOTION	Kofi ba biɛ Kofi came here

v-tr-partwhlsubj_idobj-locth_exper-SENS	o-he jɔɔ bo your-self rest you 'you are at ease, relaxed'
v-ditr-unifobj2-agsens_locus_materialzr-PERCPT	wɔ-bo lɛ toi we-listen him ear 'we listened to him'
sv-vtr_ag*PRONPREF_th*-vditr_endpnt-PLACEMENT	wɔ-tsi amɛ wɔ-gbee shi we-push them we-fell [them] down

Table 3. *Some labels for syntactically and semantically identified construction type in Ga*

On layout consistency grounds, the first (for syntactic annotation) and third system seem preferable. Which one to choose among these of course depends on what kind of information the parse grammar in question exposes (or is made to expose).

Although the matter of labelling may seem theoretically insignificant, for the building of a test suite, an instructive label system is quite helpful, and far easier to relate to than feature structures of the types illustrated earlier.

Such a labelling design may also be helpful in the building of a construction inventory not (yet) linked to a computational grammar, both by its potential for enhancing clarity of organization, and for linking up to feature structures of the type provided by the Daemon. Before a consolidated system of labelling can be considered, the usefulness of candidate systems outside of the domain of grammar engineering test suites would thus also be important.

5.3 Structure of the ontology

The ontology is organized into one syntactic and one semantic part, with construction types inheriting from both sides. Below is an example of how a syntactic and a semantic specification are joined to define a constructional type, for *nonargsubj-intr-result-secpr-adj*, illustrated by *bɔtta regner full* 'the bucket rains full' (= 'it rains such that the bucket gets full', cf. fig. 5 above):

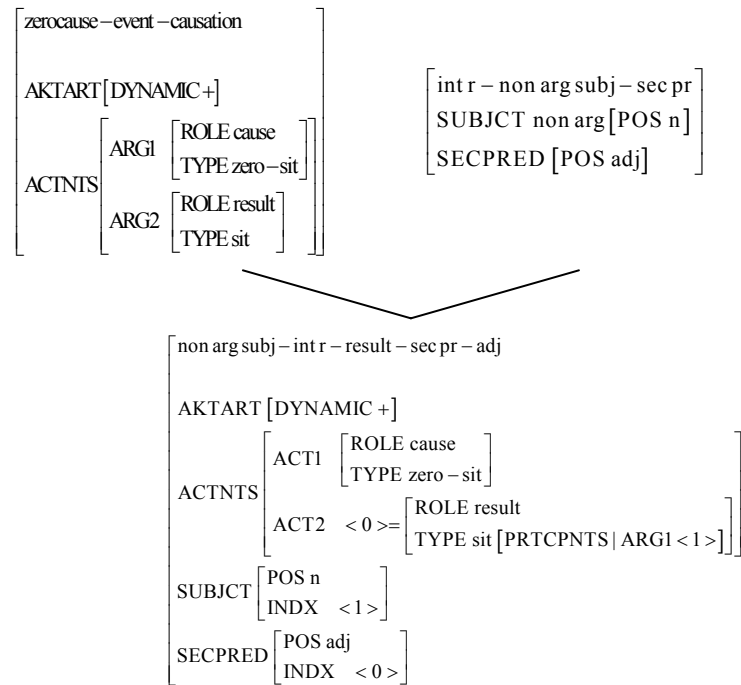


Fig. 6 Example of constructional type inheritance

In the same vein, the following is an approximate view of some of the top level syntactic types in the Norwegian system, and some of the semantic ones, with one example of the stitching together of the syntactic and the semantic side:

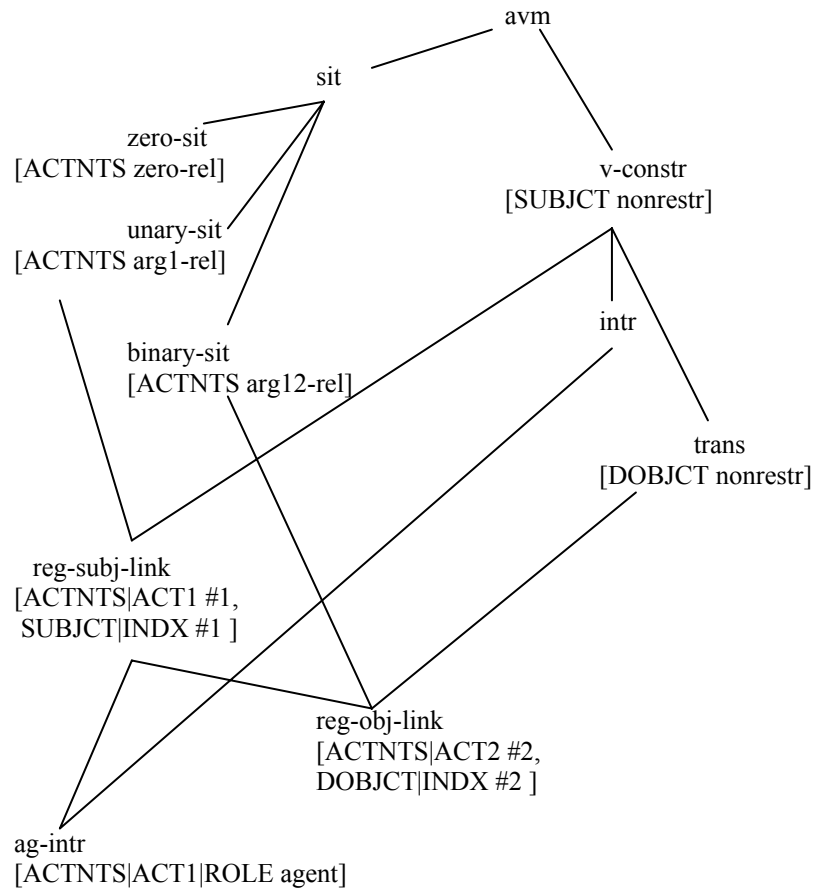


Fig. 7 Partial view of constructional type inheritance

With the ontology populated by more languages, the syntactic part of the system would most likely have to grow, while the semantic part (apart from improvements of the system as such) in principle might remain constant. The design might then make it possible to view, for any identified situation type, which construction types in various languages embody that situation type, and likewise for going in from a given syntactic specification to see construction types across languages supplying different situation types.

In order for a construction ontology of this kind to be implemented for grammar engineering frameworks less likely to employ an LKB system, it will be an interesting question to see to what extent the present ontology can be ported, e.g., to OWL.

6 Summary

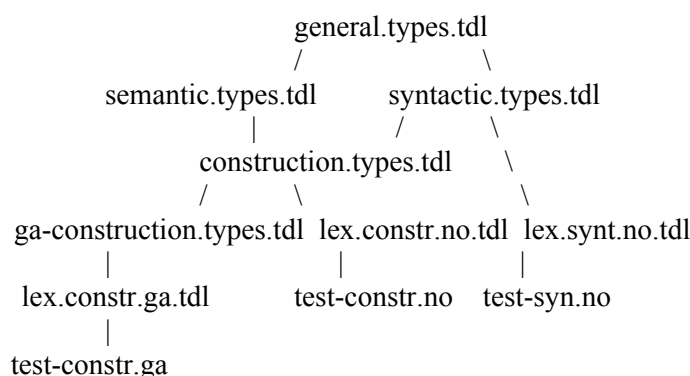
In order to expose the content of grammars for evaluation across frameworks, very much of their formalism and terminology has to be abstracted away from, in order to attain a common ground of comparison. A requirement on such a common ground is that it is still identifiably sound according to both theoretical and empirical criteria of linguistics. If such a ground can be found, in turn, it is helpful not only for the construction of grammars and the evaluation and development of the various frameworks, but it may also help mediate the contribution of computational grammars to a wider linguistic community.

In the system presented above, the concern of designing a common ground has been combined with a strategy of creating systematic test suites for use in individual grammars. This strategy, in turn, tries to improve classification of construction types in a way that may be useful for linguists outside the computational community, not the least for purposes of grammar documentation.

An attempt is made to embed the system in an actual mechanism, which on the one hand exposes an ontology of verbal constructions, and on the other can be integrated in a computational grammar so as to 'witness' how the specifications produced by a parse relate to declared aims phrased in terms of a 'common ground' analytic language. The mechanism is lightweight, and is only a means by which one can *start* becoming more systematic about evaluation in analytic respects. As the mechanism is built on the LKB platform, it - especially in the latter respect - is confined mainly to HPSG-based grammars. However, these two functions can clearly be split, and hopefully the present system may serve as a partial starting point for the construction of systems based on different platforms.

Appendix

Below is a sketch of the structure of the Daemon as realized by an LKB system. It consists of the directory 'constructions', which, in addition to the lkb folder and roots.tdl, has the following files, here displayed so as to reflect which files depend on which:



Purely syntactic aspects of the construction specifications are defined in `syntactic.types.tdl`, purely semantic aspects in `semantic.types.tdl`, and combined constructional specifications (combining types from the former two) are defined in `construction.types.tdl`. In addition, `general.types.tdl` states over-arching general types. For Ga, there is also a type file `ga-construction.types.tdl` deriving new types from `construction.types.tdl`.

In the lexical files, each 'lexical item' is a full sentence, corresponding to the items in the test files. For Norwegian there are two test files, corresponding to the lexicon files `lex.synt.no.tdl` and `lex.constr.no.tdl`: the former has types reflecting only the syntactic specification, the latter has types reflecting a full constructional specification. For Ga, there is only one test file, representing full constructional specification.

To view the full construction hierarchy, enter 'sign' in the LKB Top View window. To view either the syntactic part of the construction hierarchy exclusively, or the situation hierarchy exclusively, comment out `construction.types.tdl` (and `ga-construction.types.tdl`) in the script file (in this case, also comment out `lex.constr.no.tdl` and `lex.constr.ga.tdl`), and then, to view the syntactic construction hierarchy, enter 'syncons' in the View window, and to view the situation hierarchy, enter 'sit' in the View window. Whichever type hierarchy is displayed, to see the feature specification of the type, do normal leftclick on the type label and view 'Expanded type'.

Exemplifying sentences can also be viewed. From any of the test-files, any sentence can be selected and entered in the LKB Top parse window (or '(do-parse-tty "...")' in the commonlisp buffer), and a minimal parse tree emerges, rooted by 'constr' (or '?').⁴ When clicking for 'Enlarged tree' and then 'Feature structure', one sees the type and feature structure specification (the latter being the same as what one sees for that type on the previous

⁴ When, for Norwegian, a given sentence appears in both lexicons, two parses are displayed, the upper one rendering the full construction specification, the lower one the syntactic part of that specification.

view).

To create the 'fulfillment' effects described in section 3:

Mode 1 requires no further steps than the creation of a common test suite.

Mode 2 is realized in the following way:

- a. In the parse grammar, define a subdirectory 'constructions', populated by the type files and the relevant lexicon(s) of the language chosen, except the file `general.types.tdl`, since its definitions are already covered in the parse grammar.
- b. To secure that each attribute is introduced with a unique type, the attribute names in the Daemon will be largely distinct from those used in the parse grammar, since in the Daemon, information is much less complex than in the parse grammar.
- c. Two root values are defined in the parse grammar, one being the root category of the parse grammar, and one the root category of the Daemon.

Mode 3 requires the same steps as mode 2, and in addition:

- d. In the type declaration for 'sign', add a feature 'DAEMON' with value 'sign-min', which will take as value the Daemon FSs plus correspondence declarations.
- e. In the parse grammar, for those verb lexeme types for which one wants to display the Daemon correspondence, introduce a type file defining subtypes of these types which (i) provide a slot for the Daemon FS, and (ii) state exactly which feature paths in the parse FS provide counterparts of the relevant values in the Daemon FS.
- f. Corresponding to these 'Daemon'-related lexeme types, introduce a lexicon file with the verbs used in the shared test file, now defined as items of the 'Daemon'-related types.

Mode 3 has been realized for a small parse grammar for Ga, based on the HPSG Grammar Matrix. Here, step e. is implemented through the file '`ga-daem.tdl`', and step f. through the file '`lexicon-ga.tdl`'. Thus, in the '`Ga grammar/lkb/script`' file, the following lines are active on mode 3,

```
(lkb-pathname (parent-directory) "ga-daem.tdl")
(lkb-pathname (parent-directory) "constructions/syntactic.types.tdl")
(lkb-pathname (parent-directory) "constructions/semantic.types.tdl")
(lkb-pathname (parent-directory) "constructions/construction.types.tdl")
(lkb-pathname (parent-directory) "constructions/ga-construction.types.tdl")
....
(lkb-pathname (parent-directory) "lexicon-daem.tdl")
(lkb-pathname (parent-directory) "constructions/lex.constr.ga.tdl")
```

and are commented out to return the grammar to its mode 1 operation (then also deleting the extra 'Daemon' line in roots.tdl). To go from mode 3 to mode 2, only ga-daem.tdl and lexicon-ga.tdl are commented out (leaving roots.tdl with both root definitions).

References

- Bender, Emily M., Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starterkit 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 Esther Kropp, 2004: Ga clauses without syntactic subjects. *Journal of African Languages and Linguistics* 25.1: 1-40.
- Essegbey, James. 1999. Inherent Complement Verbs Revisited. MPI Series in Psycholinguistics, Nijmegen.
- Goldberg, Anne. *Constructions. A Construction Grammar Approach to Argument Structure*. Chicago University Press, Chicago.
- Hellan, Lars., Lars Johnsen and Anneliese Pitz. 1989. The TROLL Lexicon. Ms, NTNU.
- Hellan, Lars and Dorothee Beermann. 2005. Classification of Prepositional Senses for Deep Grammar Applications. In Kordoni, Valia and Aline Villavicencio (eds) *Proceedings of the Second ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*. University of Essex.
- Sag, Ivan A., Thomas Wasow and Emily Bender. 2003. *Syntactic Theory. A Formal Introduction*. CSLI Publications, Stanford.